DETECTING AND LEVERAGING CHANGES IN TEMPORAL DATA

A Dissertation
Presented to
The Academic Faculty

By

Nauman Ahad

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering
College of Engineering

Georgia Institute of Technology

August 2024

DETECTING AND LEVERAGING CHANGES IN TEMPORAL DATA

Thesis committee:

Dr. Mark Davenport, Advisor
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Eva Dyer
Biomedical Engineering
Georgia Institute of Technology

Dr. Yao Xie
Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Omer Inan
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Justin Romberg
Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Sharon Sonenblum
School of Nursing
Emory University

Date approved: June 14, 2024

ACKNOWLEDGMENTS

The overarching lesson that I have learned at the end of graduate school is that a doctorate program, more than anything else, can be a lengthy test of persistence with sparse rewards. These extrinsic rewards have been well compensated by intrinsic rewards that I have learned to find in exploring new topics and navigating failures. This essential skill was forged thanks to the supportive community I was lucky to be a part of.

First and foremost, I would like to thank my advisor, Dr. Mark Davenport. I have been fortunate to have an advisor who has been extremely supportive in all possible facets of graduate school. Mark, thank you for providing an environment where we have been encouraged to freely explore and set our own research agendas. The open environment in our lab has never made me hesitant to discuss any issue, and that has been immeasurably valuable, especially for international students like myself. I have had a wonderful mentee experience that I feel is important to pass on. Thank you for giving me a chance with the WISAT project that provided one of the most important change-points in my life. There are too many things to mention here, but (almost) ever day in graduate school has been both exciting and inspiring. Thank you for making my graduate school experience so enriching.

Speaking of WiSAT, I would like to thank Dr. Sharon Sonenblum for providing me the opportunity to work on such a challenging yet stimulating project. Dr. Sonenblum, thank you for being patient with while I learned how to work with human subject data. The lessons learned while working on this project have motivated many of the proposed methods presented in this thesis. These lessons would continue to guide me as long as I work on datasets collected in real-world settings.

I would like to express my sincere thanks and gratitude to Dr. Yao Xie, whose help has been foundational in shaping this thesis. Dr. Xie, thank you for your time and patience in teaching me about change-point detection and all it's associated machinery. Your help is the sole reason I was able to finish the symmetrical CUSUM project. Our work seeded ideas that helped shape the

subsequent chapters in this thesis. For all this, I am very thankful.

I was fortunate to have had the chance to work with Dr. Eva Dyer in the latter stages of graduate school. Eva, thank you for your time and patience as we explored different research directions. Though these paths haven't always been fruitful, the exploratory wandering has been invaluable. Thank you for pushing me when I most needed pushing. Your energy, generosity, and dedication to research is inspiring. Thank you for providing me the opportunity to work closely with the NerDS lab. These collaborations have helped me broaden my horizons for which I am very thankful.

I would like to thank Dr. Justin Romberg and Dr. Omer Inan for agreeing to serve on my committee. Dr. Romberg, thank you for teaching and designing great courses that have provided me the foundations to conduct research here at Georgia Tech. Omer, thank you for providing me opportunities to apply what I have learned on challenging real-world problems, and more importantly, for teaching me how to effectively communicate with a wider audience.

Our lab-group and the wider COTN community have been instrumental in transforming by initial socially impoverished graduate school life into an experience that has been supportive, engaging, and filled with a strong sense of community. Liangbei Xu, Austin Xu, Namrata Nadagouda, Peimeng Guan, Santosh Karnik, Matt O'Shaughnessy, Rakshith Sharma, Andrew McRae, Tomer Hamam, Jihui Jin, Mehdi Abazou, Carolina Urzay, and others that I might have missed; It has been very rewarding to engage in conversations that have ranged from research ideas to whether terms such as "Chai Tea" or "Naan Bread" form an acceptable body of culinary nomenclature. The hikes, the potlucks, the video games and the free-food scavenging hunts have been a pleasure. You have all made graduate school fun. Thank you!

Outside the lab, I was always fortunate to have great roommates that have helped me get a semblance of home away from home. Asif, Usama, and Nathan: thank you for making 511 Lynch the warm place it was. Bastian, the Colo-Colo games, the Spanish lessons, and the weekly BeltLine bike rides to Little Tart were very helpful in preserving my sanity. These last few years would have

been very difficult without all these. Thank you for being a great roommate and for introducing me to the rich world that is South America.

Growing up, I never had to leave Pindi and live on my own before starting graduate school. Atlanta and Georgia Tech have provided me the space to establish my own little home for a major part of the past decade. This has been made possible by the kindness and generosity extended by numerous people, especially the Pakistani community at Georgia Tech. I am very thankful to this community for all the support it has provided.

Finally, I would like to thank my family. My elder brother Hasaan who has always been a shield to deflect my troubles. Thank you for always being there. Azka, who has always been extremely supportive. My niece and nephew, Aleena and Taimoor, for bringing so much joy in our lives. To dado, Fatima, and dadabbu, Zahoor-ul-Haq, whose prayers are still guiding me, long after they have departed this world. And to mama, Shabana, and baba, Abdul Ahad, words can't express my gratitude for your countless sacrifices. You have always made sure we are provided the right environment to succeed and pursue our interests. This work is dedicated to you.

SUMMARY

Many tasks in statistics and machine learning depend on accurately modeling data distributions. In real-world settings, data distributions often evolve with time and many applications require us to detect when data distributions change. The statistics community has extensively developed methods that allow us to detect these changes. Distributional changes are also relevant for time series machine learning tasks as class labels often change when underlying distributions change.

This thesis explores the interplay between change-point detection and machine learning. We propose new methods which show how change-point detection can be of benefit to machine learning, and in return, how machine learning can be of benefit to change-point detection. We discuss these contributions in four main chapters.

In the first chapter, we propose a new sequential change-point detection method called "Data adaptive symmetrical CUSUM" (DAS-CUSUM) [1, 2]. The change statistics produced by DAS-CUSUM are symmetrical which means that the change statistic for a change from distribution θ_0 to distribution θ_1 would be equal to the change from θ_1 to θ_0 .This makes it easier to set a detection threshold for detecting multiple change points. In the second chapter, we show how unsupervised change point detection methods can provide weak supervision for time series classification tasks. This weak supervision can be used in conjunction with available labels for semi-supervised time series classification which can help alleviate challenges in obtaining expensive labeled data [3]. In the third chapter aim we propose a method that leverages supervision from true change-points to devise improved change-point detection methods [4, 5]. We show that this supervision can be used to learn a ground metric. This learned metric can be incorporated into change-point detection methods which helps these methods ignore certain *kinds* of changes which are not of interest to us. This is particularly useful when change-points have to be detected in multi-variate time series signals. In the fourth chapter we propose a new method for unsupervised domain adaption that we call "Selective screening and signal selection for time series domain adaptation" (SSSS-

TSA). Supervised machine learning models often fail when the distribution of unlabeled test data is different from labeled training data, Unsupervised domain adaptation methods try to adapt these trained models to unlabeled test data. Different time series channels have different severity of distribution shifts between training and target domains. Our method can ignore channels with larger shifts. This leads to superior time-series domain adaptation performance.

These methods collectively show that change-point detection and machine learning methods are mutually beneficial to each other.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

A fundamental task in statistics and machine learning is to understand *properties* of collected data. One way to infer these properties is to model the data's underlying distribution. Such models provide us tools to summarize data, to make predictions, and to compare how different sets of samples are related. These comparisons can help us identify anomalous data samples.

We can illustrate this through an example. Consider a clinical trial studying a drug's effect on blood pressure. Researchers can measure blood pressure before and after drug administration, and model the difference in these readings as a normal distribution. This distribution model serves multiple purposes: it allows us to provide probabilistic predictions of a certain blood pressure reduction for future patients. We can create a similar distribution model for a placebo group and compare the two. This statistical distance between these models helps us gauge how effectively the drug reduces blood pressure compared to the placebo. If the statistical difference between the drug and placebo distributions is substantial, the drug model can be used to flag unusual data points from other drugs, potentially indicating patients who might have taken a different drug.

Tools that model relationships between data samples are also of fundamental importance in machine learning. Supervised machine learning methods leverage optimization tools on large datasets to train models that find associations between data and provided categories of interest. Similarly, unsupervised methods try to learn models that cluster data into homogeneous regions such that *similar* data is expected to be closer than *dissimilar* data in terms of statistical distance. Some machine learning models, like variational autoencoders, explicitly model input data as normal distributions [6]. The input data is encoded by these models in such a way that similar data points are associated with distributions which are relatively closer. For example, the distribution associated with an input MNIST image sample for digit 1 would be closer to the distribution associated with

input sample of 7 than the distribution associated with the number 6. Supervised labels are used to learn associations between these encoded data distributions and provided class labels to solve classification tasks. Other machine learning models, such as vanilla auto encoders, SVMs, and neural networks directly model relationships between data samples and associated class categories without explicitly modeling data probability distributions. These models still implicitly model data distribution information when learning class association relationships.

In real-world settings, data distributions and subsequently, data properties *change* over time. This is particularly true for time series datasets where samples at different instances have different distributions. Consequently, these different distributions are associated with different class labels. In temporal sequences, changes in distributions give meaning to different data categories. A DC signal on its own doesn't carry much value unless there is subsequent change in the signal over a period of interest.

Changes in data distributions can also cause machine learning models to fail. This happens because the relationships learned between data and labels might not hold true when the underlying data patterns change. As a result, the model struggles to perform well on new data, failing to generalize in real-world deployed settings.

Many events of interest are also associated with changes in data distribution. For example, cyber attacks are accompanied by changes in network performance metrics. Abnormalities in industrial processes are accompanied with changes in key process metrics such as pressure and temperature. These scenarios require us to devise methods that can identify changes in data distributions.

1.1 Change-point detection

Change-points are time instances where there is a change in the underlying distribution. For example, Figure 1.1 shows a sequence in which initial points $x_1, x_2, ..., x_{n_c-1}$ are generated through distribution f_{θ_0}, where f could be a class of distributions, such as normal distribution, and θ are the

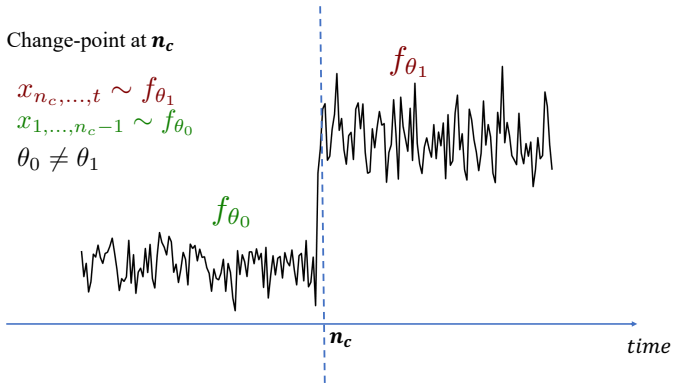


Figure 1.1: An example of a Change-point at time instance n_c

associated distribution parameters such as mean and variance. At time instance n_c, the data generating distribution changes and subsequent data points $x_{n_c}, x_{n_c+1}..$ are generated from f_{θ_1}. The task of change-point detection is to identify the time instance, n_c in this example, where the data generating distribution changes.

Change-point detection has wide applications in time series analysis. They are often used to segment time series data in intervals where data follows uniform properties(as they are generated from the same distribution) [7]. Other applications, such as anomaly detection, utilize change-point detection to identify instances of interest [8].

Broadly speaking, change-point detection is often categorized into two main categories: (1) Offline change-point detection and (2) Online change-point detection,

In the offline setting, the complete sequence of interest is available to us when processing and detecting change points [9]. In online settings, change-points have to be detected in newly revealed data in streaming settings. In such streaming settings minimizing detection delay is critical.

Despite being developed almost 70 years ago, CUSUM is still a widely popular method for detecting online change points [10].Based on sequential hypothesis tests of the log likelihood ratio between post-change and pre-change data distributions, it's optimality in minimizing detection

delay for a given false positive rate have resulted in its wide usage, and numerous methods in recent past continue to extend it for modern use cases [11].

However, such *quickest* change-point detection methods are often used to detect a single change point. In many real-world applications, *multiple* change-points need to be detected in the quickest way. Such settings often require change-point detection methods to run with minimal intervention, giving rise to new challenges on how to set detection threshold for identifying multiple change points in streaming settings in the quickest possible way. This leaves room for developing improved methods that can effectively detect multiple change points quickly.

Another class of online change-point detection methods employ a statistical distance between distributions computed over two sliding windows to detect change points [7, 12]. For such methods, the focus is not on detecting change points in the quickest possible manner. Instead, the focus is on detecting multiple change points with a fixed detection delay, with the detection delay equivalent to the window size being used.

What all these change-point detection methods have in common is that they are all fundamentally *unsupervised*. These methods try to detect how different two sets of samples are when deciding if a change has occurred or not. In modern applications, with time series being very high dimensional, what *kinds* of change-points to detect can be very subjective. Some applications might require us to ignore certain *types* of changes in some dimensions while focus on identifying certain other kind of changes. There has been very little work that explores how this can be done. Most change-point detection methods detect all types of changes. This leaves room to develop new class of change detection methods that can incorporate information on what kind of changes should be detected and what kind of changes should be ignored.

As we discussed in the previous section, change-points are often associated with changes in class categories. These changes provide information on how class labels change, information that can be beneficial for machine learning tasks.

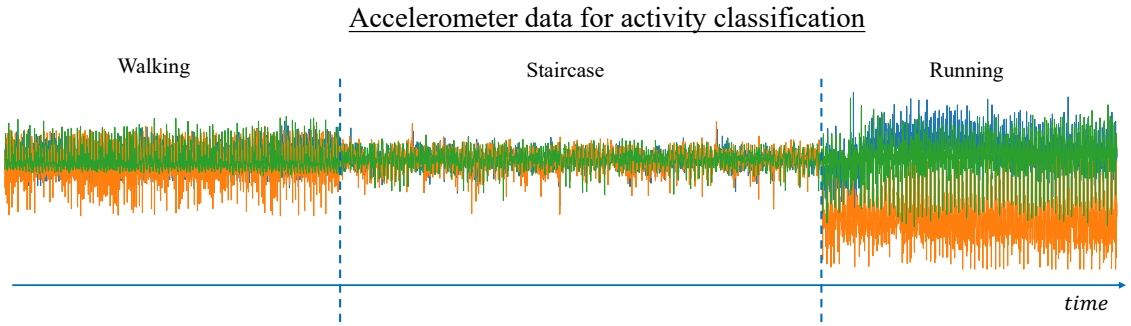


Figure 1.2: Accelerometer time series data where classes switch from walking to going down a staircase to running. Supervised machine learning methods are required to classify data from each class correctly.

1.2 Distribution changes and machine learning for temporal data

In the past few decades, the field of machine learning has become a dominant tool for identifying patterns in data. For time series datasets, the categories of interest,and as a consequence corresponding data distributions, change over time.

However, machine learning methods are mostly *supervised*. They require extensive examples of what types of patterns need to be identified in time series sequences. Supervision from vast labeled examples on the internet are at the heart of recent explosion in image classification and large language models. Such extensive labeled examples are unfortunately not available when developing time series classification models. As compared to image classification tasks, where the underlying structure of natural images is similar, different time series classification tasks can be very unrelated. Additionally, obtaining curated labeled datasets for training classifiers can be much more challenging. Different applications require application specific labels, the nature of which can vary a lot of between different applications. For example, the data associated with the task of distinguishing abnormal heart beats from normal heart beats is a very different problem than accelerometer data used for distinguishing running from bicycling. This underscores the need to devise machine learning methods which are label efficient. There has been extensive work over the past few decade exploring semi-supervised, and more recently, self-supervised methods to devise

label efficient machine learning models [13].

Machine learning methods also assume that data distributions at train time and test time are identical. A classifier that learns a function g_ϕ to map input X to output Y, i.e. $g_\phi(X) = Y$ makes the underlying assumption that the relationship between inputs X and class categories Y remains unchanged. These methods also make the assumption that the input output pairs (X and Y) contained in datasets used to train machine learning models capture all possible type of input-output pairs a machine learning model would encounter when deployed in real world settings. This often isn't true as data encountered at test time can be different than data used for training. For example a machine learning model trained to classify walking from jogging can fail when it is used on new users who jog at a different pace. Such *distribution shifts* can cause machine learning models to fail catastrophically [14].

1.3 Interplay between change-point detection and machine learning

Our discussion in the previous sections suggests that there are common themes between change-point detection and machine learning. Figure 1.1 and Figure 1.2 show that both machine learning and change detection require recognizing differences in data distributions. Change-point detection is concerned with detecting changes/separability in temporally adjacent distributions, while machine-learning methods are concerned with finding separability between different distributions encountered in a training set (either through supervision for distinguishing categories or unsupervised clustering methods).

This common goal of finding separability between distribution provides promising grounds for exploring ways in which machine learning and change point-detection can be mutually beneficial to each other.

This thesis explores the interplay between machine learning and change-point detection, and uses this interplay to devise new methods that utilize change-point detection for improving machine learning and in the reverse case, utilizes machine-learning to devise improved change point

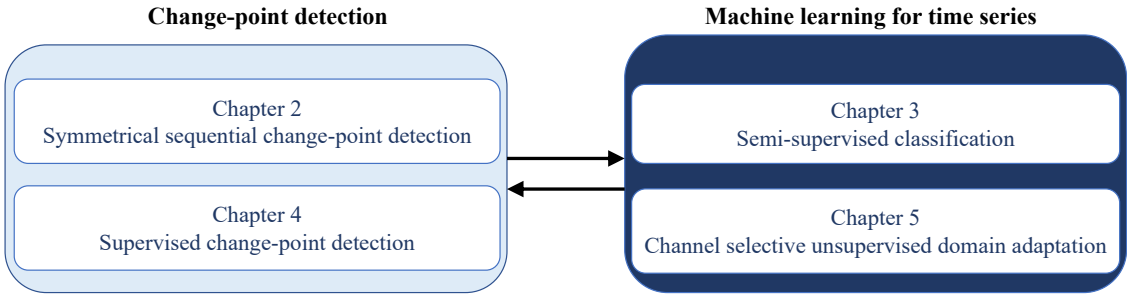


Figure 1.3: Overview of thesis - this thesis explores how improving change point detection can improve machine learning methods and in return, how can machine learning help improve change-point detection

detection methods.

Figure 1.3 provides an overview of this thesis. Many of these contributions are a product of exploring this mutually beneficial relation between change-point detection and machine learning models which are represented by by the arrow signs.

This thesis consists of 4 main chapters, each of which that explores this interplay, and sees how these can be mutually beneficial each other.

Improving Classical Change-point Detection Methods

Chapter 2: Data-adaptive Symmetrical CUSUM for Change-point Detection

Detecting change-points sequentially in a streaming setting, especially when both the mean and the variance of the signal can change, is often a challenging task. A key difficulty in this context often involves setting an appropriate detection threshold, which for many standard change statistics may need to be tuned depending on the pre-change and post-change distributions. This presents a challenge in a sequential change detection setting when a signal switches between multiple distributions. For example, consider a signal where change-points are indicated by increases/decreases in the mean and variance of the signal. In this context, we would like to be able to compare our change statistic to a fixed threshold that will be symmetric to either increases or decreases in the mean and variance. Unfortunately, change-point detection schemes that use the log-likelihood ratio, such as CUSUM and GLR, are quick to react to changes but are not symmetric when both

the mean and the variance of the signal change. This makes it difficult to set a single threshold to detect multiple change-points sequentially in a streaming setting.

In this chapter we propose a modified version of CUSUM that we call Data-Adaptive Symmetric CUSUM (DAS-CUSUM). The DAS-CUSUM change-point detection procedure is symmetric for changes between distributions, making it suitable to set a single threshold to detect multiple change-points sequentially in a streaming setting. We provide results that relate to the expected detection delay and average run length for our proposed procedure. Extensive simulations are used to validate these results. Experiments on real-world data further show the utility of using DAS-CUSUM over both CUSUM and GLR.

Leveraging Change-point Detection for Improving Machine Learning

Chapter 3: Semi-supervised Sequence Classification through Change-point Detection

Sequential sensor data is generated in a wide variety of real-world applications. A fundamental machine learning challenge involves learning effective classifiers for such sequential data. While deep learning has led to impressive performance gains in recent years within domains such as speech, this has relied on the availability of large datasets of sequences with high-quality labels. In many applications, however, the associated class labels are often extremely limited, with precise labeling/segmentation being too expensive to perform in a high volume. However, large amounts of unlabeled data may still be available. In this paper we propose a novel framework for semi-supervised learning in such contexts. In an unsupervised manner, change-point detection methods can be used to identify instances where classes change within in a sequence. We show that change points provide examples of similar/dissimilar pairs of sequences which, when coupled with class labels, can be used in a semi-supervised classification setting. Pairs from labels and change points are used by a neural network to learn improved representations for classification. We provide extensive synthetic simulations and show that the learned representations are better than those learned through an autoencoder and obtain improved results on simulations and human activity recognition datasets.

Leveraging Machine Learning for Improving Change-point Detection

Chapter 4: Learning Sinkhorn Divergences for Change-point Detection

As discussed previously, most existing methods for change point detection are unsupervised and, as a consequence, lack any information regarding what kind of changes we want to detect or if some kinds of changes are safe to ignore. This often results in poor change detection performance. In this chapter we present a novel change point detection framework that uses true change point instances as supervision for learning a ground metric such that Sinkhorn divergences can be then used in two-sample tests on sliding windows to detect change points in an online manner. Sinkhorn divergences are entropic regularized variants of Wasserstein distances [15]. As Wasserstein distances incorporate a ground metric, similar dissimilar pairs from change-points provide a natural way to incorporate supervision through a learned metric. Our method can also be used to learn a sparse metric which can be useful for both feature selection and interpretation in high-dimensional change point detection settings. Experiments on simulated as well as real world sequences show that our proposed method can substantially improve change point detection performance over existing unsupervised change point detection methods using only few labeled change point instances.

Adaptaing Temporal Machine-Learning Models to Distributional Changes

Chapter 5: Channel Selective Unsupervised Domain Adaptation

We also previously discussed how machine learning models fail when data distributions change. Though we can use change-point detection methods for detecting when machine learning models fail, the end goal is to develop models that adapt to distributional changes. This is the goal of unsupervised domain adaptation. Unsupervised domain adaptation methods try to use available training data in conjunction with unlabeled test data to improve model generalization in test time settings. Such methods involve learning a model that tries to minimize classification loss on available training data while simultaneously minimizing a statistical distance between the labeled training and unlabeled test time datasets.

Building generalizable and robust unsupervised domain adaptation methods for multivariate

time series presents significant challenges, particularly due to variations in channel-level information and shifts between training and testing sets. To overcome these obstacles, we introduce a method called *Signal Selection and Screening via Sinkhorn alignment for Time Series domain Adaptation* (SSSS-TSA) Our approach hinges on aligning learned representations of individual channels while simultaneously aligning a pooled global representation across all channels. This dual alignment strategy not only ensures effective adaptation to new domains but also maintains robustness in scenarios with training and testing set shifts or when certain channels are absent or corrupted. We evaluate our method on several time-series classification benchmarks and find that it consistently achieves state-of-the-art performance. Furthermore, our results demonstrate the importance of adaptively selecting and screening different channels to enable more effective alignment across domains.

We end this dissertation with a chapter that discusses possible future directions in Chapter 6.

CHAPTER 2

DATA-ADAPTIVE SYMMETRIC CUSUM FOR SEQUENTIAL CHANGE DETECTION

2.1 Introduction

For a sequence of observations $x_1, \ldots, x_t$, the goal of change-point detection is to detect whether there exists an instance n_c such that $x_1, \ldots, x_{n_c-1}$ are generated according to a different distribution than $x_{n_c}, \ldots, x_t$, and if so, estimating n_c. This is typically accomplished by computing a simple change statistic based on the log-likelihood ratio, which can be compared to a threshold to detect changes or optimized to estimate n_c. Sequential change-point detection involves sequentially detecting multiple changes in streaming data. Many real-world world applications require sequential detection of change-points within streaming signals. Healthcare, communication, and finance are just a few areas where sequential change detection is widely used [16–18]. An extended discussion of applications of change-point detection can be found in [7].

Despite being devised more than half a century ago, the CUSUM statistic is still one of the most popular methods for detecting change-points [10]. This is chiefly due to two reasons. First, it has a simple recursive implementation which makes it computationally efficient to apply. Second, it has been shown to be optimal in minimizing the detection delay for a given false alarm rate [19]. However, computing the CUSUM statistic requires complete knowledge of both the pre-change and post-change distributions. This is not feasible in many real-world scenarios where the post-change distribution can be unknown. In such settings, a more common approach is to use the GLR statistic, which involves estimating the post-change distribution for all possible change-points [20]. Both the CUSUM and GLR statistics leverage the log-likelihood ratio for the known/estimated pre- and post-change distributions.

Work proposed in this chapter was published in [1]. This work was motivated by our parallel work that involved developing an activity tracker for wheelchair users [2].

Most work on change-point detection has focused on identifying a *single* change-point in the quickest possible manner. Though this has been useful for some applications, especially those that monitor a process for abnormal behavior such as machine fault detection and network intrusion detection, many modern applications require the detection of *multiple* change-points sequentially in streaming data. In sequential change-point detection, the detection procedure must be restarted and continued after each change-point is detected, resulting in multiple change-points being detected. Examples of such settings include segmentation of signals for activity recognition where change-points are used to identify transitions from one activity to another in a streaming setting [21]. In such settings, the pre-change and post-change distributions themselves change after each change-point and cannot be assumed to be known *a priori*. This presents a significant challenge to most standard change detection approaches because the detection threshold must be set without any knowledge of these distributions (with the threshold typically being fixed in advance and held constant throughout the procedure).

The machine learning community has been addressing this problem of identifying multiple change-points in data streams [12]. Such works show that procedures employed to detect change-points should be *symmetric*. This means the magnitude of a change from a distribution θ_0 to a distribution θ_1 should be the same for a change from θ_1 to θ_0. Using a procedure that has a similar power in detecting such changes makes it easy to select a threshold for detecting multiple changes sequentially. Statistics such as the GLR and CUSUM are not symmetric when distribution changes involve a change in variance. This makes it difficult to use these in detecting multiple changes.

In this chapter, we present an adaptive symmetric version of CUSUM that we call Data-Adaptive Symmetric CUSUM (DAS-CUSUM). DAS-CUSUM uses a window to estimate the post-change distribution and employs a symmetric change statistic to make it easier to select a fixed threshold to detect multiple change-points in streaming data. We provide theoretical results for our proposed method that relate the expected detection delay (EDD) (average delay in detecting true changes) to the average run length (ARL) (average time until a false alarm occurs).

The rest of this chapter is organized as follows. After reviewing related literature in Section 2.2, we formalize the change detection problem in Section 2.3 and further motivate the need to have a symmetric change statistic for detecting multiple changes. Section 2.4 provides a description of the proposed procedure. Theoretical results that relate EDD versus ARL are described in Section 2.5, where a sketch of the related proofs is also given. Section 2.6 contains simulations that empirically validate the theoretical results in a practical setting. Experiments on real-world data are summarized in Section 2.7.

2.2 Related work

The CUSUM statistic is known for being asymptotically optimal in minimizing the maximum average detection delay as the average time to false alarm reaches infinity [19]. CUSUM was later shown to be optimal in minimizing the expected detection delay for a provided (non-asymptotic) expected time to false alarm [22]. There has been extensive work done to further investigate and generalize the optimality property of CUSUM. These results, however hold when both pre-change and post-change distribution are completely known. A summary of such work can be found in [23]. A two-sided CUSUM test can be used to detect either an increase or decrease in mean [24], but this approach still assumes a fixed and known variance. When the post-change distribution is unknown, the generalized log-likelihood ratio test (GLR) can be used by estimating both the change location and the post-change distribution through maximum likelihood estimation. However, CUSUM, GLR, and their variants are often used to detect only a *single* change-point [8]. The few works that do use these methods to detect multiple changes do so by only detecting changes in the mean of normally distributed data [25, 26]. It is more challenging to detect multiple changes when both the mean and the variance of a signal change. There is limited prior work that detects joint changes in both the mean and the variance of the signal [27], however, this has not been considered in the context of detecting multiple changes.

Recently, there has been increasing interest in the machine learning community to detect mul-

tiple change-points sequentially within streaming data [12, 28–30]. Most of these methods use non-parametric change statistics, which are symmetrical. This means that the magnitude of the change statistic for a change from θ_0 to θ_1 is equivalent in magnitude for a change from θ_1 to θ_0. The need for this symmetrical statistic was noted by [12], who use a symmetric KL-divergence to detect multiple changes within streaming data where both the mean and variance of the normally distributed signal are changing. The symmetric statistic makes it easy to set a single detection threshold before the procedure is started to detect multiple changes within streaming data. At each time instance, a pre-change distribution is estimated using a "past window," and the post-change distribution is estimated using a "future window." These methods, however, do not incorporate data samples directly. These samples are incorporated through estimates of the distribution, which makes these methods slow to react to changes. None of these methods characterize the relationship between detection delay and false alarm rate.

The need to use symmetric statistics for change detection was also earlier noticed in [31–33], where the authors noted the asymmetry in change statistics when there are changes in both mean and variance. These works used a log-likelihood ratio with a drift term to make the expected value of the change statistic symmetric under the post-change distribution. However, this drift term meant that the expected value of the statistic is zero under the pre-change distribution, which can lead to more false positives. A slightly modified version of this technique was mentioned in [34], where false alarm rates were reduced by adding a fixed drift term which made the expected value of the statistic negative under the pre-change distribution. However, no details were provided about setting this drift term. These methods also provided no characterization of the relationship between detection delay and false alarm rate.

In this work, we investigate a suitable choice for this fixed drift to make the statistic symmetric under the post-change distribution while also ensuring the expectation is negative under the pre-change distribution. Our proposed change detection procedure provides a symmetric change statistic for different families of probability distributions, however, the theoretical results relating

detection delay and false alarm rate consider the more restricted setting of i.i.d. univariate normally distributed data.

2.3 Problem statement

Change-points are instances in a signal where the underlying distribution of data changes, e.g., the parameters of the signal-generating distribution change from θ_0 to θ_1. Most change-point detection methods rely on hypothesis tests based on the log-likelihood ratio. Specifically, suppose we are given a sequence of observations $x_0, \ldots, x_t$. We will assume that each element x_i is drawn independently from a distribution f_θ where θ represents some (possibly changing) parameters. To detect a change we compare the null hypothesis ($\mathcal{H}_0$) that all x_i are drawn according to f_{θ_0} for some (known) θ_0 to the alternate hypothesis ($\mathcal{H}_1$) that the time series distribution changes from f_{θ_0} to f_{θ_1}, at time n_c, for some $\theta_1 \neq \theta_0$.

The likelihood of X under these two hypotheses is given by $\prod_{i=1}^{t} f_{\theta_0}(x_i)$ (under $\mathcal{H}_0$) and $\prod_{i=1}^{n_c-1} f_{\theta_0}(x_i) \prod_{i=n_c}^{t} f_{\theta_1}(x_i)$ (under $\mathcal{H}_1$), respectively. By computing the likelihood ratio and taking the logarithm, we obtain the likelihood-ratio statistic at instance t for a possible change-point at n_c:

$$\ell_{n_c}^{t} = \sum_{i=n_c}^{t} \log \frac{f_{\theta_1}(x_i)}{f_{\theta_0}(x_i)}.$$

Since the location of the change-point n_c is unknown, the maximum over all possible change-point locations are taken to compute the change statistic at instance t:

$$\ell^{t} = \max_{1 \leqslant n_c < t} \ell_{n_c}^{t}. \tag{2.1}$$

In (2.1), we are maximizing over n_c to find the maximum log-likelihood ratio. Instead of maximizing (2.1) with respect to n_c, we can also maximize the log-likelihood ratio by minimizing, over n_c, the expression:

$$\ell^t = \sum_{i=1}^{t} \log \frac{f_{\theta_1}(x_i)}{f_{\theta_0}(x_i)} - \min_{1 \leqslant n_c < t} \sum_{i=1}^{n_c} \log \frac{f_{\theta_1}(x_i)}{f_{\theta_0}(x_i)}. \tag{2.2}$$

The CUSUM statistic [10] provides a computationally attractive recursive implementation of the test in (2.2). It assumes both pre-change parameters θ_0 and post-change θ_1 distribution parameters are known. In such a setting, a recursive implementation of (2.2) can be obtained as shown below in (2.3):

$$S_t = S_{t-1}^{+} + \log \frac{f_{\theta_1}(x_t)}{f_{\theta_0}(x_t)}, \tag{2.3}$$

where $(x)^+ = \max(0, x)$ and $S_0 = 0$. The detection procedure is a stopping time T; a change-point is detected at the first time when the detection statistic S_t exceeds a pre-set threshold b:

$$T = \inf\{t > 0 : S_t > b\}. \tag{2.4}$$

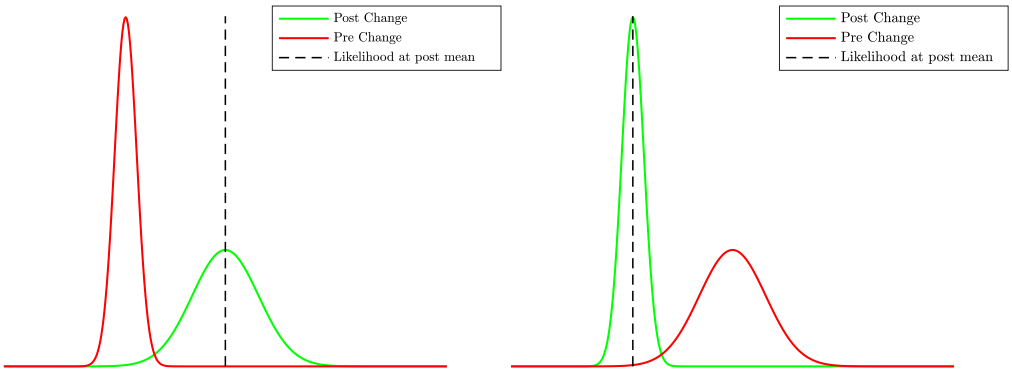
The post-change distribution is often unknown in real-world settings. In such cases, the GLR [20] can be used to obtain the change statistic ℓ_t. GLR maximizes the change statistic in (2.5) over both the post-change distribution, θ_t at instance t, and the change instance n_c. Let

$$\ell_{n_c}^t := \max_{\theta} \sum_{i=n_c}^{t} \log \frac{f_{\theta}(x_i)}{f_{\theta_0}(x_i)}.$$

And the GLR detection statistic is defined as

$$\ell^t = \max_{1 \leqslant n_c < t} \ell_t^{n_c}. \tag{2.5}$$

Once the change statistic, ℓ_t, crosses the threshold b, a change is detected at a similarly defined stopping time T as in (2.4), and the estimated change-point location n_c^* corresponds to the maximizing parameter at T. The corresponding post-change estimate $\hat{\theta}_T^{n_c^*}$ is used as the new pre-change esti-



(a) $\mathcal{N}(1,1) \rightarrow \mathcal{N}(10,3)$ (b) $\mathcal{N}(10,3) \rightarrow \mathcal{N}(1,1)$

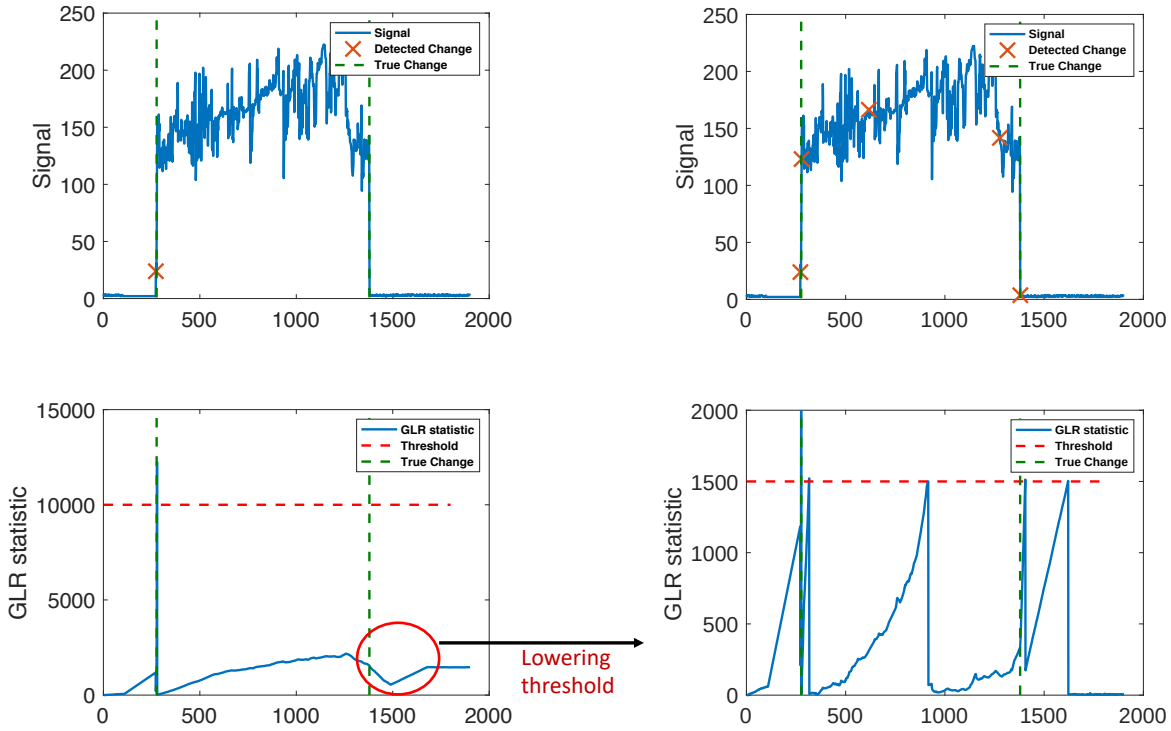
Figure 2.1: Joint changes in mean and variance lead to asymmetric Likelihood Ratios. In Figure 2.1a, the pre-change likelihood is in the tail, leading to a large likelihood ratio. In Figure 2.1b, the post-change likelihood is higher than it is in Figure 2.1a, leading to a relatively smaller likelihood ratio

mate θ_0 and the sequential change-point detection procedure is repeated to detect the next change. This way *multiple* change-points are detected. It is important to note that the GLR procedure is non-recursive and can be computationally expensive to run.

2.3.1 Asymmetry of log-likelihood ratio

The log-likelihood ratio statistic, employed by GLR and CUSUM, is quick to react to changes but is asymmetric for detecting joint changes in mean and variance. Figure 2.1 illustrates this asymmetry. This difference becomes more pronounced when one of the two distributions has a much smaller variance.

Figure 2.2 shows a real-world example where this asymmetry makes it difficult for GLR to detect multiple change-points. The log-likelihood ratio for the first change-point is much larger than that for the second change-point. This makes it difficult to set a detection threshold *a priori* to detect multiple change-points in a streaming data setting. In the first sub-figure, the fixed detection threshold misses the second change-point, which has a much smaller statistic. As seen in the second sub-figure, a blackuction in the detection threshold leads to many false change-point detections.



