APPLYING KERNEL CHANGE POINT DETECTION TO FINANCIAL MARKETS

Tyler Manning-Dahan

A thesis
in
The Department
of
Engineering and Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science
Concordia University
Montréal, Québec, Canada

June 2020

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Tyler Manning-Dahan**

Entitled: **Applying Kernel Change Point Detection to Financial Markets**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

______________________________ Chair

______________________________ Examiner

______________________________ Examiner

______________________________ Examiner

______________________________ Supervisor

Approved ______________________________
Chair of Department or Graduate Program Director

____________ 20 ______ ______________________________

Abstract

Applying Kernel Change Point Detection to Financial Markets

Tyler Manning-Dahan

Text of abstract.

Acknowledgments

I would like to thank my supervisor Dr. Jia Yuan Yu for accepting me into his lab and giving me the opportunity to further develop my academic career with this Master's thesis. I have become a much better researcher thanks to him and I am very grateful for his mentorship. Thank you to everyone in our research lab who read over the drafts and provided valuable feedback along the way.

Several fellow researchers have been very helpful in writing this thesis. Thank you especially to Thomas Flynn, Nicholas Kriven, and Damien Garreau who took a lot time of out their schedule to answer my questions and inform particular directions of this thesis.

I would also like to thank all my colleagues at DRW who have helped me learn about the finance industry over the past three years. I am especially grateful to Yves, Neil and Laura, who have been very patient with me and have me taught me a lot about understanding financial markets, writing clean code and building practical, statistical models.

Lastly, I would like to thank my fiancée, Tanya, who initially pushed me to go back to school while I was still young and had the opportunity.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the first half of the twentieth century, Walter Shewart pioneered the use of statistical control charts for detecting real-time changes in variation. Shewart was interested in reducing the unexpected causes of variation in the manufacturing processes that produced faulty manufacturing equipment [54]. Shewart's method involved charting the process measurements over time and detecting when a statistical process was no longer exhibiting an expected level of variation. Once this detection occurred, the process was stopped and was not restarted until the cause of the variation was fixed. Shewart's control charts were one of the first formal methods to solve the problem of detecting changes in a distribution of a sequence of random variables. This problem is now known more generally as the *change point detection problem.* Many industries make use of change point techniques for real-time decision support systems. The following are a few motivating examples.

1.1.1 Health Care

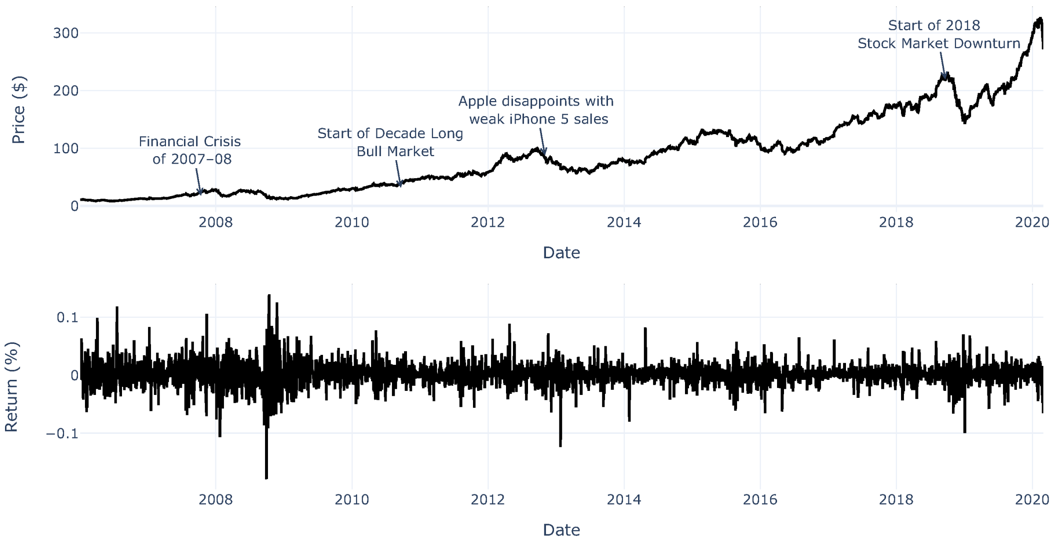
Health care is an important area for quickly detecting signal changes. Some recent studies include applications to heart rate monitoring [63] [57], epilepsy signal segmentation [41], and multi-modal MRI lesion detection [6] to name a few. Quickly detecting changes to a patient's health is absolutely necessary for any system to be of practical use. However, this quick detection must be balanced with high accuracy as false positives or missed detections could have life-threatening consequences. Therefore, balancing missed change points with falsely identified change points is a central theme to online change point detection.