(a) $\mathcal{N}(1,1) \rightarrow \mathcal{N}(10,3)$ (b) $\mathcal{N}(10,3) \rightarrow \mathcal{N}(1,1)$

Figure 2.2: Joint changes in mean and variance lead to asymmetric likelihood ratio. In Figure 2.2a, the likelihood ratio (in GLR) for the second change is much smaller than the likelihood for the first change. This can lead to a missed change-point when the detection threshold is set to be large. When the detection threshold is loweblack to detect this missed change-point, many false change-points are detected, as shown in Figure 2.2b.

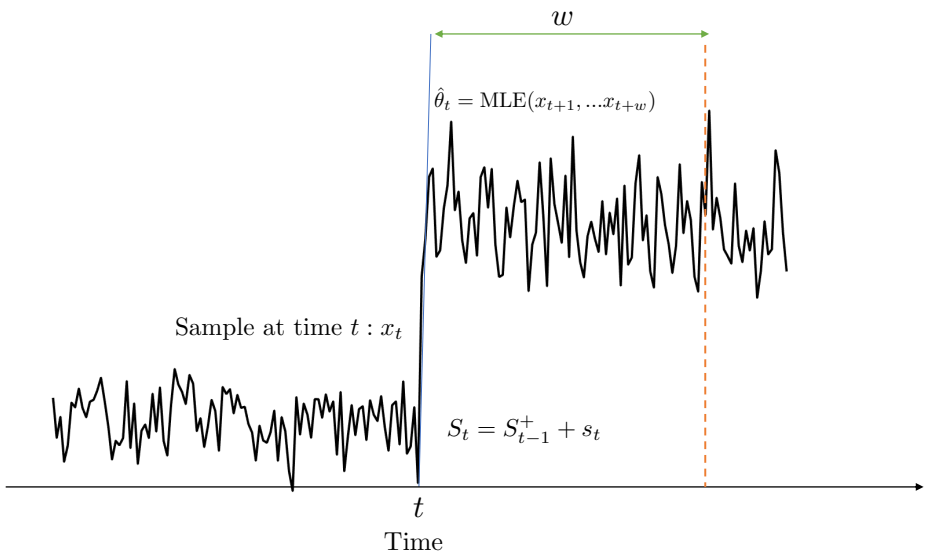


Figure 2.3: Adaptive version of CUSUM. Using a "future" window to estimate post-change parameters $\hat{\theta}_t$ could be used in place of post-change distribution θ_1 for CUSUM update.

2.4 Data-adaptive symmetric DAS-CUSUM

2.4.1 Adaptive post-change estimation

When the post-change distribution is unknown, another way to estimate the post-change distribution is to use a window of size w to perform a sequential estimate of the post-change parameters $\hat{\theta}_t$ at time t for the CUSUM statistic S_t, called the window-limited CUSUM in [11] and used in [35] where a window is used to estimate post-change distribution change distribution for subspace change detection. Figure 2.3 illustrates the procedure. For normally distributed i.i.d. data, the post-change distribution estimate $\hat{\theta}_t = (\hat{\mu}_t, \hat{\sigma}_t^2)$ at time t can be calculated conveniently as

$$\hat{\mu}_t = \sum_{i=t+1}^{t+w} x_i, \quad \hat{\sigma}_t^2 = \sum_{i=t+1}^{t+w} \frac{1}{w}(x_i - \hat{\mu}_t)^2.$$

Using "future" samples to calculate post-change estimates $\hat{\theta}_t$ may initially seem unreasonable. Still, detection decisions can be delayed by w samples so that data is available for calculating these estimates (provided that w is not excessively large). These estimates can be substituted for θ_1

in (2.3) to obtain an adaptive form of CUSUM where the post-change distribution is estimated.

Such estimates are also independent of the change statistic S_t. In comparison to GLR, adaptive CUSUM leads to a more computationally efficient method for detecting change points when the post-change distribution is unknown. CUSUM has been extensively studied to develop tools that characterize the detection average run length (ARL), the average time till false detection under the pre-change distribution, and the expected detection delay (EDD), which is the expected time till true detection under the post-change distribution. Adaptive CUSUM can utilize the similar technique to characterize the ARL and EDD performance.

2.4.2 Proposed procedure

As discussed in Section Subsection 2.3.1, the log-likelihood ratio test is asymmetric for changes between two distributions having different variances. This makes it difficult to select a single threshold for adaptive CUSUM to detect multiple changes.

To address this problem, we introduce a symmetric version of adaptive CUSUM called DAS-CUSUM. To begin, we recall that the Kullback-Leibler (KL) Divergence between the distribution f_{θ_0} and f_{θ_1} is given by:

$$D_{\mathrm{KL}}(\theta_0, \theta_1) = \int f_{\theta_0}(x) \log \frac{f_{\theta_0}(x)}{f_{\theta_1}(x)} dx.$$

The DAS-CUSUM-based change detection statistic is defined as:

$$S_t = (S_{t-1})^+ + s_t,$$

with the incremental update statistic is given by:

$$s_t = \log \frac{f_{\hat{\theta}_t}(x_i)}{f_{\theta_0}(x_i)} + D_{\mathrm{KL}}(\theta_0, \hat{\theta}_t) - v, \tag{2.6}$$

where $v > 0$ is a constant in the drift term.

Comparing to the incremental update for CUSUM, which only contains the log-likelihood ratio, the DAS-CUSUM update statistic has two additional terms, which can be seen in (2.6). The first of these terms is a KL divergence, which makes the incremental statistic almost symmetric under the post-change distribution.

When w is sufficiently large, $\hat{\theta}_t \approx \theta_1$, and thus

$$\mathbb{E}_{\theta_1}[s_t] \approx D_{\mathrm{KL}}(\theta_1, \theta_0) + D_{\mathrm{KL}}(\theta_0, \theta_1) - v. \tag{2.7}$$

The second of these additional terms, v, is a drift term that makes the expectation of the incremental statistic negative under the pre-change distribution. This allows our proposed statistic to match the property of CUSUM, which requires that the increment term should be negative under the pre-change distribution to avoid false alarms.

We want to point out that by a symmetrical change statistic, we mean that the change statistic value when moving from θ_0 to θ_1 is similar to the change statistic value when moving from θ_1 to θ_0. Though this symmetrical statistic assumes the sequence switches between two distributions, the change statistic values, as compared to CUSUM or GLR, are relatively similar when a sequence switches between multiple distributions having different variances. This makes it easier for our proposed method to use a single threshold to detect multiple distributional changes in a sequence.

2.4.3 Practical implementation

1 shows how to implement DAS-CUSUM for detecting multiple change-points. This algorithm uses values for the window size w^* and drift term v^*, based on theoretical results presented in Section 2.5. However, these results require complete knowledge of the post-change distribution θ_1 to compute the KL divergences, necessary to compute the desiblack values for w^* and v^*. Since this post-change distribution is unknown, we can set a minimum symmetric KL divergence, which

corresponds to the minimum change in distribution that is to be detected in a streaming data setting. This minimum symmetric KL divergence can be used to set window size values w^* and drift term v^*. The optimal window size w^* can be found by minimizing an expression. This expression is discussed in more detail in Remark 2. Despite this expression being convex with respect to w, a closed-form expression of w^* is difficult to obtain. This optimal window size w^* can be solved numerically. When a change-point is detected, the previous post-change estimate $\hat{\theta}_t$ is used as the pre-change distribution θ_0 for detecting the subsequent change-point. The initial pre-change distribution θ_0 used at the start of the change detection procedure can either be determined through prior knowledge or estimated through samples at the beginning of sequence.

Algorithm 1 DAS-CUSUM for multiple change-point detection

Inputs: Sequence: X, Threshold b, Target ARL: γ, Min sym div: s',

Pre-change distribution: $\theta_0 := (\mu_0, \sigma_0^2)$

Output CpList : List containing change-points

Choose window size

$$w^* = \arg\min_w \frac{\log \gamma}{-1 + (1 + ws'^2)^{\frac{1}{2}} + \log\left(1 - \frac{\left(-1+(1+ws'^2)^{\frac{1}{2}}\right)^2}{ws'^2}\right)} + w$$

$$\delta_0^* = -\frac{1}{s'} + \left(\frac{1}{s'^2} + w^*\right)^{1/2}, \quad v^* = \frac{-\log(1 - \frac{\delta_0^{*2}}{w})}{\delta_0^*}$$

for $t = 1$ to length$(X)]$ **do**

$\hat{\mu}_t = \sum_{i=t+1}^{t+w} x_i, \quad \hat{\sigma}_t^2 = \sum_{i=t+1}^{t+w} \frac{1}{w}(x_i - \hat{\mu}_t)^2, \quad \hat{\theta}_t = (\hat{\mu}_t, \hat{\sigma}_t^2)$

Compute CUSUM recursion

$$S_t = (S_{t-1})^+ + \log \frac{f_{\hat{\theta}_t}(x_i)}{f_{\theta_0}(x_i)} + D_{\mathrm{KL}}(\theta_0, \theta_1) - v^*$$

if $S_t > b$ **then**

Add t to CpList

$\mu_0 := \hat{\mu}_t, \quad \sigma_0^2 := \hat{\sigma}_t^2$

end if

end for

2.5 Theoretical results: EDD versus ARL

With the definition of the stopping time T, the detection procedure takes τ samples to detect a change. The average run length (ARL) is the expected value of τ under the pre-change distribution θ_0 such that a false change is detected: $\mathbb{E}_\infty[T]$, where $\mathbb{E}_\infty$ is the expectation under the probability measure on observations without a change. A commonly considered metric is the worst expected detection delay (EDD) [19] conditioned on the worst possible realizations:

$$\bar{\mathbb{E}}_1[T] = \sup_{k \geqslant 1} \operatorname{ess\,sup} \mathbb{E}_k([T - k + 1]^+ | X_1, ..., X_{k-1}), \tag{2.8}$$

where k denotes the change-point location and $\mathbb{E}_k$ is the expectation under the probability measure of observations when the change occurs at k.

Using a similar argument as showing Lemma 4 in [11], we can show that $\mathbb{E}_1[T]$ (when the change happens at the first time instance) provides an upper bound to the worst case expected detection delay (2.8). Thus, in our analysis, we focus on $\mathbb{E}_1[T]$ which we call the EDD. Our first result relates DAS-CUSUM's ARL with its EDD, using similar techniques as those in [11, 36]

Theorem 1. *Let $f_{\theta_0}(x)$ and $f_{\theta_1}(x)$ be the normal density functions of x under the pre-change distribution θ_0 and post-change distribution θ_1, which is unknown and estimated using a window of size w. Assume the ARL $\geqslant \gamma$. When $\gamma \to \infty$ and for large window size w, the EDD of DAS-CUSUM is given by:*

$$\mathbb{E}_1[T] = \frac{\log \gamma + o(1)}{\delta_0 \left(D_{\mathrm{KL}}(\theta_1, \theta_0) + D_{\mathrm{KL}}(\theta_0, \theta_1)\right) + \log(1 - \frac{{\delta_0}^2}{w})} + w. \tag{2.9}$$

where $\delta_0 > 0$.

Corollary 1. *The value of δ_0 that minimizes the EDD for a given ARL in* (2.9) *is given by:*

$$\left(\frac{1}{\left(D_{\mathrm{KL}}(\theta_0,\theta_1)+D_{\mathrm{KL}}(\theta_1,\theta_0)\right)^2}+w\right)^{1/2}-\frac{1}{\left(D_{KL}(\theta_0,\theta_1)+D_{\mathrm{KL}}(\theta_1,\theta_0)\right)}. \tag{2.10}$$

Remark 1. The value of δ_0^* from Corollary 1 can be used in the result of Theorem Theorem 1 to obtain the minimum EDD for a given ARL.

Corollary 2. *The optimal drift term v^* which minimizes the EDD in* (2.9) *for a given ARL is given by*

$$-\log(1-\delta_0^{*2}/w)/\delta_0^*. \tag{2.11}$$

Remark 2. The expression in Theorem Theorem 1 can be minimized with respect to w (at a provided value of average run length and symmetric KL divergence) to find the optimal window size w^*. A closed-form expression for w^* cannot be obtained, but w^* can be solved numerically. Figure 2.4 shows how EDD relates to window size w. The curve has a minimum point corresponding to a window size of $w = 11$. When this solution is too small, the results in Theorem Theorem 1 do not hold, which assume w to be large (so that post-change estimates converge to true post-change distribution). More details can be found in Subsection 2.6.2. Additionally, the window size should be large enough for $\delta_0^* < w$ for the logarithmic term in Theorem Theorem 1 to be real.

2.5.1 Comparison to CUSUM results

[19] provides the asymptotic lower bound for EDD for CUSUM given $\mathbb{E}_\infty \geqslant \gamma$ and $\gamma \to \infty$,

$$\mathbb{E}_1[T] \geqslant \frac{\log\gamma(1+o(1))}{D_{\mathrm{KL}}(\theta_1,\theta_0)}. \tag{2.12}$$

For the proposed detection procedure, it can be seen in Theorem Theorem 1 that the EDD at a set ARL value would be similar for a change from θ_0 to θ_1 and a change from θ_1 to θ_0. This is

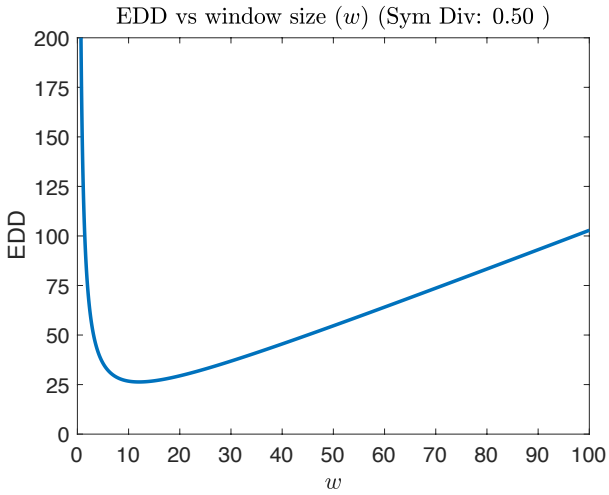


Figure 2.4: Expected detection delay (EDD) versus window size for a change with symmetric KL divergence of 0.5 at an ARL value of 5,000. This shows that the EDD is a convex function of w which can be minimized to obtain the optimal window w^*.

not true for CUSUM, where the detection delay for a change from θ_0 to θ_1 will not be equal to a change from θ_1 to θ_0.

The expression in Theorem 1 also has an additional w term, which takes into account the time delay for obtaining the window to estimate post-change parameters, but this is a consequence of the post-change distribution being unknown.

2.5.2 Sketch of the proof

The increment of the CUSUM statistic in (2.3) consists of a log-likelihood ratio which has a negative expectation under the pre-change distribution θ_0. The proposed increment statistic for DAS-CUSUM in (2.6) has a negative drift under the post-change distribution but is not a log-likelihood ratio. One way to find the optimal value v in our proposed update statistic is to convert it into a valid log-likelihood ratio. Once this is done, ARL and EDD results from CUSUM can be used for our proposed statistic. This expression would consist of the negative drift term v, which could be minimized to find the optimal value for v. It can been seen in [19] that for a detection

threshold b , the CUSUM procedure has the following asymptotic average run length:

$$\mathbb{E}_{\infty}[T] = \frac{e^{b}(1 + o(1))}{K}, \tag{2.13}$$

where K is a constant.

For CUSUM, the EDD matches the lower bound as shown in (2.12).

Using the techniques developed in proposed in [11, 35], an equivalence term δ_0 can be introduced to our incremental statistic, which satisfies the equation

$$\mathbb{E}_{\theta_0}[\exp(\delta_0 s_t)] = 1. \tag{2.14}$$

When (2.14) is satisfied, $\exp(\delta_0 s_t)$ can be consideblack to be the likelihood ratio between distributions $\tilde{f}_{\theta_1} = \exp[\delta_0 s_t] f_{\theta_0}$ and f_{θ_0} which then allows us to use (2.13) to obtain the ARL performance for DAS-CUSUM. The threshold b can be expressed in terms of the average run length (γ)

$$b = \frac{\log \gamma(1 + o(1))}{\delta_0}. \tag{2.15}$$

This expression is obtained through (2.13) where the constant K is absorbed within $o(1)$ and the introduced scaling factor δ_0 for the incremental statistic is appropriately scaled. Similarly, δ_1 can be introduced such that $\delta_1 s_t$ is the log-likelihood ratio between f_{θ_1} and $\tilde{f}_{\theta_0} = \exp[-\delta_1 s_t] f_{\theta_1}$. Thus, we can relate change between f_{θ_1}, where the δ_1 term is observed in $o(1)$ as shown below:

$$\mathbb{E}_1[T] = \frac{b(1 + o(1))}{\mathbb{E}_{\theta_1}[s_t]}. \tag{2.16}$$

Substituting (2.15) in the above equation, we obtain

$$\mathbb{E}_1[T] = \frac{\log \gamma(1 + o(1))}{\delta_0 \mathbb{E}_{\theta_1}[s_t]}. \tag{2.17}$$

Substituting (2.7) yields

$$\mathbb{E}_1[T] = \frac{\log \gamma(1 + o(1))}{\delta_0 \left(D_{\mathrm{KL}}(\theta_0, \theta_1) + D_{\mathrm{KL}}(\theta_1, \theta_0) - v\right)}. \tag{2.18}$$

Our expression above assumes that our statistic is converted to a log-likelihood ratio by satisfying the martingale property in (2.14). Lemma 1 satisfies this requirement by finding an expression that relates the drift value v with the equivalence factor δ_0

Lemma 1. *As* $w \to \infty, \mathbb{E}_{\theta_0}[\exp(\delta_0 s_t)] = 1$ *asymptotically when* v *takes the value in* (2.11).

Then the value for v, for which (2.14) is satisfied, can be substituted. As w samples are needed to estimate the post-change distribution $\hat{\theta}_t$, when $w \to \infty$, the EDD approaches to

$$\frac{\log \gamma + o(1)}{\delta_0 \left(D_{KL}(\theta_0, \theta_1) + D_{KL}(\theta_1, \theta_0)\right) + \log(1 - \frac{{\delta_0}^2}{w})} + w. \tag{2.19}$$

This EDD expression can be minimized with respect to δ_0 by equating the derivative to 0. The optimal value of δ_0^* that minimizes the expression is given by (2.10).

Sketch of Proof for Lemma 1

The left side of (2.14) can be written as shown below by substituting the proposed update statistic from (2.6):

$$\mathbb{E}_{\theta_0}\left[\exp(\delta_0 \tilde{s}_t)\right] = \mathbb{E}_{\theta_0}\left[\exp\left(\delta_0\left(-\frac{(x_t - \hat{\mu}_t)^2}{2\hat{\sigma}_t^2} + \frac{(x_t - \mu_0)^2}{2\sigma_0^2} + \frac{\sigma_0^2 + (\mu_0 - \hat{\mu}_t)^2}{2\hat{\sigma}_t^2} - \frac{1}{2} - v\right)\right)\right].$$

Since a future window $(x_{t+1}, \ldots, x_{t+w})$ is used to estimate $\hat{\mu}_t$ and $\hat{\sigma}_t$, these estimates are independent from x_t. These estimates can be treated as constants while introducing a conditional expectation through the tower rule. The equation above can be written as

$$\mathbb{E}_{x_{t+1},\ldots,x_{t+w}\sim f_{\theta_0}}\left[\exp\left(\delta_0\left(\frac{\sigma_0^2+(\mu_0-\hat{\mu}_t)^2}{2\hat{\sigma}_t^2}-\frac{1}{2}-v\right)\right)\mathbb{E}_{x_{t+1},\ldots,x_{t+w}\sim f_{\theta_0}}\left[r(x_t)\middle|\hat{\mu}_t,\hat{\sigma}_t\right]\right]$$
$$=\exp(\delta_0(-\frac{1}{2}-v))\mathbb{E}_{x_{t+1},..,t+w\sim\theta_0}\left[\exp\left(\delta_0\left(\frac{\sigma_0^2+(\mu_0-\hat{\mu})^2}{2\hat{\sigma}_t^2}\right)\right)\mathbb{E}_{x_t\sim\theta_0}\left[r\left(x_t\right)\middle|\hat{\mu}_t,\hat{\sigma}_t\right]\right], \tag{2.20}$$

where

$$r\left(x_t\right)=\exp\left(\delta_0\left(-\frac{(x_t-\hat{\mu}_t)^2}{2\hat{\sigma}_t^2}+\frac{(x_t-\mu_0)^2}{2\sigma_0^2}\right)\right).$$

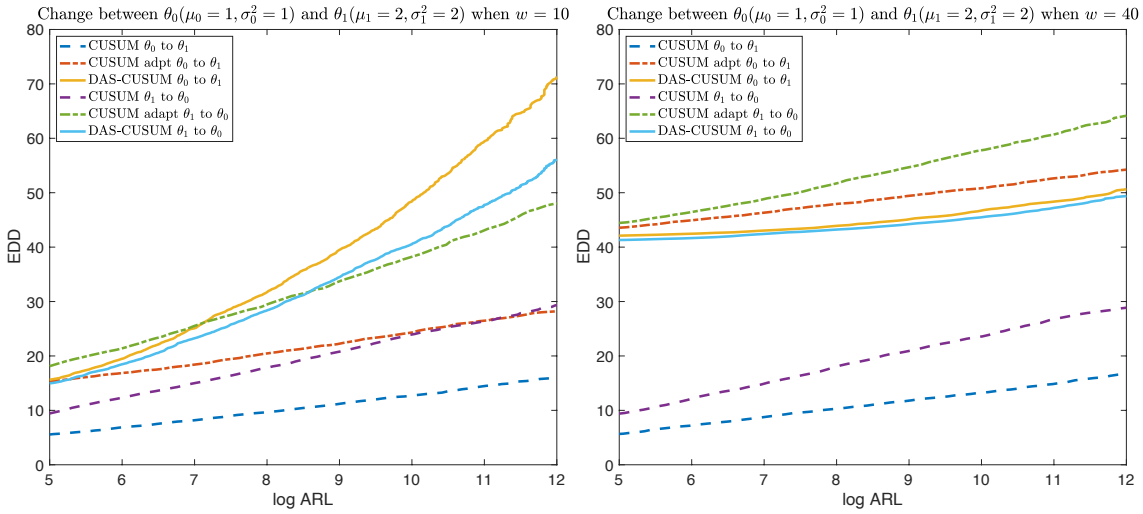
Further details for these calculations can be found in the Appendix.

2.6 Simulations

2.6.1 ARL and EDD

As discussed in Section 2.4, the DAS-CUSUM change-point detection procedure is designed to have a symmetric change statistic. Due to this symmetric property, DAS-CUSUM should have similar ARL versus EDD performance for changes from the distribution θ_0 to θ_1 and from θ_1 to θ_0. This symmetry is studied in ARL versus EDD plots in Figure 2.5. This figure also contains plots for CUSUM and an adaptive version of CUSUM where a future window of size w is used to estimate the post-change parameters. CUSUM curves for changes from $\theta_0(\mu_0=1,\sigma_0^2=1)$ to $\theta_1(\mu_1=2,\sigma_1^2=2)$ and θ_1 to θ_0 are far away from one another, while DAS-CUSUM curves are closer to each other. These DAS-CUSUM curves become closer when the post-change estimates become more accurate with increasing window size, as shown in Figure 2.5b. These results align with Subsection 2.5.1, which compares the results of DAS-CUSUM in Theorem Theorem 1 with corresponding results for CUSUM. Specifically, EDD at a given ARL is the same for a change from θ_0 to θ_1 and vice versa when the window length w becomes asymptotically large.

Now we validate the accuracy of theoretical approximation by comparing it against simulation



(a) (b)

Figure 2.5: EDD versus ARL performance comparison for DAS-CUSUM and CUSUM for changes between $\theta_0(\mu_0 = 1, \sigma_0^2 = 1)$ and $\theta_1(\mu_1 = 2, \sigma_1^2 = 2)$ which corresponds to a symmetric KL divergence of 1. Figure 2.5a shows the relationship when a window size of 10 is used for the post-change estimate. while Figure 2.5b shows the case when the window size is 40. Notice the similar performance for DAS-CUSUM for changes from θ_0 to θ_1 and θ_1 to θ_0. This similarity increases with window size w.

results. Figure 2.6 shows DAS-CUSUM plots for EDD versus ARL at different window lengths (w to estimate post-change distribution). For each window length, plots for the theoretical relationship (from Theorem Theorem 1) are compablack to simulated plots. For a small window size ($w = 10$), the theoretical and simulated results grow apart as ARL increases. The difference between the theoretical and simulated plots decreases as the window size increases. This is expected as the results in Theorem Theorem 1 hold when w grows asymptotically. When $w = 120$, the difference between theoretical and simulated EDD is approximately 1 sample for the shown ARL range.

2.6.2 Optimal window length

DAS-CUSUM results that relate EDD with ARL in Theorem Theorem 1 depend on the estimation window size w (at provided values of ARL and symmetric KL divergence). This equation can be minimized for w to find the optimal window length (w^*). Unfortunately, there is no closed-form

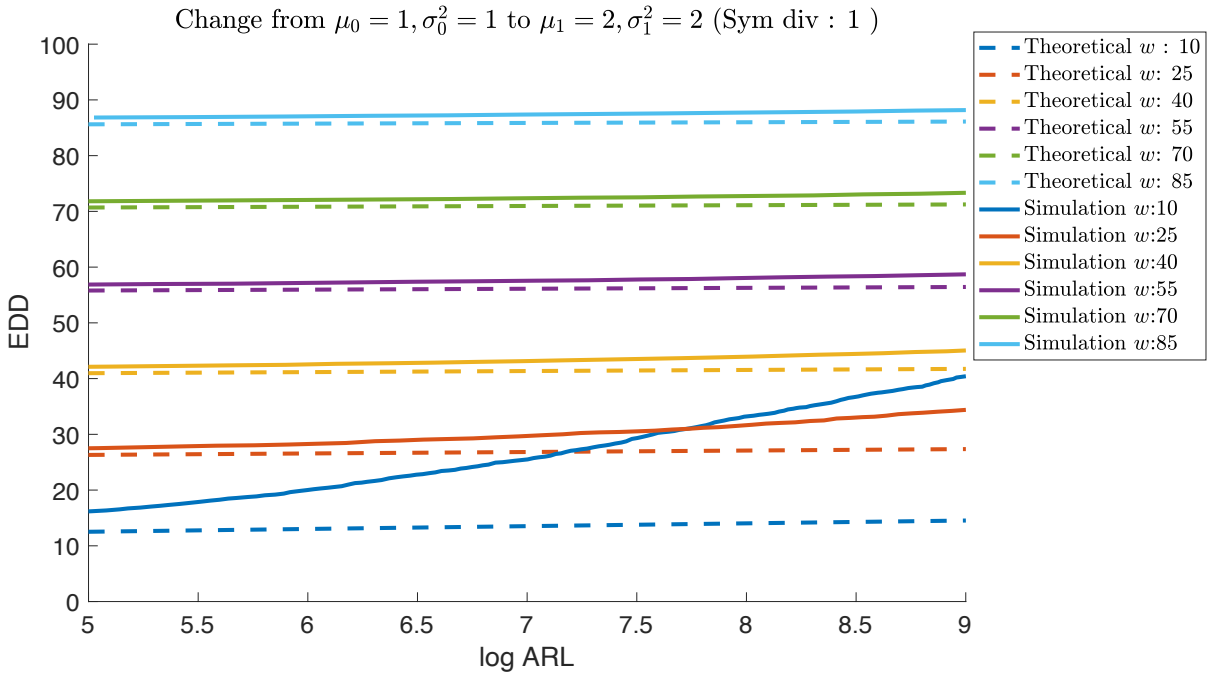
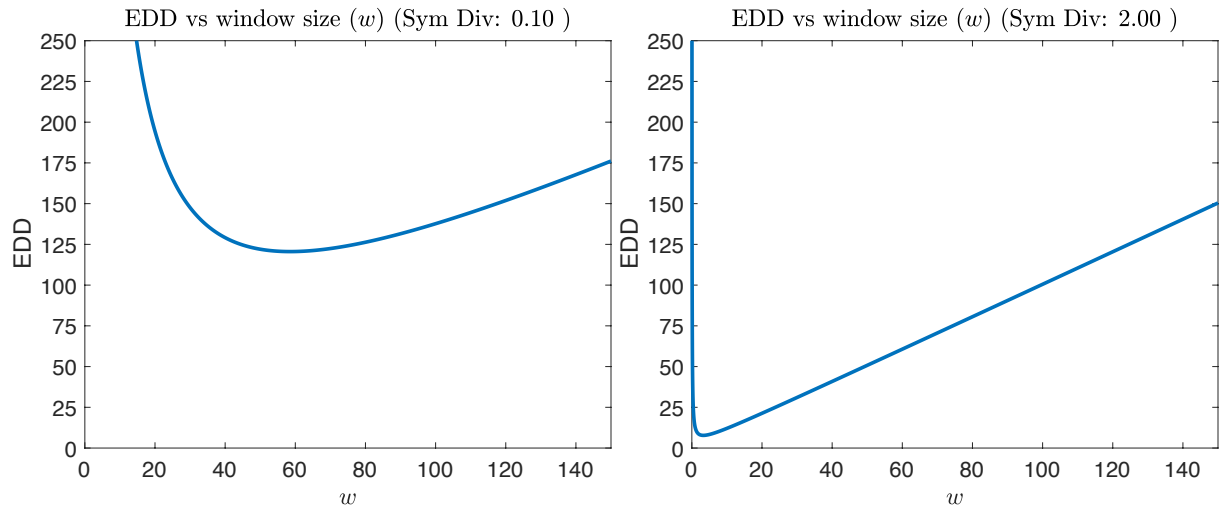


Figure 2.6: Comparison between theoretical and simulated DAS-CUSUM results for different post-change estimation window sizes w. The change in this example has a symmetric KL divergence of 1.

expression for this optimal value. Nevertheless, this equation can be minimized numerically to obtain w^*. Figure 2.7 shows this relationship at an ARL of 5,000 for changes with two different symmetric KL divergence values.

Figure 2.7 shows this relationship for a smaller change in distribution (a symmetric diverge of 0.11), while Figure 2.7b shows this relationship for a larger change (a symmetric KL divergence of 2). Intuitively, a larger change (with a larger symmetric KL divergence) would be easier to detect, requiring a shorter window length as compared to a smaller change (with a smaller symmetric KL divergence). However, for larger changes, the window size corresponding to the minimum EDD value could be too small, as seen in Figure 2.7b where this window is of size 4. The theoretical results start to match simulated results at a window size of about 30 while results at a window size of 10 divergences. For this reason, when the optimal window size (w^*) is below 20, a rule should be in place for a minimum window size.

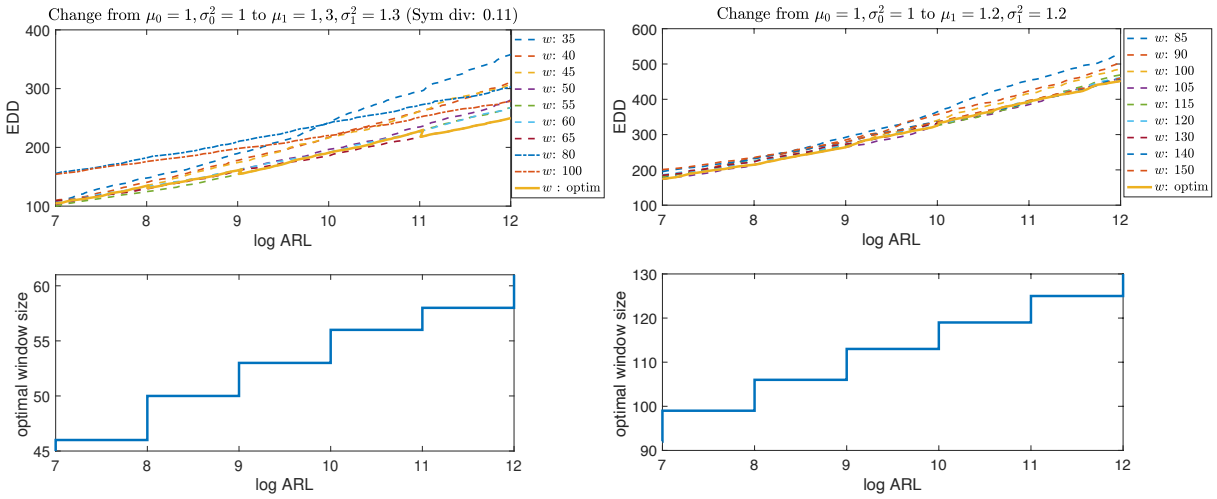


(a) (b)

Figure 2.7: Relation between EDD and window size for changes with different symmetric KL divergence. The ARL has been set to 5,000 in both figures. The optimal window size corresponds to the minimum EDD values. Figure 2.7a shows the relationship for a change with symmetric KL divergence of 1 while Figure 2.7b shows the relationship for a symmetric KL divergence of 2

.

Results that relate the optimal window length for different ARL values can be seen in Figure 2.7. The changes in this figure have small divergence values, which lead to w^* that is greater than a size of 20. The curves for w^* are in yellow and seem to provide better EDD versus ARL performance than most other window sizes. As the optimal window size, w^* increases in Figure 2.8b, the corresponding ARL versus EDD curve often performs best (or close to best) when compared with other window sizes.



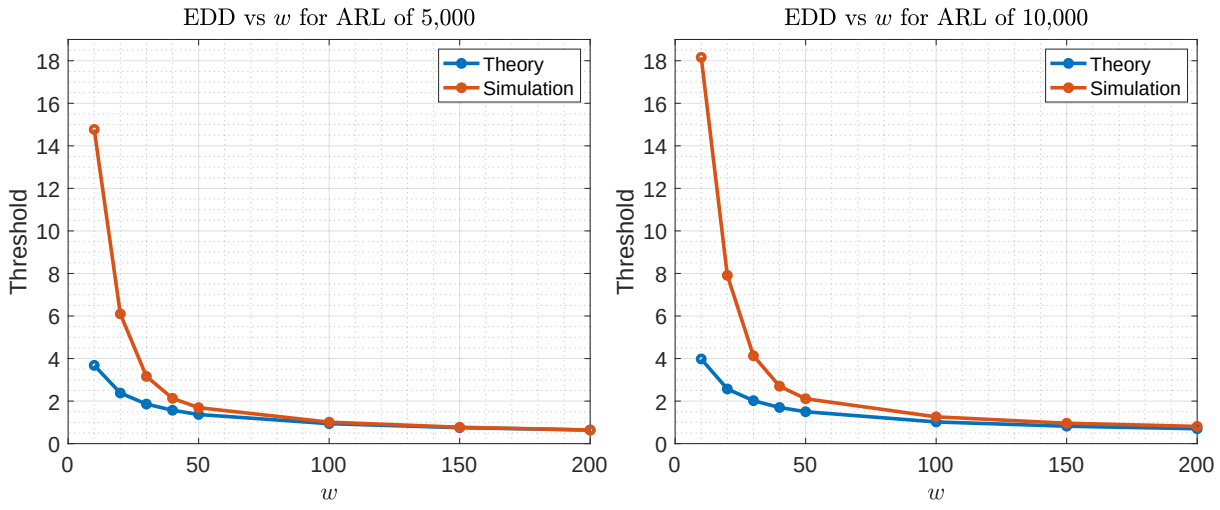
(a) (b)

1

Figure 2.8: ARL versus EDD performance for different window length (w) sizes. Figure 2.8a shows plots for a change from $\theta_0(\mu_0 = 1, \sigma_0^2 = 1)$ to $\theta_1(\mu_1 = 1.3, \sigma_1^2 = 1.3)$ and Figure 2.8b shows plots for changes from $\theta_0(\mu_0 = 1, \sigma_0^2 = 1)$ to $\theta_1(\mu_1 = 1.2, \sigma_1^2 = 1.2)$. Optimal window size ($w$*) provides optimal performance as w* increases.

2.6.3 Setting the detection threshold

The table below compares the simulated and theoretical detection threshold (b) to achieve different ARL values. The theoretical relationship between ARL and the detection threshold is provided in (2.15). These experiments were done on a distribution change from $\theta_0(\mu_0 = 1, \sigma_0^2 = 1)$ to $\theta_1(\mu_1 = 2, \sigma_1^2 = 2)$ which corresponds to a symmetric KL divergence of 1. For ARL, false alarms occur when data points generated from pre-change distribution (θ_0) are falsely detected as change-points. Intuitively ARL values should depend only on the pre-change θ_0 but the post-change distribution (θ_1) is used to set the δ_0^* value, which is used within the theoretical (2.15) as well as for setting the drift term v for the simulations. The difference between theoretical and simulated results is large for small values of post-change estimate window w, but these results become closer as this window size increases. This is expected as the relationship between the detection threshold, and average run length is obtained using (2.14), which is satisfied asymptotically.



(a) (b)

Figure 2.9: Plots for results in table Table 2.1. Figure 2.9a shows the relationship for an ARL value of 5,000 while Figure 2.9b shows the relationship for an ARL of 10,000. The gap between simulation and theoretical results gets small at a window value of about 30.

Table 2.1: Comparison between theoretical and simulated detection thresholds for a change with symmetric KL -divergence of 1

		$w = 10$	$w = 20$	$w = 30$	$w = 40$	$w = 50$	$w = 100$	$w = 150$
ARL = 5,000	Thr.	3.68	2.38	1.86	1.57	1.37	0.94	0.75
	Sim.	14.77	6.10	3.16	2.13	1.69	1.01	0.77
ARL = 10,000	Thr.	3.98	2.57	2.02	1.70	1.50	1.02	0.82
	Sim.	18.16	7.91	4.13	2.70	2.11	1.26	0.96

2.6.4 Moving between multiple distributions

In our previous section, as demonstrated in (2.6), we showed that the DAS-CUSUM method exhibits symmetry when transitioning between distributions θ_0 and θ_1. This desirable property allows for the implementation of a single threshold when detecting multiple change-points, even

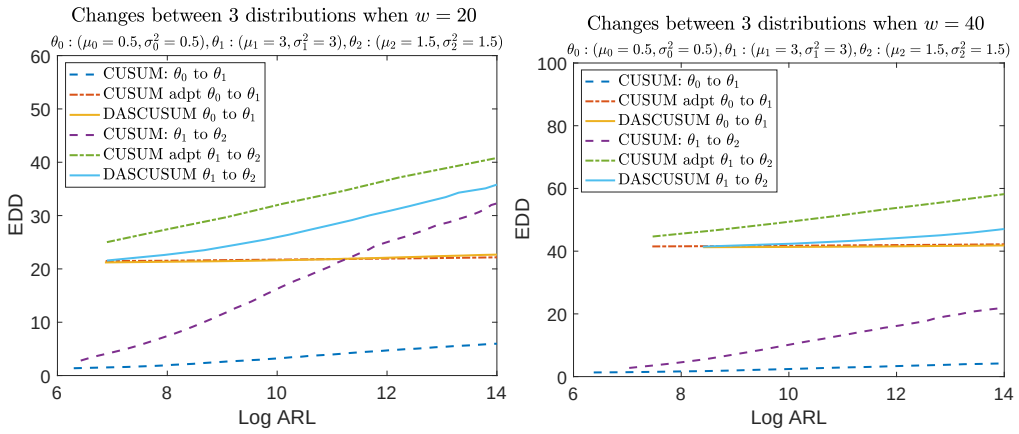


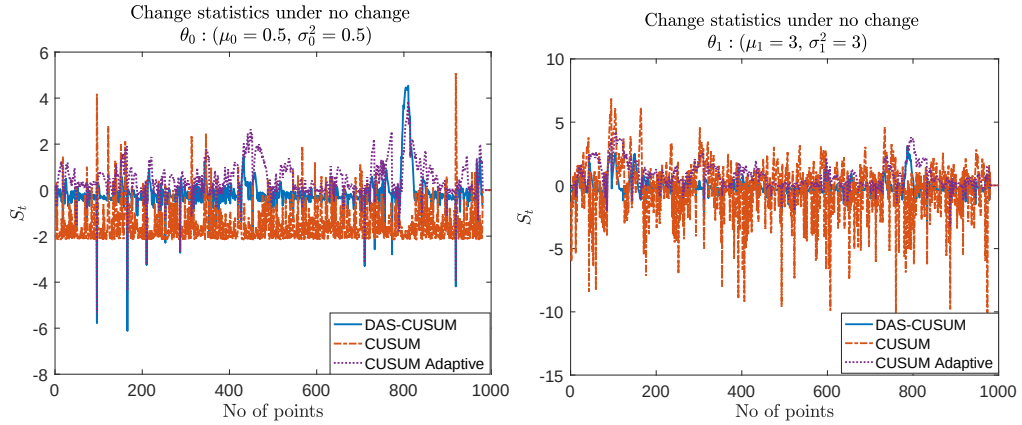
Figure 2.10: ARL vs. EDD relationship when moving between $\theta_0 : (\mu_0 = 0.5, \sigma_0^2 = 2)$ to $\theta_1 : (\mu_1 = 3, \sigma_1^2 = 3)$ and from $\theta_1 : (\mu_1 = 3, \sigma_1^2 = 3)$ to $\theta_2 : (\mu_2 = 1.5, \sigma_2^2 = 1.5)$. For both figures, which show different window sizes used, curves for DAS-CUSUM are closer, indicating it is easier to set a threshold to detect changes from θ_0 to θ_1 and from θ_1 to θ_2 with closer EDD vs. ARL performance.

when the data sequence involves more than two distinct distributions.

Consider a sequence that alternates among three distributions: $\theta_0 = (\mu_0 = 0.5, \sigma_0^2 = 0.5), \theta_1 = (\mu_1 = 3, \sigma_1^2 = 3), \theta_2 = (\mu_2 = 1.5, \sigma_2^2 = 1.5)$. In this case, the change statistics for transitions between θ_0 and θ_1, as well as between θ_1 and θ_2, would differ. However, the DAS-CUSUM method generates more similar change statistics for these varying distributional transitions when compablack to the CUSUM and adaptive CUSUM approaches.

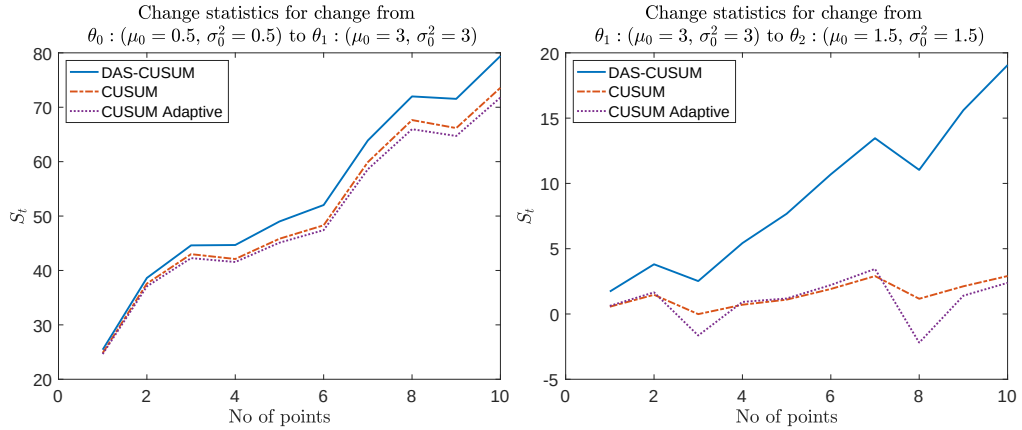
This similarity in change statistics for diverse distributional transitions facilitates the selection of a single threshold capable of detecting true change-points across multiple types of distributional shifts while minimizing the identification of false changes. Figure 2.10 illustrates the EDD versus ARL curves for changes from θ_0 to θ_1 and from θ_1 to θ_2. The proximity of these curves for the DAS-CUSUM method (represented by light blue and yellow) is noticeably greater than that observed for the other methods.

To further illustrate this example, consider the case when this sequence persists in θ_0 for 1000 samples and then switches from θ_0 to θ_1. This sequence then persists in θ_1 for 1000 samples, after which it changes from θ_1 to θ_2. When this sequence changes from θ_0 to θ_1, the change statistics



(a) (b)

Figure 2.11: The change statistic S_t for DAS-CUSUM, CUSUM, and adaptive CUSUM is presented under the assumption of no distributional shifts. The left figure displays a pre-change distribution with $\mu_0 = 0.5$ and $\sigma_0^2 = 0.5$, while the right figure illustrates a pre-change distribution with $\mu_0 = 3$ and $\sigma_0^2 = 3$. A window size of 20 is employed for estimating the post-change distribution in both DAS-CUSUM and adaptive CUSUM methods.



(a) (b)

Figure 2.12: The change statistic S_t for DAS-CUSUM, CUSUM, and adaptive CUSUM is presented under different distributional shifts. The left figure displays the change statistic under the post-change distribution of $\theta_1 = (\mu_1 = 3, \sigma_1^2 = 3)$, transitioning from a pre-change distribution of $\theta_0 = (\mu_0 = 0.5, \sigma_0^2 = 0.5)$. In contrast, the right figure illustrates the change statistic under the post-change distribution of $\theta_2 = (\mu_2 = 1.5, \sigma_2^2 = 1.5)$, originating from a pre-change distribution of $\theta_1 = (\mu_1 = 3, \sigma_1^2 = 3)$. Both DAS-CUSUM and adaptive CUSUM methods employ a window size of 20 for estimating the post-change distribution.

after 10 samples can be seen in Figure 2.12a. Figure 2.11b shows the change statistics when the sequence persists in θ_1 for 1000 samples, while Figure 2.12b shows the change statistics after 10 samples of distribution changing from θ_1 to θ_2. The change statistic for CUSUM and adpative CUSUM under no change in Figure 2.11b goes up to 6.01 and 4.3, respectively. These are smaller than their respective change statistics after 10 samples in Figure 2.12b. This means that CUSUM and adaptive CUSUM can't set a threshold that can correctly detect a change within 10 samples of the sequence switching from θ_1 to θ_2 without detecting a false change when the sequence is in θ_1. This is not a problem for DAS-CUSUM. Note that the change statistics when the sequence persists in θ_0 for 1000 samples in Figure 2.11a is very similar to change statistics when the sequence persists in θ_1. Thus, all CUSUM, adaptive CUSUM, and DAS-CUSUM can correctly detect a change when moving from θ_0 to θ_1 without detecting a false change when this sequence persists in θ_0. However, only DAS-CUSUM, with a threshold value greater than 5 (but lower than 20), can detect changes from θ_0 to θ_1 and θ_1 to θ_2 without detecting any false changes when the sequence persists in θ_0 and θ_1.

2.7 Real data

Real-world sequences often involve signals that switch between multiple distributions. These distributions may also persist for relatively short intervals. DAS-CUSUM's symmetric statistic is more useful for detecting multiple changes as compablack to GLR and adaptive CUSUM. This is favorable for detecting multiple changes in real-world problems, as seen in Figure 2.14 which shows readings from a pressure mat that can be seen in Figure 2.13. The mat is inserted beneath a wheelchair cushion and is used to characterize in-seat movement for wheelchair users. When the wheelchair is occupied, the sensor signal has a high mean and variance, whereas when the chair is unoccupied, the signal has a low mean and variance. Detecting changes in occupancy can be treated as a change detection problem. As discussed previously, the asymmetric log-likelihood ratio makes it difficult for both GLR and adaptive CUSUM to detect these changes.

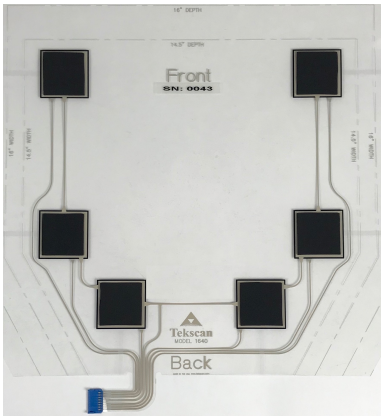
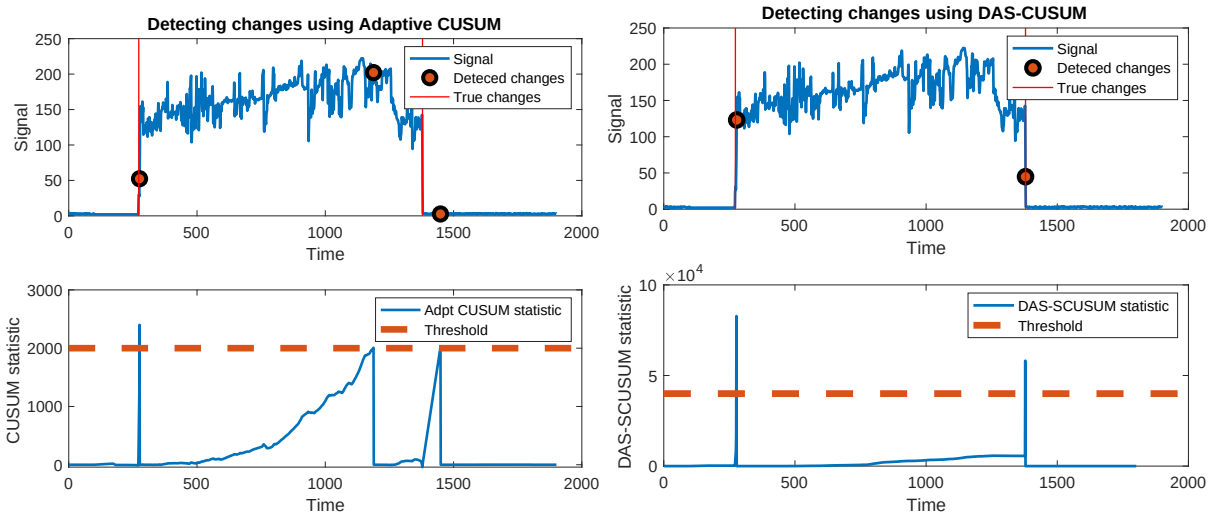


Figure 2.13: Sensor mat used for characterizing in-seat behavior for wheelchair users. Sequential change-point detection can be used to identify changes in wheelchair occupancy.

For both Figure 2.2a and Figure 2.2b, the statistic for getting into the chair (low variance to high variance) is not equal to the statistic for getting out of the chair. For this reason, it is difficult to select a threshold that detects both changes. It can be seen that there is a larger delay in detecting the change while still detecting a false positive change-point. Because of the asymmetric statistics, the change for the first statistic is extremely large as compablack to the second change. To detect both changes, a lower threshold is set, which causes the first change to be detected really quickly (where the signal is in the middle of the transition). This causes incorrect signal estimates to be used as pre-change estimates causing false change-points to be detected. Figure 2.14b shows the performance of DAS-CUSUM on this signal. The symmetric change statistic provides similar power for detecting both changes without detecting any false positive changes. The symmetric statistic makes it easy to select a threshold to detect multiple changes. This is attractive for real-world scenarios where numerous changes need to be detected when the signal changes to unforeseen distributions.

Figure 2.15 provides an extended example of the occupancy problem. The signal sensors develop drift, and the post-change distribution can change to different unknown distributions at different times. This makes it difficult to use 2-sided CUSUM or other variants as the post-change distribution is not known. In such an example, it can be seen that with symmetric statistics, DAS-



(a) (b)

Figure 2.14: Figure 2.14a and Figure 2.14b show how DAS-CUSUM improves over CUSUM for detecting multiple change-points

CUSUM performs much better than GLR and adaptive CUSUM. The in-chair distribution is not static. The mean and the variance of the signal changes within the chair, however, these changes are much smaller than the changes in distribution when there is a change in wheelchair occupancy. Symmetric DAS-CUSUM's change statistic is much larger for these occupancy change events, which makes it easy to detect these events without detecting any false alarms. For all methods, a window size of 300 was to estimate the post-change distribution.

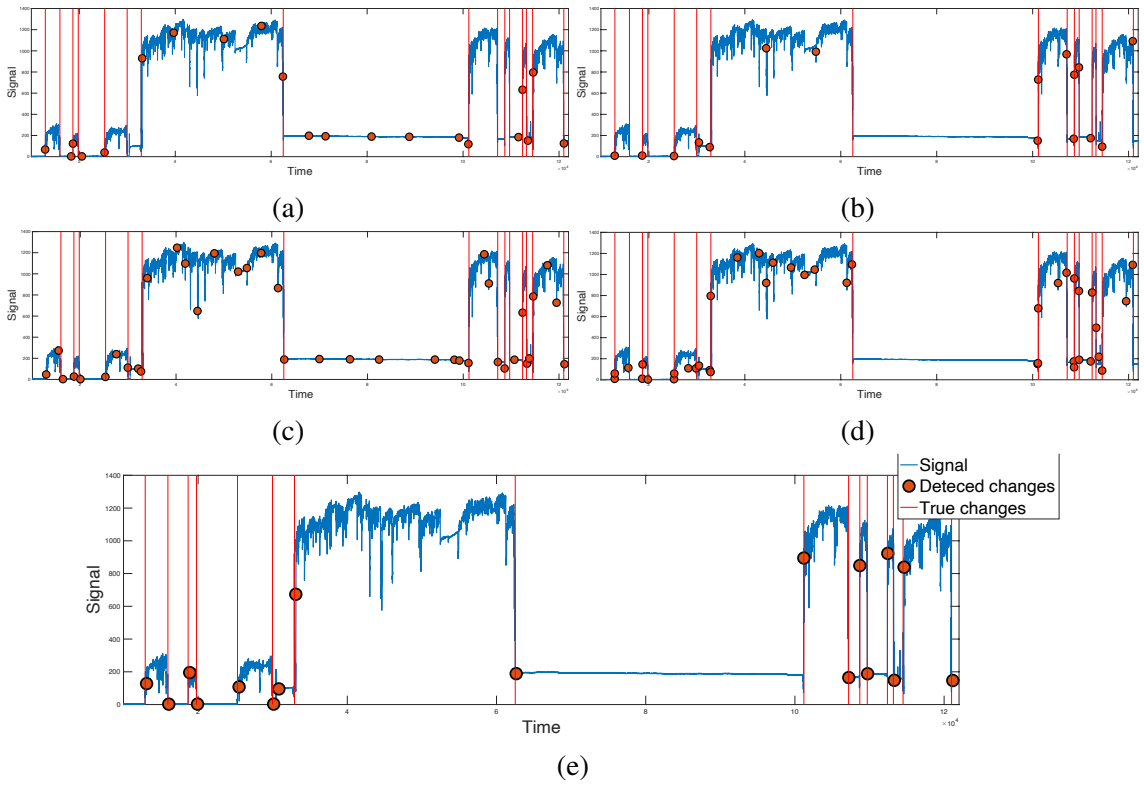


Figure 2.15: Comparison of GLR, Adaptive CUSUM, and DAS-CUSUM for detecting multiple change-points. The asymmetric log-likelihood ratio makes it difficult for CUSUM and GLR to detect all changes correctly without any false alarms. In Figure 2.15a and Figure 2.15b, a large detection threshold to avoid false change-points results in many missed change-points while still detecting a few false change-points. Figure 2.15c and Figure 2.15d show how a lower threshold results in many false change-points. The symmetric DAS-CUSUM is able to correctly detect all true change-points without detecting any false change-points

2.8 Conclusion

In this work, we have presented DAS-CUSUM, which is a symmetric change-point detection procedure. Due to DAS-CUSUM's symmetric incremental statistic, the EDD versus ARL relationship is the same for changes from a distribution θ_0 to θ_1 and from θ_1 to θ_0. This symmetric change statistic is helpful when identifying multiple changes in both the mean and variance of a signal. A single threshold can be easily set to detect multiple change-points. This is extremely helpful for identifying change-points in real-world settings where log-likelihood ratio-based approaches such

as GLR and adaptive CUSUM struggle. We have derived results that characterize DAS-CUSUM's expected detection delay (EDD) and average run length (ARL). Extensive simulations are used to validate these results.

CHAPTER 3

SEMI-SUPERVISED SEQUENCE CLASSIFICATION THROUGH CHANGE-POINT DETECTION

3.1 Introduction

As devices ranging from smart watches to smart toasters are equipped with ever more sensors, machine learning problems involving sequential data are becoming increasingly ubiquitous. Sleep tracking, activity recognition and characterization, and machine health monitoring are just a few applications where machine learning can be applied to sequential data. In recent years, deep networks have been widely used for such tasks as these networks are able automatically learn suitable representations, helping them achieve state-of-the-art performance [37]. However, such methods typically require large, accurately labeled training datasets in order to obtain these results. Unfortunately, especially in the context of sequential data, it is often the case that despite the availability of huge amounts of unlabeled data, labeled data is often scarce and expensive to obtain.

In such settings, *semi-supervised* techniques can provide significant advantages over traditional supervised techniques. Over the past decade, there have been great advances in semi-supervised learning methods. Impressive classification performance – particularly in the fields of computer vision – has been achieved by using large amounts of unlabeled data on top of limited labeled data. However, despite these advances, there has been comparatively much less work on semi-supervised classification of sequential data.

A key intuition that most semi-supervised learning methods share is that the data should (in the right representation) exhibit some kind of clustering, where different classes correspond to different clusters. In the context of sequential data, the equivalent assumption is that data segments

Work in this chapter was published in [3].

within a sequence corresponding to different classes should map to distinct clusters. In the context of sequential data, the challenge is that exploiting this clustering would require the sequence to be appropriately segmented, but segment boundaries are generally unknown *a priori*. If the start/end points of each segment were actually known, it would be much easier to apply traditional semi-supervised learning methods.

In this chapter, we show that standard (unsupervised) *change-point detection* algorithms provide a natural and useful approach to segmenting an unlabeled sequence so that it can be more easily exploited in a semi-supervised context. Specifically, change-point detection algorithms aim to identify instances in a sequence where the data distribution changes (indicating an underlying class change). We show that the resulting change-points can be leveraged to learn improved representations for semi-supervised learning.

We propose a novel framework for semi-supervised sequential classification using change point detection. We first apply unsupervised change-point detection to the unlabeled data. We assume that segments between two change-points belong to the same distribution and should be classified similarly, whereas adjacent segments which are on opposite sides of a change-point belong to different distributions and should be classified differently. These similar/dissimilar pairs, derived from change-points, can then be combined with similar/dissimilar pairs derived from labeled data. We use these combined similar/dissimilar constraints to train a neural network that preserves similarity/dissimilarity. The learned representation can then be fed into a multilayer feedforward network trained via existing semi-supervised techniques.

We show that this approach leads to improved results compared to sequential auto-encoders in a semi-supervised setting. We show that even if the final classifier is trained using standard supervised techniques that ignore the unlabeled data, the learned representations (which utilize both label and unlabeled data pairs) result in competitive performance, indicating the value of incorporating change-points to learning improved representations. The proposed method method is completely agnostic with respect to the change-point detection procedure to be used – any detection

procedure can be used as long as it does well in detecting changes.

Our main contribution is to show that pairwise information generated via change-points helps neural networks achieve improved classification results in settings with limited labeled data. This, to the best of our knowledge, is the first work to recognize the utility of change-points within the context of semi-supervised sequence classification. The proposed method should not be considered a substitute for existing semi-supervised methods, but should be taken as a complementary procedure that produces representations which are better suited for existing semi-supervised methods.

3.2 Related Work

The fundamental idea of semi-supervised learning is that unlabeled data contains useful information that can be leveraged to more efficiently learn from a small subset of labeled data. For example, in the context of classification, an intuitive justification for why this might be possible might involve an implicit expectation that instances belonging to different classes will map to different clusters. More concretely, most semi-supervised approaches make assumptions on the data such as: that instances corresponding to different classes lie on different submanifolds, that class boundaries are smooth, or that class boundaries pass through regions of low data density [38].

Perhaps the simplest semi-supervised learning method is to use transductive methods to learn a classifier on the unlabeled data and then assign "pseudo labels" to some or all of the unlabeled data, which can be used together with the labeled data to retrain the classifier. Transductive SVMs and graphical label propagation are examples of such methods [39, 40]. See [41] for a survey of such methods. However, such *self-training* semi-supervised methods struggle when the initial model trained from limited labels is poor.

A more common approach to semi-supervised learning is to employ methods that try to learn class boundaries that are smooth or pass through areas of low data density[42]. Entropy regularization can be used to encourage class boundaries to pass through low density regions[43].

Consistency-based methods such as denoising autoencoders, ladder networks [44] and the π method [45] attempt to learn smooth class boundaries by augmenting the data. Specifically, unlabeled instances can be perturbed by adding noise, and while both the original and perturbed instances are unlabeled, we can ask that they both be assigned the same class. This approach is particularly effective in computer vision tasks, where rather than using only noise perturbations, we can exploit class-preserving augmentations such as rotation, mirroring, and other transformations [46]. By enforcing the classifier to produce the same labels for original and transformed images, decision boundaries are encouraged to be smooth, leading to good generalization.

Unfortunately, due to a lack of natural segmentation and the difficulty of defining class-preserving transformations, there has been comparatively little work on semi-supervised classification of sequences. Most prior work (e.g., [44, 47]) use sequential autoencoders (or their variants) as a consistency-based method to learn representations that lead to improved classification performance. Such autoencoders have been exploited successfully in the context of semi-supervised classification for human activity recognition [48]. However, while such consistency-based approaches do encourage smooth class boundaries, they do not necessarily promote the kind of clustering behavior that we need in cases where there are extremely few labels available.

An alternative approach that more explicitly separates different classes involves learning representations that directly incorporate pairwise similarity information about different instances. One example of this approach is *metric learning* – as an early example, [49] showed that improved classification could be achieved by learning a Mahalanobis distance using pairwise constraints based on class membership. The learned metric leads to a representation in which different classes map to different clusters. A similar approach learns a more general non-linear metric to encourage the formation of clusters while adhering to the provided pairwise constraints [50]. Neural networks such as Siamese [51] and Triplet networks also learn representations from available similar/dissimilar pairs. In [52] it was shown that such similar/dissimilar pairs (obtained from labeled data) can be used for clustering data where each cluster belongs to a different class in the dataset.

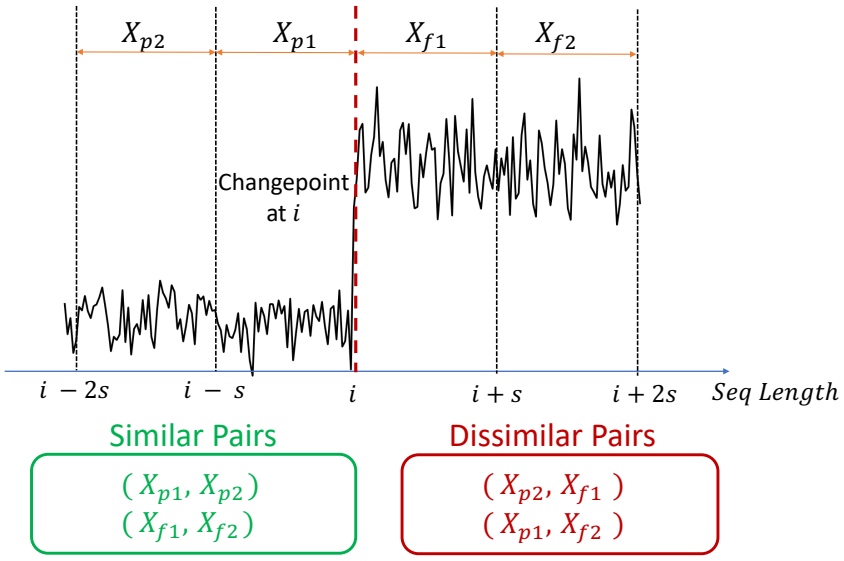


Figure 3.1: Using change-points to generate similar and dissimilar pairs of size s.

Our approach is similar in spirit to that of [52]. While this prior work used pairwise similarity constraints derived from labeled images to learn clustered representations, our goal is to apply this idea in the semi-supervised context. At the core of our approach is the observation that pairwise similarity constraints on sequential data can be derived through unsupervised methods. Specifically, change-point detection can be used to identify points within a sequence corresponding to distribution shifts, which can then be used to obtain pairwise similarity constraints. When the availability of labeled data is limited, this can be a valuable source of additional information.

3.3 Proposed method

3.3.1 Change-point detection

Given a sequence $X : x_1, \ldots, x_N$ of N vectors $x_i \in \mathbb{R}^D$, the first step in our procedure is to detect *all* change-points within X in an unsupervised way. Note that this is a different problem than *quickest* change detection, where only a *single* change-point is to be detected in the fastest possible manner. To detect a change at a point i in the sequence, two consecutive length-w windows (X_p^i

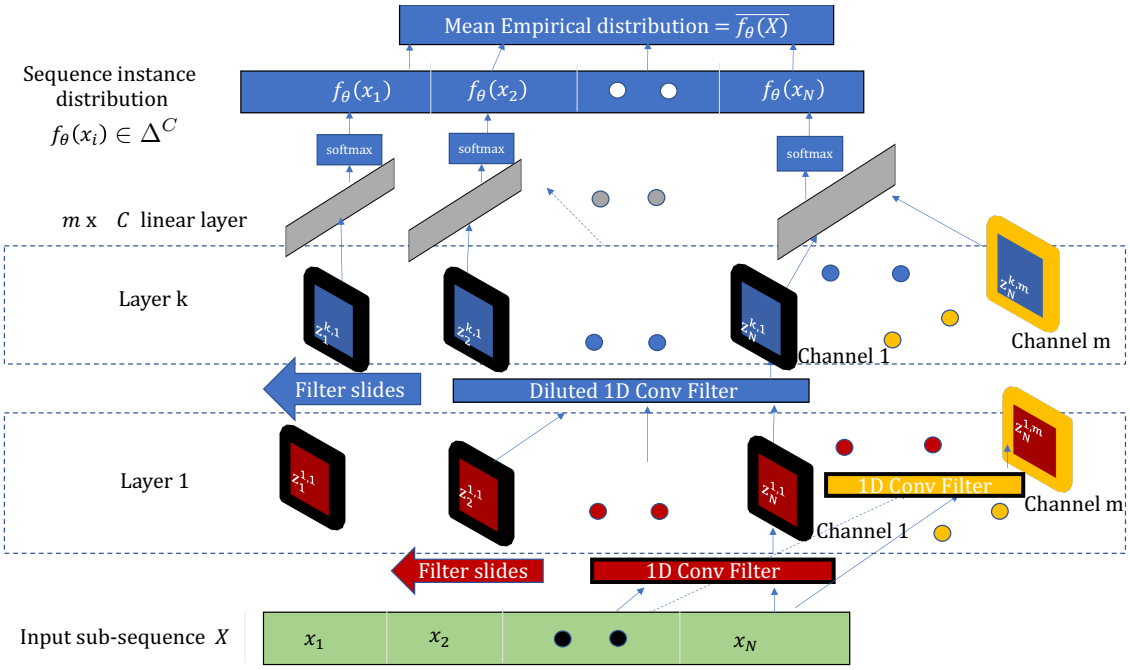


Figure 3.2: Neural network diagram (f_θ) for learning representations.

and X_f^i) are first formed:

$$X_p^i = x_{i-1}, x_{i-2} \ldots x_{i-w} \qquad X_f^i = x_i, x_{i+1} \ldots x_{i+w}.$$

A change statistic, m_i, is then computed via some function that quantifies the difference between the distributions generating X_p^i and X_f^i. If m_i is greater than a specified constant τ, a change-point is detected at the point i.

As one example, many change-point detection procedures assume a parametric form on the distributions generating X_p^i and X_f^i. In this case, the distribution parameters ($\hat{\theta}_p^i$ and $\hat{\theta}_f^i$) can be estimated from X_p^i and X_f^i via, e.g., maximum likelihood estimation. Given these parameter estimates, a symmetrical KL-divergence can be used to quantify the difference between the distributions [12]:

$$m_i = \mathrm{KL}(\hat{\theta}_p^i, \hat{\theta}_f^i) + \mathrm{KL}(\hat{\theta}_f^i, \hat{\theta}_p^i). \tag{3.1}$$

More commonly in practice, the underlying distributions generating the sequence are unknown. In this case, non-parametric techniques can be used to estimate the difference between the distributions of X_p^i and X_f^i. One such approach uses the *maximum mean discrepancy* (MMD) as a change statistic [53]. The MMD has been used to identify change-points in [54] and [30]. The MMD statistic is given below, where $K_{a,-b}^i := k(x_{i+a}, x_{i-b})$ represents a kernel-based measure of the similarity between x_{i+a} and x_{i-b}:

$$\begin{aligned} m_i &= \text{MMD}(X_f^i, X_p^i) \\ &= \frac{1}{\binom{w}{2}} \sum_{\substack{a,b=1 \\ a \neq b}}^{w} (K_{a,b}^i + K_{-a,-b}^i) + \frac{1}{w^2} \sum_{a,b=1}^{w} 2K_{a,-b}^i. \end{aligned}$$

Throughout this chapter, MMD with a radial basis function kernel is used to detect change-points unless otherwise specified. However, we again emphasize that any change-point detection method can be used as long as it performs well in identifying changes points.

The labeled data can be used to set the change-point detection threshold τ and the window size w to balance between false and missed change-points. While we simply fix these parameters in advance using labeled data, these could also be considered as tuning parameters whose values can be set based on performance on a hold-out validation dataset.

3.3.2 Pairwise constraints via change-point detection

Equipped with the detected change-points, similar and dissimilar pairs of sub-sequences can be obtained in an unsupervised manner as shown in Figure 3.1. The idea is to form four consecutive non-overlapping sub-sequences. The first two sub-sequences (X_{p1}, X_{p2}) both occur before the change-point. Since the change-point detection algorithm did not determine that there was a change-point in the combined segment of (X_{p1}, X_{p2}), we assume these two segments are generated by the same distribution and should be classified similarly. Similarly, the last two sub-sequences (X_{f1}, X_{f2}) both occur after the change-point and are also taken as a similar pair. In contrast,

the segments on opposite sides of the change-point have been identified as having different underlying distributions. In order to lead to a balanced distribution of similar/dissimilar pairs, we only use the constraints that (X_{f1}, X_{p2}) and (X_{f2}, X_{p1}) should be classified differently. Each of the subsequences above is chosen to be of a fixed length s (determined by the spacing between change-points).

3.3.3 Clustered representations via pairwise constraints

Using the approach described above, we can obtain similarity constraints from the unlabeled data. We can also obtain such constraints from labeled data via the assumption that sub-sequences corresponding to the same (different) class labels are similar (dissimilar) respectively. We can represent these as a set $\mathcal{P}_S$ consisting of sub-sequence pairs (X_1, X_2) that are similar and a set $\mathcal{P}_D$ of dissimilar pairs. For compactness, we use the notation $P = (X_1, X_2)$ to refer to a sub-sequence pair belonging to $\mathcal{P}_S$ or $\mathcal{P}_D$.

These sub-sequences are then fed into a 1D temporal convolutional neural network [55], as illustrated in Figure 3.2. The neural network consists of 6 convolutional layers (or 3 temporal blocks as defined by [55]) followed by 1 linear layer. We use a RELU activation function after every convolutional layer. We choose this architecture because the dilated filter structure leads to improved performance at classifying time series while being less computationally expensive than recurrent networks such as RNNs and LSTMs, although our framework could also easily accommodate either of these alternate network architectures.

Each instance x_i, in the input sub-sequence X, is passed through the neural network where the final linear layer transforms the output from the last convolutional layer into $\mathbb{R}^C$, where C is the number of classes. A softmax function is then applied to obtain the empirical distribution $f_\theta(x_i)$ for each instance x_i. For a length-N sequence X, we define the mean empirical distribution as:

$$\overline{f_\theta(X)} = \frac{1}{N} \sum_{i=1}^{N} f_\theta(x_i).$$

We then compute the KL divergence between the mean empirical distributions for each sub-sequence within a pair $P = (X_1, X_2)$. Our loss function is constructed applying a hinge loss (with margin parameter ρ) to this KL divergence:

$$h_\theta(P) = \begin{cases} \mathrm{KL}(\overline{f_\theta(X_1)}, \overline{f_\theta(X_2)}) & P \in \mathcal{X}_S, \\ \rho - \mathrm{KL}(\overline{f_\theta(X_1)}, \overline{f_\theta(X_2)}) & P \in \mathcal{X}_D. \end{cases}$$

The network is then trained according to the loss function:

$$\mathcal{L}_{\mathrm{R}}(\theta) = \frac{1}{|\mathcal{P}_L|} \sum_{P \in \mathcal{P}_L} h_\theta(P) + \frac{\lambda_R}{|\mathcal{P}_U|} \sum_{P \in \mathcal{P}_U} h_\theta(P).$$

Here, $\mathcal{P}_L$ and $\mathcal{P}_U$ denote the sets of sub-sequence pairs in $\mathcal{P}_S \cup \mathcal{P}_D$ formed from the labeled and unlabeled data, respectively, and λ_r is a tuning parameter which controls the influence of the unsupervised part of the loss function.

3.3.4 Training a classifier

Once trained, the network f_θ is fixed. The mean empirical distribution for an input sub-sequence X, $\overline{f_\theta(X)}$, can then be used as a representation of X that can serve as input to classifier network f_ψ. We use a 2-layer feedforward neural network followed by a softmax function to obtain a distribution over the different classes. Labeled as well as unlabeled sub-sequences (which correspond to the generated pairs from change-points) are passed through this classification network. Since the learned representations encourage unlabeled data points to cluster around provided labeled data points, known semi-supervised methods can be also used to incorporate unlabeled data while training f_ψ. We use entropy regularization [43] to exploit the unlabeled data by encouraging the classifier boundary to pass through low density regions.

The training data is comprised of two sets: $\mathcal{X}_L$ and $\mathcal{X}_U$. Each element of $\mathcal{X}_L$ consists of a pair (X, Y), where X denotes a sequence $x_1, \ldots, x_N$ of vectors in $\mathbb{R}^D$ and Y denotes a one-hot

encoding of the class label for X (and is hence in $\mathbb{R}^C$ where C is the number of classes). Each element of $\mathcal{X}_U$ consists of a sub-sequence X identified by the change-point detection step (i.e., the individual sub-sequences in the set $\mathcal{P}_U$). The loss function that we use to train f_ψ is given by:

$$\mathcal{L}_{\text{C}}(\psi) = \frac{1}{|\mathcal{X}_L|} \sum_{(X,Y)\in\mathcal{X}_L} \mathcal{L}_{\text{CE}}(X,Y) + \frac{\lambda_C}{|\mathcal{X}_U|} \sum_{X\in\mathcal{X}_U} \mathcal{L}_{\text{NE}}(X).$$

Here, λ_C is a tuning parameter, $\mathcal{L}_{\text{CE}}$ is the cross entropy loss, and $\mathcal{L}_{\text{NE}}$ is the negative entropy loss:

$$\mathcal{L}_{\text{CE}}(X,Y) = -\sum_{c=1}^{C} Y_c \log f_\psi\left(\overline{f_\theta(X)}\right)_c$$
$$\mathcal{L}_{\text{NE}}(X) = -\sum_{c=1}^{C} f_\psi\left(\overline{f_\theta\left(X\right)}\right)_c \log f_\psi\left(\overline{f_\theta(X)}\right)_c.$$

Above, f_ψ represents the output of the feedforward classification network which ends with a softmax distribution over C classes. The input to f_ψ is the mean empirical representations learned by network f_θ for input sequence X. The negative entropy loss encourages the network f_ψ to produce low entropy empirical class distributions for unlabeled data. This encourages unlabeled data to be mapped to a distribution that concentrates on a single class, pushing the classifier boundary to f_ψ towards low-density regions.

A summary of our overall approach to semi-supervised learning via change-point detection is given in Algorithm 2.

3.4 Experiments

3.4.1 Baselines

All of the following baselines use the same representation network f_θ and classification network f_ψ architectures.