1.1.2 Financial Applications

The application of accurate and timely change point detection is popular in the finance sector where shifts in asset prices can suddenly happen. Change point detection is particularly hard in financial applications because of the non-stationary data typically observed in asset price time series. A common goal is detecting key, historical moments in the market such as stock market crashes [3] or the sub-prime mortgage crisis [66]. Note, in the financial literature, change points are also referred to as structural breaks, but for this thesis we will use the broader term change points.

One example of an online, quick detection technique is proposed in [48], where a modified Shiryaev - Roberts procedure is used to detect a change point in a single stock's daily returns. See Figure 1 for the kind of of change points the authors try to detect. They compare their non-parametric method with other classic control chart methods using speed of detection and false alarm rate as measures of performance.

Figure 1: Daily stock prices (**top**) and daily returns (**bottom**) for Apple Inc. (NYSE: AAPL) for the period from January 1, 2006 through February 28, 2020.



Detecting changes in variance is explored in [33]. The authors propose an offline change point algorithm that minimizes a global cost function by using an adaptive regularization function. The algorithm is applied to the absolute returns of the FTSE 100 stock index and the US dollar-Japanese Yen foreign intra-day exchange rate to detect changes in asset price volatility. The change points identified in the FTSE 100 coincided with key market events such as the stock market crash that occurred on October 14th, 1987 and breaking the 5000 price barrier in August 1997.

For more applications to options markets and arbitrage opportunities, see section 1.3.6 of [59].

1.2 Characteristics of the change point problem

A number of surveys of the literature already exist [1] [46], therefore we will not cover all existing methods but rather touch upon several, important factors to consider when approaching a change point detection problem. Across the literature, these factors determine what methods are available to practitioners.

The first factor is selecting between *parametric* and *non-parametric* techniques. Deciding between these two broad techniques is dependent on the prior knowledge one wants to encode into the problem. For example, if it is known that data is generated by a distribution from an exponential family of distributions, then the problem can be subsetted from the space of all possible distributions to a smaller space of distributions. For example, Shewart control charts and CUSUM change point techniques are both parametric techniques based on the Gaussian-family of distributions [47] [10]. In other settings, it is not possible to leverage information about the data and non-parametric techniques must be used instead [7].

The second factor is deciding whether change points should be detected *offline* or *online*. Some algorithms are offline—also referred to as batch algorithms or retrospective or *posteriori* change point detection—and they are applied in an ex-post fashion after the dataset has been completely acquired [61]. If change points must be detected as soon as possible, then waiting for the entire dataset to be acquired is not feasible and methods that operate on data streams must be used. Methods that fall into the category are referred to as online change point detection methods. The aforementioned Shewart control chart and CUSUM algorithm are both designed for data that is streamed in real-time. In the statistical literature, online methods of change point detection are also referred to sequential change point detection [59]. For this thesis, the terms will be used interchangeably.

The third factor is determining if there are multiple change points or only one change point to detect. This is an important factor for offline change point detection where the decision to detect one or more change points is often chosen at the outset [28]. Detecting multiple change points could also be relevant for the online case if a situation arises where the window of time series under consideration may contain more than one change point. However, most online change point methods are designed to detect a single change point at a time.

Finally, the last factor to address is determining what statistical changes a change point detection algorithm should detect. Many methods focus solely on detecting changes in the mean of a distribution [35] or changes in variance [22] [27]. Methods like kernel change point detection do not focus on detecting a specific statistical parameter, but rather detecting that a change occurred in some moment of the distribution [2]. This is especially useful in situations where very little is known about the data.

This thesis will concern itself with online change point detection, where data is received in a streaming nature. We assume no prior distributional characteristics on the data and operate in a completely non-parametric setting.

1.3 Our Contributions

The contributions of this work are several fold. First, to our knowledge, no paper has explored the state of the book over time or applied change point detection techniques to order book liquidity levels. Second, given the strong emphasis of theoretical results in the change point detection community, we test several, recent online kernel change point algorithms on synthetic data and real-world data. Classification error as well as time to detection are compared across the various methods. The focus of recent algorithms is on kernel methods such as KCUSUM, NEWMA and Mstats-CPD that will be discussed in detail in Chapter 3.

1.4 Chapter Overview

Below is short description of each chapter and its contents. Ideally, each chapter should be read chronologically, but effort has been made so that each chapter is as self-contained as possible.

Chapter 2 provides a background on hypothesis testing and its relation to the change point detection problem. The online change point problem is formulated along with measures for evaluating performance. The chapter closes out by reviewing classic methods for detecting change points on streaming data.