Algorithm 2 SSL via change-point detection

Inputs: Unlabeled sequence X, labeled sequences $\{X_l, Y_l\}$, CP detection parameters τ, w,
Output: Trained networks: f_θ, f_ψ
Init: Add similar/dissimilar pairs from $\{X_l\}$ to $\mathcal{P}_S, \mathcal{P}_D$
for $i = 1$ to length(X) **do**
  Form windows: X_p^i, X_f^i
  $m_i = \text{MMD}(X_p^i, X_f^i)$
  if $m_i > \tau$ **then**
    Form two segments before CP: X_{p1}^i, X_{p2}^i
    Form two segments after CP: X_{f1}^i, X_{f2}^i
    Add pairs (X_{p1}^i, X_{p2}^i) and (X_{f1}^i, X_{f2}^i) to $\mathcal{P}_S$
    Add pairs (X_{f1}^i, X_{p2}^i) and (X_{f2}^i, X_{p1}^i) to $\mathcal{P}_D$
  end if
end for
for $j = 1$ to num_epochs **do**
  Train network f_θ by optimizing loss $\mathcal{L}_{\text{R}}$
end for
for $j = 1$ to num_epochs **do**
  Train network f_ψ by optimizing loss $\mathcal{L}_{\text{C}}$
end for

Supervised

In the supervised setting, only the labeled sequence is passed through through both the representation f_θ and classifier networks f_ψ. We train the two networks in an end-to-end manner by minimizing:

$$\mathcal{L}_{\text{S}}(\theta, \psi) = \frac{1}{|\mathcal{X}_L|} \sum_{(X,Y) \in \mathcal{X}_L} \mathcal{L}_{\text{CE}}(X, Y).$$

Denoising autoencoder

A denoising autoencoder [47] or its variants such as the ladder network (where the reconstruction error for intermediate layers is also minimized) [48] are often employed for semi-supervised learning with sequential data. Since it has been previously shown that the performance gap between these approaches is marginal [48] – which we have observed as well – we focus only on the au-

toencoder as a baseline. In this approach, for every $X \in \mathcal{X}_U$, we also consider a perturbed version $\widehat{X}$ produced by adding noise to X. Both X and $\widehat{X}$ are passed through an encoder network f_θ to obtain embeddings which are used by a decoder network f'_θ to reconstruct the unlabeled data. A reconstruction loss of the form $\mathcal{C}(X) = \|X - f'_\theta(f_\theta(\widehat{X}))\|^2$ is incorporated into the loss function to exploit the unlabeled data. The labeled data is first passed through the encoder network f_θ to obtain embeddings, which are then fed into a classifier network f_ψ. We train the two networks in an end-to-end manner by minimizing:

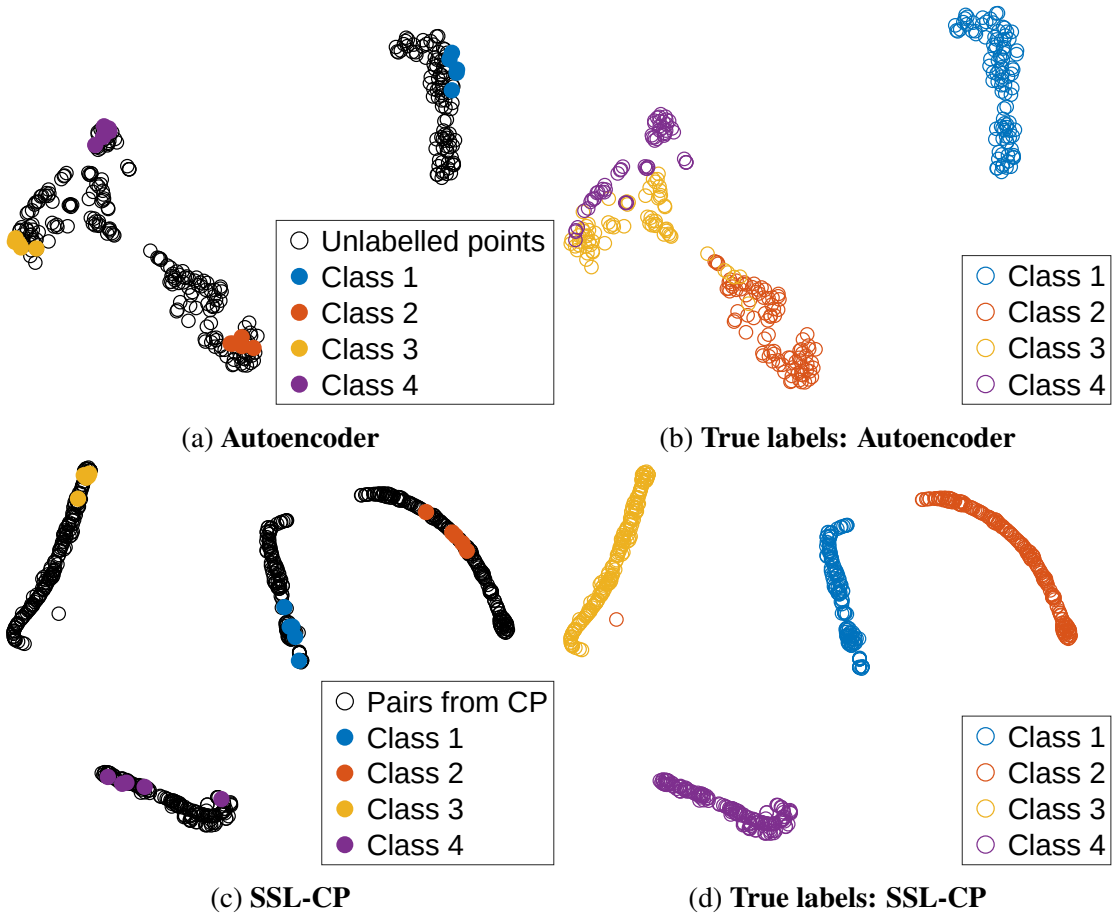
$$\mathcal{L}_{\text{AE}}(\theta, \psi) = \frac{1}{|\mathcal{X}_L|} \sum_{(X,Y) \in \mathcal{X}_L} \mathcal{L}_{\text{CE}}(X, Y) + \frac{\lambda_C}{|\mathcal{X}_U|} \sum_{X \in \mathcal{X}_U} \mathcal{C}(X).$$

Table 3.1: Classifier performance for mean, variance change

Method	10 labels	30 labels
Supervised	0.90 $\pm$0.02	0.98 $\pm$0.01
Autoencoder	0.87 $\pm$0.03	0.99 $\pm$0.01
SSL-CP	0.99 $\pm$ 0.01	0.99 $\pm$ 0.01
SSL-CP (ER)	0.99 $\pm$ 0.01	0.99 $\pm$ 0.01

3.4.2 Synthetic experiments

In all of the results below, we use the mean F1 score (unweighted) as an evaluation metric. In all synthetic simulations, we split the data in a 70/30 ratio where we use the larger split for training and the smaller split as a test dataset. We further split the training data in a ratio of 10/60/30. We use the smallest of these splits to obtain labeled data, the largest as unlabeled data for the semi-supervised setting, and the last split for validation. We use a small sub-sequence (comprising of 20 segments) in the unlabeled split to tune the parameters for change-point detection. In our results, SSL-CP denotes our approach to semi-supervised learning via change-points, but without the inclusion of the negative entropy term in the loss function. SSL-CP (ER) denotes our approach



(a) **Autoencoder** (b) **True labels: Autoencoder**

(c) **SSL-CP** (d) **True labels: SSL-CP**

Figure 3.3: T-SNE visualizations for the representations learned by the representation network (f_θ) on the Mackay-Glass example when 5 labels are provided from each class. Figure 3.3a shows representations learned by an autoencoder using both labeled and unlabeled data. It can be seen in Figure 3.3b that different classes overlap in this representation. Figure 3.3c show the representations learned by SSL-CP, which are clustered and non-overlapping. This leads to improved classification when limited labels are provided. True labels for these representations are shown in Figure 3.3d.

when including this entropy regularization term.

This example consists of data generated by a univariate normal distribution that switches its parameters (μ, σ^2) every 500 samples. We use 1500 such random switches to produce a sequence of data with five classes, correspond to the parameter settings $\{(2, 0.1), (4, 0.1), (4, 0.7), (10, 0.1), (0, 0.1)\}$. We use the symmetrical KL divergence from (3.1) to detect change-points in the unlabeled data. This is a simple change-point detection problem where we detect all change-points correctly. We

use small sub-sequences of length 20 as labeled and unlabeled data. and we show the resulting performance in Table Table 3.1. This is a relatively simple sequence classification problem as it requires merely learning that the mean and variance determine class membership. Both the supervised and autoencoder baselines do reasonably well. However, classes 2 and 3 have the same mean but different variance, and both baselines struggle compared to SSL-CP in separating these classes when only 10 labels are provided.

Mackay-Glass equation

The Mackay-Glass equation [56] is a non linear time delay differential equation defined as

$$\frac{d(x(t))}{d(t)} = -0.1x(t) + \frac{\beta x(t)(t-\tau)}{1 + x(t-\tau)^{10}}.$$

In a manner similar to [57], we generate a sequence by randomly switching between parameters $(\beta, \tau) \in \{(0.2, 8), (0.18, 16), (0.2, 22), (0.22, 30)\}$ every 1400 samples. We define class membership according to the parameter settings of each segment. We generated 2000 such segments and added $\mathcal{N}(0, 0.1)$ noise to the entire sequence. A small sub-sequence is shown in Figure 3.4. We obtained pairs of sequences of size 100 using change points detected on the unlabeled dataset, where almost all true change-points were detected correctly. There were about 4000 such pairs. We obtained 8100 non-overlapping windows of size 100 from the unlabeled-split for use by the autoencoder. Labeled data is also formed using non-overlapping windows of size 100 were used as labels. Table Table C.1 shows results for different numbers of provided labels. We see that SSL-CP approach significantly outperforms the baselines. The representations learned by the autoencoder and SSL-CP are visualized in Figure 3.3, which illustrates that the autoencoder does not perform as well because it fails to learn representations that exhibit sufficient clustering. The influence of varying the number of provided pairs is shown in Table Table 3.3. We note that entropy regularization enhances the performance of SSL-CP when amount of unlabeled data is large.

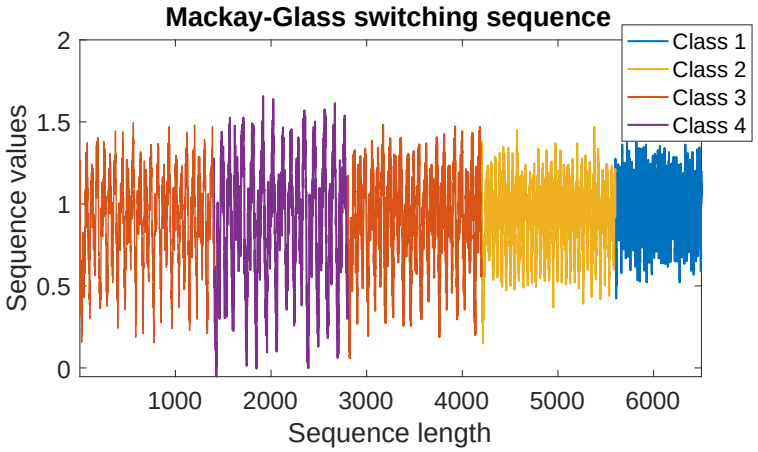


Figure 3.4: Example switching Mackay-Glass sequence.

Table 3.2: Mackay-Glass: Classifier performance for different number of labeled examples

Model	20 labels	30 labels	60 labels
Supervised	0.55 $\pm$0.07	0.86 $\pm$ 0.04	0.95 $\pm$ 0.02
Autoencoder	0.73 $\pm$ 0.04	0.90 $\pm$ 0.02	0.98 $\pm$ 0.01
SSL-CP	0.96 $\pm$0.02	0.98 $\pm$ 0.01	0.99 $\pm$ 0.01
SSL-CP (ER)	**0.99 $\pm$ 0.02**	**0.99 $\pm$ 0.01**	**0.99 $\pm$ 0.01**

3.4.3 Real-world datasets

HCI: Gesture recognition

The HCI gesture recognition dataset consists of a user performing 5 different gestures using the right arm [58]. Data is obtained from 8 IMUs placed on the arm. The gestures recorded included drawing triangle up, circle, infinity, square, and triangle down. We also consider the null case (where the user is not performing an activity) as a class. We use the free-hand subset from this dataset as it presents a relatively challenging classification problem when compared with the more controlled subset. Rather than using consecutive non-overlapping windows (as the resulting sub-sequences are too small to contain a single class, since the duration of the null class can be very small), the sequential data is first divided into 100 segments using the labels. 30 segments are left as test data.

Table 3.3: Mackay-Glass: Classifier performance for different amounts of unlabeled data

Model	600 Pairs	1800 Pairs	4000 Pairs
SSL-CP	0.87 $\pm$0.2	0.94 $\pm$0.1	0.96 $\pm$0.1
SSL-CP (ER)	0.87 $\pm$0.2	0.95 $\pm$0.1	0.99 $\pm$0.2

Table 3.4: HCI: Mean classifier performance when using one label per class

Supervised	Autoencoder	SSL-CP
0.63	0.68	0.72

This dataset presents a challenge to the SSL-CP approach in that most classes never appear adjacent to each other in the data set as they are always separated by a period in the null class. To obtain similarity constraints involving class pairs that do not include the null class, we generate a sequence by repeating a randomly sampled segment and concatenating it with another repeated randomly sampled segment. Change detection is then applied on this concatenated sequence to provide similar and dissimilar pairs. 600 of such similar dissimilar pairs were obtained.

When all labels within the dataset are provided, the mean F1 score for the supervised approach is 0.88. Such a score can actually sometimes be achieved by the supervised classifier even when only 1 label from each class is provided. However in this setting, the results can vary dramatically depending on exactly which instances are labeled. We obtained classification results across 30 trials, with a different random choice of which instance in each class were labeled. We show the average results in Table Table 3.4. In Table Table 3.5 we show the percentage of trials in which each method performed best.

WISDM: Activity recognition

The WISDM activity recognition dataset [58] consists of 36 users performing 6 activities which include running, walking, ascending stairs, descending stairs, sitting, and standing. Data is collected through an accelerometer mounted on the participant's chest which provides 3 dimensional

Table 3.5: HCI: Trial Percentage for method performing best when using 1 label per class

Supervised	Autoencoder	SSL-CP
11%	26%	63%

data sampled at 20Hz. For our experiments, we retained data from users 33, 34, and 35 as test set. We split the data from the rest of the users in a 70/30 ratio, using the large split for training and the small split for validation. We used a small sub-sequence (consisting of about 20 change-points) to tune the change detection parameters. Once tuned, we obtained change-points on the entire training set to obtain pairs of size 50. We obtained a total of about 4000 such pairs. We used about 7000 non-overlapping windows of size 50 as unlabeled data for the autoencoder. We used non-overlapping windows of size 50 as labeled data. In all experiments, we used a balanced number of labels from each class.

Table Table 3.6 shows results when 48 labels (6 from each class). When pairs from all detected change-points within the training set (4000 in number) are used, the performance of SSL-CP is slightly worse than that of the autoencoder. This is because many false change-points are detected (up to about 40% false change-points) for a small number of users, leading to erroneous similarity constraints. After the removal of 10 such users, the number of falsely detected change-points is reduced (to below 10% across all users) and about 1600 pairs are obtained. The performance of SSL-CP for this case (filtered users) is notably better than the autoencoder. The performance further improves when all true change-points are provided. In such a case, the number of unlabeled pairs are larger leading to improved performance of entropy regularization as well. Figure 3.5 shows the relationship between classification performance and the number of labels available. In this experiment, only pairs derived from change-points on the filtered users are used.

Table 3.6: WISDM: Classifier performance with 48 labels

Method	F1 score
Supervised	0.45 ± 0.04
Autoencoder	0.54 ± 0.02
SSL-CP (All users)	0.53 ± 0.03
SSL-CP (Filtered users)	0.65 ± 0.02
SSL-CP (True CPs, all users)	0.66 ± 0.01
SSL-CP-ER (Filtered users)	0.65 ± 0.01
SSL-CP-ER (True CPs, all users)	0.69 ± 0.01

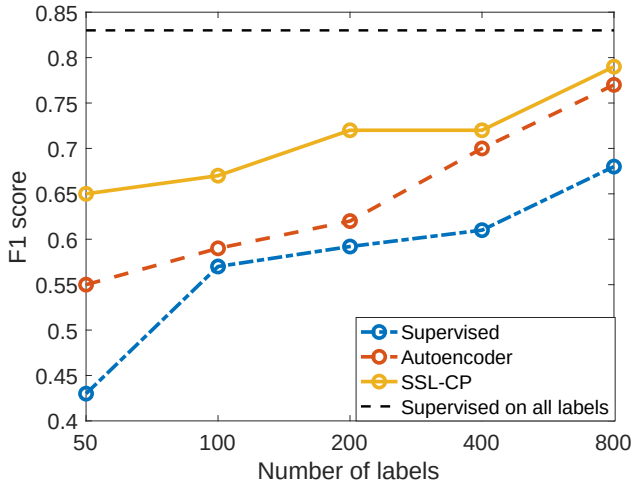


Figure 3.5: Performance on WISDM as the the number of provided labels increases. (Filtered users)

3.5 Discussion and conclusion

As highlighted by the performance on the WISDM dataset, the performance of our proposed method depends critically on the successful detection of change-points. The detection of too many false change-points can lead to corrupt similarity/dissimiarity constraints, that can potentially dete-

riorate performance. The other main limitation of the SSL-CP approach is that obtaining a rich set of similarity/dissimilarity constraints across all possible combinations of classes requires that these classes appear adjacent in the data. However, as we observed in the HCI dataset, the generation of additional sequences can provide a synthetic solution to this problem that is effective in practice.

Despite these limitations, SSL-CP consistently outperformed our baselines on both synthetic and real-world datasets. This clearly shows the potential utility of incorporating information from change-points in semi-supervised learning. Moreover, the results on the WISDM dataset clearly illustrate the potential improvement that could be realized by a more robust change-point detection procedures.

CHAPTER 4

LEARNING SINKHORN DIVERGENCES FOR CHANGE-POINT DETECTION

4.1 Introduction

As we discussed in the introduction of this thesis, change-point detection approaches can be broadly classified as either *offline* methods that focus on partitioning a complete sequence [9] or *online* methods that that can operate in a streaming setting. Online methods can vary in whether their focus is to rapidly detect a single change point (e.g.,[59]) or to detect a sequence of multiple change points (e.g.,[12, 30, 60, 61]). What nearly all of these online approaches have in common is that they are fundamentally *unsupervised*. This makes it particularly challenging to identify subtle changes, especially in the high-dimensional setting. Most of these methods employ a fixed measure of divergence over sliding windows to detect changes in a streaming setting, and there is not a mechanism for incorporating information about what kinds of changes should be detected and what kinds of changes should be ignored. Consider the example in Figure 4.1A, where true change point instances are shown by vertical red lines. These vertical lines indicate that true change points are associated with changes in the variance of the yellow signal. Most existing online change point methods cannot exploit this information, and as a result, often detect spurious changes. Incorporating labeled change point information could facilitate change detection methods in identifying what kinds of changes we wish to detect and what kind of changes we may wish to ignore.

In this chapter, we propose a contrastive metric learning framework to both improve change point detection performance and improve interpretability. We do this by learning a *ground metric* for Sinkhorn divergences to discriminate between different types of changes in our data. Sinkhorn divergences are computationally efficient variants of optimal transport distances that compute dis-

The work in this chapter is currently under peer-review and is available as a preprint at [4]. Parts of this chapter consisted of collaborations with Carolina Urzay and Mehdi Azabou that were presented in [5].

tances over a given metric. We show that we can use available change points to learn this metric to highlight changes of interest. This can both improve change point detection over unsupervised and non-parametric based approaches, and also reveal interpretable maps of which features are most important.

After training the metric on labeled change points, we show that the method can be deployed on sliding windows to capture change points with high accuracy and limited amounts of data and computation.

4.2 Background and related work

4.2.1 Change point detection

Change points are instances within a sequence where the underlying data generating distribution changes. Concretely speaking, let $\boldsymbol{X}$ denote a sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t \in \mathbb{R}^d$. We say that $\boldsymbol{X}$ has a change point at index n_c if $\boldsymbol{x}_{n_c}, \boldsymbol{x}_{n_c+1}, \ldots$ are generated according to a different distribution from $\boldsymbol{x}_{n_c-1}, \boldsymbol{x}_{n_c-2}, \ldots$. The most widely used algorithms for sequential change point detection are parametric approaches such as CUSUM and GLR. Both of these methods model changes as parametric shifts in the underlying distributions and operate on statistics formed from the log-likelihood ratio between the pre-change and post-change distributions [10, 20, 34, 59]. These methods are best suited to detecting a single change in the *quickest* time (given some false alarm constraint) mostly in settings where simple parametric models are realistic.

Over the past decade, there has been an increasing focus on *non-parametric* change point detection methods that operate over sliding windows. Integral probability metrics such as the kernel ratio, the maximum mean discrepancy (MMD), and Wasserstein distances are then used in a two-sample test detect change points [12, 60, 61]. Other kernel methods such as the Hilbert Schmidt Independence Criterion with ℓ_1 regularization (HSIC Lasso) have used sliding windows for feature selection in high dimensional change point settings [62]. Wasserstein distances provide an

alternate way to conduct two-sample tests without needing to tune kernel bandwidth parameters. Some recent works have also used neural network autoencoders and GANs to learn (unsupervised) representations that can then be used in sliding windows to detect change points [30, 63]. Other methods try to learn neural network representations that minimize representation distance for nearby sub-sequences while maximizing representation distance for far away sub-sequences [64]. This is different than our method where use triplet pairs from true change points to learn a ground metric for Sinkhorn divergences.

All of these methods detect change points in an unsupervised way. There is, however, limited work that incorporates supervision in the form of labeled change points. Most of this work (see [7, 65–67]) is restricted to the offline change point setting where a provided sequence is partitioned into segments, as opposed to the sliding window setting that we consider here. These segmentation methods often require predetermining the number of change points to detect. Other supervised methods treat change point detection as a classification problem where the goal is to predict the time instances where a change point may be located [68, 69]. All of these methods address a different problem of detecting change points in a provided sequence in an offline manner. Our method on the other hand addresses the problem of detecting change points through sliding windows in streaming data settings. As compared to existing supervised change point detection methods, our method provides a framework for learning a metric that can be used by divergence measures for detecting change points over sliding windows. There has been relatively little work in literature that explores how metric learning can be used to complement change point detection performance. This makes our contributions to the literature novel. The most similar work to our approach seems to be a method developed in parallel where true change points are used learn a sparse Mahalanobis metric for change point detection using provided triplet sub-sequences [70]. Though this method also attempts to address the offline change point detection task where a given sequence is to be partitioned into different segments, as compared to our method which equips divergences over sliding window with a learned metric for change detection on streaming data.

4.2.2 Wasserstein distances and Sinkhorn divergences

Wasserstein distances compute the minimal cost of transporting mass from one distribution to another. Concretely, consider two discrete multivariate distributions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ on $\mathbb{R}^d$. We can express these distributions as

$$\boldsymbol{\alpha} = \sum_{i=1}^{n} a_i \boldsymbol{\delta}_{\boldsymbol{x}_i} \quad \text{and} \quad \boldsymbol{\beta} = \sum_{j=1}^{m} b_j \boldsymbol{\delta}_{\boldsymbol{y}_j},$$

where $\boldsymbol{\delta}_{\boldsymbol{x}}$ is the Dirac measure at position $\boldsymbol{x} \in \mathbb{R}^d$, so that the $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ denote the mass locations for the distributions and $a_i, b_i \in \mathbb{R}_+$ are the weights at these mass locations for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ respectively. The ground cost metric $\boldsymbol{C} \in \mathbb{R}^{n \times m}$ represents the transportation cost between each pair of distribution mass locations. In this work, we consider **Wasserstein 2** ($\mathcal{W}^2$) distances that use a squared distance ground cost metric, where the $(i, j)^{\text{th}}$ entry of $\boldsymbol{C}$ is given by

$$\boldsymbol{C}_{i,j} = \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2^2.$$

As the goal is to minimize the cost of moving mass between two distributions, Wasserstein distances require computing a transport plan $\boldsymbol{P}$ that dictates how mass is transported between the distributions. This is done by solving the following optimization problem:

$$\begin{aligned} \mathcal{W}(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \min_{\boldsymbol{P}} \langle \boldsymbol{C}, \boldsymbol{P} \rangle, \\ \text{subject to } \boldsymbol{P} &\in \mathbb{R}_+^{n \times m}, \boldsymbol{P}^\top \mathbb{1}_n = \boldsymbol{b}, \boldsymbol{P} \mathbb{1}_m = \boldsymbol{a}, \end{aligned}$$

where $\langle \boldsymbol{C}, \boldsymbol{P} \rangle$ is the Frobenius inner product between the cost matrix $\boldsymbol{C}$ and the transport plan $\boldsymbol{P}$, $\boldsymbol{a}$ and $\boldsymbol{b}$ contain the mass weights for the distributions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and $\mathbb{1}_n \in \mathbb{R}^n$ is the vector of all ones.

Wasserstein distances can be unstable and computationally expensive to compute, requiring

$O(n^3 \log n)$ computations to evaluate in the case where n and m are of the same order. This makes it difficult to use Wasserstein distances repeatedly in two-sample tests. Additionally, the minimization problem can also be sensitive to slight changes in the input. One solution to these problems is to add a regularization term $\boldsymbol{H}(\boldsymbol{P})$ to form the entropic regularized Wasserstein distance $\mathcal{W}_\gamma$ [15, 71]. This is also known as the Sinkhorn distance and is defined as

$$\begin{aligned} \mathcal{W}_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \min_{\boldsymbol{P}} \langle \boldsymbol{C}, \boldsymbol{P} \rangle - \gamma \boldsymbol{H}(\boldsymbol{P}), \\ \text{subject to } \boldsymbol{P} &\in \mathbb{R}_+^{n \times m}, \boldsymbol{P}^\top \mathbb{1}_n = \boldsymbol{b}, \boldsymbol{P} \mathbb{1}_m = \boldsymbol{a}, \end{aligned} \tag{4.1}$$

where $\boldsymbol{H}(\boldsymbol{P})$ is the entropy of the transport plan matrix $\boldsymbol{P}$ and is given by

$$\boldsymbol{H}(\boldsymbol{P}) = -\sum_{i=1}^{n} \sum_{j=1}^{m} \boldsymbol{P}_{i,j} (\log \boldsymbol{P}_{i,j} - 1),$$

while γ is a regularization parameter. This regularization terms makes the minimization problem convex, which makes it less sensitive to changes in input, and can be solved with $O(n^2)$ computations using the Sinkhorn algorithm [15]. Note that the regularized Wasserstein distance is biased as $\mathcal{W}_\gamma^2(\boldsymbol{\alpha}, \boldsymbol{\alpha}) \neq 0$. An unbiased divergence can be constructed from these regularized Wasserstein distances and is called the **Sinkhorn divergence**:

$$S_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathcal{W}_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}) - \frac{1}{2}\mathcal{W}_\gamma(\boldsymbol{\alpha}, \boldsymbol{\alpha}) - \frac{1}{2}\mathcal{W}_\gamma(\boldsymbol{\beta}, \boldsymbol{\beta}). \tag{4.2}$$

The regularization parameter γ allows Sinkhorn divergences to interpolate between Wasserstein distances and energy distances [72, 73].

4.2.3 Learning a ground metric for optimal transport

While the squared distance is a natural choice for the ground cost metric, when it is available side information can also be used to learn an improved ground metric. This idea was first explored

to directly estimate the ground cost given similarity/dissimilarity information for nearest neighbor classification tasks [74]. Similarity/dissimilarity information was also used to learn a Mahalanobis ground metric to compare word embeddings through Wasserstein distances in [75]. Unsupervised ground metric learning has been also leveraged to devise subspace robust Wasserstein distances [76] that lead to better Wasserstein distance performance in high dimensional settings. This is done by finding an orthogonal projection of a given rank on the input data such that the Wasserstein distance between samples is maximized. Ground metric learning has also been used to compare entire time series/sequences using order preserving Wasserstein distances [77]. In such settings, time series labels are used to learn a ground metric. Other applications involving ground metric learning include domain adaptation and label distribution learning [78, 79].

4.2.4 Sinkhorn divergence with learned ground metric

A learned ground metric can be readily incorporated into the calculation of the Sinkhorn divergence. Suppose that we have learned a Mahalanobis metric parameterized by an inverse covariance matrix $\boldsymbol{M}$ with rank r, and consider the factorization $\boldsymbol{M} = \boldsymbol{L}^\top \boldsymbol{L}$, where $\boldsymbol{L}$ is an $r \times d$ matrix. For mass weight locations $\boldsymbol{x}_i, \boldsymbol{y}_j \in \mathbb{R}^d$, we can express the ground cost matrix in terms of this Mahalanobis distance as the matrix $\boldsymbol{C}_{\boldsymbol{L}}$ with $(i,j)^{\text{th}}$ element given by

$$\boldsymbol{C}_{\boldsymbol{L}i,j} = \|\boldsymbol{L}(\boldsymbol{x}_i - \boldsymbol{y}_j)\|_2^2.$$

$\boldsymbol{C}_{\boldsymbol{L}}$ can then be used to compute the Sinhkorn distance:

$$\mathcal{W}_{\boldsymbol{L},\gamma}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \min_{\boldsymbol{P}} \langle \boldsymbol{C}_{\boldsymbol{L}}, \boldsymbol{P} \rangle - \gamma \boldsymbol{H}(\boldsymbol{P}), \tag{4.3}$$
$$\text{subject to } \boldsymbol{P} \in \mathbb{R}_+^{n \times m}, \boldsymbol{P}^\top \mathbb{1}_n = \boldsymbol{b}, \boldsymbol{P}\mathbb{1}_m = \boldsymbol{a}.$$

As before, these parameterized Sinkhorn distances can be used to obtain the parameterized Sinkhorn divergence:

$$S_{\boldsymbol{L},\gamma}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \mathcal{W}_{\boldsymbol{L},\gamma}(\boldsymbol{\alpha},\boldsymbol{\beta}) - \frac{1}{2}\mathcal{W}_{\boldsymbol{L},\gamma}(\boldsymbol{\alpha},\boldsymbol{\alpha}) - \frac{1}{2}\mathcal{W}_{\boldsymbol{L},\gamma}(\boldsymbol{\beta},\boldsymbol{\beta}) \tag{4.4}$$

4.3 Proposed method

4.3.1 Sinkhorn divergence on sequences

Before considering how supervised change detection can be performed by combining Sinkhorn divergence with a learned ground metric, we first clarify how Sinkhorn divergences can be applied to compare two sequences. Consider two sequences $\boldsymbol{X} \in \mathbb{R}^{T\times d}$ and $\boldsymbol{Y} \in \mathbb{R}^{T\times d}$, where T is the length of the two sequences (which for simplicity we assume to be equal) and d is the dimension of each sample in the two sequences. We can construct the empirical distribution of $\boldsymbol{X}$ and $\boldsymbol{Y}$ via

$$\sum_{i=1}^{T} \frac{1}{T}\boldsymbol{\delta}_{\boldsymbol{x}_i} \quad \text{and} \quad \sum_{i=1}^{T} \frac{1}{T}\boldsymbol{\delta}_{\boldsymbol{y}_i}$$

These empirical distributions take uniform weights of $\frac{1}{T}$ at the mass locations. While there are many other ways to represent the sequences as discrete distributions, this scheme is often [72] used because of it simplicity. The parameterized Wasserstein distance between the two sequences can be computed as:

$$\mathcal{W}_{\boldsymbol{L},\gamma}(\boldsymbol{X},\boldsymbol{Y}) = \min_{\boldsymbol{P}} \sum_{i,j=1}^{T} \boldsymbol{P}_{i,j}\|\boldsymbol{L}(\boldsymbol{x}_i - \boldsymbol{y}_j)\|_2^2 - \gamma\boldsymbol{H}(\boldsymbol{P}) \tag{4.5}$$

$$\text{subject to } \boldsymbol{P} \in \mathbb{R}_+^{T\times T}, \boldsymbol{P}^\top \mathbb{1}_T = \mathbb{1}_T, \boldsymbol{P}\mathbb{1}_T = \mathbb{1}_T.$$

Note that we will slightly abuse notation in writing $\mathcal{W}_{\boldsymbol{L},\gamma}(\boldsymbol{X},\boldsymbol{Y})$ by letting $\boldsymbol{X}$ and $\boldsymbol{Y}$ denote both the empirical distribution of the sequences as well as the sequences themselves.

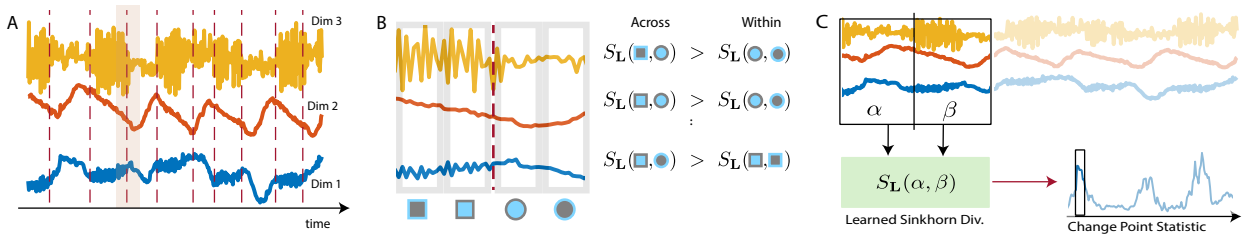


Figure 4.1: Overview of our approach. In (A-B), we show how labeled change point instances (red vertical lines) are used to obtain similarity triplets from pre-change (red) and post-change (green) windows as shown in B. These triplets are then used to learn a linear transformation $\boldsymbol{L}$ that ensures samples across the change points are far apart and samples on the same side of a change point are closer. (C) After learning $\boldsymbol{L}$, we use a two-sample test in a sliding window to perform online change point detection.

4.3.2 Generating similarity triplets from change points

As shown in Figure Figure 4.1, a true change point can be used to generate similar and dissimilar pairs of sub-sequences. In Figure 4.1(B), we obtain two sub-sequences before a change point that are labeled 1 and 2 in red. Mathematically, we refer to these as $\boldsymbol{X}_1^p$ and $\boldsymbol{X}_2^p$ respectively. Similarly we can obtain two sub-sequences after the change point that are labeled 1 and 2 in green labels. We refer to these as $\boldsymbol{X}_1^f$ and $\boldsymbol{X}_2^f$. Sub-sequences on the same side of the change should be similar, whereas the sub-sequences on the opposite side of the change points should be dissimilar. This can be captured mathematically via the Sinkhorn divergence via a constraint that, for example, $S_{\boldsymbol{L},\gamma}(\boldsymbol{X}_1^p, \boldsymbol{X}_2^p)$ should be smaller than $S_{\boldsymbol{L},\gamma}(\boldsymbol{X}_1^p, \boldsymbol{X}_2^f)$. Such constraints can be represented as triplets $(\boldsymbol{X}_i, \boldsymbol{X}_i^s, \boldsymbol{X}_i^d)$, where $\boldsymbol{X}_i^s$ represents a sequence that is nearer to $\boldsymbol{X}_i$ than $\boldsymbol{X}_i^d$. From each labeled change point we can construct such triplets.

4.3.3 Learning a ground metric for change detection

Our goal is to leverage the triplets generated from the labeled change points to learn a ground metric such that the Sinkhorn divergence $S_{\boldsymbol{L},\gamma}$ does a better job of highlighting change points. The metric learning community has been considering similar problems in a range of works [80, 81]. When comparing distributions, the Wasserstein distance, or its variants such as the Sinkhorn

divergence, can capture differences in geometry of samples in a distribution, helping detect samples with dissimilar distributions. Equipping Sinkhorn divergences with a learned ground metric can further improve this ability by transforming the data in a way that highlights dissimilarities (and de-emphasizes similarities). This can be done by using the triplet loss

$$l(\boldsymbol{L}) = \sum_{i \in \text{Triplets}} \left[c - (\mathcal{S}_{\boldsymbol{L},\gamma}(\boldsymbol{X}_i, \boldsymbol{X}_i^d) - \mathcal{S}_{\boldsymbol{L},\gamma}(\boldsymbol{X}_i, \boldsymbol{X}_i^s)) \right]^+, \tag{4.6}$$

where c is the triplet margin, $[.]^+$ is the hinge loss defined as $[d]^+ = \max(0, d)$, and $S_{\boldsymbol{L},\gamma}$ is the parameterized Sinkhorn divergence from (4.4).

The gradient of the parameterized Sinkhorn divergence between two sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ with respect to $\boldsymbol{L}$ can be computed as:

$$\frac{\partial \mathcal{S}_{\boldsymbol{L}}(\boldsymbol{X}, \boldsymbol{Y})}{\partial \boldsymbol{L}} = 2\boldsymbol{L} \sum_{i,j=1}^{T} \boldsymbol{P}^*_{i,j} (\boldsymbol{x}_i - \boldsymbol{y}_j)(\boldsymbol{x}_i - \boldsymbol{y}_j)^\top,$$

where $\boldsymbol{P}^*$ is the optimal transport plan computed by solving (4.5). The gradient of the triplet loss function in (4.6) is

$$\begin{aligned} \frac{\partial l(\boldsymbol{L})}{\partial \boldsymbol{L}} = \sum_{v \in \text{Viol}} 2\boldsymbol{L} \sum_{i,j=1}^{T} \boldsymbol{P}^*_{v_i, v_j^s} (\boldsymbol{x}_{v_i} - \boldsymbol{x}^s_{v_j})(\boldsymbol{x}_{v_i} - \boldsymbol{x}^s_{v_j})^\top \\ - 2\boldsymbol{L} \sum_{i,j=1}^{T} \boldsymbol{P}^*_{v_i, v_j^d} (\boldsymbol{x}_{v_i} - \boldsymbol{x}^d_{v_j})(\boldsymbol{x}_{v_i} - \boldsymbol{x}^d_{v_j})^\top, \end{aligned} \tag{4.7}$$

where v is the index for similarity triplets that violate the hinge loss constraint in (4.6), $\boldsymbol{P}^*_{v,v^s}$ is the transport plan between $\boldsymbol{X}_v$ and its similar pair $\boldsymbol{X}^s_v$, and $\boldsymbol{P}^*_{v,v^d}$ is the transport plan between $\boldsymbol{X}_v$ and its dissimilar pair $\boldsymbol{X}^d_v$. Algorithm Algorithm 3 shows how this gradient can be used to learn the linear transform matrix $\boldsymbol{L}$. Algorithm Algorithm 4 shows how this learned matrix can be used for change detection over sliding windows. Our method, like all methods based on sliding

Algorithm 3 Learn transform matrix $\boldsymbol{L}$ for ground metric using similarity triplets from true change points

Inputs: Set of triplets $(\boldsymbol{X}_i, \boldsymbol{X}_i^s, \boldsymbol{X}_i^d)$ from true change points, Sinkhorn regularization parameter γ, Gradient descent rate μ, Triplet loss constant c

Output: Trained $\boldsymbol{L}$

Initialize: $\boldsymbol{L}_0$

for $t = 1$ to number of iterations **do**

Identify triplet indices v that violate the hinge constraint

Compute transport plan between similar pairs $\boldsymbol{P}^*_{v,v^s}$ by solving $\mathcal{S}_{\boldsymbol{L}_{t-1}}(\boldsymbol{X}_v, \boldsymbol{X}_v^s)\ \forall\ v$

Compute transport plan between dissimilar pairs $\boldsymbol{P}^*_{v,v^d}$ by solving $\mathcal{S}_{\boldsymbol{L}_{t-1}}(\boldsymbol{X}_v, \boldsymbol{X}_v^d)\ \forall\ v$

Use computed transport plans $\boldsymbol{P}^*_{v,v^d}$ and $\boldsymbol{P}^*_{v,v^s}$ to form gradient $\frac{\partial l(\boldsymbol{L})}{\partial \boldsymbol{L}}$

$\boldsymbol{L}_t = \boldsymbol{L}_{t-1} - \mu \frac{\partial l(\boldsymbol{L})}{\partial \boldsymbol{L}}$

end for

windows, assumes that the length of the windows is large enough to capture the statistics of the pre-change and post-change distributions. We also assume that the triplet sub-sequences should be of sufficient length to capture distributional information for learning the ground metric.

Algorithm 4 Using Sinkhorn divergence with learned metric for change detection

Inputs: Sequence $\boldsymbol{X}$, Window length w, Change threshold τ, Learned $\boldsymbol{L}$

Output: Detected changes

for $n = 1$ to length of sequence $\boldsymbol{X}$ **do**

Form consecutive windows $\boldsymbol{X}_p^n, \boldsymbol{X}_f^n$ at index n

$m_n = S_{\boldsymbol{L},\gamma}(\boldsymbol{X}_p^n, \boldsymbol{X}_f^n)$ using (4.2)

if $m_n > \tau$ **then**

Add n to change points

end if

end for

4.3.4 Learning a sparse ground metric

Additional regularization terms can be used in conjunction with the triplet loss to learn a ground metric that is suitable for different change detection settings. For example, adding a regularizing with an ℓ_1 or a mixed norm loss has been used for learning a sparse metric [82]. We can use the

same idea for learning a sparse ground metric by considering, for example,

$$\min_{\boldsymbol{L}} l(\boldsymbol{L}) + \lambda \|\boldsymbol{L}\|_1.$$

Such an approach aims to learn a metric that depends on only a sparse subset of the original features in $\mathbb{R}^d$.

4.3.5 Learned metric for two-sample tests using Sinhkorn divergences

Sample complexity results for Sinkhorn distances, $\mathcal{W}_\gamma$, were given in [83]. A straightforward extension of these results can be obtained for Sinkhorn divergences.