Chapter 3 examines kernel change point detection as it is a focus of this thesis. A short background on the maximum mean discrepancy and its use in two-sample hypothesis tests is covered first. This is followed by a review of the most competitive online, kernel change point detection methods.

Chapter 4 applies our change point detection algorithm to liquidity in financial markets. A model of the limit order book is presented and the construction of the financial dataset is explained.

Chapter 5 concludes the thesis by summarizing all the results and discussing future avenues for research.

Chapter 2

Background

This chapter describes how the change point problem will be formulated in this thesis and, by extension, how all methods will be described using the change point detection problem notation. Because online change point detection is closely related to two-sample testing, a background on statistical hypothesis testing is presented first.

2.1 Hypothesis Testing

Let x and y be random variables defined on the topological space $\mathcal{X}$ with probability distributions P and Q respectively. Assume we draw n observations from P and m observations from Q resulting in two samples $X = \{x_1, x_2, ..., x_n\} \sim P$ and $Y = \{y_1, y_2, ..., y_m\} \sim Q$, where each sample is independently and identically distributed with respect to P and Q. The main question posed in this thesis is can we determine if P and Q are statistically the same or different distributions.

To answer this question, we use the *statistical hypothesis testing* framework as it is described in [8]. Generally, a *hypothesis* is a statement about a population parameter, θ. Examples of a population parameter are the population mean, variance or other higher order moments. In hypothesis testing, we trying to determine whether one of two complementary hypotheses is true. The first, denoted by H_0, is called the *null hypothesis* and it states that $\theta \in \Theta_0$ where Θ_0 is some subset of the parameter space. The second hypothesis, denoted by H_1, is called the *alternative hypothesis* and it states that $\theta \in \Theta_0^c$. For instance, if it is believed P and Q are distinguishable by their population means, $\mathbb{E}[P]$ and $\mathbb{E}[Q]$, then the possible hypotheses are:

$$\begin{cases} H_0 : \mathbb{E}[Q] \in \{\mathbb{E}[P]\} & \text{(same mean, i.e. } P = Q) \\ H_1 : \mathbb{E}[Q] \notin \{\mathbb{E}[P]\} & \text{(different mean, i.e. } P \neq Q). \end{cases} \tag{2.1}$$

How do we pick between H_0 and H_1 in equation 2.1? Every hypothesis test relies on a corresponding *test statistic* T that is a real-valued random variable. Because P and Q are unknown in our context, an estimate $\hat{T}$ can be calculated using X and Y such that $T : \mathcal{X}^n \times \mathcal{X}^m \to \mathbb{R}$, which yields:

$$\hat{T} = T(X, Y). \tag{2.2}$$

The test statistic $\hat{T}$ is then compared to some *threshold* selected by the user. There many ways to set such a threshold, but usually if the test statistic exceeds the threshold then H_0 is rejected and H_1 is accepted. Otherwise, the hypothesis test fails to reject H_0.

A common way to set the threshold is through a significance level, $\alpha \in [0, 1]$ that is chosen at the outset. Common choices for α are $0.1, 0.05$ and 0.01 [43]. The test statistic is compared to α by computing a *p-value* that is estimated by $\hat{p} = P(T \geq \hat{T}|H_0)$. The p-value is the probability of observing $\hat{T}$ under the null hypothesis. A p-value $< \alpha$ would be improbable under the null hypothesis, therefore, in this situation it is rejected and the alternative hypothesis is accepted.

Given the binary outcome of a two-sample test, it is clear the hypothesis test can fail in the two following ways. The first is rejecting the null hypothesis when it is correct. This is known as a false positive or a type-I error and is upper-bounded by the chosen significance level, α. It is equivalent to following conditional probability: $\mathbb{P}(\text{reject } H_0 | H_0 \text{ is true})$. The second possible source of error is a false negative or type-II error. The probability of committing a type-II error is denoted as $\beta = \mathbb{P}\left(\text{accept } H_0 | H_0 \text{ is false}\right)$. The quantity $1 - \beta$ is referred to as the *power* of a test. Maximizing test power is an important part of designing new algorithms and is typically used to compare different methods.

Often there is a trade-off between type-I and type-II errors and the practitioner must decide how to balance the two given their domain-specific knowledge of the problem. In some cases, it may be desirable to sacrifice one for the other. For example, in the medical field [37], a false positive diagnosis (type-I error) may be more desirable than missing a diagnosis (type-II error) which would result in never giving treatment to a patient.