Proposition 1. *If n samples are used to compute the empirical distributions $\widehat{\boldsymbol{\alpha}}_n \sim \boldsymbol{\alpha}$ and $\widehat{\boldsymbol{\beta}}_n \sim \boldsymbol{\beta}$ on $\mathbb{R}^d$, where $\widehat{\boldsymbol{\alpha}}_n = \sum_{i=1}^n \frac{1}{n}\boldsymbol{\delta}_{\boldsymbol{x}_i}$ and $\widehat{\boldsymbol{\beta}}_n = \sum_{j=1}^n \frac{1}{n}\boldsymbol{\delta}_{\boldsymbol{y}_j}$, then the deviation of the Sinkhorn divergence between these empirical distributions from the true distributions is bounded with probability $1-\delta$:*

$$|S_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}) - S_\gamma(\widehat{\boldsymbol{\alpha}}_n, \widehat{\boldsymbol{\beta}}_n)| \leqslant 12B\frac{\rho K}{\sqrt{n}} + C\sqrt{\frac{8\log(\frac{2}{\delta})}{n}}, \tag{4.8}$$

where $\rho = O(\max(1, \frac{1}{\gamma^{d/2}})$, K is the maximum value taken by the kernel associated with the dual Sinkhorn potentials, B is the Lipschitz constant of the function depicting the dual formulation of entropic regularized Wasserstein distance in (C.3)*, while C is the upper bound of this function. Here $C = \kappa + \gamma\exp(\kappa/\gamma)$, $\kappa = 2L|\mathcal{X}| + \|c\|_\infty$ and L is the Lipschitz constant of the cost/distance c between mass locations. $|\mathcal{X}|$ represents the diameter of the space of mass locations $\boldsymbol{x} \in |\mathcal{X}|$. Additionally, the Lipschitiz constant B is upper bounded by $\leqslant 1 + \exp(2\kappa/\gamma)$.*

More details for these constants are provided in the Appendix. These sample complexity results can be used to obtain deviation bounds for the Sinkhorn divergence under the null distribution.

Corollary 3. *With probability $1-\delta$, the Sinkhorn divergence between two n samples $\boldsymbol{\alpha}_n^1, \boldsymbol{\alpha}_n^2 \sim \boldsymbol{\alpha}$,*

is bounded by

$$|S_\gamma(\widehat{\boldsymbol{\alpha}}_n^1, \widehat{\boldsymbol{\alpha}}_n^2)| \leqslant 12B\frac{\rho K}{\sqrt{n}} + C\sqrt{\frac{8\log(\frac{2}{\delta})}{n}}. \tag{4.9}$$

As the metric learning loss in (4.6) contains similar pairs from the same distribution, an ideally learned metric would ensure that $S_{\boldsymbol{L},\gamma}(\boldsymbol{\alpha}, \boldsymbol{\alpha}) = 0$, while $S_{\boldsymbol{L},\gamma}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \geqslant c$ for dissimilar pairs, where c is the triplet loss margin. In other words, the Sinkhorn divergence between similar pairs from the same distribution would be 0 and the Sinkhorn divergence between dissimilar pairs would be greater than the margin c. If this margin c is set such that $c > S_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta})$, i.e., greater than the Sinkhorn divergence without a learned metric, then it is likely that $S_{\boldsymbol{L},\gamma}(\boldsymbol{\alpha}, \boldsymbol{\beta}) > S_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta})$. From (4.8), this will likely ensure that $S_{\boldsymbol{L},\gamma}(\widehat{\boldsymbol{\alpha}}_n, \widehat{\boldsymbol{\beta}}_n) > S_\gamma(\widehat{\boldsymbol{\alpha}}_n, \widehat{\boldsymbol{\beta}}_n)$, resulting in a Sinkhorn divergence with increased testing power.

Under the case where both samples come from the same distribution, the results in (4.9) show that when the regularization parameter γ is small, a large dimension of the input distributions can lead to a large Sinkhorn divergence, which would result in false change points. Learning a ground metric allows us to enforce a structure on the distribution that improves sample complexity results. This can be done by projecting the distribution into low-dimensional subspace, as explored in [76, 84]. Learning either a low-dimensional transformation matrix $\boldsymbol{L}$ or a sparse transformation matrix $\boldsymbol{L}$ reduces the effective dimension of the data distribution, leading to improved performance by detecting fewer false change points. By reducing the effective dimension we mean that the output dimensionality r of $\boldsymbol{L}$ could be chosen such that it transforms input data to a space that is lower dimensional than the original input dimension, or a sparse transformation where the number of non-zero dimensions is lower than the original number of dimensions. A lower dimensional output space of $\boldsymbol{L}$ results in a smaller λ for the results in Proposition 1 and Corollary 1, leading to smaller difference between the empirical and true Sinkhorn divergences, resulting in fewer false change points. Though transforming input to a lower dimensional space through $\boldsymbol{L}$ can decrease the probability of detecting false change points, it reduces the number of parameters that are learned by $\boldsymbol{L}$.

This may diminish the ability to learn a metric that satisfies the triplet margin constraints between dissimilar samples, particularly samples from distributions that are close in terms of statistical distance. Learning $\boldsymbol{L}$ that makes the Sinkhorn distance large between samples from such distributions would require a more expressive $\boldsymbol{L}$ with higher degrees of freedom.

4.4 Experiments and Results

4.4.1 Evaluation Metrics

We use true change point labels and predicted change scores, which are the computed change statistics, to generate ROC curves [85] using scikit-learn[86]. These curves relate the false positive and true positive rates for different detection thresholds on provided change detection scores. We then report the area under these ROC curves to evaluate change detection performance using the same protocol as that used in [30, 61]. We also provide F1 scores at detection margins d, such that a true change point is detected whenever the distance between a true change point location and a predicted change point location is less than d. F1 scores provide the harmonic mean of precision and recall for detecting change points, and are more suitable for measuring change detection performance than accuracy. These F1 scores were calculated using the same method that was used for other change point detection methods in the literature [61].

4.4.2 Datasets

Switching variance. We simulate the AR process

$$x_1(t) = 0.6x_1(t-1) - 0.5x_1(t-2) + \epsilon_t,$$

where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ and σ switches between $\sigma = 1$ and $\sigma = 5$ every 100 time steps. We also generate a noise vector $[x_2(t), \ldots, x_{50}(t)] \sim \mathcal{N}(\mathbf{0}_{49}, \boldsymbol{I}_{49}))$. We concatenate $x_1(t)$ with $[x_2(t), \ldots, x_{50}(t)]$ to obtain a 50 dimensional vector $[x_1(t), \ldots, x_{50}(t)]$ where changes are only happening in the first

Table 4.1: AUC and F1 scores for different change point detection methods on simulated datasets.

Method	GMM		Swt Freq	
	AUC	F1	AUC	F1
HSIC	0.493	0.423	0.426	0.561
M-stats	0.947	0.664	0.553	0.759
GauSymKL	0.823	0.742	0.428	0.405
TIRE_{T}	0.501	0.318	0.551	0.727
TIRE_{F}	0.677	0.329	0.647	0.732
KLCPD	0.802	0.486	0.941	0.951
SinkDv	0.778	0.476	0.797	0.759
SinkDivLM	0.974	0.985	0.981	0.972

dimension.

Switching Gaussian mixtures Two 100 dimensional Gaussian mixture distributions $\boldsymbol{\alpha} = \mathcal{N}(\mathbf{0}, \boldsymbol{I}) + \mathcal{N}(\mathbf{1}, \boldsymbol{\Sigma}_0)$ and $\boldsymbol{\beta} = \mathcal{N}(\mathbf{0}, \boldsymbol{I}) + \mathcal{N}(\mathbf{1.5}, \boldsymbol{\Sigma}_1)$ were used to simulate a sequence where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ switched every 100 samples. $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Sigma}_1$ are diagonal covariance matrices where the first 3 entries on the diagonal are 3 and 5 respectively, while the rest of the diagonal entries are 1. A training sequence consisting of 25 changes, of which 80 percent used for training and 20 percent were used for validation, was used to train the ground metric. A separate testing sequence consisting of 25 changes was used to evaluate performance.

Switching frequency mixture A two dimensional sequence where the first dimension switches between $\sin(2\pi(0.1)t) + \sin(2\pi(0.5)t) + \sin(2\pi(0.3)t)$ and $\sin(2\pi(0.1)t) + \sin(2\pi(0.5)t) + \sin(2\pi(0.35)t)$ every $t = 100$. The second dimension switches between $\sin(2\pi t) + \sin(2\pi(1.5)t) + \sin(2\pi(1.70)t)$ and $\sin(2\pi t) + \sin(2\pi(1.5)t) + \sin(2\pi(0.35)t)$ every $t = 100$. These sequences are generated such that there are 10 samples per second. Both these dimension have $\mathcal{N}(0, 0.1)$ noise added. 15 changes points are used to train the ground metric, 4 are used for validation. A different sequence consisting of 19 changes points is used as the test set.

Table 4.2: AUC and F1 scores for different change point detection methods on real-world datasets

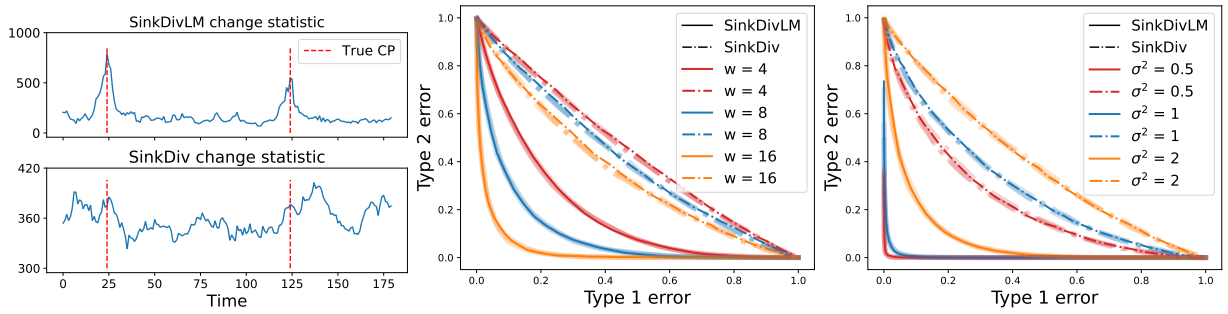
Method	BeeDance		HASC('11)		HASC('16)		Yahoo		ECG	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
HSIC	0.543	0.796	0.603	0.643	0.591	0.697				
M-stats	0.494	0.741	0.605	0.651	0.751	0.772	0.737	0.580	0.844	0.321
GauSymKL	0.485	0.312	0.779	0.737	0.750	0.791	0.780	0.349	0.803	0.606
TIRE_T	0.539	0.693	0.659	0.712	0.643	0.727	0.865	0.531	0.747	0.381
TIRE_F	0.556	0.693	0.725	0.710	0.712	0.731	0.871	0.546	0.900	0.411
KLCPD	0.632	0.805	0.663	0.692	0.742	0.771	0.932	0.580	0.810	0.566
SinkDv	0.556	0.773	0.757	0.815	0.717	0.750	0.942	0.654	0.900	0.680
SinkDivLM	0.682	0.854	0.803	0.824	0.759	0.801	0.946	0.675	0.899	0.679

Switching frequency mixture with slopes This includes the frequency switching dataset concatenated with 48 additional dimensions whose slopes change every 1000 samples, resulting in a 50 dimensional dataset. True change point are only labeled at instances where the frequencies in the first two dimensions change. For 24 of the additional dimensions the slope changes from a gradient of -0.06 to 0.06, and for the other 24 dimensions the slopes change from 0.06 to -0.06. All of the slope dimensions have $\mathcal{N}(0, 0.0001)$ noise added.

Real world datasets

Real world datasets

Bee Dance Bees are tracked using videos to obtain three dimensional sequences, where the first two dimensions represent the x,y coordinates for bee location, while the third dimension shows the bee heading angle. Instances where the bee waggle dance changes from one stage to the other are labeled as change points. The dataset consists of 6 sequences. We used two sequences for training and validation, while the rest of the sequences are used as test datasets. In total there are 15 change points that are used for training, of which 12 are used for training and 3 for validation.



(a) Detected changes (b) Errors across window sizes (c) Errors across noise levels

Figure 4.2: *Results for the switching GMM dataset*. On the left, we show the change point statistic for the Sinkhorn divergence with (SinkDivLM) and without (SinkDiv) a learned metric. The red vertical lines show true change point locations. To the right, we show Type 1 vs Type 2 errors for both approaches as we vary the (B) window sizes and (C) amount of added noise. The experiment is repeated 5 times with different seeds which are shown in fainter lines.

HASC (Activity Detection) HASC-2011 and HASC-2016 datasets consists of people performing different activities such as walking, running, skipping, staying. An accelerometer provides a three dimensional sequence where changes are labeled when there is a change in activity. A single sequence from HASC-2016 was used for true labels and training the ground metric. This sequence provided 15 true change points of which 80 percent are used for training and 20 percent for validation. The rest of the 89 sequence datasets in the HASC-2016 are used as test datasets. A single sequence dataset, the same used by [30], from HASC-2011 is also used as a test dataset.

Yahoo 15 sequences containing change points that indicate a change in different metrics, such as CPU usage, memory usage, etc. All sequences are one dimensional. We use 3 change points from one of the 15 sequences to train our metric and use the rest of the sequences for evaluation.

ECG A single dimensional sequence containing change point labels at Ischemia, or abnormal heartbeat, instances. We split the dataset for training and testing, and use 21 change points for training and validation, in an 80-20 split.

Mouse Sleep Stage This dataset consists of a mouse going through different sleep stages [87] (REm, nREM, and wake) over 12 hours. The dataset consists of the spiking activity of 42 neurons detected from multi-electrode arrays implanted in the hippocampus. A sub-sequence that contained 14 change points (between REM and nREM) were used to train and validate the learned ground metric. A different sub-sequence consisting of 28 change points was used as a test dataset.

Baselines We compare the performance of our method (SinkDivML) with different methods that we classify into two categories: those that require access to true change points for learning a model and those that do not.

Baselines not requiring access to true change points: These baselines include Sinkhorn divergence without learning a metric (SinkDiv), M-stats [60], HSIC [62], and GauSymKL. GausSymKL is based on applying symmetrical KL divergences between two sliding windows that are assumed to follow a Gaussian distribution. The distribution parameters for the windows are estimated through samples within these windows. It provides a strong baseline for simulated examples involving Gaussian data. We use a symmetrical KL divergence as it is symmetrical for changes in variance which makes it easier to set detection thresholds [1]. SinkDiv and M-stats do not involve any model learning whereas HSIC learns a model using pseudo labels only.

Baselines requiring access to true change points: These includes generative neural network based kernel change point (KLCPD), autoencoder based methods in time domain TIRE_T, and frequency domain TIRE_F[63]. Though these models are trained in an unsupervised manner, they need access to true change labels to tune and validate the learned model. We use the same sequence datasets to train and validate these models that we use to train and validate our method. We also use a supervised version of HSIC (sHSIC) for experiments that involve feature selection in high dimensional change settings. Rather than using pseudo labels, which HSIC uses, we use true change point labels for feature selection.

All of these baselines consist of non-parametric change point detection method which make

the dominant class of recent change methods that are suitable for real-world datasets.

4.4.3 Results

AUC and F1 scores Table 4.1 and Table 4.2 shows the AUC and F1 performance for different methods on these datasets. As these datasets are mostly either low dimensional or involve changes in all dimensions , the ground metric is learned without ℓ_1 regularization. SinkDivML performs best on almost all datasets. The Yahoo and ECG datasets involve abrupt instantaneous changes (and thus involve a very small window size of size 2 and 3 respectively). Our method assumes that changes persist for some time to learn a metric, and these datasets involve transient changes leaving very small sub-sequence windows to learn a metric. For this reason, the performance gain is not prominent on these datasets. Additionally the performance is already relatively strong for other methods on these datasets, leaving relatively little room for improvement. Results for HSIC were not available for one dimensional datasets such as Yahoo and ECG as HSIC requires a multidimensional sequence out of which the most important dimensions are selected. When the one dimensional sequence is repeated across new artificial dimensions to create new dimensions, HSIC fails to distinguish between sequences and fails. Details on the detection margins used for these F1 scores can be found in Appendix Section C.1. Our proposed method, SinkDivLM, performs particularly well on BeeDance and HASC datasets where it outperforms all other baselines. SinkDivLM performs particularly well on the GMM dataset by learning a metric that involves projection onto a lower dimension, leading to much better results. This is further discussed in the next section.

Learned Metric improves Type1 vs Type 2 error performance When a ground metric is correctly learned, the Sinkhorn divergence between dissimilar samples is larger than the provided margin constant c in (4.6). This results in the change point statistics between dissimilar sub-sequences being larger than the change point statistics between similar sub-sequences, which makes it easier to set a threshold that correctly detects true change points with out detecting many false change

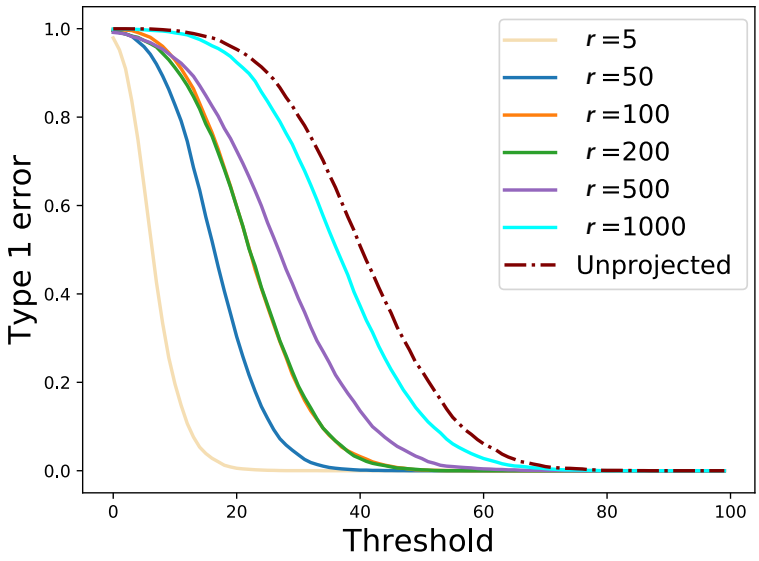
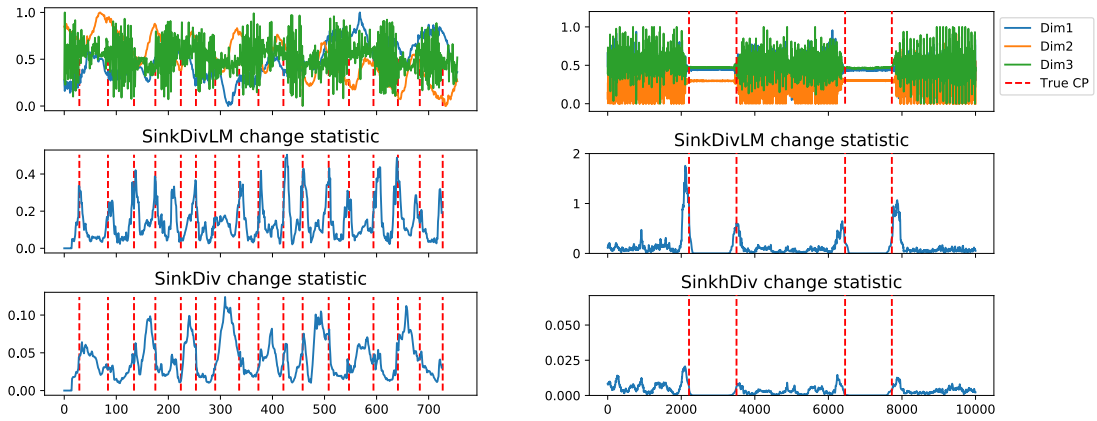


Figure 4.3: Type 1 errors vs detection threshold for learned $\boldsymbol{L}$. As we decrease the dimensionality r of the output space of $\boldsymbol{L}$, type 1 errors decrease for the same detection threshold.

points.Figure 4.2a shows how a learned metric leads to a larger change statistic between dissimilar sequences, which improves change point detection performance. A larger Sinkhorn divergence between dissimilar sequences also leads to improved results on real-world datasets such as BeeDance and HASC. SinkDivLM leads to major improvements on real world datasets such as BeeDance and HASC(2011,2016) which are widely used in other change point detection works [30, 61, 63].

Figures Figure 4.2b and Figure 4.2c show the relationship between the probability of having type 1 and type 2 errors between samples from the two 100 dimensional Gaussian Mixture Models, that are used in the GMM switching dataset, at different noise and window sizes. A type 1 error occurs when a sample from the null hypothesis is incorrectly judged to be from the alternate hypothesis, while a type 2 error occurs when a sample from the alternate hypothesis is judged to be from the null hypothesis. Samples for the null hypothesis were generated using $\boldsymbol{\alpha}$, while samples for the alternate hypothesis were generated using $\boldsymbol{\beta}$ described in Subsection 4.4.2. Solid lines represent the Sinkhorn divergence with a metric that is learned without added noise using 10 samples from each distribution, while dashed lines represent Sinkhorn divergence without a learned ground metric. For Figure 4.2b, noise of $\mathcal{N}(\mathbf{0}, 2\boldsymbol{I})$ was added. InFigure 4.2c 10 samples



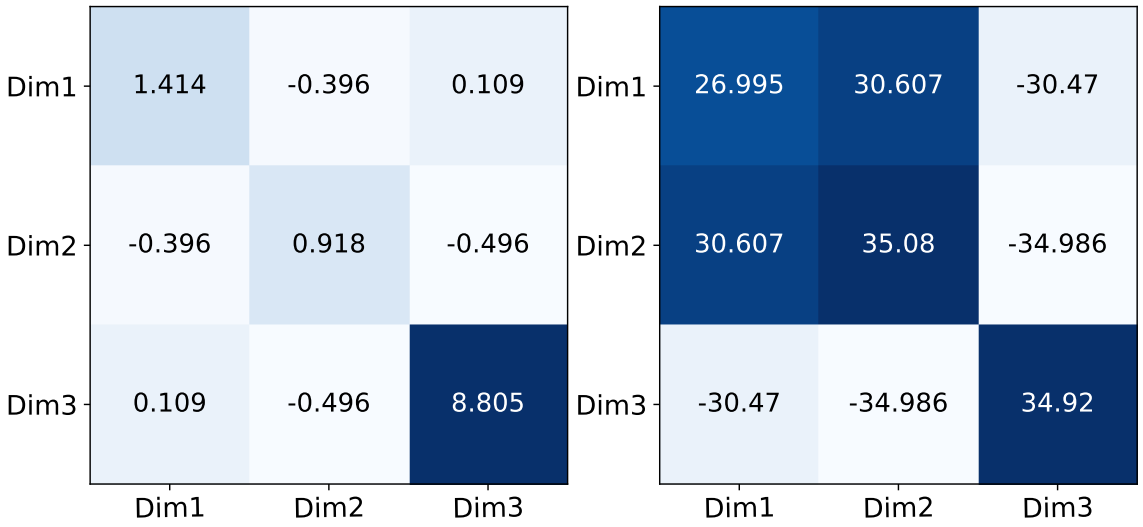
(a) Sequence from Beedance

(b) A sequence from the HASC dataset

Figure 4.4: Figure 4.4a shows an example sequence with true change points shown by vertical lines. It can be seen that changes are often associated by changes in the variance of the third dimension signal. The learned metric in Figure 4.5a captures this information as the 3 dimension is associated with a much larger value as compared to other dimensions. The 2nd subplot in Figure 4.4a shows the learned change statistic for Sinkhorn divergence with the learned metric. It does a much better job at identifying change points than Sinkhorn divergences without a learned metric shown in the subplot. Similarly, Figure 4.4b shows an example sequence from the HASC dataset where the learned metric leads to much better performance for SinkDivLM over SinkDiv.

from each distribution were used. Both these figures show that the learned metric achieves lower type 2 errors at the same type 1 error rates

Type 1 error vs output dimension of $\boldsymbol{L}$ Here we further analyze two samples tests between samples from two 100 dimensional Gaussian mixture models; $\boldsymbol{\alpha} = \mathcal{N}(\mathbf{0}, \boldsymbol{I}) + \mathcal{N}(\mathbf{1}, \boldsymbol{\Sigma}_0)$ and samples from $\boldsymbol{\beta} = \mathcal{N}(\mathbf{0}, \boldsymbol{I}) + \mathcal{N}(\mathbf{1.5}, \boldsymbol{\Sigma}_1)$. $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Sigma}_1$ are diagonal covariance matrices where the first 3 entries on the diagonal are 3 and 5 respectively, while the rest of the diagonal entries are 1. $\boldsymbol{\alpha}$ is chosen to be the null distribution and $\boldsymbol{\beta}$ is used as the alternate distribution for these experiments. An additional noise of $\mathcal{N}(0, 2)$ was added to these samples. Figure Figure 4.3 shows the relationship between type 1 errors and detection thresholds at various values of r, which is the dimensionality of the output space of matrix $\boldsymbol{L}$. Type 1 errors incorrectly reject the null hypothesis as the test erroneously concludes that the noisy samples from the null distribution are generated by the alternate distribution. For the same threshold, larger output dimensional spaces have larger



(a) Metric learned for Beedance (b) Metric learned for Beedance

Figure 4.5: Figures showing the learned metrics for Beedance and HASC datasets

Table 4.3: Results for different change-point detection method on high-dimensional datasets

Dataset	HSIC	sHSIC	GauSymKL	SinkDiv	**SinkDivLM**
Sleep Stage	0.668	0.941	0.857	0.925	0.946
Swch Var	0.868	0.934	0.782	0.567	0.931
Swch Frq/Slp	0.592	0.521	0.12	0.331	0.672

type 1 errors. These results support our discussion in Subsection 4.3.5 where we describe how high dimensional data can cause Sinkhorn divergences to detect false change points.

Learning a sparse metric for feature selection When ℓ_1 regularization is used the learned ground metric, our method can identify time series dimensions that correspond to changes of interest. For the switching variance dataset, the learned metric $\boldsymbol{L}$ is large in magnitude for indices corresponding to the dimension in which the variance is changing. The true changes between REM and non-REM sleep in the mouse sleep stage dataset also correspond to changes in only some of the 48 dimensions (different neurons), which the ℓ_1 regularized ground metric is able to select. Table 4.3 shows results on switching variance and sleep stage datasets. As these datasets are high-dimensional, ℓ_1 regularization is used to learn a sparse ground metric.

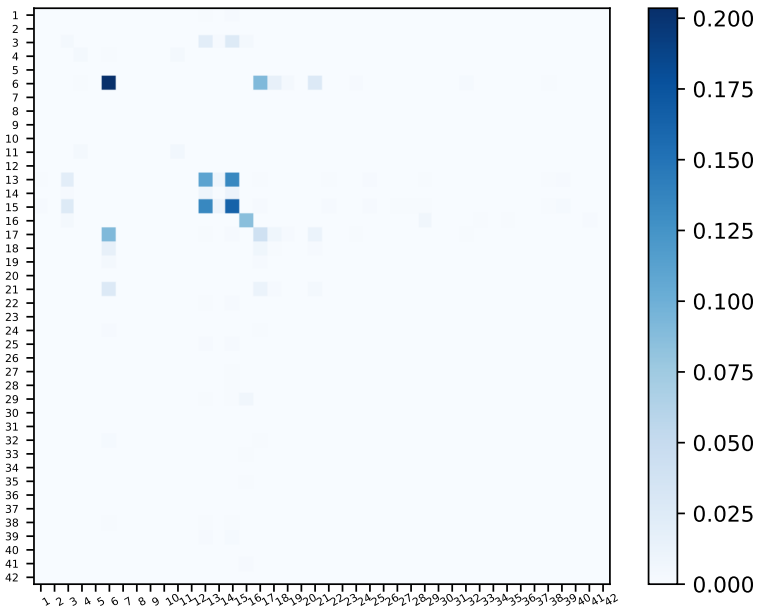


Figure 4.6: Learned sparse metric for the Sleep Stage dataset.

Though HSIC is presented in [62] as an unsupervised method that finds features that maximize separation by using pseudolabels at every time instance, we can also use HSIC for feature selection in a supervised manner by focusing on true change points. Both HSIC and our method aim to focus on finding a small number of features that can predict changes of interest. However, our method can also identify multivariate patterns (or correlations in different variables) that must be present for a change point to be detected. For this reason, when we tested both methods on the switching frequency with slopes dataset. HSIC fails to identify the correct feature in switching frequency dataset and mistakenly identifies other dimensions with constant slope as features causing the change. Moreover, through using a triplet loss in our approach, we can identify features whose Sinkhorn divergence is smaller for sub-sequences before the change than sub-sequences after the change. This allows it to correctly to identify the feature of interest causing the change.

Affect of number of change points on performance To better understand how many change points are needed to learn an effective metric, we conducted experiments for sequences that switch between 100 dimensional Gaussian Mixture distributions $\boldsymbol{\alpha} = \mathcal{N}(\mathbf{0}, \boldsymbol{I}) + \mathcal{N}(\mathbf{1}, \boldsymbol{\Sigma}_0)$ and $\boldsymbol{\beta} =$

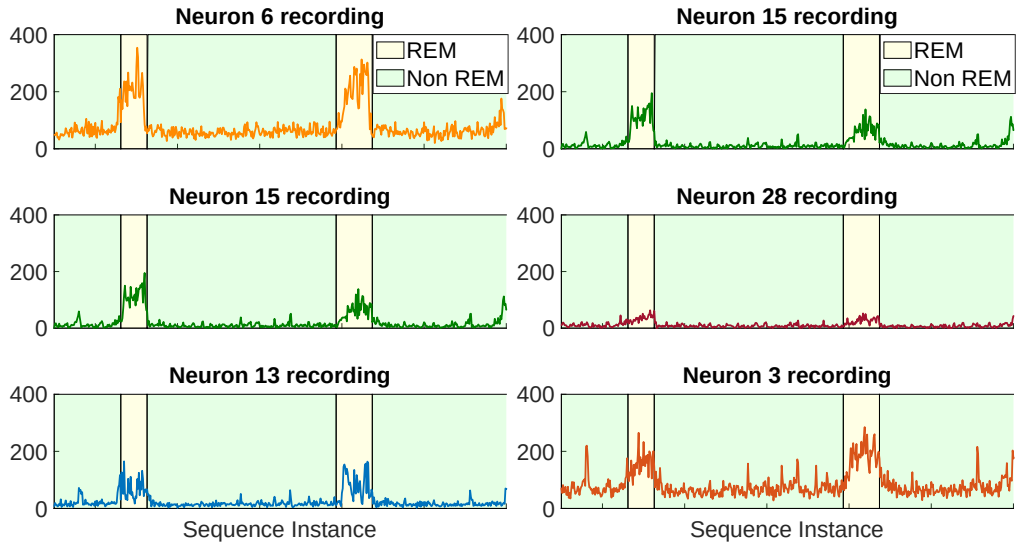
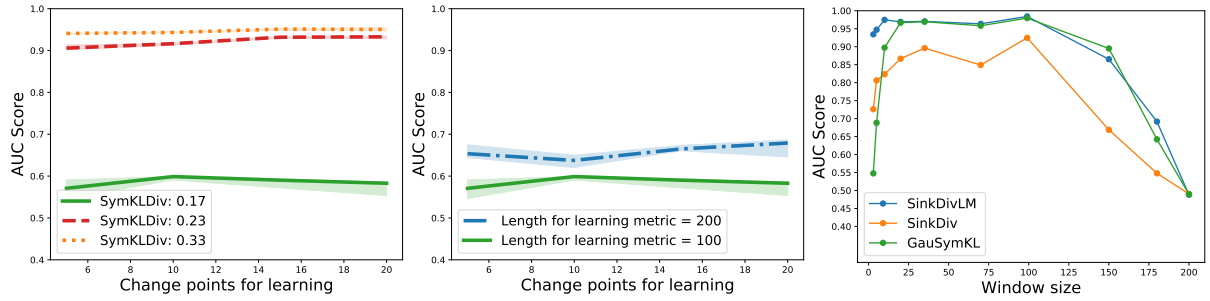


Figure 4.7: The top 3 features, or neurons, from the learned metric in Figure 4.6 are visualized in Figure 4.7a, while Figure 4.7b visualizes top 3 features identified by sHSIC.

$\mathcal{N}(\mathbf{0}, \boldsymbol{I}) + \mathcal{N}(\mu\mathbf{1}, \boldsymbol{\Sigma}_1)$, where $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Sigma}_1$ are diagonal covariance matrices where the first 3 entries on the diagonal are 3 and 5 respectively, while the rest of the diagonal entries are 1. We vary the value of μ to simulate different change detection scenarios. These experiments involved learning a metric with $\boldsymbol{L} \in R^{50\times 100}$. The sub-sequences used to learn the metric were of length 100. Figure 4.8a shows the area under the curve (AUC) scores versus the number of change points used to learn $\boldsymbol{L}$. The blue curve has $\mu = 1.1$ (a change of symmetrical KL divergence of 0.17), red has $\mu = 1.3$ (a change of symmetrical KL divergence of 0.23), and orange has $\mu = 1.5$ (a change of symmetrical KL divergence of 0.33). For all these cases, a different test-sequence were used to test and report numbers. It can be seen that the AUC performance doesn't change much as the number of change points are increased. The AUC performance depends on how difficult the change detection problem is. For sequences involving larger changes, which are associated with larger KL divergences, a small number of change points can be used to learn a metric that provides high AUC scores. Figure 4.8c shows the affect of changing the length of the sub-sequences used to learn a metric. Difficult change detection problems, such as the one involving a symmetrical KL



(a) (b) (c)

Figure 4.8: Figure 4.8a shows the area under the curve (AUC) scores versus the number of change points used to learn $\boldsymbol{L}$ for three different switching GMM sequences, where each of these sequences is 100 dimensional. The mean shift, and hence the symmetrical KL divergence, associated with these sequences is smallest for the green sequence and largest for the red sequence. Figure 4.8c shows the affect of changing the length of the sub-sequences used for learning $\boldsymbol{L}$ for the most difficult GMM switching sequence which is associated with small symmetrical KL divergence. Figure 4.8c shows the relationship between window size and AUC for the red switching GMM sequence shown in Figure 4.8a .

divergence of 0.17, are difficult to detect, and increasing the number of change points used to learn a metric doesn't improve performance. A larger sub-sequence length is instead more important for learning a metric from the provided triplet supervision. Like all learning problems, our method would require available change points to represent the types of changes expected to be encounter in testing settings. These experiments show that our method can learn an effective metric with a small number of representative change points.

Window size vs performance As our method uses a divergence test over sliding windows to compute change point statistics, the detection delay would depend on the window size. A shorter window size, while reducing detection delay, would decrease the number of samples used to estimate change statistics, leading to false change points. Figure 4.8c provides AUC vs window size plots on the 100 dimenstional switching GMM dataset. We compare our method with SinkDiv and GauSymKL. This plot shows that the AUC performance for all methods improves as the window size is increased. However, when the window size becomes large enough to consume multiple distributions, which is 100 for this dataset, the performance suffers. For these experiments, SinkDi-

Table 4.4: Top Neurons identified by SinkDivLM and HSIC along with their normalized importance values

SinkDivLM	Neuron 6	Neuron 15	Neuron 13
	1.00	0.81	0.54
HSIC	Neuron 15	Neuron 28	Neuron 3
	1.00	0.45	0.45

vLM learns an $\boldsymbol{L} \in \mathbb{R}^{5\times 100}$ which maps 100 dimensional input to a 5 dimensional space. Thus applying Sinkhorn divergences on such a low dimensional space allows our method to achieve high AUC scores with only a window size of 5. This can allow our method to significantly reduce detection delays over other change detection methods that also use sliding windows.

Interpretability A learned linear metric can also allows us to interpret how different input features contribute to detecting changes. For example, figure Figure 4.5a shows how our learned metric identifies the third dimension of the input sequence as the most important dimension for detecting changes. This helps us get a better understanding of what *kinds* of changes in sequences are of interest. For the Human activity dataset, the learned metric in Figure 4.5b shows what input features are positively correlated and what features are negatively correlated. These learned metrics increase the change statistic at true change points, leading to better change detection performance as seen in Figure 4.4. Figure 4.6 shows the sparse learned metric for the Sleep Stage dataset. This helps us identify what features, or neurons in the hippocampus, are responsible for causing changes between REM and non-REM sleep stages. Table 4.4 show the top 3 neurons identified by SinkDivLM and HSIC respectively for causing changes between REM and Non-REM sleep stages. These neurons are visualized in Figure 4.7a and Figure 4.7b.

4.5 Dicussion

In this chapter, we introduce SinkDivLM, a method that uses labeled change points to learn a metric for Sinkhorn divergences. This learned metric allows Sinkhorn divergences to perform improved change point detection, which we demonstrate on numerous real world datasets. This metric also enables interpretation by highlighting the aspects of a time series that most clearly indicate changes and allows us to learn a sparse metric, which is key in high-dimensional settings.

It can be argued that the availability of true change points could enable a scheme that switches between different distribution models for change point detection. However, it is not clear how we can identify what types of distributional models to use, especially in real world settings. These efforts would require human intervention which often is infeasible in large scale/high-dimensional datasets. Our metric learning method allows a data-driven approach to learn such a model.

While Sinkhorn divergences provide a powerful tool for distinguishing between samples, they do not naturally incorporate temporal structure. This means that Sinkhorn divergences cannot, for example, distinguish a sequence from its temporally inverted counterpart. In some change detection problems this may prove to be somewhat limiting. One potential way to address this challenge is to leverage tools from order preserving Wasserstein distances in [77, 88].However, such distances assume that sequences of the same kind have identical starting instances, a condition that is violated for two adjacent sub-sequences. Another direction could use recurrent networks to learn a metric that captures these temporal dynamics. Such a metric, however, would result in reduced interpretability.

Finally, we note that by incorporating supervision, there is always the possibility that the learned metric could become unsuitable for sequences when there is a shift in the kind of change points to be detected so that the supervision is no longer relevant to the desired task. Thus, it is important to not naïvely apply this approach when the labeled change points are not indicative of the types of changes one hopes to detect.

There are numerous ways in which we might further improve our method. For example, we currently train only using similar dissimilar pairs formed using true change points. We can increase the number of training pairs by using the learned metric to detect additional possible change points on the training sequences. Falsely detected change points could be used to obtain similar pairs which could in turn be used to increase the number of similarity triplets for retraining the metric.

We have proposed a method that uses true change information to learn a ground metric for improved change point detection. While requiring only relatively few true change points for training, we have shown through examples on simulated as well as real-world datasets that our method can improve change point detection performance. As some examples of change points are often available in many settings, we believe our proposed method has the potential to be widely applicable in the real-world.

CHAPTER 5

CHANNEL SELECTIVE UNSUPERVISED DOMAIN ADAPTATION

5.1 Introduction

Time series analysis is increasingly pivotal in diverse fields such as astronomy, climate science, neuroscience, healthcare, finance, and industrial monitoring [89–92]. However, due to a variety of factors including drift, sensor differences, and measurement limitations, there can often be significant shifts in the data between the training and testing times [93]. Traditional methods often struggle with the variability inherent in time-series data, leading to suboptimal performance and limited generalization. This challenge underscores the need for more robust and adaptable models that can effectively manage these complexities and leverage the full potential of time-series data.

Recent work has shown the promise of domain adaptation approaches for time series [93–95] to help address some of these challenges. In this setting, we combine labels on the training set with unlabeled test data to build a unified encoder even in light of significant shift across the two sets. Unfortunately, in the context of multivariate (multi-channel) time series, these models fail when presented with missing channels or when there are significant shifts in individual channels that cannot be easily corrected for when solving a joint alignment objective across all channels. This limits the effectiveness of these approaches in real-world scenarios.

To address these challenges, we introduce a novel approach for time series domain adaptation. Our approach centers on constructing a *separable alignment plan* between the labeled (source) and unlabeled (target) data, where the goal is to first *align each channel* and then *align the joint embeddings* formed after pooling across channels. To achieve a sparse and selective attention of channels when pooling, we employ a simpler variant of self-attention to select and combine channels, enabling the fusion of the channel latent representations into a comprehensive global representation.