Note there is not a lot of information at our disposal for constructing a non-parametric two-sample hypothesis test. Distributions P and Q are unknown, X and Y are the observed data and the other settings are chosen according to the specific hypothesis test. This means the main decision left up to the practitioner is determining what statistical test to use. In other words, what T should be used for evaluating equation 2.2? The choice should depend on how P and Q may differ from each other. For example, the Student t-test is a two-sample test for determining if samples of univariate data come from a population with the same mean [58]. A generalization of the Student t-test for the multivariate case is the Hotelling T^2 test that compares whether the means of two multivariate samples are significantly different [25]. Both of these are parametric tests as they assume the samples are normally distributed. This means, in the context of our problem where nothing is known about P and Q, they are not suitable tests for comparing two samples.

Alternatively, non-parametric tests make no assumptions about the distributions P and Q. For example, the Kolmogorov-Smirnov test (KS test) [42] can determine whether or not two univariate samples come from the same distribution. This is done by computing the *supremum* of the difference of the empirical cumulative distribution functions from each sample. The KS test does not specify what distribution the samples come from, only if they differ according to the KS statistic. More recently in [17] , the kernel two-sample test is introduced as a flexible, non-parametric test. It is not limited to one dimensional data, and can be applied to non-numeric data. It is based on the *maximum mean discrepancy* (MMD) statistic and is capable of detecting any kind of change in distribution. It is a focus in this thesis and is discussed in more detail in section 3.1.

2.2 Problem Formulation

Because there is no official standard formulation for the online change point detection problem in the literature, we use the description in [32] as the basis for the following problem description.

2.2.1 Change Point Detection Problem

Consider a data stream X to be a sequence of random variables, $X_1, X_2, X_3, ...$ where each X_t for $t = 0, 1, 2, ...$ is generated by some probability distribution P_t and each X_t is independent of the one that came before it. A change point occurs at time $t+1$ if $P_t \neq P_{t+1}$. The goal of online change point detection is detecting this change point as soon as possible.

Consider the case where we don't know when $P_t \neq P_{t+1}$ and we rely only on the observed values X_t to answer this question. We can re-frame the online change point problem to the problem of comparing two sets from X. The intuition behind this is simple. If indeed a change point occurs then any observed values after the change point should exhibit a statistical difference with observed values prior to the change point. We simply need to decide what observations should be part of each set so that a significant comparison can be made. Using this approach, a general formulation can now be made.

Let w be the number of most recent observations of X. Let this set be denoted by $X_t^w = \{X_{t-w}, ..., X_{t-1}, X_t\}$ whose length is the window, w where $w \in \mathbb{Z}^+$. This recent set will be compared to some reference set of X denoted by $X_t^{ref} = \{X_0, X_1, ..., X_{t-w}\}$ whose distributions $P_0 = P_1 = ... = P_{t-w}$ are sampled from a common distribution P. What is this reference set? It is data that is known to be distributed consistently prior to a change point. In the process control literature, it is known as the in-control distribution [5]. We will use these terms interchangeably as they capture the same idea. We do not explicitly state how many observations are stored for X_t^{ref} because this depends on the change point detection method used.

Let θ_t be the parameter we expect to change in our data stream at a particular time. The most recent parameter θ_t^w will be compared to the set reference parameters, $\{\theta_0, \theta_1, ..., \theta_{t-w}\}$, to determine if a change point has occurred. In other words, at each time, t, the online change point detection problem is performing the following hypothesis test:

$$\begin{cases} H_0 : \theta_t^w \in \{\theta_0, \theta_1, ..., \theta_{t-w}\} & \text{(no change point occured)} \\ H_1 : \theta_t^w \notin \{\theta_0, \theta_1, ..., \theta_{t-w}\} & \text{(a change point occured)} \end{cases} \tag{2.3}$$

If the null hypothesis is true then the streaming data is distributed consistently along P and no change point exists. If the null hypothesis is rejected, the time series may be partitioned by a change point, t^*, that signifies all data from $t \geq t^*$ are distributed differently than data from $t < t^*$. Because we are operating in a non-parametric setting, the parameters used for comparison are estimated using the streamed data and aren't assumed to follow any particular family of distributions.

Many change point detection algorithms define a statistic that is computed using each set before and after the possible change point [1]. If the statistic is above a threshold, $h \in \mathbb{R}$, then time t is classified as a change point. The estimated change point time is denoted as $\hat{t}$ and can be compared to underlying true change point, t^*, if it is known.

The size of the window, w, is an important consideration that is typically chosen based on the problem being solved. A small window will detect change points more frequently resulting in fast change point detection but at the cost of more incorrect detections. A large window will have the opposite problem. The change point method will have less incorrect detections but will be slower to detect real change points.