This method not only enhances adaptability across domains but also allows for discernment in channel selection and screening, ensuring that only the most relevant and informative channels are utilized for alignment and inference.

We evaluate the performance of the model on a number of time series benchmarks and achieve state-of-the-art performance on most tasks. In benchmark tests on a human activity recognition dataset called WISDM – a fundamental application of time-series analysis – our method achieved a nearly 6% improvement over the existing state-of-the-art models. Our results and ablations not only demonstrate the effectiveness of our approach in dealing with complex, multi-channel time-series data but also highlights its potential in identifying the most informative channels across two datasets for diagnostic and interpretation purposes.

Our contributions include:

Novel method for time series domain adaptation. We develop a new method that builds a separable alignment plan that *aligns each channel* independently before pooling across channels and *aligning the fused representations*.

Channel selection and screening via a self-attention layer. Our method employs a self-attention layer for sparse and selective attention of channels. This allows for the efficient selection and combination of only the most relevant channels, leading to the formation of a robust global representation that is representative of the essential features in the data even in the context of significant shifts in some channels (see Fig. Figure 5.1).

State-of-the-art performance in time-series classification benchmarks. Our approach achieves state-of-the-art performance on a number of datasets, and achieves a 6% improvement in accuracy over existing state-of-the-art methods on the WISDM human activity dataset. This result underscores the effectiveness of our method in handling shifts in complex, multi-channel time-series data.

Interpretability. Our approach not only excels in performance but also provides insights into the most informative channels across datasets. This feature is particularly beneficial for diagnostic

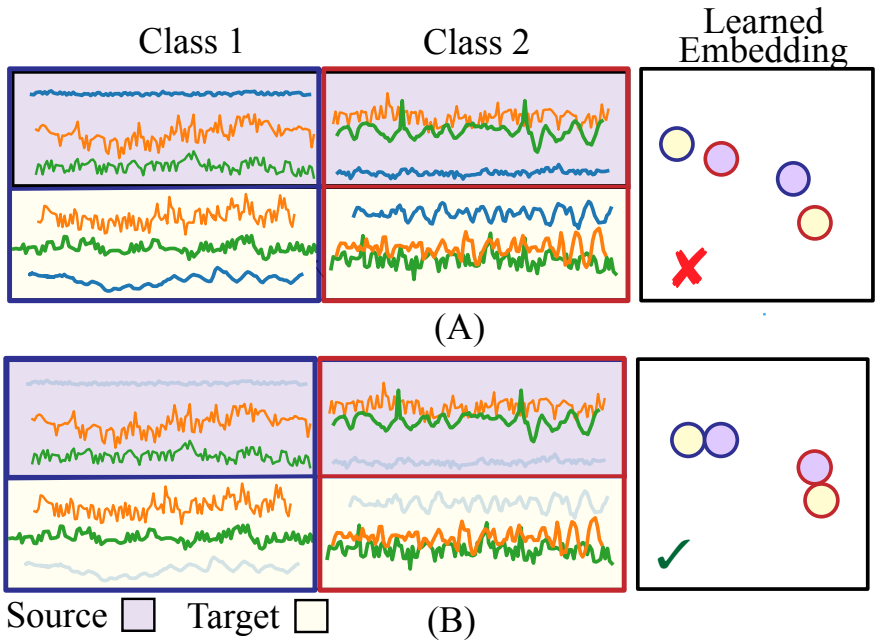


Figure 5.1: Large domain shifts in certain channels can cause incorrect classes to be aligned between the source and the target domains. (A) shows such a case where the blue channel causes class 1 in the source domain to be aligned with class 2 of the target domain (and vice versa). (B) shows how ignoring the blue channel leads to correct class alignment between domains. Our proposed method aims to address such cases by selecting and screening channels to enable more robust alignment and adaptation.

purposes, allowing practitioners to understand which channels provide the most joint information across the source and target.

5.2 Background and Related Work

5.2.1 Domain adaptation

Many real-world scenarios require adapting a model which is trained on a source labeled dataset to a *related* unlabelled target dataset. This related dataset can have a shift in either the unlabeleld data (feature shift), or the (unavailable) labels in the target domain.

Domain adaptation methods try to improve prediction performance on unlabelled target domain data by leveraging source domain labelled data. Most methods addressing feature shift, which is the domain shift we address in this chapter, make the assumption that the shifted class conditioned

data in the target domain is closer to the corresponding class conditioned source data in the representation space. This means that the source domain representations for a class should be closer to the target domain representations of the same class than the target representations of other classes [96–98]. When this assumption is met, making the source and target domain representations *invariant* while simultaneously minimizing the source classification loss can help models adapt to target domains.

These source and target representations can be made invariant through adversarial learning [99, 100], or minimizing distances such as the maximum mean discrepancy or Wasserstein distance [101–103]. Other methods take an alternate approach where source domain labels are used to generate pseudo labels in the target domain, which are then used to train a model to classify unlabelled target domain data [104]. Related methods have incorporated augmentations with contrastive learning on both source and domain representations to better adapt models to the target domain [105].

5.2.2 Domain adaptation techniques in time series

Time series domain adaptation methods have mostly adopted methods from vision while utilizing encoders more suited to time-series data such as RNNs and 1D temporal CNNs. Many methods utilize adversarial learning [14], or use kernels more suited to time-series data to align source and target representations [95, 106]. Recent methods have also additionally incorporated pseudo-labelling and self-supervised contrastive learning through augmentations [107, 108]. As frequency domain information can be in some cases useful for time series classification tasks, newer methods have also incorporated source and target domain frequency representations while learning domain invariant representations [93]. All of these methods pass all time series channels collectively into a common neural network encoder. There has been no work, to the best of our knowledge, that develops different representations and domain adaptation schemes for each channel in multivariate time series.

5.2.3 Attention mechanisms in time-series analysis

Many machine learning models have incorporated attention mechanism for improved classification and forecasting. Some of these methods process different time series channels independently by passsing through neural encoders separately. [109–111]. However, most existing literature focuses on channel-selection in the context of single-domain data, leaving a gap for methods that apply these principles effectively in domain adaptation scenarios.

5.2.4 Optimal transport and Sinkhorn divergences

To build aligned representations, we use the Sinkhorn divergence, a robust measure of distributional similarity. The Sinkhorn divergence is an entropic regularized variant of Wasserstein distances. The entropic regularization, with regularization parameter γ, allows a computationally efficient transport plan solution to be obtained through Sinkhorn iterations [15]. This divergence can then be used to measure how similar two different samples sets are, and thus can be used as a loss function to make two sample sets similar [112].

More formally, the Sinkhorn distance between two distributions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is defined as

$$\mathcal{S}_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \min_{\boldsymbol{P}} \langle \boldsymbol{C}, \boldsymbol{P} \rangle - \gamma \boldsymbol{H}(\boldsymbol{P}),$$
$$\text{s.t } \boldsymbol{P} \in \mathbb{R}_+^{n \times m}, \boldsymbol{P}^T \mathbb{1}_n = \boldsymbol{b}, \boldsymbol{P} \mathbb{1}_m = \boldsymbol{a},$$

where $\boldsymbol{P}$ is called the transport plan, $\boldsymbol{H}(\boldsymbol{P})$ is the entropy of the transport plan matrix $\boldsymbol{P}$ and is given by $\boldsymbol{H}(\boldsymbol{P}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \boldsymbol{P}_{i.j}(\log \boldsymbol{P}_{i,j} - 1)$, while γ is a regularization parameter. This regularization terms makes the minimization problem strongly convex and makes it less sensitive to changes in input, and can be solved with $O(n^2)$ computations using the Sinkhorn algorithm [15].

5.3 Proposed Method

5.3.1 Overview

Different channels in multivariate time series can capture different information about the phenomenon of interest. In many cases, the end classification or regression task of interest might often depend on the information contained in a handful of channels. Additionally, individual channels in a time series might shift in very different manners across domains. Figure Figure 5.1 shows a real world example in human activity recognition where class 1 (sitting – blue) and class 2 (standing – red) need to be adapted from the source domain shown in purple to target domain in yellow. In Fig. Figure 5.1(A), a large domain shift in the blue channel can cause the representations of the source and target classes to be misaligned which can result in disastrous performance on the target domain. If this blue channel is simply ignored as in Fig. Figure 5.1B, the target representations are much likely to align with their respective classes in the source domain, resulting in greatly improved performance in the target domain.

This example emphasizes the need to develop domain adaptation method that account for shifts in each channel differently while also having the ability to screen channels involving larger domain shifts.

5.3.2 Step 1: Align individual channel representations

An input time series $\boldsymbol{X} \in \mathbb{R}^{C \times T}$ with length T, and C channels is first split into C different, one dimensional, time series $\boldsymbol{x}^c \in \mathbb{R}^T$, where the superscript c represents the c$^{\text{th}}$ channel. Each of these one dimensional time series is then fed into a channel specific encoder f_θ^c, to obtain channel specific representations $\boldsymbol{z}^c \in \mathbb{R}^d$. These encoders are used to encode both source domain time series $\boldsymbol{X}_s$ and target domain time series$\boldsymbol{X}_t$ into $\boldsymbol{z}_s^c$ and $\boldsymbol{z}_t^c$ for all channels $c \in C$. The source domain representations for the channels, $\boldsymbol{z}_s^C$, are then linearly transformed by $\boldsymbol{W}_c \in \mathbb{R}^{d \times M}$ for all $c \in C$, where M is the number of classes. A softmax function is then applied on this linear

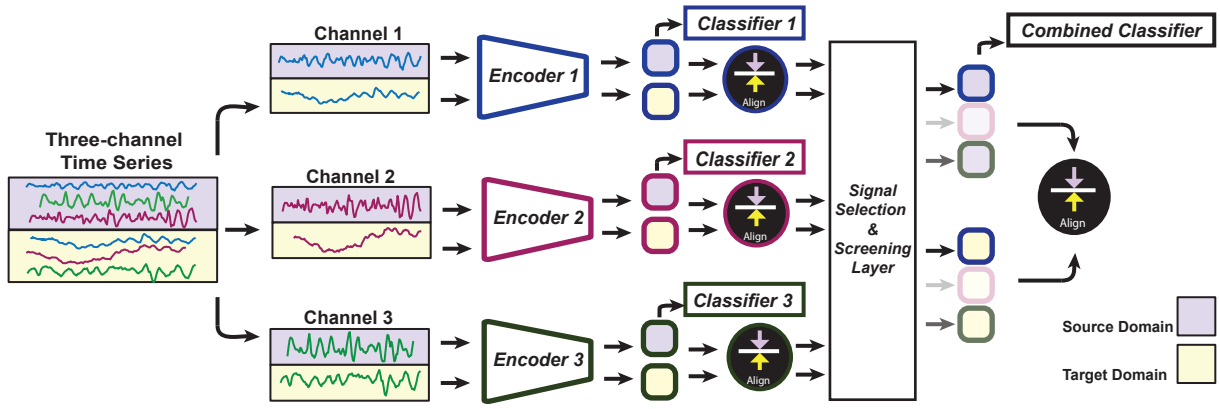


Figure 5.2: Overview of our proposed approach. For both the source and the target domain time series, each of the input channels is fed into channel specific encoders. The source and target domain outputs of each of these encoders is domain adapted by minimizing source classification and alignment loss. All of the domain adapted channel representations are then provided to a channel weighting layer, which reweights source and target channel representations. These weighted source and target representations are combined, and then again domain adapted using source classification and alignment loss.

transformation to produce the class prediction vector $\hat{\boldsymbol{y}}_s^c$.

The target representations for all channels can then be aligned with their respective source domain representations by minimizing the Sinkhorn distance [15] between the individual source and target domain channel representations while minimizing the source classification loss. Concretely speaking, this can be done by solving the following optimization problem for all channels:

$$\min_{\theta_c, \boldsymbol{W}_c} S_\gamma(f_\theta^c(\boldsymbol{x}_s^c), f_\theta^c(\boldsymbol{x}_s^c)) + \mathcal{L}_{\text{CE}}(f_\theta^c(\boldsymbol{x}_s^c)\boldsymbol{W}^c, \boldsymbol{y}_s),$$

where f_θ^c and $\boldsymbol{W}^c$ are the encoder and linear classifier for channel c whereas $\boldsymbol{x}_s^c, \boldsymbol{x}_t^c$ are the source and target inputs for channel c. This can be expressed more succinctly using the representations $\boldsymbol{z}$ described previously:

$$\min_{\theta_c, \boldsymbol{W}_c} S_\gamma(\boldsymbol{z}_s^c, \boldsymbol{z}_t^c) + \mathcal{L}_{\text{CE}}(\hat{\boldsymbol{y}}_s, \boldsymbol{y}_s) \; \forall c \in C$$

where S_γ is the Sinkhorn distance with regularization parameter γ and $\mathcal{L}_{\text{CE}}$ is the cross entropy loss.

5.3.3 Step 2: Compute a global multi-channel alignment

When combining representations for individual channels, a weighting vector $\boldsymbol{w} \in \mathbb{R}^C$ is computed to ascertain the importance of each channel. This can help filter unimportant/noisy channels while selecting channels that are (a) important for classifying source data and (b) are invariant between the source and target domains.

We obtain the weighting value for the c[th] channel in this vector $\boldsymbol{w}$, w^c, by linearly transforming all individual channel representations $\boldsymbol{z}^c$ into $\boldsymbol{k}^c$ and $\boldsymbol{q}^c$ using linear transformations $\boldsymbol{K}$ and $\boldsymbol{Q} \in \mathbb{R}^{d \times d}$. The inner product between these linear transformations,$(\boldsymbol{q}^c)^\intercal \boldsymbol{k}^c$, is then computed for each channel after which a softmax non-linear transformation is applied across the vector consisting of these inner products for all channels to obtain channel weighting scores. Specifically, the weighting vector $\boldsymbol{w}$ can be obtained by:

$$\boldsymbol{w} = \text{softmax}\left(\frac{1}{\tau}\left[\frac{1}{\sqrt{d}}\left((\boldsymbol{q}^1)^\intercal \boldsymbol{k}^1, \ldots, (\boldsymbol{q}^C)^\intercal \boldsymbol{k}^C\right)\right]\right)$$

where τ is the softmax scaling factor. A larger value of τ leads to a weighting vector that is uniform in its value, leading to the selection of more channels, while a smaller value of τ encourages a sparser weighting vector. Additionally, the computed inner products are also normalized by the embedding dimension $\sqrt{d}$ in a manner similar to the way it is done for self-attention. The representations for all individual channels are concatenated to form the matrix $\boldsymbol{Z} \in \mathbb{R}^{C \times d}$. A Hadamard product between the weighting vector $\boldsymbol{w}$ and the concatenated representations matrix $\boldsymbol{Z}$ is then computed and vectorized into a vector to obtained combined channel representations $\boldsymbol{z}^a \in \mathbb{R}^{dC}$:

$$\boldsymbol{z}^a = \text{vec}\left(\boldsymbol{w} \odot \boldsymbol{Z}\right) = \text{vec}\left(\begin{pmatrix} w^1\boldsymbol{z}^1 \\ \vdots \\ w^C\boldsymbol{z}^C \end{pmatrix}\right).$$

We can use this selection and screening operation to obtain collective channel representations for both the source ($\boldsymbol{z}_s^a$) and target ($\boldsymbol{z}_t^a$) time series. The source representations can then be transformed by a linear classifier $\boldsymbol{W}_s^a$ followed by a softmax non-linear transformation for source class prediction.

Relation to self-attention. This channel selection operation can also be expressed as the output of a modified self attention layer

$$\boldsymbol{w} = \text{softmax}\left(\text{diag}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\intercal}{\sqrt{d}}\right)\right)\mathbf{I},$$

where the *values* matrix is replaced by the identity, only the diagonal terms of the attention matrix, which is computed by the inner product of the *keys* and *queries* matrices, are kept and the softmax is computed along this diagonal.

5.3.4 Putting it all together

Our overall alignment and classification loss is a function of all channel encoders (f_θ^c) and channel classifiers ($\boldsymbol{W}^c$):

$$\mathcal{L} = \underbrace{S_\gamma(\boldsymbol{z}_s^a, \boldsymbol{z}_t^a) + \mathcal{L}(\hat{\boldsymbol{y}}_s^a, \boldsymbol{y})}_{\text{Dom. adapt. for combined reps)}} + \sum_{c=1}^{C} \underbrace{S_\gamma(\boldsymbol{z}_s^c, \boldsymbol{z}_t^c) + \mathcal{L}_\rfloor(\hat{\boldsymbol{y}}_s^c, \boldsymbol{y}_s)}_{\text{Dom. adapt. for each channel}},$$

Figure 5.2 shows the overall model for our proposed method. Time series are first split into different channels, and each of the split channels is fed to a channel specific encoder and a channel specific classifier (the border color of the encoder and the classifiers is used to indicate how these networks serve as channel specific modules). The source and target representations for each channel are aligned through Sinkhorn distances, which are shown in black, while simultaneously minimizing the source classification loss for the representations of these channels. All channel representations are then used to obtain channel weighting scores which are used to reweight chan-

nel representations. These reweighted channel representations for the source and target domain are concatenated and then aligned across domains in a similar manner as before.

5.3.5 Intuition behind our method

Different time series channels can carry more diverse information than channels within other modalities such as images. As a result, many real-world time series classification problems can often heavily depend on a limited number of channels.

Learning separate classifiers for each channel, which is required as part of our method in Subsection 5.3.2, leads to individual channel representations that try to maximize the mutual information between each channel input and source label data. The weights produced by the signal selection and screening layer can select the most informative of these representations across all channels for the final classification output of the model. These weights can lead to sparse selection of individual channel representations for supervised classification, helping ignore uninformative channels.

At the same time, a supervised loss on the source domain is combined with a loss that minimizes the Sinkhorn distance between source and target data for each channel, the signal selection and screening layer now produces weights that still aim to be informative in terms of classification on the source domain, but also leads to better overall alignment between the source and target domains. Thus, in addition to helping to ignore channels that are uninformative for classification in the source domain, this also helps de-emphasize channels that do not align well between the source and target domains. As we will see, this can ultimately significantly improve domain adaptation performance.

Table 5.1: Mean accuracy and macro F1 scores for different domain adaptation methods

Method	Simulation		UCIHAR		HHAR		PXECG		WISDM		WISDM-Bal	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Sup	43.12	0.423	77.04	0.750	59.40	0.543	63.51	0.366	64.90	0.504	65.84	0.521
DANN	71.32	0.70	82.91	0.857	71.27	0.678	62.87	0.347	67.94	0.567	73.86	0.683
AdvSKM	74.31	0.712	85.12	0.813	63.25	0.616	62.98	0.372	69.92	0.581	71.19	0.611
CoDATS	54.31	0.531	86.34	0.856	68.79	0.686	66.30	0.366	68.35	0.548	75.15	0.665
CDAN	79.54	0.81	84.59	0.836	70.06	0.704	64.29	0.375	70.12	0.517	70.29	0.661
DeepCoral	82.34	0.841	86.53	0.851	66.16	0.690	62.60	0.346	72.72	0.605	74.31	0.649
CLUDA	78.21	0.802	82.45	0.854	67.03	0.641	64.92	0.324	65.57	0.504	73.77	0.699
SinkDiv	73.11	0.713	85.13	0.876	69.64	0.720	64.97	0.376	67.16	0.578	70.98	0.648
Raincoat	73.11	0.713	89.13	0.873	62.11	0.603	66.22	0.357	62.11	0.523	69.09	0.727
SSSS-TSA	**99.01**	**0.985**	**90.12**	**0.901**	**72.19**	**0.737**	**66.38**	**0.419**	**75.19**	**0.635**	**83.57**	**0.816**

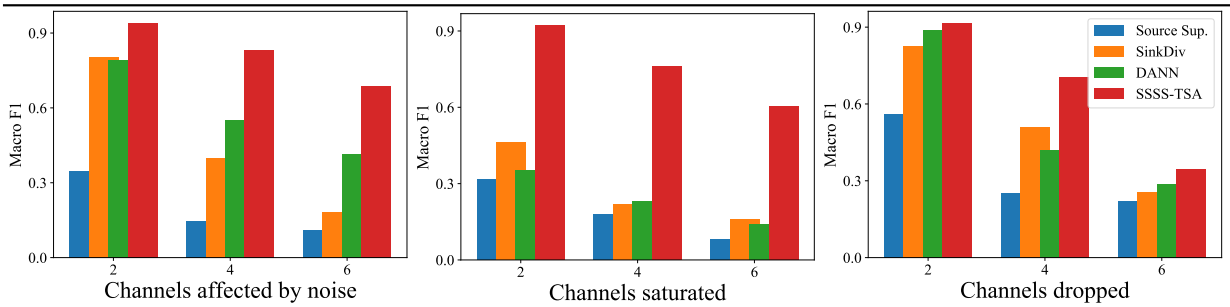


Figure 5.3: Domain adaption performance when target domain in HHAR dataset is further shifted through corrupted channels. From left to right, we compare our method to baselines in an (a) additive Gaussian noise setting, (b) saturated channels setting, and (c) when channels are dropped.

5.4 Experiments and Results

5.4.1 Experimental setup

Simulated mean shift data. We first consider simulated data consisting of 4 dimensional sequences of length 128. Source domain data consists of Gaussian i.i.d. data with variance 1, and the means of these channels shifts between 4 classes. The target domain data is generated by randomly selecting and shifting one channel mean for each class.

Real world datasets. We consider several real world multivariate time series datasets that are often used to benchmark domain adaptation methods [94].

UCI Human Activity Recognition (UCIHAR). Data was collected from a group of 30 volunteers

performing six different activities (walking, walking upstairs, walking downstairs, sitting, standing, and laying). Each participant was equipped a smartphone with embedded accelerometer and gyroscope that provides 9 channel data (3 axis body acceleration, 3 axis angular velocity, and 3 axis total acceleration).

Heterogeneity Human Activity Recognition (HHAR). Data consists of different users carrying various types of smartphones and smartwatches while performing common activities like walking, sitting, standing, etc. This dataset poses a significant challenge in domain adaptation due to the variability in sensor outputs across different devices. This data consists of 3 axis accelerometer data.

PXECG. This is a 12 channel ECG dataset with 5 diagnostic classes. The data is collected from 5 different sites, each of which constitutes a different domain [113].

Wireless Sensor Data Mining (WISDM). Involves time-series data collected from wireless sensors embedded in smartphones. The data typically includes activities like jogging, walking, ascending and descending stairs, sitting, and standing. WISDM can be a highly imbalanced dataset across subjects, which makes it particularly challenging for domain adaptation.

WISDM-Bal. We also take the WISDM dataset and balance classes across source and target domains to better analyze the performance of our proposed method. We do this to ignore issues in domain adaptation that arise because of imbalanced datasets, an issue that we do not aim to address in this work.

As the possible number of domain adaptation scenarios on WISDM, HHAR, and UCIHAR can be as large as $\binom{30}{2}$, we select 10 domain adaptation scenarios for our experiments. More details on the chosen domain adaptation scenarios and their corresponding results can be found in the Appendix.

Baselines. The models compared in these experiments include: supervised learning on the source domain (no domain adaptation), Domain-Adversarial Neural Networks (DANN) [99], Adversarial Spectral Kernel Matching (AdvSKM) [95], CoDATS [14], Conditional Domain Adver-

sarial Networks (CDAN) [100], DeepCoral, CLUDA [108], Sinkhorn Divergence (SinkDiv) , and Raincoat [93], alongside our proposed method, SSSS-TSA. The Sinkhorn divergence baseline is the Raincoat baseline without the frequency domain information. It aligns the Sinkhorn distance between source and target representations while minimizing the source classification loss. We select these baselines as these are the methods considered in more recent domain adaptation papers [93, 108]. The Adatime benchmarking suite is adapted and used to run these baselines [94].

Experiments and evaluation details. We use a 1D CNN neural network as an encoder for all of our baselines. Most of these datasets provide standardized splits to train models and test splits to report numbers. As unlabelled target domain data is not available in real world domain adaptation settings, there is some uncertainty in the community regarding the best way to evaluate domain adaptation methods. We run all methods for a fixed number of epochs and report numbers at the end of these. While less common, this scheme has been used by other papers (e.g., [93]) and most accurately depicts real-world domain adaptation settings. We also report test-set numbers when models attain their best performance on a validation holdout from the training data in the appendix (as that is a popular evaluation criteria in literature). We report both macro F1 and accuracy scores for better evaluation across datasets with varying class imbalances. Each model is run five times on each dataset to ensure statistical reliability, and the results are averaged to produce the mean accuracy and macro F1 scores.

5.4.2 Results on time-series classification benchmarks

The results, as shown in Table Table 5.1, demonstrate the superior performance of SSSS-TSA across a range of datasets. We first note that it is interesting to see how poor the performance of most baselines is on the simulated dataset. Our method's performance of 0.98 F1 score underscores how alignment of *channels* can lead to significant improvements over approaches that only align fused or global representations. Our method achieves the highest accuracy and macro F1 score on

every dataset considered. It often significantly outperforms the second best algorithm, and there is no alternative algorithm that is consistently close to the performance of our approach.

These results underscore the effectiveness of our approach in domain adaptation for time-series data. Our method consistently outperform other state-of-the-art models in various complex and real-world scenarios. The significant improvement in mean macro F1 scores across these datasets highlights the robustness and adaptability of our approach, particularly in handling the challenges posed by multi-channel and noisy time-series data. The Appendix contains additional results when the target labels in validation sets are used to determine the stopping time. While this is not a realistic metric in a true domain adaptation scenario (as it allows labels in the target domain to influence the training process), we find that using this evaluation criteria our method continues to be either competitive or superior to baselines.

5.4.3 Testing on corrupted data

To further test the capability of our method to handle different shifts, we create additional domain shifts with in the 9 channel HAR dataset. For the domain adaptation setting where the source domain is subject 2 and the target domain is subject 11, a scenario in which most domain adaptations do well, we enforce three types of additional shifts in the target domain: 1) additive Gaussian noise, (2) saturating channels, (3) dropping channels. For all these three cases, we can increase the number of channels affected by these shifts to see how robust domain adaptation methods are to these channel corruptions.

For the first setting, we add Gaussian noise of mean 0 and variance 2 to randomly selected channels in the target domain. To simulate channel saturation, we set randomly selected channels to a large value (we set these channels to 2, which is a larger value for these prepossessed normalized datasets). Similarly, we drop channels by setting randomly selected channels to 0. All these corruptions depict domain shifts likely to be experienced in real world settings.

We repeat our experiment five times, each time using a different random set of selected chan-

Figure 5.4: The top row shows channel weights learned by our model on the HHAR dataset for the corresponding input data below. The overall weight distribution across the two domains is mostly similar. The colored boxes highlight how weights learned for different classes help mask channels with larger shifts between the domains, contributing to improved domain adaptation performance.

nels. The average macro F1 scores are shown in Figure 5.3. We can see that SSSS-TSA consistently outperforms baselines such as DANN and SinkDiv. We selected these baselines as most other baselines are variants of these. We observe that our baselines exhibit significant performance degradation when only 2 channels are saturated. Our method achieves a macro F1 score of 0.922 while the best next baseline, SinkDiv Alignment, falls to 0.46. Even when 6 out of the 9 channels are saturated, SSSS-TSA still attains an F1 score of 0.60. The third figure in Figure 5.3 shows the results of the setting where randomly selected channels are dropped. Though our method performs better than other baselines, the margin is much smaller for SSSS-TSA when six channels are dropped. As this dataset was preprocessed and normalized, many channels in the source domain would have values closer to 0, which would make it likely for some of these channels to be aligned with dropped channels. Our method performs much better in the additive Gaussian setting as the corrupted channels are less similar to the source channels. Overall, these results show the our method often has the ability to screen and select important unaffected channels to improve time

series domain adaptation performance.

5.4.4 Ablations

How important is it to align independent channels?

In our first ablation study, we evaluate the importance of aligning independent channels in our method. Our approach, with independent channel alignment, achieves an F1 score of 0.816 on the balanced WISDM dataset. When this alignment was removed (denoted as W/O Ind Align), the performance significantly decreased to 0.639. This drop in performance supports our claim in Subsection 5.3.5 on the importance of obtaining informative channel representations.

How important is the selection and screening module?

The second ablation focuses on the impact of the selection and screening module, specifically our implementation of the attention mechanism. With the full model, including the attention mechanism, the F1 score stood at 0.816. However, when the attention mechanism was removed (W/O Attn), the score dropped to 0.688. This reduction highlights the significance of the selection and screening module in our method. The attention mechanism enables the model to focus on the most relevant and informative channels, thereby improving the both the quality of the merged representation for classification in the source domain as well as the relevance of this representation in the target domain, and consequently, improving the overall performance of the model. This result demonstrates that the selective attention to channels is not just beneficial but is a crucial aspect of achieving high accuracy in domain adaptation for time-series data.

Table 5.2: Ablations on WISDM-Bal

Our Method	W/O Ind Align	W/O Attn
0.816	0.639	0.688

5.4.5 Visualizing the learned weights

Figure 5.4 provides a domain shift example from the HHAR dataset. The matrices on the top show the channel weights learned by our model for both source and target domain data which is shown below. Note that the overall distribution of the weight matrices across the classes is mostly similar between the source and the target domains. We use source and domain data from three classes to illustrate how these learned weights help improve domain adaptation.

The source and target weights for class 1, bounded in red boxes, select channel 3 as an important channel for the domain adaptation task. We can see in the input data for class 1 that these chosen weights indeed help ignore the blue and the orange channels which encounter major shifts between the source and the target domains. The selected green channel is most similar between the two domains. We observe similar phenomena for classes 2 and 5. While not depicted in the figure, it is also noteworthy that the weights learned for class 4 are quite different in the source and target domains. Upon detailed inspection, each channel is quite informative, and in this case the precise selection of channels is not particularly important for ensuring correct classification.

5.4.6 Visualizing the latent representations

Finally, we also examine the latent representations learned by our models and compare them to a standard Sinkhorn alignment across all channels in Fig. Figure 5.5. In this example, we can see that the representations formed by our method provide good overall alignment across all the classes globally and also gives a good local alignment of each class. In contrast, for the Sinkhorn baseline we see that some classes are often fractured and can be mapped to different parts of the latent space.

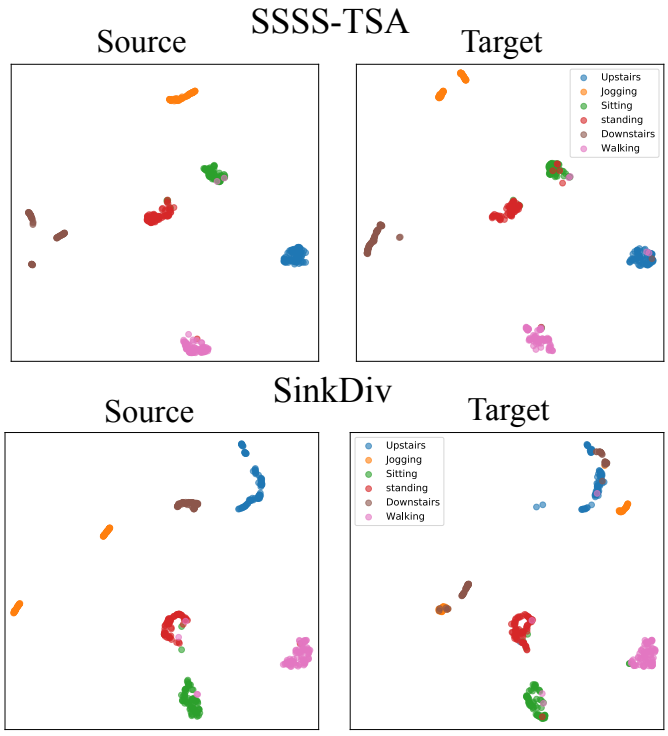


Figure 5.5: Umap embeddings for domain adaptation from subject 0 to subject 2 on the HHAR dataset, where SSSS-TSA achieves an F1 score of 0.94 as compared to 0.69 for the Sinkhorn baseline.

5.5 Conclusion

A key component of our method is its contrast with traditional encoders that indiscriminately process all channels jointly. Such encoders often fail to exploit the inherent structure and importance of different channels in time-series data, potentially leading to suboptimal performance, especially in the presence of irrelevant or noisy channels. Our approach avoids this pitfall by focusing on the most relevant channels, ensuring that the model is not only more efficient but also more effective in capturing the nuanced relationships within the data. At the same time, our work provides new insights into the significance of channel invariance in domain adaptation in time-series. This adds to a growing body of work in time-series transformers and other channel-invariant encoders in time series classification and forecasting. Here, we show how channel invariance can be a powerful tool for alignment, especially in noisy channel settings and across different subsets of channels in the training and testing sets.

Of course our approach is not without limitations. One such limitation is the potential for overfitting in scenarios with extremely noisy or sparse datasets, where channel selection might become biased towards non-representative features. Additionally, our current model uses different encoders for each channel which restricts the application to new or unknown channels. Nevertheless, our method has shown the advantages of selectively screening and aligning channels representations. This can likely be used in conjunction with other recent methods that explore augmentations [108] and label correction [93] for further improvements in the future.

CHAPTER 6
CONCLUSION AND FUTURE DIRECTIONS

6.1 Conclusion

Different chapters in this thesis propose ways through which machine learning can help improve change-point detection and ways in which machine learning can improve change-point detection.

In Chapter 2, we proposed a new method to improve sequential change-point detection for detecting multiple change-points. In Chapter 3, we showed how similar dissimilar pairs can be generated from detecting unsupervised change-points. These similar dissimilar pairs provide weak supervision which can be used in conjunction with available supervised labels for semi-supervised sequence classification. This method showed how tools from change-point detection can help machine learning. In Chapter 4, we saw how similar dissimilar could be generated from provided examples of change-points. These similar dissimilar pairs can be used to learn a distance metric which can be leverage by entropic regularized Wasserstein distances for detecting change points over sliding windows. This shows how tools from machine learning can help improve change-point detection. Finally, in Chapter 5, we proposed a method that could allow time series machine learning models to better adapt to distributional changes. Our method improves over existing methods for unsupervised time series domain adaptation by ignoring channels for which there is a large shift between the source and the target domains. This allows our method to ignore channels with much larger shifts, helping our method to perform better.

The methods proposed in this thesis provide fertile grounds for future work.

6.2 Future directions

6.2.1 Change-points for self-supervised learning

In Chapter 2, we saw how change-points can help semi-supervised classification of time series data. There is also space for new methods that explore how change-point detection can improve self-supervised learning for temporal data. Self-supervised learning methods for time series data try to learn effective neural network representations through contrastive learning tasks [13]. These methods take input sample sub-sequences and obtain *similar* views to these samples through either augmenting the original input sub-sequences, or choosing sub-sequences that are temporally proximal to the inputs. Temporally distant input sub-sequences are chosen as *dissimilar* views. Contrastive learning tries to learn representations that are *closer* for similar pairs than dissimilar pairs.

A critical challenge for self-supervised learning methods arises when dissimilar pairs are drawn from the same underlying distribution. These methods can also falter when temporally proximal sub-sequences chosen as similar views are in fact generated by different underlying distributions. Change-point detection can help avoid these pitfalls; temporally proximal sub-sequences should only be chosen as a similar sub-sequence only if there is no change-point between the original sub-sequence and a candidate similar sub-sequence. Hypothesis tests and statistical distances are at the heart of change-point detection methods. These tools can also be used to guide augmented similar pairs; augmented sub-sequences should be chosen such that they are not statistically similar to adjacent sub-sequences that are generated from different distributions. Such sub-sequences can be identified through change-points.

6.2.2 Learning multiple metrics for change-point detection

In Chapter 3, we showed how metric learning can be used for improved change-point detection. We also showed how we could learn sparse metrics which can help select and identify what chan-

nels are important for detecting change-points. There can be cases where different types of metrics are required for detecting different types of change-points. For example, when we used our metric learning method for detecting switches between multiple types of sleep stages, we observed that learning separate metrics for REM-nonREM changes and REM-awake changes led to better performance [5].

This shows that there is space to explore how different metrics can be combined for effectively detecting change-points. One potential scheme for this could involve learning to switch between different metrics. Another potential approach could be to learn an ensemble of metrics and change-point detection schemes, out of which the most confidently detected change-points could be kept.

6.2.3 Active learning for time series classification

Active learning is a popular strategy for reducing the amount of labeled data needed to train machine learning models. Recent active learning methods for deep learning select uncertain and diverse set of samples for labeling [114, 115]. However, there is little work that explores how active learning methods need to cater for time series data. Deep learning methods often divide time series into smaller sub-sequences, and each of these sub-sequences is arranged into batches. These batches are then fed into deep models to obtain a set of representations. As time series data is temporally related, sample representation within an input batch are often correlated and periodic. There is room to exploit this correlation to design active learning methods catered to time series data. We have presented preliminary work in which we devise a strategy to select frequently encountered periodic uncertain samples in time series data [116]. Though these preliminary active learning methods are based on intuitive design principles. There is a need to devise principled strategies for active learning that are grounded in theory.

Change-point detection can also be potentially helpful in devising labeling strategies. Change-points can help identify homogeneous data segments that can help guide diverse sample selection active learning methods. Homogeneous data segments also help provide class representative sam-

ples which could further help with label efficiency.

6.2.4 Continual learning

Most machine-learning methods suffer from catastrophic forgetting when they are required to learn new tasks. For example, when a model, that been trained on classifying cats from dogs, is required to learn a new task of classifying cats from sharks, its performance on previously learned tasks could suffer. Recently, the machine learning community has been focusing on continual learning methods which can continue to learn new tasks without impacting performance on previously learned tasks [117]. Most continual learning methods involve learning tasks that evolve and change with time. These methods often store representative samples from previous learning in a memory bank that could be revisited when learning a new task [118]. Other methods constrain neural networks to learn task-specific parameters that lie in different orthogonal subspaces [119]. However most continual learning methods have predefined knowledge about time instances at which tasks change. In real-world settings, this information isn't available and change-point detection methods can be used to identify when learning tasks change [120, 121].

Change-point detection can provide additional information for learning tasks. There is room to explore how change-point detection can identify how distributions, and consequently learning tasks change between tasks. This can help better decide whether samples from new learning tasks are related to previous learning tasks, helping develop continual learning methods that are more memory efficient. Change-point detection can also help guide what parts of a network to update when learning tasks change.

Appendices

APPENDIX A

SUPPLEMENTARY MATERIAL FOR CHAPTER 2

A.1 Proof of Lemma 1

A.1.1 Computing the inner expectation

For the expression in (2.20), the inner expectation is first simplified

$$\begin{aligned}\mathbb{E}_{x_t\sim\theta_0}\left[r\left(x_t\right)\middle|\hat{\mu}_t,\hat{\sigma}_t\right] &= \mathbb{E}_{x_t\sim\theta_0}\left[\exp\left(\delta_0\left(-\frac{(x_t-\hat{\mu})^2}{2\hat{\sigma}^2}+\frac{(x_t-\mu_0)^2}{2\sigma_0^2}\right)\right)\middle|\hat{\mu}_t,\hat{\sigma}_t\right]\\ &= \frac{\sigma_{12}}{\sigma_0}\exp\left(\frac{-(\mu_0-\hat{\mu}_t)^2}{2\frac{\hat{\sigma}_t^2}{\delta_0}+2\frac{\sigma_0^2}{1-\delta_0}}\right),\end{aligned} \tag{A.1}$$

where

$$\sigma_{12}^2=\frac{\hat{\sigma}_t^2\sigma_0^2}{\delta_0\sigma_0^2+(1-\delta_0)\hat{\sigma}_t^2}. \tag{A.2}$$

Note: δ_0 should be such that $\sigma_{12}^2>0$ in (A.2).