2.2.2 Performance Measures

An important crux of change point detection is how to evaluate an algorithm. Real life streaming data does not always announce when its distribution has changed even when clearly it has. In some cases, it may be possible to trace a change in variation back to a deterministic machine that is no longer operating as expected, but in other cases this may be impossible. For this reason most past research has focused on theoretical results rather than empirical ones. In the cases where empirical data is used, it is usually generated synthetically. This allows the practitioner to control exactly where the distribution changes and compare a proposed method to ground truth change points. Therefore, the following measures of performance will be presented in the context of synthetic data when all change points are known.

Suppose we continuously generate synthetic data according to some distribution and some point in time, t_1^*, we generate the data from a different distribution. If we repeat this process K^* times then we can let the set of true change points be denoted by $\mathcal{T}^* = \{t_1^*, t_2^*, ..., t_{K^*}^*\}$ whose size is $|\mathcal{T}^*| = K^*$. If this particular dataset is fed into an online change point detection algorithm in a streaming fashion then the algorithm can generate an estimated set of change points. Let this set of estimated change points be denoted by $\hat{\mathcal{T}} = \{\hat{t}_1, \hat{t}_2, ..., \hat{t}_{\hat{K}}\}$ whose size is $|\hat{\mathcal{T}}| = \hat{K}$. Note, $\hat{K}$ does not necessarily equal K^*.

Armed with both sets of change points, we can now assess performance of an algorithm. A change point is considered detected if it occurs within a user-defined margin of error $M > 0$. That is, if $\hat{t}$ is detected within M samples of t^* then it is an estimated change point. Therefore, the set of true positives TP can be written as,

$$\mathrm{TP}(\mathcal{T}^*, \hat{\mathcal{T}}) = \{t^* \in \mathcal{T}^* | \exists \hat{t} \in \hat{\mathcal{T}} \text{ s.t. } 0 < \hat{t} - t^* \leq M\}. \tag{2.4}$$

Notice by this definition, an estimated change point occurring just prior to a real one is a missed detection in online detection. This is in stark contrast to an offline case that is more concerned with segmentation a time series and being as close as possible to an actual change point is sufficient. This is unrealistic in the online scenario.

If an estimated change point falls outside the margin, then it is deemed a false positive. These are typically called *false alarms* or false detections in the change point detection literature [35]. The set of false alarms can be written as:

$$\mathrm{FA}(\mathcal{T}^*, \hat{\mathcal{T}}) = \{\hat{t} \in \hat{\mathcal{T}} | \nexists t^* \in \mathcal{T}^* \text{ s.t. } 0 < \hat{t} - t^* \leq M\}. \tag{2.5}$$

Missed change points are referred to as missed detections and they occur when no estimated change point is flagged following an actual change point. This is equivalent to a false negative in machine learning classification. Therefore the set of missed detections are:

$$\mathrm{MD}(\mathcal{T}^*) = \{t^* \in \mathcal{T}^* \text{ s.t. } t^* \notin \mathrm{TP}\}. \tag{2.6}$$

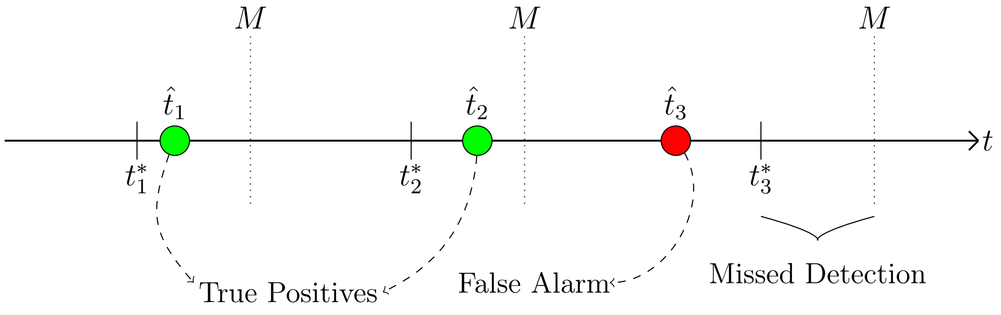
In figure 2, the relationship between true positives, false alarms and missed detections is shown.