A.1.2 Computing the outer expectation

Plugging in the results of the inner expectation from (A.1) in (2.20):

$$\mathbb{E}_{\theta_0}\left[\exp(\delta_0\tilde{s}_t)\right]=\exp(\delta_0(-\frac{1}{2}-v))\mathbb{E}_{x_{t+1..w}\sim\theta_0}\left[\exp\left(\delta_0\left(\frac{\sigma_0^2+(\mu_0-\hat{\mu}_t)^2}{2\hat{\sigma}_t^2}\right)\right)\frac{\sigma_{12}}{\sigma_0}\exp\left(\frac{-(\mu_0-\hat{\mu}_t)^2}{2\frac{\hat{\sigma}_t^2}{\delta_0}+2\frac{\sigma_0^2}{1-\delta_0}}\right)\right].$$

Expressing and simplifying the above equation yields:

$$\mathbb{E}_{\theta_0}\left[\exp(\delta_0\tilde{s}_t)\right] = \exp(\delta_0(-\frac{1}{2}-v))\mathbb{E}_{x_{t+1..w}\sim\theta_0}\left[\exp\left(\delta_0\left(\frac{\sigma_0^2+(\mu_0-\hat{\mu}_t)^2}{2\hat{\sigma}_t^2}\right)\right)\frac{\sigma_{12}}{\sigma_0}\exp\left(\frac{-(\mu_0-\hat{\mu}_t)^2}{2\frac{\hat{\sigma}_t^2}{\delta_0}+2\frac{\sigma_0^2}{1-\delta_0}}\right)\right]$$
$$= \exp(\delta_0(-\frac{1}{2}-v))\mathbb{E}_{x_{t+1..w}\sim\theta_0}\left[\exp\left(\frac{1}{2}\log\frac{\sigma_{12}^2}{\sigma_0^2}+\delta_0\left(\frac{\sigma_0^2+(\mu_0-\hat{\mu}_t)^2}{2\hat{\sigma}_t^2}\right)-\frac{(1-\delta_0)(\delta_0)(\mu_0-\hat{\mu}_t)^2}{2(1-\delta_0)\hat{\sigma}_t^2+2\delta_0\sigma_0^2}\right)\right]$$
$$= \exp(\delta_0(-\frac{1}{2}-v))\mathbb{E}_{x_{t+1..w}\sim\theta_0}\left[\exp\left(g(\hat{\mu}_t,\hat{\sigma}_t^2)\right)\right]. \tag{A.3}$$

A.1.3 Asymptotic distribution of $g(\hat{\mu}_t, \hat{\sigma}_t)$

Now the asymptotic distribution for the argument of the exponent ($g(\hat{\mu}_t, \hat{\sigma}_t)$) within the expectation would be found (when sample mean and sample variance are estimated under the pre-change distribution). This argument is defined as:

$$g(\hat{\mu}_t,\hat{\sigma}_t) = \underbrace{\frac{1}{2}\log\left(\frac{\hat{\sigma}_t^2}{\delta_0\sigma_0^2+(1-\delta_0)\hat{\sigma}_t^2}\right)+\frac{\delta_0\sigma_0^2}{2\hat{\sigma}_t^2}}_{a(\hat{\sigma}_t^2)} + \underbrace{\frac{(\mu_0-\hat{\mu}_t)^2}{2}\left(\frac{\delta_0}{\hat{\sigma}_t^2}-\frac{(1-\delta_0)\delta_0}{(1-\delta_0)\hat{\sigma}_t^2+\delta_0\sigma_0^2}\right)}_{b(\hat{\mu}_t,\hat{\sigma}_t)}. \tag{A.4}$$

We now find the distribution of $g(\hat{\mu}_t, \hat{\sigma}_t)$ when samples samples $x_t..x_{t+w}$ used to calculate $\hat{\mu}_t$ and $\hat{\sigma}_t$ are distributed by θ_0. Decomposing $g(\hat{\mu}_t, \hat{\sigma}_t)$ into two terms:

$$a(\hat{\sigma}_t^2) = \frac{1}{2}\log\left(\frac{\hat{\sigma}_t^2}{\delta_0\sigma_0^2+(1-\delta_0)\hat{\sigma}_t^2}\right)+\frac{\delta_0\sigma_0^2}{2\hat{\sigma}_t^2},$$
$$b(\hat{\mu}_t,\hat{\sigma}_t^2) = \frac{(\mu_0-\hat{\mu}_t)^2}{2}\left(\frac{\delta_0}{\hat{\sigma}_t^2}-\frac{(1-\delta_0)\delta_0}{(1-\delta_0)\hat{\sigma}_t^2+\delta_0\sigma_0^2}\right).$$

A.1.4 Asymptotic distribution of first term $a(\hat{\sigma}_t^2)$

To find the asymptotic distribution of $a(\hat{\sigma}_t^2)$, we first recall some results. The asymptotic distribution of sample variance is $\hat{\sigma}_t^2$:

$$\hat{\sigma}_t^2 = \frac{1}{w}\sum_{1=1}^{w}(X_i - \hat{\mu}_t)^2,$$
$$\frac{\hat{\sigma}_t^2}{\sigma_0^2} = \frac{1}{w}\sum_{i=1}^{w}\underbrace{\frac{(X_i - \hat{\mu}_t)^2}{\sigma_0^2}}_{\chi_1^2 \text{ variables}}.$$

(Note sample variance is divided by $(w-1)$ instead of w. Though as $w \to \infty$, the the sample variance is similar when divided by w or $w-1$. We divide by w to use the tools of central limit theorem which can be found below.) By the central limit theorem, as $\frac{\hat{\sigma}_t^2}{\sigma_0^2}$ is a mean of sum of χ_1^2 variables. These variables have a mean 1 and variance 2:

$$\sqrt{N}(\frac{\hat{\sigma}_t^2}{\sigma_0^2} - 1) \xrightarrow{d} Z \sim \mathcal{N}(0, 2).$$

Or equivalently

$$\sqrt{N}(\hat{\sigma}_t^2 - \sigma_0^2) \xrightarrow{d} Z \sim \mathcal{N}(0, 2\sigma_0^4). \tag{A.5}$$

An asymptotically normal estimator $\hat{\theta}$, for the parameter θ, is distributed through:

$$\sqrt{n}(\hat{\theta} - \theta) \xrightarrow{d} W \sim \mathcal{N}(0, \sigma^2)$$

For a function $g(\hat{\theta})$, of an asymptotically normal estimator $\hat{\theta}$ of θ, the delta method states that:

$$\sqrt{n}(g(\hat{\theta}) - g(\theta)) \xrightarrow{d} W^* \sim \mathcal{N}(0, g'(\theta)^2\sigma^2)$$

This result is however true only when $g'(\theta)$ exists and is not 0. Since the sample variance, $\hat{\sigma}_t^2$, is asymptotically normal (as shown in (A.5)), we can try applying the delta method with $a(\hat{\sigma}_t^2)$ in place of $g(\theta)$:

$$\sqrt{n}(a(\hat{\sigma}_t^2) - a(\sigma_0^2)) \xrightarrow{d} W^* \sim \mathcal{N}(0, 2(a'(\hat{\sigma}_t^2))^2\sigma_0^4)$$

$$a'(\hat{\sigma}_t^2) = \frac{\delta_0\sigma_0^2}{-2\hat{\sigma}_t^4} + \frac{\delta_0\sigma_0^2}{2\hat{\sigma}_t^2(\delta_0\sigma_0^2 + (1-\delta_0)\hat{\sigma}_t^2)}$$

$$a'(\sigma_0^2) = \frac{\delta_0\sigma_0^2}{-2\sigma_0^4} + \frac{\delta_0}{2\sigma_0^2}$$

$$= 0.$$

As $a'(\sigma_0^2) = 0$, the delta method cannot be used. In such a case, the second order delta method can be used if $a''(\sigma_0^2) \neq 0$.

A.1.5 Second order delta method

For an asymptotically normal estimator $\hat{\theta}$ for the parameter θ, i.e.,

$$\sqrt{n}(\hat{\theta} - \theta) \xrightarrow{d} W \sim \mathcal{N}(0, \sigma^2),$$

the second order delta method [122] states that if there is a function g on these estimates $\hat{\theta}$, and both $g(\hat{\theta})$ and $g''(\theta_0)$ exist and are non 0, then

$$n(g(\hat{\theta}) - g(\theta_0)) \xrightarrow{d} W \sim \sigma_0^2 \frac{g''(\sigma_0^2)}{2}\chi_1^2.$$

Since the sample variance, $\hat{\sigma}_t^2$, is asymptotically normal (as shown in (A.5)), we can try applying the second order delta method with $a(\hat{\sigma}_t^2)$ in place of $g(\theta)$

$$n(a(\hat{\sigma}_t^2) - a(\sigma_0^2)) \xrightarrow{d} W \sim \sigma_0^4 a''(\sigma_0^2)\chi_1^2. \tag{A.6}$$

Finding the double derivative of $a(\hat{\sigma}_t^2)$ with respect to $\hat{\sigma}_t^2$

$$a''(\hat{\sigma}_t^2) = \frac{\delta_0\sigma_0^2}{\hat{\sigma}_t^6} - \frac{\delta_0\sigma_0^2\hat{\sigma}_t^{-2}(1-\delta_0) + \delta_0\sigma_0^2\hat{\sigma}_t^{-4}(\delta_0\sigma_0^2 + (1-\delta_0)\hat{\sigma}_t^2)}{2(\delta_0\sigma_0^2 + (1-\delta_0)\hat{\sigma}_t^2)^2}.$$

Plugging in $\hat{\sigma}_t = \sigma_0$

$$\begin{aligned} a''(\sigma_0^2) &= \frac{\delta_0}{\sigma_0^4} - \frac{\delta_0(1-\delta_0) + \delta_0}{2\sigma_0^4} \\ &= \frac{\delta_0^2}{2\sigma_0^4}. \end{aligned}$$

Plugging this result in (A.6), we have

$$\begin{aligned} n(a(\hat{\sigma}_t^2) - a(\sigma_0^2)) &\xrightarrow{d} W \sim \frac{\delta_0^2}{2}\chi_1^2, \\ a(\hat{\sigma}_t^2) &\xrightarrow{d} W^* \sim \frac{\delta_0^2}{2n}\chi_1^2 + a(\sigma_0^2). \end{aligned}$$

As $a(\sigma_0^2) = \frac{\delta_0}{2}$,

$$a(\hat{\sigma}_t^2) \xrightarrow{d} W^* \sim \frac{\delta_0^2}{2n}\chi_1^2 + \frac{\delta_0}{2}, \tag{A.7}$$

$$a(\hat{\sigma}_t^2) \xrightarrow{d} \frac{\delta_0^2}{2n}Z + \frac{\delta_0}{2} \text{ (where } Z \sim \chi_1^2).$$

A.1.6 Looking at the 2nd term $b(\hat{\mu}_t, \hat{\sigma}_t^2)$

The second term is defined as:

$$b(\hat{\mu}_t, \hat{\sigma}_t^2) = \frac{(\mu_0 - \hat{\mu}_t)^2}{2}\left(\frac{\delta_0}{\hat{\sigma}_t^2} - \frac{(1-\delta_0)\delta_0}{(1-\delta_0)\hat{\sigma}_t^2 + \delta_0\sigma_0^2}\right).$$

Manipulating the second term:

$$
\begin{aligned}
b(\hat{\mu}_t, \hat{\sigma}_t^2) &= \frac{(\mu_0 - \hat{\mu}_t)^2}{2} \left(\frac{\delta_0}{\hat{\sigma}_t^2} - \frac{(1-\delta_0)\delta_0}{(1-\delta_0)\hat{\sigma}_t^2 + \delta_0 \sigma_0^2} \right) \\
&= \frac{\frac{\sigma_0^2}{w} \frac{(\mu_0 - \hat{\mu}_t)^2}{\frac{\sigma_0^2}{w}}}{2} \left(\frac{\delta_0}{\hat{\sigma}_t^2} - \frac{(1-\delta_0)\delta_0}{(1-\delta_0)\hat{\sigma}_t^2 + \delta_0 \sigma_0^2} \right) \\
&= \frac{\frac{\sigma_0^2}{w} \frac{(\mu_0 - \hat{\mu}_t)^2}{\frac{\sigma_0^2}{w}}}{2} \left(\frac{\delta_0}{\frac{\sigma_0^2}{w-1}\frac{w-1}{\sigma_0^2}\hat{\sigma}_t^2} - \frac{(1-\delta_0)\delta_0}{(1-\delta_0)\frac{\sigma_0^2}{w-1}\frac{w-1}{\sigma_0^2}\hat{\sigma}_t^2 + \delta_0 \sigma_0^2} \right).
\end{aligned} \tag{A.8}
$$

Recall that the distribution of the sample mean and the sample variance are given by:

$$
\begin{aligned}
\hat{\mu}_t &\sim \mathcal{N}(\mu_0, \frac{\sigma_0^2}{n}), \\
\frac{(w-1)}{\sigma_0^2}\hat{\sigma}_t^2 = V &\sim \chi_{w-1}^2.
\end{aligned} \tag{A.9}
$$

Also,

$$
\frac{(\mu_0 - \hat{\mu}_t)^2}{\frac{\sigma_0^2}{w}} = Z \sim \chi_1^2.
$$

Using these results in (A.8) yields

$$
b(\hat{\mu}_t, \hat{\sigma}_t^2) \xrightarrow{d} \frac{\frac{\sigma_0^2}{w} Z}{2} \left(\frac{\delta_0}{\frac{\sigma_0^2}{w-1} V} - \frac{(1-\delta_0)\delta_0}{(1-\delta_0)\frac{\sigma_0^2}{w-1} V + \delta_0 \sigma_0^2} \right).
$$

As $w \to \infty$, by law of large numbers $\frac{\chi_{w-1}^2}{w-1} \to 1$. Thus as $w \to \infty$, $\frac{V}{w-1} \to 1$ leading to:

$$
\begin{aligned}
b(\hat{\mu}_t, \hat{\sigma}_t^2) &\to \frac{\frac{\sigma_0^2}{w} Z}{2} \left(\frac{\delta_0}{\sigma_0^2} - \frac{(1-\delta_0)\delta_0}{(1-\delta_0)\sigma_0^2 + \delta_0 \sigma_0^2} \right) \\
&= \frac{\delta_0^2 Z}{2w}.
\end{aligned}
$$

As $Z \sim \chi_1^2$, then when $w \to \infty$,

$$b(\hat{\mu}_t, \hat{\sigma}_t^2) \xrightarrow{d} \frac{\delta_0^2 Z}{2w} \sim \frac{\delta_0^2 \chi_1^2}{2w}. \tag{A.10}$$

A.1.7 Combining the two terms

Note that

$$\begin{aligned} g(\hat{\mu}_t, \hat{\sigma}_t) &= \underbrace{\frac{1}{2}\log\left(\frac{\hat{\sigma}_t^2}{\delta_0\sigma_0^2 + (1-\delta_0)\hat{\sigma}_t^2}\right) + \frac{\delta_0\sigma_0^2}{2\hat{\sigma}_t^2}}_{a(\hat{\sigma}_t^2)} + \underbrace{\frac{(\mu_0 - \hat{\mu}_t)^2}{2}\left(\frac{\delta_0}{\hat{\sigma}_t^2} - \frac{(1-\delta_0)\delta_0}{(1-\delta_0)\hat{\sigma}_t^2 + \delta_0\sigma_0^2}\right)}_{b(\hat{\mu}_t, \hat{\sigma}_t)} \\ &= a(\hat{\sigma}_t^2) + b(\hat{\mu}_t, \hat{\sigma}_t^2). \end{aligned}$$

Since

$$\begin{aligned} a(\hat{\sigma}_t^2) &\xrightarrow{d} \frac{\delta_0^2}{2n} Z + \frac{\delta_0}{2}, \\ b(\hat{\mu}_t, \hat{\sigma}_t^2) &\xrightarrow{d} \frac{\delta_0^2 U}{2w}. \end{aligned}$$

we have,

$$\begin{aligned} g(\hat{\mu}_t, \hat{\sigma}_t^2) &\xrightarrow{d} \frac{\delta_0^2}{2n} Z + \frac{\delta_0}{2} + \frac{\delta_0^2}{2n} U \text{ (where } Z, U \sim \chi_1^2) \\ &= \frac{\delta_0^2}{2n} Y + \frac{\delta_0}{2} \text{ (where } Y \sim \chi_2^2). \end{aligned} \tag{A.11}$$

A chi square variable of v degrees of freedom can be written as a gamma variable, with shape parameter $v/2$, and scale parameter 2. Also if $X \sim$ Gamma(a, b), then $k.X \sim$ Gamma$(a, k.b)$. Thus the asymptotic distribution of $g(\hat{\mu}_t, \hat{\sigma}_t^2)$ can be written as:

$$g(\hat{\mu}_t, \hat{\sigma}_t^2) \xrightarrow{d} X + \frac{\delta_0}{2} \text{ (where } X \sim \text{Gamma}(a(\text{shape}) = 1, b(\text{scale}) = \frac{\delta_0^2}{w})). \tag{A.12}$$

A.1.8 Finding the equivalence factor δ_0

The results from (A.12) can be used within (A.3) which can be written as:

$$\mathbb{E}_{\theta_0}\left[\exp(\delta_0\tilde{s}_t)\right] = \exp(\delta_0(-\frac{1}{2}-v))\mathbb{E}_{X\sim\text{Gamma}(1/2,2\delta_0^2/n)}\left[\exp\left(X+\frac{\delta_0}{2}\right)\right].$$

Using this result to solve for δ_0 in the statement of Lemma 1

$$\begin{aligned}\exp(\delta_0(-\frac{1}{2}-v))\mathbb{E}_{X\sim\text{Gamma}(1/2,2\delta_0^2/n)}\left[\exp\left(X+\frac{\delta_0}{2}\right)\right] &= 1\\ \exp\left(-\delta_0 v\right)\mathbb{E}_{X\sim\text{Gamma}(1/2,2\delta_0^2/n)}\underbrace{\left[\exp\left(X\right)\right]}_{\text{MGFunc}} &= 1. \qquad \text{(A.13)}\end{aligned}$$

The moment generating function of the gamma distribution is:

$$\mathbb{E}_{X\sim\text{Gamma}(a,b)}\left[\exp\left(tX\right)\right] = (1-tb)^{-a}.$$

Using the moment generating function results in

$$\begin{aligned}\exp\left(-\delta_0 v\right)\left(1-\frac{\delta_0^2}{n}\right)^{-1} &= 1\\ \delta_0 v + \log(1-\frac{\delta_0^2}{n}) &= 0\\ v &= \frac{-\log(1-\frac{\delta_0^2}{n})}{\delta_0} \quad \square\end{aligned}$$

APPENDIX B

SUPPLEMENTARY MATERIAL FOR CHAPTER 3

B.1 Technical details on training neural networks

B.1.1 Preprocessing

Inputs to all the networks are scaled to be between 0 and 1.

B.1.2 Network architecture and training details

The neural network f_θ used to learn representations from change points consists of three temporal blocks, where each of the blocks consists of 100 channels. Here a temporal block is defined as in [55] where each temporal block consists of two convolution layers with the same filter dilation. Each convolution layer is followed by weight normalization which is followed by a RELU activation and a dropout layer of 0.2.

For each successive temporal block, the filter was dilated by a factor of 2. The number of epochs needed to minimize training loss for f_θ was reduced by multiplying a constant (temperature [123]) to the embedding provided to the final softmax layer. Details for these parameters can be found in Table Table B.2. The first column refers to the different filters sizes (without dilation) used for each experiment. The second column lists the ρ parameter for the hinge loss while the third parameter lists the temperature values used.

2000 epochs were provided to train f_θ through pairwise change points. The ADAM optimizer with a learning rate of 0.0001 was used for all experiments to train f_θ.

The feedforward fully connected network f_ψ was trained using a learning rate of 0.001 through the ADAM optimizer and had two hidden layers of sizes 400 and 100 respectively. A RELU activation is used after each of these hidden layers. 400 epochs were provided to train the feed

Table B.1: Parameters for training f_θ

Experiment	Filter size	Hinge param ρ	Temp
Mean var	5	4	5
Mackay-Glass	10	8	10
HCI	30	8	5
WISDM	10	4	10

forward network.

For both the autoencoder and the supervised baselines, 1000 epochs were provided for training as the loss (for both validation and training, the loss became constant at the 700^{th} iteration and was constant until the 1000^{th} epoch.)

Each of the reported experiments was repeated 5 times, with mean and deviation (difference from the largest deviation from the mean) reported. The seed for functions based on randomness was fixed to a value 5.

B.2 Detecting change-points

Change points are also detected on sequences that are scaled between 0 and 1. This is not necessary but makes it convenient to get scaled similar/dissimilar pairs for the neural network f_θ directly from the sequence on which change points are detected.

An example of detecting change points on the Mackay-Glass sequence can be seen in Figure Figure B.1. This is a short sequence consisting of about 15 segments which can be used to set parameters needed for detecting change points. The first subplot shows the Mackay-Glass sequence. The second subplot shows the values of the MMD function as well as the detected changes while the third subplot shows the labels corresponding to different segments within the sequence. Note the mountain/hill like features for the MMD statistic in the second subplot. These hills arise because the MMD function starts increasing when the future window X_f starts overlapping with the segment belonging to the next class in the sequence. The peak value within this hill corresponds to the change point. The MMD function starts decreasing when the previous window X_p starts

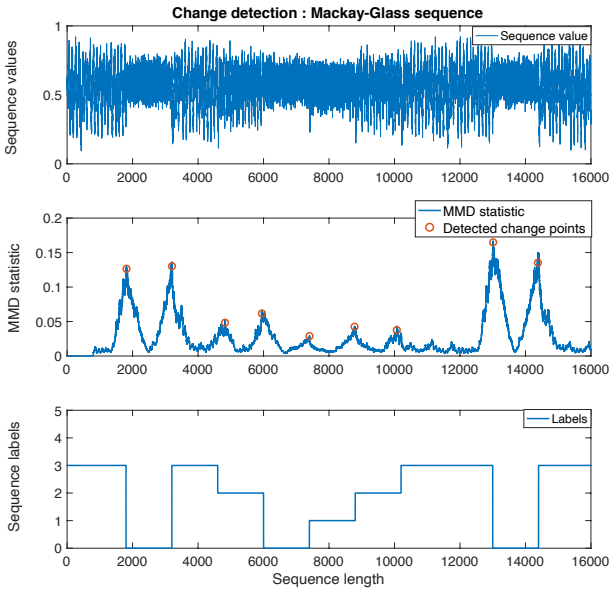


Figure B.1: Detected change points on Mackay-Glass sequence

overlapping with the sequence class corresponding to X_f

The peak function within the scipy [124] python package can be applied on change statistics m_i to obtain change points. The peak function is used with two options. One is the 'peak height' which is equivalent to the change detection threshold τ. The second argument is distance which specifies the minimum distance between detected change points.

The parameters used for detecting change points are listed below. τ is the detection threshold, w is the size of the windows for X_f and X_p and distance is the minimum distance between change points provided to the peak function.

Table B.2: Parameters for detecting change points

Experiment	τ	w	distance
Mean var	3	100	100
Mackay-Glass	0.025	800	800
HCI	0.18	600	600
WISDM	0.025	200	200

APPENDIX C
SUPPLEMENTARY MATERIAL FOR CHAPTER 4

C.1 Additional Experiment Details

For all experiments, a total of 2000 iterations were used and the model with best validation loss was saved. Table Table C.2 shows the projection dimension , sliding window size, triplet margins and regularization parameter γ used for all datasets. The triplet margin τ and regularization parameter γ were set through validation held out data.

For baselines that involved two-sample tests (such as M-Stats, KLCPD, SinkDiv), the same window sizes were used. For TIRE different window sizes were used till the best performance was attained.

For time efficiency, sliding windows were used to obtain batched batched two-sample tests. two-sample tests using Sinkhorn divergence libraries for Pytorch were conducted on these batches.

For the Sleep stage dataset, the true change points between REM and non-REM sleep stages are often not labeled perfectly (There might not be any prominent change at a true labeled change point for very short window sizes). For these reasons, when learning features through both sHSIC and SinkDivLM, we select windows on the opposite side of change points with a buffer of size 10. This buffer is not needed when detecting change points over sliding windows

In contrast to other methods, we do not post-process or filter score statistics before computing F1 and AUC scores [61, 63]. The AUC scores are obtained by first computing the receiver operator curves (ROC) curves, and then calculating the area under these curves. Figure Figure C.1 shows ROC curves for the Beedance and HASC sequences shown in figure Figure 4.4. The AUC for all sequences within a dataset were averaged and provided in table Table 5.1. For AUC scores, we do not use a detection margin (where a change is correctly detected if it is within a certain

Table C.1: F1 scores. Detection margins used for computing F1 scores

Dataset	Detection Margin	KLCPD	M-stats	HSIC	TSCP	TIRE_T	TIRE_F	SinkDiv	SinkDivLM
Beedance	15	0.805	0.741	0.796	0.541	0.693	0.744	0.773	0.853
HASC -2011	200	0.692	0.651	0.643	0.632	0.712	0.701	0.815	0.824
Yahoo	25	0.580	0.271	-	0.571	0.546	0.531	0.645	0.675
GMM	10	0.486	0.664	0.423	0.313	0.318	0.329	0.476	0.985
SwchFreq	200	0.951	0.759	0.561	0.431	0.727	0.732	0.759	0.848

Table C.2: Parameter settings for experiments

Dataset	Proj. dim	Win size	τ	γ
GMM switch	5	10	1.0	0.1
Freq switch	50	200	2.0	0.1
Freq switch w slope	50	100	2.0	0.1
Beedance	3	15	0.1	0.01
HASC	3	200	1	1.0
Yahoo	5	2	0.5	0.1
ECG	2	3	1.0	0.0001
Sleep stage	42	15	0.1	1

margin of the true change point). However, a detection margin is used for obtaining F1 scores. The detection margin, along with the threshold τ in table Table C.2, is used to get true positives, false positives and false negatives. These can then be used to compute Precision and Recall scores, with Precision $= \frac{\text{True Positive}}{\text{True Positive}+\text{False Positive}}$, and Recall $= \frac{\text{True Positive}}{\text{True Positive}+\text{True Negative}}$. F1 score can then be computed as F1 $= \frac{2(\text{Precision}\times\text{Recall})}{\text{Precision}+\text{Recall}}$.These F1 scores are calculated in the same way as other works in literature [61]. These detection margins are provided in table Table C.1. This table also lists additional F1 scores for recent work using contrastive representations for change point detection [64]. The available implementation for this method computes change point metrics on partitioned windows, which makes AUC score comparison infeasible with methods presented in the main paper, whose values for AUC are computed in a point by point manner (the original authors of [64] didn't report AUC numbers).

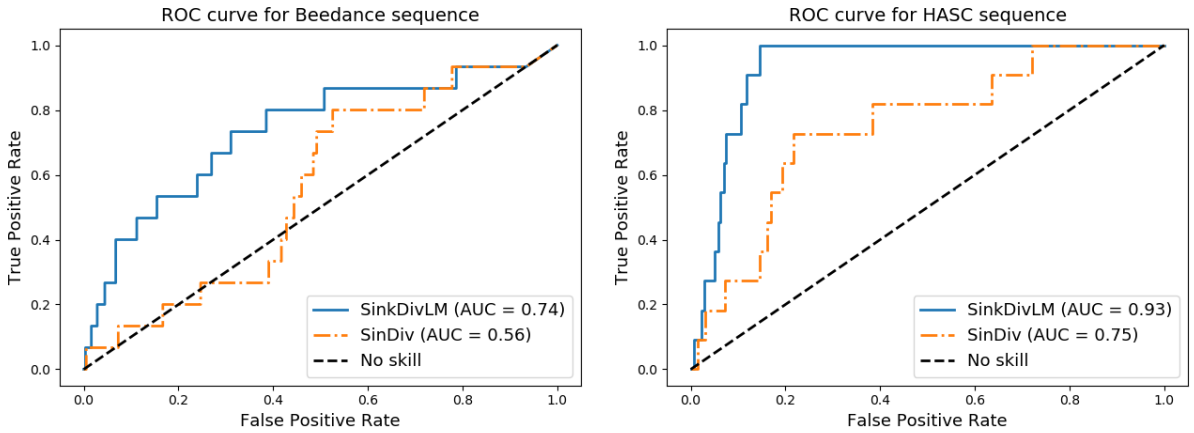


Figure C.1: ROC curves for Beedance and HASC sequences.

C.2 Proof of Proposition 1

Starting with the definition of Sinkhorn divergence

$$|S_\gamma(\alpha,\beta) - S_\gamma(\hat{\alpha}_n,\hat{\beta}_n)| = |\mathbb{E}\bar{f}_\gamma^{X,Y}(u^*,v^*) - \frac{1}{n}\sum_{i=1}^{n}\bar{f}_\gamma^{X_i,Y_i}(u,v)|, \tag{C.1}$$

where

$$\bar{f}_\gamma^{X,Y}(u,v) = f_\gamma^{X,Y}(u,v) - 0.5f_\gamma^{X,X}(u,v) - 0.5f_\gamma^{Y,Y}(u,v), \tag{C.2}$$

and where $f_\gamma(X,Y)(u,v)$ is the dual formulation of the entropy regularized Wasserstein distance.

$$f_\gamma^{(X,Y)}(u,v) \quad = \quad u(X) \quad + \quad v(Y) \quad - \quad \gamma\exp\left(\frac{u(X)+v(Y)-c(X,Y)}{\gamma}\right) \quad + \quad \gamma \tag{C.3}$$

Here u and v are the dual potentials, that are the Lagrange multipliers for the constraints in the primal formulation. X and Y are the mass locations and $c(X,Y)$ is the cost between mass locations X and Y.

The dual/Sinkhorn potentials u and v lie uniformly in the ball of Sobolov space with radius λ - represented by $(\mathcal{H}_\lambda^s)$, where the radius λ is of the order $(1+\frac{1}{\gamma^{s-1}})$ (Theorem 2 in [83]). The

Sobolov space , with $s = d/2 + 1$ is a reproducing kernel hilbert space (RKHS). Therefore the Sinkhorn potentials lie in an RKHS with a radius λ that is of the order $(1 + \frac{1}{\gamma^{d/2}})$.

We can apply results similar to those given in Lemma 3 in [83]. This leads to

$$\begin{aligned}|S_\gamma(\alpha, \beta) - S_\gamma(\hat{\alpha}_n, \hat{\beta}_n)| &= |\mathbb{E}\bar{f}_\gamma^{X,Y}(u^*, v^*) - \frac{1}{n}\sum_{i=1}^{n} \bar{f}_\gamma^{X_i,Y_i}(u, v)| \\ &\leqslant 3 \sup_{(u,v)\in(\mathcal{H}_\lambda^s)} |\mathbb{E}\bar{f}_\gamma^{X,Y}(u, v) - \frac{1}{n}\sum_{i=1}^{n} \bar{f}_\gamma^{X_i,Y_i}(u, v)|.\end{aligned}$$

Expanding the Sinkhorn divergence from (C.2),

$$\begin{aligned}\mathbb{E}|\mathbb{E}\bar{f}_\gamma^{X,Y}(u, v) - \frac{1}{n}\sum_{i=1}^{n} \bar{f}_\gamma^{X_i,Y_i}(u, v)| &\leqslant |\mathbb{E}f_\gamma^{X,Y}(u, v) - \frac{1}{n}\sum_{i=1}^{n} f_\gamma^{X_i,Y_i}(u, v)| \\ &+ 0.5|\mathbb{E}f_\gamma^{X,X}(u, v) - \frac{1}{n}\sum_{i=1}^{n} f_\gamma^{X_i,X_i}(u, v)| \\ &+ 0.5|\mathbb{E}f_\gamma^{Y,Y}(u, v) - \frac{1}{n}\sum_{i=1}^{n} f_\gamma^{Y_i,Y_i}(u, v)|\end{aligned}$$

Using results from Theorem 3 and Proposition 2 in [83], each of the absolute difference term on the right is bounded by $\frac{6B\lambda K}{\sqrt{n}}$. Using Proposition 2 in [83] , this leads to

$$\mathbb{E}|S_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}) - S_\gamma(\hat{\boldsymbol{\alpha}}_n, \hat{\boldsymbol{\beta}}_n)| \leqslant \frac{12B\lambda K}{\sqrt{n}}, \tag{C.4}$$

where n is the number of samples used to estimate $\hat{\boldsymbol{\alpha}}$. B is Lipschitz constant of the function f_γ , K is the maximum value taken by the kernel associated with the dual potentials and λ is the radius of the RKHS ball and is $O(1 + \frac{1}{\gamma^{d/2}})$.

Now we can apply McDiarmid's inequality on the function

$$f(x_1, ..., x_n) = |S_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}) - S_\gamma(\hat{\boldsymbol{\alpha}}_n, \hat{\boldsymbol{\beta}}_n)|. \tag{C.5}$$

The McDiarmid's inequality states that if $X_1, ..., X_n$ are independent random variables, and

$(x_1, x_2, ..., x_n)$ and $(\hat{x}_1, \hat{x}_2, ..., \hat{x}_n)$ differ only at the i^{th} index ($x_j = \hat{x}_j$ for all $j \neq i$)

$$|f(x_1, ..., x_n) - f(\hat{x}_1, ..., \hat{x}_n)| \leqslant c_i,$$

then

$$Pr(f(X_1, ..., X_n) - \mathbb{E}((f(X_1, ..., X_n) \geqslant t) \leqslant \exp(\frac{-2t^2}{\sum_{i=1}^{n} c_i^2}).$$

In our case the function $f(X_1, ..., X_n) = |\mathbb{E}\bar{f}_{\gamma}^{X,Y}(u, v) - \frac{1}{n}\sum_{i=1}^{n} \bar{f}_{\gamma}^{X_i,Y_i}(u, v)|$. With $f_{\gamma}^{X,Y}(u, v) \leqslant C$, changing one of the variables in $f(X_1, ..., X_n)$ changes the value of the function by at most $\frac{4C}{n}$. Thus $c_i = \frac{4C}{n}$, as compared to $\frac{2C}{n}$ for regularized Wasserstein distance in [83].Thus with probability $1 - \delta$,

$$f(X_1, ..., X_n) \leqslant \mathbb{E}(f(X_1, ..., X_n)) + C\sqrt{\frac{8\log(\frac{1}{\delta})}{n}}$$

Substituting (C.5), followed by (C.4) in this expression leads to with probability 1-δ

$$|S_{\gamma}(\boldsymbol{\alpha}, \boldsymbol{\beta}) - S_{\gamma}(\hat{\boldsymbol{\alpha}}_n, \hat{\boldsymbol{\beta}}_n)| \leqslant \frac{12B\lambda K}{\sqrt{n}} + C\sqrt{\frac{8\log(\frac{1}{\delta})}{n}}) \tag{C.6}$$

where n is the number of samples used to estimate $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$, λ is the radius of the RKHS ball in which the dual potentials u, v lie and is $O(1 + \frac{1}{\gamma^{d/2}})$. $C = \kappa + \gamma\exp(\kappa/\gamma)$, $\kappa = 2L|\mathcal{X}| + \|c\|_{\infty}$, L is the Lipschitz constant of the cost/distance c between mass locations, and $|\mathcal{X}|$ represents the diameter of the of the space of mass locations $X \in |\mathcal{X}|$. B is the Lipschitz constant of f_{γ} and is upper bounded by $\leqslant 1 + \exp(2\kappa/\gamma)$ [83].

C.3 Obtaining Transport Plans for Sinkhorn distances

The transport plan $\boldsymbol{P}$ for the regularized Wasserstein distances (or Sinkhorn distances), can be obtained using the Sinkhorn iteration [71].

C.3.1 Dual formulation

We can incorporate the constraints into a Lagrangian dual function

$$\max_{\mathbf{f},\mathbf{g}} \min_{\mathbf{P}} L(\mathbf{P},\mathbf{f},\mathbf{g}) = \max_{\mathbf{f},\mathbf{g}} \min_{\mathbf{P}} \langle \mathbf{C},\mathbf{P}\rangle - \gamma H(\mathbf{P}) \tag{C.7}$$

$$+ \langle \mathbf{f}, \mathbf{a} - \mathbf{P}\mathbb{1}_m\rangle + \langle \mathbf{g}, \mathbf{b} - \mathbf{P}^T\mathbb{1}_n\rangle$$

$$= \max_{\mathbf{f},\mathbf{g}} \langle \mathbf{f},\mathbf{a}\rangle + \langle \mathbf{g},\mathbf{b}\rangle + \min_{\mathbf{P}} \langle \mathbf{C} - \mathbf{f}\mathbb{1}_m - \mathbf{g}\mathbb{1}_m, \mathbf{P}\rangle - \gamma H(\mathbf{P})$$

$$= \max_{\mathbf{f},\mathbf{g}} \langle \mathbf{f},\mathbf{a}\rangle + \langle \mathbf{g},\mathbf{b}\rangle + \min_{\mathbf{P}} \langle \mathbf{C} - \mathbf{f}\mathbb{1}_m - \mathbf{g}\mathbb{1}_m, \mathbf{P}\rangle \tag{C.8}$$

$$- \gamma\langle \mathbf{P}, \log \mathbf{P} - \mathbb{1}_{n\times m}\rangle \tag{C.9}$$

By solving $\frac{\partial L(\mathbf{P},\mathbf{f},\mathbf{g})}{\partial \mathbf{P}} = 0$, we can obtain $\mathbf{P}$ such that:

$$\mathbf{P}_{i,j} = e^{\mathbf{f}_i/\gamma} \underbrace{e^{-\mathbf{C}_{i,j}/\gamma}}_{\text{Kernel}} e^{\mathbf{g}_j/\gamma}$$

Substituting $\mathbf{P}$ in (C.9), we can obtain after simplification the equivalent dual problem

$$\max_{\mathbf{f},\mathbf{g}} \langle \mathbf{f},\mathbf{a}\rangle + \langle \mathbf{g},\mathbf{b}\rangle - \gamma\langle e^{\mathbf{f}/\gamma}, e^{-\mathbf{C}/\gamma} e^{\mathbf{g}/\gamma}.\rangle \tag{C.10}$$

For non-discrete distributions, a more generalized dual formulation can be seen below

$$\begin{aligned}
\mathcal{W}_\gamma^p(\mathbf{a},\mathbf{b}) &= \sup_{(f(x)\in\mathcal{C}(\mathcal{X}), g(y)\in\mathcal{C}(\mathcal{Y}))} \int_{\mathcal{X}} f(x) d\alpha(x) \\
&\quad + \int_{\mathcal{Y}} g(y) d\beta(y) - \gamma \int_{\mathcal{X},\mathcal{Y}} e^{\frac{f(x)+g(y)-c(x,y)^p}{\gamma}} d\alpha(x) d\beta(y) \\
&= \sup_{(f(x)\in\mathcal{C}(\mathcal{X}), g(y)\in\mathcal{C}(\mathcal{Y}))} \mathbb{E}_{\alpha\beta}[z_\epsilon^{X,Y}(f,g)]
\end{aligned}$$

where $z_\epsilon^{x,y}(f,g) = f(x) + g(y) - \gamma e^{\frac{f(x)+g(y)-c(x,y)^p}{\gamma}}$.

C.3.2 Sinkhorn Algorithm

We can rewrite this as in vectors $\mathbf{u} = e^{\mathbf{f}_i/\gamma}, \mathbf{v} = e^{\mathbf{g}_i/\gamma}$ and $\mathbf{K} = e^{-\mathbf{C}_{i,j}/\gamma}$ as:

$$\mathbf{P} = \text{diag}(\mathbf{u})\mathbf{K}\text{diag}(\mathbf{v}) \tag{C.11}$$

Also from constraints:

$$\text{diag}(\mathbf{u})\mathbf{K}\text{diag}(\mathbf{v})\mathbb{1}_m = \mathbf{a} \text{ and } \left(\text{diag}(\mathbf{u})\mathbf{K}\text{diag}(\mathbf{v})\right)^T \mathbb{1}_n = \mathbf{b}.$$

$$\mathbf{u} \odot \mathbf{K}\mathbf{v} = \mathbf{a}. \text{ and } \mathbf{v} \odot \mathbf{K}^T\mathbf{u} = \mathbf{b}$$

An alternating update scheme can be used to update the dual potentials until convergence

$$\mathbf{u}^{l+1} = \frac{\mathbf{a}}{\mathbf{K}\mathbf{v}^l} \text{ and } \mathbf{v}^{l+1} = \frac{\mathbf{b}}{\mathbf{K}^T\mathbf{u}^l}$$

Matrix multiplications for these iterates can be parallelized through passing them as tensors on GPUs through libraries such as Pytorch. This allows a large number of two sample tests to be simultaneously computed, which speeds up gradient computation for learning $\boldsymbol{L}$. We use the Python Optimal Transport library which leverages Pytorch for computing Sinkhorn distances [125]. The Sinkhorn iterations are computed with a tolerance of 1e-3. For scenarios where the Sinkhorn distance time complexity $O(n^2)$ might be prohibitive to use for detecting repeated changes over sliding windows in a sequence, there is room to speed up computations by leveraging sliced Wasserstein distances on data that is transformed by the learned metric $\boldsymbol{L}$. This can be explored in the future.

APPENDIX D

SUPPLEMENTARY MATERIAL FOR CHAPTER 5

D.1 Hyper-parameters and other training details

For all of our runs, we used a Sinkhorn regularization parameter, $\gamma = 1e-3$.
Learning rate for all experiments was also set to $1e-3$. Our loss function doesn't take a constant sum of the supervised classification loss and Sinkhorn alignment values. The ratio of these two terms in the loss function wasn't tuned for any experiment. We were mindful of how our experiments settings should reflect real world scenarios where true labels from the target domain aren't available. All datasets were trained for 300 epochs before reporting numbers in table Table 5.1 For the HHAR and WISDM datasets, the temperature parameter τ for the softmax non linearity was set to 3. For HHAR this was set to 10 (as a larger number of channels were involved).

For the remaining baselines, we used the already set parameters for these different datasets in the Adatime benchmarking suite. We used Raincoat's adaptation of this benchmarking suite https://github.com/mims-harvard/Raincoat to run both raincoat and other baselines.

All experiments were performed on a Single NVIDIA Quadro RTX 5000.

D.2 Additional results

Table Table D.1 shows macro F1 scores when target domain labels in the validation hold out set were used to report evaluation numbers. The model paramaters corresponding to the best macro F1 performance on the target domain holdout validation set were saved. The performance of these saved models on test sets was then reported. We believe this method doesn't accurately reflect real world performance, but report numbers here as this is a common evaluation scheme used in literature.

Table D.1: Mean macro F1 scores over 5 runs for different domain adaptation methods

Model	Simulations	UCIHAR	HHAR	WISDM	WISDM-Bal	PXECG
Supervised Src	0.262	0.836	0.66	0.504	0.521	0.366
DANN	0.7	0.891	0.701	0.681	0.626	0.361
AdvSKM	0.712	0.88	0.671	0.616	0.665	0.389
CoDATS	0.531	0.907	0.744	**0.685**	0.816	0.366
CDAN	0.812	0.647	0.731	0.632	0.742	0.363
DeepCoral	0.843	0.892	0.697	0.621	0.701	0.346
CLUDA	0.802	0.857	0.661	0.491	0.760	0.325
SinkDiv	0.713	0.876	0.720	0.602	0.648	0.376
Raincoat	0.713	0.889	0.714	0.519	0.746	0.354
SSSS-TSA	**0.98**	**0.915**	**0.787**	0.677	**0.857**	**0.422**

It can be seen that there is a big difference in methods that employ adversarial learning (such as DANN, CoDATS etc.) between numbers in Table Table D.1 and Table 5.1. Methods employing Sinkhorn Distance have a smaller margin difference between these two approaches. Adversarial methods can be very unstable, and the best target domain validation scores often do not correspond to higher source domain classification scores. Source domain F1 scores on validation holdout sets are another way to stop model training and report domain adaptation methods on target domain test sets, but source classification scores reach their best values very early on in the training regime, way before target domain performance improves to their best values.

For HHAR and UCIHAR datasets, we use the same domain adaptation scenarios used in [93]. For WISDM we use a different set of domain adaptation scenarios as the raincoat paper addresses the universal domain shift problem, where some source domain calsses are not present in the target domain. As that is not the focus of our paper, we use a different set of domain adaptation scenarios for the WISDM dataset.

D.2.1 WISM domain adaptation scenarios

Table D.2: WISDM scenario test scores at end of training. Mean macro F1 scores for each domain adaptation scenario over 5 runs

Case	Supervised	DANN	AdvSKM	CoDATS	CDAN	CLUDA	SinkDiv	Raincoat	SSSS-TSA
20 to 30	0.5545	0.5668	0.6142	0.6144	0.4954	0.4862	0.7039	0.2414	0.6116
12 to 19	0.3668	0.2901	0.3380	0.2427	0.2758	0.2715	0.3153	0.2636	0.4451
30 to 20	0.6525	0.6540	0.6630	0.7912	0.7156	0.3591	0.4941	0.4506	0.7961
2 to 32	0.5897	0.4067	0.6914	0.5667	0.5004	0.3940	0.5003	0.3392	0.6345
7 to 30	0.7673	0.7471	0.8101	0.6666	0.7329	0.5073	0.6329	0.5073	0.8932
12 to 7	0.5132	0.5467	0.5200	0.5674	0.4905	0.3755	0.6804	0.4070	0.4785
18 to 20	0.5604	0.5212	0.5855	0.6886	0.5812	0.4400	0.6477	0.3912	0.6258
19 to 30	0.5627	0.3386	0.5657	0.3860	0.4865	0.6343	0.3454	0.4131	0.5105
4 to 19	0.1971	0.3404	0.3725	0.2932	0.3196	0.4181	0.6344	0.3119	0.6313
26 to 2	0.6626	0.6625	0.6492	0.6672	0.5730	0.3222	0.6162	0.2797	0.7321

Table D.3: WISDM scenario test scores when validation target domain labels to stop early. Mean macro F1 scores over 5 runs

Case	Supervised	DANN	AdvSKM	CoDATS	CDAN	CLUDA	SinkDiv	Raincoat	SSSS-TSA
20 to 30	0.6088	0.6303	0.6227	0.6980	0.6112	0.4862	0.6701	0.5716	0.7303
12 to 19	0.3661	0.3884	0.3306	0.3687	0.3247	0.2715	0.3800	0.2904	0.4479
30 to 20	0.7447	0.8452	0.7678	0.7959	0.8117	0.3591	0.5757	0.5818	0.8220
2 to 32	0.6810	0.7153	0.7160	0.7392	0.6914	0.3940	0.5660	0.6019	0.6295
7 to 30	0.7801	0.8041	0.8326	0.8528	0.7985	0.5073	0.6907	0.7195	0.9406
12 to 7	0.5335	0.7322	0.5245	0.7215	0.6271	0.3755	0.6925	0.5385	0.5154
18 to 20	0.6014	0.7349	0.6403	0.7316	0.6879	0.4400	0.7145	0.5469	0.6621
19 to 30	0.6847	0.6636	0.5898	0.5889	0.6716	0.6343	0.5201	0.5136	0.5792
4 to 19	0.3741	0.5121	0.4780	0.4719	0.4212	0.4181	0.5868	0.4243	0.6326
26 to 2	0.6716	0.7616	0.6627	0.8833	0.6743	0.3222	0.6311	0.4040	0.8181

D.2.2 WISDM-Balanced

Table D.4: WISDM-balance scenario test scores at end of training. Mean macro F1 scores for each domain adaptation scenario over 5 runs

Case	Supervised	DANN	AdvSKM	CoDATS	CDAN	CLUDA	SinkDiv	Raincoat	SSSS-TSA
20 to 30	0.5667	0.6495	0.6239	0.6067	0.5970	0.6230	0.6638	0.3382	0.8855
12 to 19	0.3898	0.6230	0.4848	0.8289	0.6673	0.5816	0.6306	0.5004	0.8333
30 to 20	0.5412	0.7388	0.6059	0.7048	0.7660	0.7410	0.6126	0.4495	0.7644
2 to 32	0.6133	0.7250	0.7004	0.6928	0.6007	0.6538	0.5506	0.4664	0.7205
7 to 30	0.7890	0.7526	0.7600	0.6975	0.7524	0.9223	0.6591	0.3515	0.9667
12 to 7	0.6028	0.7442	0.5418	0.5971	0.6109	0.7500	0.7541	0.5294	0.8966
18 to 20	0.5671	0.4784	0.6178	0.5805	0.4810	0.6663	0.7239	0.6197	0.5801
19 to 30	0.5572	0.6440	0.6824	0.5357	0.7061	0.6586	0.6913	0.3526	0.7218
4 to 19	0.1911	0.7800	0.4673	0.7174	0.6834	0.8079	0.7589	0.5225	0.9284
26 to 2	0.6456	0.6997	0.6281	0.6936	0.7472	0.5836	0.5529	0.3287	0.8598

Table D.5: WISDM-balance scenario test scores when validation target domain labels used to stop early. Mean macro F1 scores over 5 runs

Case	Supervised	DANN	AdvSKM	CoDATS	CDAN	CLUDA	SinkDiv	Raincoat	SSSS-TSA
20 to 30	0.6094	0.6610	0.6204	0.7028	0.6752	0.6783	0.6586	0.7077	0.8904
12 to 19	0.3969	0.7043	0.4847	0.9336	0.8137	0.6398	0.6889	0.6532	0.8765
30 to 20	0.6752	0.8003	0.7157	0.8089	0.8301	0.7684	0.6623	0.7803	0.7048
2 to 32	0.6761	0.8401	0.7600	0.8243	0.7414	0.7519	0.6099	0.7106	0.7316
7 to 30	0.7767	0.7578	0.7628	0.7576	0.7346	0.8588	0.7167	0.7903	0.9830
12 to 7	0.6375	0.7893	0.6232	0.8190	0.6694	0.8709	0.7723	0.8969	0.9695
18 to 20	0.6402	0.6508	0.6591	0.7890	0.6001	0.7542	0.7116	0.7820	0.7508
19 to 30	0.7005	0.7950	0.7336	0.8048	0.7834	0.7585	0.7170	0.8508	0.8642
4 to 19	0.5702	0.8186	0.6876	0.8425	0.7823	0.8247	0.7878	0.5886	0.9529
26 to 2	0.6621	0.7455	0.6111	0.8788	0.7979	0.6944	0.5934	0.6992	0.8528

D.2.3 HHAR domain adaptation scenarions

Table D.6: HHAR scenario test scores at end of training. Mean macro F1 scores for each domain adaptation scenario over 5 runs

Case	Supervised	DANN	AdvSKM	CoDATS	CDAN	CLUDA	SinkDiv	Raincoat	SSSS-TSA
0 to 2	0.6103	0.6879	0.6611	0.6550	0.7097	0.6975	0.6654	0.7291	0.8423
1 to 6	0.8185	0.9482	0.8703	0.9296	0.9402	0.8505	0.9193	0.9010	0.9157
2 to 4	0.3948	0.6147	0.4327	0.5501	0.6009	0.6029	0.6763	0.3900	0.6581
4 to 0	0.2688	0.2749	0.2585	0.3431	0.3104	0.3241	0.3487	0.2324	0.4575
4 to 5	0.8245	0.9564	0.9283	0.9633	0.9558	0.8973	0.8914	0.9069	0.9406
5 to 1	0.9011	0.9789	0.9000	0.9695	0.9557	0.9335	0.9457	0.8873	0.9794
5 to 2	0.2905	0.3671	0.3435	0.3112	0.4011	0.4939	0.4076	0.2422	0.5563
7 to 2	0.3604	0.4264	0.3817	0.2890	0.4376	0.4490	0.4351	0.3776	0.5978
7 to 5	0.6025	0.8752	0.6194	0.8606	0.6830	0.6075	0.7413	0.7186	0.7142
8 to 4	0.7085	0.9727	0.7661	0.9679	0.9714	0.5585	0.7858	0.6518	0.6459

Table D.7: HHAR scenario test scores when validation target domain labels used to stop early. Mean macro F1 scores over 5 runs

Case	Supervised	DANN	AdvSKM	CoDATS	CDAN	CLUDA	SinkDiv	Raincoat	SSSS-TSA
0 to 2	0.6971	0.7520	0.7107	0.7207	0.7774	0.7118	0.7285	0.7436	0.8923
1 to 6	0.9022	0.9541	0.8986	0.9414	0.9475	0.8739	0.9372	0.9198	0.9344
2 to 4	0.4846	0.6613	0.4835	0.6119	0.6264	0.6288	0.8093	0.5962	0.7558
4 to 0	0.3090	0.4000	0.3181	0.4166	0.3723	0.3991	0.3683	0.4408	0.5506
4 to 5	0.8819	0.9709	0.9445	0.9747	0.9551	0.9064	0.9254	0.9182	0.9382
5 to 1	0.9248	0.9776	0.9224	0.9766	0.9599	0.9176	0.9470	0.9472	0.9771
5 to 2	0.3699	0.5157	0.3874	0.4631	0.4503	0.5153	0.4521	0.4527	0.6217
7 to 2	0.4074	0.5232	0.4078	0.4525	0.4668	0.4744	0.4378	0.5047	0.6245
7 to 5	0.7230	0.9008	0.7561	0.9077	0.7767	0.6334	0.7688	0.9253	0.8084
8 to 4	0.8425	0.9739	0.8777	0.9757	0.9783	0.5552	0.8279	0.6933	0.7706

D.2.4 UCIHAR domain adaptation scenarions

Table D.8: UCIHAR scenario test scores at end of training. Mean macro F1 scores for each domain adaptation scenario over 5 runs

Case	Supervised	DANN	AdvSKM	CoDATS	CDAN	CLUDA	SinkDiv	Raincoat	SSSS-TSA
2 to 11	0.5842	0.9653	0.9967	0.9509	0.8339	0.9401	0.9732	0.9741	0.9768
6 to 23	0.7358	0.9076	0.8739	0.9722	0.9465	0.8969	0.8984	0.8812	0.9212
7 to 13	0.7897	0.8758	0.8469	0.8678	0.9025	0.8497	0.9264	0.9123	0.9407
9 to 18	0.3897	0.5603	0.5786	0.6434	0.5345	0.5156	0.5837	0.5921	0.6938
12 to 16	0.4853	0.4907	0.6208	0.5418	0.5660	0.6082	0.6662	0.6733	0.8155
13 to 19	0.9155	0.7635	0.9297	0.8883	0.9426	0.9336	0.9685	0.9540	0.9283
18 to 21	0.9947	0.9531	0.9967	0.9939	1.0000	0.9175	0.9966	0.9990	1.0000
20 to 6	0.9694	0.9446	1.0000	0.9552	0.8159	0.8321	1.0000	0.9578	0.9463
23 to 13	0.7981	0.7455	0.7867	0.7464	0.8379	0.7543	0.8003	0.8732	0.8228
24 to 12	0.8380	0.9810	0.8535	0.9967	0.9810	0.9944	0.9555	0.9532	0.9669

Table D.9: UCIHAR scenario test scores when validation target domain labels used to stop early. Mean macro F1 scores over 5 runs

Case	Supervised	DANN	AdvSKM	CoDATS	CDAN	CLUDA	SinkDiv	Raincoat	SSSS-TSA
2 to 11	0.7290	0.9967	0.9967	0.9935	0.9935	0.9843	1.0000	1.0000	1.0000
6 to 23	0.7326	0.9772	0.8800	0.9793	0.9948	0.9252	0.9219	0.9224	0.9742
7 to 13	0.8753	0.9232	0.8524	0.8748	0.9556	0.8377	0.8994	0.8610	0.9209
9 to 18	0.7231	0.6750	0.6695	0.7402	0.7152	0.5045	0.5634	0.6544	0.6521
12 to 16	0.6534	0.5922	0.6351	0.6743	0.6624	0.6131	0.6560	0.6754	0.8372
13 to 19	0.9468	0.9758	0.9514	0.9921	0.9654	0.9212	0.9740	0.9547	0.9681
18 to 21	0.9978	1.0000	0.9967	0.9973	1.0000	0.9434	1.0000	0.9957	1.0000
20 to 6	0.9755	0.9911	1.0000	0.9782	0.9016	0.9956	1.0000	0.9578	0.9622
23 to 13	0.7642	0.7928	0.8283	0.7698	0.8945	0.8534	0.8491	0.8867	0.8531
24 to 12	0.9396	0.9941	0.9706	0.9967	0.9810	0.9943	0.9785	0.9645	0.9844

D.3 Datasets

HHAR: https://archive.ics.uci.edu/dataset/344/heterogeneity+activity+recognition

WISDM: https://archive.ics.uci.edu/dataset/507/wisdm+smartphone+and+smartwatch+activity+and+biometrics+dataset

UCIHAR: https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones

PXECG: https://physionet.org/content/ptb-xl/1.0.3/

REFERENCES

[1] Nauman Ahad, Mark A Davenport, and Yao Xie. "Data-adaptive symmetric CUSUM for sequential change detection". In: *Seq. Anal.* 43.1 (2024), pp. 1–27.

[2] Nauman Ahad et al. "Validating a wheelchair in-seat activity tracker". In: *Assist. Techn.* 34.5 (2022), pp. 588–598.

[3] Nauman Ahad and Mark A Davenport. "Semi-supervised Sequence Classification through Change Point Detection". In: *Proc. AAAI Conf. Artif. Intell. (AAAI)*. Online, 2021.

[4] Nauman Ahad et al. "Learning Sinkhorn divergences for supervised change point detection". In: *arXiv preprint arXiv:2202.04000* (2022).

[5] Carolina Urzay et al. "Detecting change points in neural population activity with contrastive metric learning". In: *Int. IEEE/EMBS Conf. on Neur. Engineering (NER)*. Baltimore, Maryland, 2023.

[6] Diederik P Kingma, Max Welling, et al. "An introduction to variational autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[7] Samaneh Aminikhanghahi and Diane Cook. "A survey of methods for time series change point detection". In: *Knowl. Inf. Syst.* 51.2 (2017), pp. 339–367.

[8] Alexander G Tartakovsky et al. "A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods". In: *IEEE Trans. Signal Process.* 54.9 (2006), pp. 3372–3382.

[9] Charles Truong, Laurent Oudre, and Nicolas Vayatis. "Selective review of offline change point detection methods". In: *Signal Process.* 167.1 (2020), p. 107299.

[10] Ewan Page. "Continuous inspection schemes". In: *Biometrika* 41.1 (1954), pp. 100–115.

[11] Liyan Xie, George V, and Yao Xie. "Window-limited CUSUM for sequential change detection". In: *IEEE Trans. Inf. Theory* (2023), pp. 5990 –6005.

[12] Song Liu et al. "Change-point detection in time-series data by relative density-ratio estimation". In: *Neural Netw.* 43 (2013), pp. 72–83.

[13] Zhihan Yue et al. "Ts2vec: Towards universal representation of time series". In: *Proc. AAAI Conf. on Artificial Intell.(AAAI)*. Online, 2022.

[14] Garrett Wilson, Janardhan Rao Doppa, and Diane J Cook. "Multi-source deep domaadaptation with weak supervision for time-series sensor data". In: *Proc. Int. Conf. on Kownl. Disc. and Data Mining (KDD)*. Online, 2020.

[15] Marco Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport". In: *Proc. Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*. Lake Thaoe, Nevada, 2013.

[16] Ping Yang, Guy Dumont, and John Ansermino. "Adaptive change detection in heart rate trend monitoring in anesthetized children". In: *IEEE. Trans. Biomed. Eng.* 53.11 (2006), pp. 2211–2219.

[17] Lifeng Lai, Yijia Fan, and Vincent Poor. "Quickest detection in cognitive radio: A sequential change detection framework". In: *Proc. IEEE Glob. Telecom. Conf. (GLOBECOM)*. New Orleans, Louisiana, 2008.

[18] Yousef Al-Assaf. "Surface myoelectric signal analysis: dynamic approaches for change detection and classification". In: *IEEE. Trans. Biomed. Eng.* 53.11 (2006), pp. 2248–2256.

[19] Gary Lorden. "Procedures for reacting to a change in distribution". In: *Ann. Math. Stat.* 42.6 (1971), pp. 1897–1908.

[20] David Siegmund and ES Venkatraman. "Using the generalized likelihood ratio statistic for sequential detection of a change-point". In: *Ann. Stat.* (1995), pp. 255–271.

[21] Samaneh Aminikhanghahi and Diane Cook. "Using change point detection to automate daily activity segmentation". In: *Proc. IEEE Int. Conf. on Perv. Comp. and Comm. Workshops (PerCom Workshops)*. Kona, Hawaii, 2017.

[22] George Moustakides. "Optimal stopping times for detecting changes in distributions". In: *Ann. Stat.* 14.4 (1986), pp. 1379–1387.

[23] Venugopal V. Veeravalli and Taposh Banerjee. "Quickest Change Detection". In: *Academic Press Library in Signal Processing*. Vol. 3. Elsevier, 2014, pp. 209–255.

[24] Pierre Granjon. *The CUSUM algorithm - a small review*. Tech. rep. hal-00914697, 2014.

[25] Dean Bodenham and Niall Adams. "Continuous monitoring for changepoints in data streams using adaptive estimation". In: *Stat. Comput.* 27.5 (2017), pp. 1257–1270.

[26] Yasmin Fathy, Payam Barnaghi, and Rahim Tafazolli. "An Online Adaptive Algorithm for Change Detection in Streaming Sensory Data". In: *IEEE Syst. J.* 13.3 (2018), pp. 2688–2699.

[27] Douglas Hawkins and KD Zamba. “Statistical process control for shifts in mean or variance using a changepoint formulation”. In: *Technometrics* 47.2 (2005), pp. 164–173.

[28] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. “Detecting change in data streams”. In: *Proc. V. Large Datab. (VLDB)*. Toronto, Canada, 2004.

[29] Cesare Alippi et al. “Change detection in multivariate datastreams: likelihood and detectability loss”. In: *Proc. Int. Joint Conf. on Artif. Intelligence (IJCAI)*. New York, NY, 2016.

[30] Wei-Cheng Chang et al. “Kernel Change-point Detection with Auxiliary Deep Generative Models”. In: *Proc. Int. Conf. on Learning Representations (ICLR)*. New Orleans, Louisiana, 2019.

[31] Michele Basseville and Albert Benveniste. “Sequential detection of abrupt changes in spectral characteristics of digital signals”. In: *IEEE Trans. Inf. Theory* 29.5 (1983), pp. 709–724.

[32] Regine Andre-Obrecht. “A new statistical approach for the automatic segmentation of continuous speech signals”. In: *IEEE Trans. Accous. Speech. Signal Process.* 36.1 (1988), pp. 29–40.

[33] Fredrik Gustafsson. *Adaptive filtering and change detection*. Vol. 1. Citeseer, 2000.

[34] Michele Basseville, Igor Nikiforov, et al. *Detection of abrupt changes: Theory and Application*. Vol. 104. Prentice Hall, 1993.

[35] Liyan Xie, George Moustakides, and Yao Xie. “First-order optimal sequential subspace change-point detection”. In: *Proc. IEEE Glob. Conf. on Sig. and Inf. Process.(GlobalSIP)*. Anaheim, CA, 2018.

[36] Liyan Xie, Yao Xie, and George V Moustakides. “Sequential subspace change point detection”. In: *Seq. Anal.* 39.3 (2020), pp. 307–335.

[37] Jindong Wang et al. “Deep learning for sensor-based activity recognition: A survey”. In: *Pattern Recognit. Lett.* 119.1 (2019), pp. 3–11.

[38] Jesper Van Engelen and Holger Hoos. “A survey on semi-supervised learning”. In: *Mach. Learn.* 109.2 (2020), pp. 373–440.

[39] Thorsten Joachims. “Transductive inference for text classification using support vector machines”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. Bled, Slovenia, 1999.

[40] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. “Semi-supervised learning using gaussian fields and harmonic functions”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. Washington, D.C., 2003.

[41] Xiaojin Zhu. *Semi-supervised learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2005.

[42] Avital Oliver et al. “Realistic evaluation of deep semi-supervised learning algorithms”. In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Montreal, Canada, 2018.

[43] Yves Grandvalet and Yoshua Bengio. “Semi-supervised learning by entropy minimization”. In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Vancouver, Canda, 2005.

[44] Antti Rasmus et al. “Semi-supervised learning with ladder networks”. In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Montreal, Canada, 2015.

[45] Samuli Laine and Timo Aila. “Temporal ensembling for semi-supervised learning”. In: *Proc. Int. Conf. on Learning Representations (ICLR)*. San Juan, Peurto Rico, 2016.

[46] David Berthelot et al. “Mixmatch: A holistic approach to semi-supervised learning”. In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Vancouver, Canada, 2019.

[47] Andrew Dai and Quoc Le. “Semi-supervised sequence learning”. In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Montreal, Canada, 2015.

[48] Ming Zeng et al. “Semi-supervised convolutional neural networks for human activity recognition”. In: *Proc. IEEE Int. Conf. on Big Data (BigData)*. Boston, Massachusetts, 2017.

[49] Eric Xing et al. “Distance metric learning with application to clustering with side-information”. In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Vancouver, Canada, 2003.

[50] Mahdieh Soleymani Baghshah and Saeed Bagheri Shouraki. “Semi-supervised metric learning using pairwise constraints”. In: *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*. Pasadena, California, 2009.

[51] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. “Siamese neural networks for one-shot image recognition”. In: *Proc. Int. Conf. on Machine Learning Workshop in Deeplearning*. 2015.

[52] Yen-Chang Hsu and Zsolt Kira. “Neural network-based clustering using pairwise constraints”. In: *Proc. Int. Conf. on Learning Representations Workshops*. 2016.

[53] Arthur Gretton et al. “A kernel two-sample test”. In: *J. Mach. Learn. Res. (JMLR)* 13 (2012), pp. 723–773.

[54] Shuang Li et al. “M-statistic for kernel change-point detection”. In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Montreal, Canada, 2015.

[55] Shaojie Bai, Zico Kolter, and Vladlen Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”. In: *arXiv:1803.01271* (2018).

[56] Leon Glass and Michael Mackey. “Mackey-Glass equation”. In: *Scholarpedia* 5.3 (2010), p. 6908.

[57] Jens Kohlmorgen and Steven Lemm. “A dynamic hmm for on-line segmentation of sequential data”. In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Vancouver, Canada, 2002.

[58] Kilian Forster, Daniel Roggen, and Gerhard Troster. “Unsupervised classifier self-calibration through repeated context occurences: Is there robustness against sensor displacement to gain?” In: *Proc. IEEE Int. Symp. on Wearable Computers (ISWC)*. 2009.

[59] Liyan Xie et al. “Sequential (Quickest) Change Detection: Classical Results and New Directions”. In: *IEEE Journ. on Select. Areas in Info. Theory* 2.2 (2021), pp. 494–514.

[60] Shuang Li et al. “Scan B-statistic for kernel change-point detection”. In: *Seq. Anal.* 38.4 (2019), pp. 503–544.

[61] Kevin C Cheng et al. “Optimal transport based change point detection and time series segment clustering”. In: *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. Online, 2020.

[62] Makoto Yamada et al. “Change-point detection with feature selection in high-dimensional time-series data”. In: *Proc. Int. Joint Conf. on Artif. Intell. (IJCAI)*. Beijing, China, 2013.

[63] Tim Deryck, Maarten De Vos, and Alexander Bertrand. “Change Point Detection in Time Series Data using Autoencoders with a Time-Invariant Representation”. In: *IEEE Trans. Signal Process.* 69.1 (2021), pp. 3513 –3524.

[64] Shohreh Deldari et al. “Time series change point detection with self-supervised contrastive predictive coding”. In: *Proc. Web Conf.(WWW)*. 2021.

[65] Rémi Lajugie, Francis Bach, and Sylvain Arlot. “Large-margin metric learning for constrained partitioning problems”. In: *Proc. Int. Conf. Mach. Learn. (ICML)*. Beijing, China, 2014.

[66] Sean J Taylor and Benjamin Letham. “Forecasting at scale”. In: *Amer. Statist.* 72.1 (2018), pp. 37–45.

[67] Charles Truong, Laurent Oudre, and Nicolas Vayatis. “Supervised kernel change point detection with partial annotations”. In: *IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*. 2019.

[68] Fang Li, George C Runger, and Eugene Tuv. “Supervised learning for change-point detection”. In: *Int. Journ. of Product. Res.* 44.14 (2006), pp. 2853–2868.

[69] Victor Eruhimov et al. “Change-point detection with supervised learning and feature selection”. In: *Proc. IEEE Int. Conf. on Informatics in Control, Automation and Robotics (ICINO)*.

[70] Charles Truong and Laurent Oudre. “Supervised change-point detection with dimension reduction, applied to physiological signals”. In: *NeurIPS Workshop on Learning from Time Series for Health*. New Orleans, Louisiana, 2022.

[71] Gabriel Peyré, Marco Cuturi, et al. “Computational optimal transport: With applications to data science”. In: *Found. Trends Mach. Learn.* 11.5-6 (2019), pp. 355–607.

[72] Jean Feydy et al. “Interpolating between optimal transport and MMD using Sinkhorn divergences”. In: *Proc. Int. Conf. Art. Intell. Stat. (AISTATS)*. Naha, Japan, 2019.

[73] Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. “On Wasserstein two-sample testing and related families of non-parametric tests”. In: *Entropy* 19.2 (2017), p. 47.

[74] Marco Cuturi and David Avis. “Ground metric learning”. In: *J. Mach. Learn. Res. (JMLR)* 15.1 (2014), pp. 533–564.

[75] Gao Huang et al. “Supervised word mover’s distance”. In: *Proc. Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*. Barcelona, Spain, 2016.

[76] François-Pierre Paty and Marco Cuturi. “Subspace robust wasserstein distances”. In: *Proc. Int. Conf. Mach. Learn. (ICML)*. Long Beach, California, 2019.

[77] Bing Su and Ying Wu. “Learning distance for sequences by learning a ground metric”. In: *Proc. Int. Conf. Mach. Learn. (ICML)*. Long Beach, California, 2019.

[78] Peng Zhao and Zhi-Hua Zhou. “Label distribution learning by optimal transport”. In: *AAAI Conf. Artif. Intell. (AAAI)*. New Orleans, Louisiana, 2018.

[79] Tanguy Kerdoncuff, Rémi Emonet, and Marc Sebban. “Metric Learning in Optimal Transport for Domain Adaptation”. In: *Proc. Int. Joint Conf. on Artif. Intell. (IJCAI)*. Online, 2021.

[80] Kilian Q Weinberger and Lawrence K Saul. “Distance metric learning for large margin nearest neighbor classification.” In: *J. Mach. Learn. Res. (JMLR)* 10.2 (2009).

[81] Eric Xing et al. “Distance metric learning with application to clustering with side-information”. In: *Proc. Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*. Vancouver, Canada, 2002.

[82] Yiming Ying, Kaizhu Huang, and Colin Campbell. “Sparse Metric Learning via Smooth Optimization”. In: *Proc. Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*. Vancouver, Canada, 2009.

[83] Aude Genevay et al. “Sample complexity of sinkhorn divergences”. In: *Proc. Int. Conf. Art. Intell. Stat. (AISTATS)*. Naha, Japan, 2019.

[84] Jie Wang, Rui Gao, and Yao Xie. “Two-sample Test using Projected Wasserstein Distance”. In: *Proc. Int. Symp. Info. Theory (ISIT)*. Online, 2021.

[85] Tom Fawcett. “An introduction to ROC analysis”. In: *Pattern Recognit. Lett.* 27.8 (2006), pp. 861–874.

[86] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *J. Mach. Learn. Res. (JMLR)* 12 (2011), pp. 2825–2830.

[87] Keith B Hengen et al. “Neuronal firing rate homeostasis is inhibited by sleep and promoted by wake”. In: *Cell* 165.1 (2016), pp. 180–191.

[88] Bing Su and Gang Hua. “Order-preserving wasserstein distance for sequence matching”. In: *Proc. IEEE Conf. on Comp. Vision and Patter. Recog (CVPR)*. Honolulu, Hawaii, 2017.

[89] Paul T Baker et al. “Multivariate classification with random forests for gravitational wave searches of black hole binary coalescence”. In: *Phys, Rev. D* 91.6 (2015), p. 62004.

[90] Christian Bock et al. “Machine learning for biomedical time series classification: from shapelets to deep learning”. In: *Artifi. Neur. Networks* (2021), pp. 33–71.

[91] Muhammad Amjad and Devavrat Shah. “Trading bitcoand online time series prediction”. In: *NeurIPS workshop on Time Series*. Long Beach, California, 2017.

[92] Daniel Coelho et al. “Predictive maintenance on sensorized stamping presses by time series segmentation, anomaly detection, and classification algorithms”. In: *Procedia Computer Science* 200 (2022), pp. 1184–1193.

[93] Huan He et al. “Domain Adaptation for Time Series Under Feature and Label Shifts”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. Honolulu, Hawaii, 2023.

[94] Mohamed Ragab et al. “ADATIME: A Benchmarking Suite for DomaAdaptation on Time Series Data”. In: *ACM Trans. on Knowl. Discov. from Data* 17.8 (2023).

[95] Qiao Liu and Hui Xue. “Adversarial Spectral Kernel Matching for Unsupervised Time Series DomaAdaptation.” In: *Proc. Int. Joint Conf. on Aritificial Intelligence (IJCAI)*. 2021.

[96] Shai Ben-David et al. “A theory of learning from different domains”. In: *Mach. Learn.* 79 (2010), pp. 151–175.

[97] Gijs van Tulder and Marleen de Bruijne. “Unpaired, unsupervised domaadaptation assumes your domains are already similar”. In: *Medical Image Analysis* 87 (2023), p. 102825.

[98] Han Zhao et al. “On learning invariant representations for domaadaptation”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. Long Beach, California, 2019.

[99] Yaroslav Ganin et al. “Domain-adversarial training of neural networks”. In: *J. Mach. Learn. Res. (JMLR)* 17.59 (2016), pp. 1–35.

[100] Mingsheng Long, Zhangjie Cao, and Michael I Wang Jianmand Jordan. “Conditional adversarial domain adaptation”. In: *Proc. Neur. Info. Processing Systems (NeurIPS)*. Montreal, Canada, 2018.

[101] Wei Wang et al. “Rethinking Maximum Mean Discrepancy for Visual DomaAdaptation”. In: *IEEE Trans. on Neur. Netw. and Learning Systems* 34.1 (2023), pp. 264–277.

[102] Jian Shen et al. “Wasserstein distance guided representation learning for domain adaptation”. In: *Proc. Conf. on Artfl. Intelligence (AAAI)*. New Orleans, Louisiana, 2018.

[103] Baochen Sun and Kate Saenko. “Deep coral: Correlation alignment for deep domaadaptation”. In: *Workshops, Europ. Conf. on Computer Vision (ECCV)*. 2016.

[104] Qian Wang and Toby Breckon. “Unsupervised domaadaptation via structured prediction based selective pseudo-labeling”. In: *Proc. Conf. on Artfl. Intelligence (AAAI)*. New York City, New York, 2020.

[105] Ankit Singh. “Clda: Contrastive learning for semi-supervised domaadaptation”. In: *Proc. Neur. Info. Processing Systems (NeurIPS)*. Online.

[106] Ruichu Cai et al. “Time series domaadaptation via sparse associative structure alignment”. In: *Proc. Conf. on Artfl. Intelligence (AAAI)*. Online, 2021.

[107] Mohamed Ragab et al. “Self-supervised autoregressive domaadaptation for time series data”. In: *IEEE Trans. on Neur. Net. and Learning Systems* 35.1 (2024).

[108] Yilmazcan Ozyurt, Stefan Feuerriegel, and Ce Zhang. “Contrastive Learning for Unsupervised DomaAdaptation of Time Series”. In: *Proc. Int. Conf. on Learning Representations (ICLR)*. Kigali, Rawanda, 2023.

[109] Ran Liu et al. “Seeing the forest and the tree: Building representations of both individual and collective dynamics with transformers”. In: *Proc. Neur. Info. Processing Systems (NeurIPS)*. New Orleans, Louisiana, 2022.

[110] Yuqi Nie et al. “A Time Series is Worth 64 Words: Long-term Forecasting with Transformers”. In: *Proc. Int. Conf. on Learning Representations (ICLR)*. Online, 2022.

[111] Wei Tao et al. “EEG-based emotion recognition via channel-wise attention and self attention”. In: *IEEE Trans. on Affective Computing* 14.1 (2020).

[112] Aude Genevay, Gabriel Peyré, and Marco Cuturi. “Learning generative models with sinkhorn divergences”. In: *Proc. Int. Conf. Art. Intell. Stat. (AISTATS)*. 2018.

[113] Patrick Wagner et al. “PTB-XL, a large publicly available electrocardiography dataset”. In: *Scientific data* 7.1 (2020), p. 154.

[114] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. “Deep bayesian active learning with image data”. In: *Proc. Int. Conf. on Machine Learning (ICML)*. Sydney, Australia, 2017.

[115] Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. “Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning”. In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Vancouver, Canada, 2019.

[116] Nauman Ahad et al. “Active learning for time instant classification”. In: *Int. Conf. on Machine Learning Workshop on Data-centric Machine Learning (ICML-DMLR)*. 2023.

[117] Liyuan Wang et al. “A comprehensive survey of continual learning: Theory, method and application”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2024).

[118] David Lopez-Paz and Marc'Aurelio Ranzato. "Gradient episodic memory for continual learning". In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Long Beach, California, 2017.

[119] Arslan Chaudhry et al. "Continual learning in low-rank orthogonal subspaces". In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Online, 2020.

[120] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. "Task-free continual learning". In: *Proc. IEEE Conf. on Comp. Vision and Patter. Recog (CVPR)*. Long Beach, California, 2019.

[121] James Harrison et al. "Continuous meta-learning without tasks". In: *Proc. Adv. in Neural Inf. Proc. Sys. (NeurIPS)*. Online, 2020.

[122] George Casella and Roger Berger. *Statistical inference*. Vol. 2. Duxbury, 2002.

[123] Xu Zhang et al. "Heated-Up Softmax Embedding". In: *arXiv: 1809.04157* (2018).

[124] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nat. Methods* 17 (2020), pp. 261–272.

[125] Rémi Flamary et al. "POT: Python Optimal Transport". In: *J. Mach. Learn. Res. (JMLR)* 22.78 (2021), pp. 1–8.