Using the set of true positives, we can also use *precision* and *recall* to evaluate the performance. Precision is the proportion of the predicted change points that are true change points. Recall is the proportion of the true change points that are well predicted. This gives:

$$\mathrm{PREC}(\mathcal{T}^*, \hat{\mathcal{T}}) = \frac{|\,\mathrm{TP}(\mathcal{T}^*, \hat{\mathcal{T}})|}{\hat{K}} \quad (2.7) \qquad \mathrm{REC}(\mathcal{T}^*, \hat{\mathcal{T}}) = \frac{|\,\mathrm{TP}(\mathcal{T}^*, \hat{\mathcal{T}})|}{K^*}. \quad (2.8)$$

Precision and recall are well-defined between $(0, 1)$ if the margin M is smaller than the spacing between two true change points. This is an important consideration that will be useful for experiments conducted in section 4.1.

Figure 2: An example of actual change points t_1^*, t_2^*, t_3^* and estimated change points $\hat{t}_1, \hat{t}_2, \hat{t}_3$. The first two estimated change points in green are true positives as they are detected within the margin M. The third estimated change point in red is not within the margin making it a false alarm. Finally the last actual change point does not have any estimated change points after it making it a missed detection.



While all the above metrics are useful, they are not sufficient on their own. Suppose we have two online change point methods, one is capable of consistently detecting change points one time-step after an actual change point and the second method is capable of consistently detecting change points 20 time-steps after an actual change point. Assuming the margin of error in this case very large, is there anyway to discern between the two methods using the above evaluation metrics? The answer is no as long $20 < M$. This is a problem because clearly one method is better at capturing changes in a time series quickly. Simply counting the number of false alarms or missed detections does not capture the goal of the online change point detection problem, which is to quickly detect change points over time, while minimizing false alarms.

The second issue is not detecting a change point when one occurs, called a *missed detection*. This could be caused by detecting a change point much too late for it to be of any use or simply missing it altogether. In both cases, a running total of false alarms and missed detections is monitored for performance.

In the online change point detection literature, there are two common ways to incorporate the notion of delay when flagging change points. The first is a metric known as the *average time to false alarm* (TTFA) and it is defined as the expected amount of time that must pass before observing a false alarm is raised. It is computed as:

$$\text{TTFA} = \mathbb{E}[\ \hat{t} \mid t^* \text{ does not exist}]. \tag{2.9}$$

Clearly, a larger value of TTFA is preferable. In the literature this measure is also referred to average run length (ARL) or average run length under the null (ARL_0) [29].

Alternatively, the second method of evaluating the delay of estimated change points is by measuring the average amount of time that passes before a true change point is detected. This is known as *expected detection delay* (EDD) and is defined as

$$\text{EDD} = \mathbb{E}[\ \hat{t} \mid t^* \text{ does exists}]. \tag{2.10}$$

Because we will use EDD to assess performance in later chapters, it can be estimated by:

$$\widehat{\text{EDD}} = \sum_{i=1}^{|\,\text{TP}\,|} \sum_{j=1}^{\hat{K}} \frac{\hat{t}_j - t_i^*}{|\,\text{TP}\,|} \qquad \forall t^* \in \text{TP}, \forall \hat{t} \in \hat{\mathcal{T}} \text{ s.t.} \tag{2.11}$$

Both TTFA and EDD are two sides of the same coin. Reducing the EDD also reduces the TTFA, which is good for the former and bad for the latter. The opposite is also true. Increasing TTFA, also increases the EDD. Therefore, minimizing EDD or maximizing TTFA depends entirely on the domain-specific application, where the practitioner may want to sacrifice one for the other.

2.3 Classic Algorithms

Presented below are the fundamental approaches to online change point detection that have been very influential on modern approaches. Many modern algorithms are variants of the classic algorithms discussed below. In the following sections let X_t be defined as it is in 2.2.1.

2.3.1 Shewart Control Chart

Shewart control charts were originally designed to detect changes in the mean of a process where the values being observed are assumed to be Gaussian distributed [54]. As the data arrives, the data is batched into samples of size N. The sample mean, $\bar{X} = \frac{1}{N}\sum_{t=1}^{N} X_t$, is the then calculated and compared it to a known, true mean μ^*.

Similarly, it is assumed the standard deviation, σ, is known in advance but it can also be estimated. If the absolute difference is greater than a threshold, then a change point is declared at the current batch. Therefore, the decision rule is defined as,

$$|\bar{X} - \mu^*| > \kappa \frac{\sigma}{\sqrt{N}}, \tag{2.12}$$

where $\kappa \in \mathbb{R}$ is a constant that controls how sensitive the algorithm is. Typically, it is set to $\kappa = 3$ as this coincides with the observations within 3 standard deviations of the mean. Under the assumption that the data is distributed normally, approximately 99.7% of the observations are distributed in this region, therefore a change point is declared if it falls outside this region. The true mean is assumed to known and is defined as $\mu^* = \mathbb{E}[X_t]$. In applications, the true mean can also be replaced by some target specification that a process must adhere to.

Tuning the hyper-parameters can drastically change the performance of the algorithm. The parameter κ is used to control the trade-off between false alarms and missed detections. Choosing a lower κ makes the control chart detect change points more often and, consequently, increases the false alarms. Whereas a higher κ results in less false detections but also more missed detections. The chosen sample size, N, is also critical and its effect on the performance of Shewart control charts is studied in [21].

2.3.2 CUSUM

Similar to the Shewart control chart, the CUSUM algorithm tracks a statistic over time relative to a predetermined threshold [47]. CUSUM is best applied to a process that is already under control. It can be thought of accumulating the information of current and past samples.

The algorithm is defined by a statistic, $S_t \in \mathbb{R}$, that is recursively updated after each sample, X_t, is observed, such that:

$$\begin{cases} S_0 = 0 & \text{(Initialization)} \\ S_t = \max(0, S_{t-1} + Z_t) & \text{for t=1,2,...,} \end{cases} \tag{2.13}$$

where $Z_t = ln(\frac{f_1(X_i)}{f_0(X_i)})$ and the statistic S_t is compared to a threshold $h \in \mathbb{R}$ that is predetermined by the user. Functions f_0 and f_1 are probability density functions for the pre-change distribution and post-change distribution respectively. If $S_t \geq h$ then a change point is declared at time t and the algorithm is either stopped or restarted. Given that the statistic only flags change points when greater than a threshold, this algorithm only detects positive changes in the distribution. In [47], it is suggested to combine two CUSUM algorithms to detect positive and negative changes in a distributional parameter.

As a parametric algorithm, it is assumed f_0 and f_1 are known at the outset. In most applications, this is quite limiting and unrealistic. Therefore, in cases where they are unknown, *maximum likelihood estimates* of each distribution's parameters are usually computed. See [30] for details on maximum likelihood estimation.

Several extensions to the CUSUM algorithm have been proposed such as the filtered-derivative extension introduced in [4], which uses the change of the discrete derivative of a signal over time to detect a change point. In [40], a fast initial response (FIR) CUSUM algorithm is proposed where the starting value of initial cumulative sums adapts over time. Instead of resetting S_0 to zero as shown above, it is reset to a non-zero value, typically based on the threshold chosen. This gives the algorithm a head-start in quickly detecting when a process is out of control and is especially useful for processes that don't start in control.

Finally, since CUSUM is typically better at detecting small shifts in signals and the Shewart control chart is faster at detecting larger changes, the two can be combined [39]. The combined Shewart-CUSUM algorithm leverages the strengths of both techniques for better overall performance. See [64] and [62] for more details.

2.3.3 EWMA

First described in [51] as a *geometric average*, the exponentially weighted moving average (EWMA) is a type of moving average that applies exponential weighting to time series samples. Initially used as a forecasting technique in the econometrics field for smoothing noisy functions, the EWMA can also be used for determining out of control processes as shown in [26]. Rather than weighing all observations uniformly like the standard CUSUM algorithm or a simple moving average, a decay factor (also called a forgetting factor), $\lambda \in [0, 1]$, is used to control how much weight is distributed over the previous observations. As each new observation arrives, the EWMA statistic, $E_t \in \mathbb{R}$, is recursively updated and compared to a threshold. If the EWMA statistic exceeds the threshold then the process is deemed out of control or, in other words, a change point is detected.

The EWMA statistic is calculated as follows at each time step, t:

$$E_t = \lambda X_t + (1 - \lambda)E_{t-1}.$$

As $\lambda \to 1$, the EWMA gives more and more weight to the most recent observations similar to a Shewhart control chart, which gives weight to the last observation only. Conversely, as $\lambda \to 0$, the weights are distributed further into the past giving the EWMA a longer memory similar to the CUSUM algorithm. Therefore, a EWMA control chart can be interpreted as a trade-off between a Shewhart control chart and a CUSUM control chart.

For detecting deviations away from a mean target value, control limits may be calculated in a similar manner to the Shewart control chart. In [26], control limits for the EWMA are chosen to be $\pm 3\sigma\sqrt{\frac{\lambda}{2-\lambda}}$. Like the Shewart control chart, it is assumed the standard deviation, σ, is known in advance but it can be estimated if it is not known.

As with the other methods previously mentioned, the standard EWMA is a parametric method as it assumes the time series has some in-control average that is known prior to use. This makes is difficult to apply in situations where the data is coming from unknown distributions. It is however very fast due to its recursive structure and does not hold a lot of data in memory making it appealing for live data streams that need fast data processing.

Chapter 3

Kernel Change Point Detection

In this chapter, we focus on the class of change point models that leverage kernel techniques, known as *kernel change point detection.* A unique feature of kernel methods is the use of the maximum mean discrepancy (MMD) metric as the change point test statistic. Therefore, a short review of the MMD statistic is presented first.

3.1 Kernel-Based Test Statistic

Recall we are looking for a suitable statistic to solve the change point hypothesis test presented in 2.3. To solve this, we need a statistic that can measure the difference between two distributions, P and Q. In our setting, this is particularly hard because we do not know P or Q and only have access to data generated by them. Ideally, the metric measuring the difference should return a small value if $P \approx Q$ and large value if P and Q are very different from each other. This means we need a metric that can compute the distance between two probability distributions using only data drawn from them. It turns out the maximum mean discrepancy (MMD) introduced in [17] meets all these criteria.

The idea behind the MMD is the following: rather than comparing the expected value of P and Q in their original feature space, why not compare their expected value in a different feature space that makes discerning between P and Q more effective. Doing so would allow the detection of differences between P and Q that were not possible before. A classic technique for comparing samples in a different feature space is through the use of kernel functions. Therefore, we'll start with a brief background of kernel functions.

3.1.1 Kernel Background

Suppose we have a non-empty set $\mathcal{X}$ for which $x, y \in \mathcal{X}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *kernel* if there exists a *feature space* $\mathcal{H}$ with a corresponding *feature map* $\phi : \mathcal{X} \to \mathcal{H}$. A kernel can be expressed as,

$$k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \tag{3.1}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is the *inner product* in $\mathcal{H}$ [53]. Using the kernel function, we can feed it two points and implicitly compute the dot product of these points in the transformed space. This is the so-called *kernel trick*. It's a trick because the dot product can be computed without having to explicitly know the feature map or what the coordinates of $\phi(x)$ or $\phi(y)$ evaluate to. Lastly, because dot products are symmetric, this implies that the kernel function as defined above is also symmetric, i.e. $k(x, y) = k(y, x)$.

A related concept is the *Gram matrix* which can be computed from a set of vectors $\{\mathbf{x}_1, .., \mathbf{x}_\ell\}$ as an $\ell \times \ell$ matrix $\mathbf{G}$, whose entries are $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Using 3.1, we can define a *kernel gram matrix* or just *kernel matrix*, denoted by $\mathbf{K}$, as:

$$\mathbf{K}_{ij} := k(\mathbf{x}_i, \mathbf{x}_j). \tag{3.2}$$

By symmetry of the kernel, the kernel matrix is also symmetric $\mathbf{K}_{ij} = \mathbf{K}_{ji}$ or $\mathbf{K}^{\mathrm{T}} = \mathbf{K}$. More explicitly, it is expressed as:

$$\begin{array}{lllll}
\mathbf{K} & 1 & 2 & \cdots & \ell \\
\hline
1 & k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_\ell) \\
2 & k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_\ell) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\ell & k(\mathbf{x}_\ell, \mathbf{x}_1) & k(\mathbf{x}_\ell, \mathbf{x}_2) & \cdots & k(\mathbf{x}_\ell, \mathbf{x}_\ell)
\end{array} \tag{3.3}$$

In most applications, k is chosen to be positive definite. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel if it is symmetric and

$$\sum_{i,j=1}^{n} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0, \tag{3.4}$$

for any $n \in \mathbb{N}$, any choice of $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathcal{X}$ and any $c_1, ..., c_n \in \mathbb{R}$. This is equivalent to requiring that the constructed kernel matrix has entirely positive eigenvalues. Furthermore, there always exists $\phi : \mathcal{X} \to \mathcal{H}$ for which equation 3.1 holds if k is positive definite.

By selecting a positive definite kernel, the feature space $\mathcal{H}$ is actually a space of functions known as a *reproducing kernel Hilbert space* (RKHS). While this may seem odd, it actually simplifies a lot of algebra because elements in a RKHS can be treated like finite vectors. For the purposes of this thesis, we will only deal with positive definite kernels to ensure the corresponding feature space is a RKHS.

3.1.2 Kernel Mean Embedding

While kernel functions are useful for comparing vectors in a new feature space, this is not quite what we need because we are looking for a way to compare two probability distributions. In the above setting, ϕ accepts points or vectors as inputs, but this can be generalized to random variables in the following way: Let X be a random variable with domain Ω and distribution $P(X)$. We refer to instantiations of X as a lower case x. In [55], it is shown that ϕ can be used to represent a probability distribution as an element in a RKHS by

$$\mu_X = \mathop{\mathbb{E}}_{X \sim P}[\phi(X)], \tag{3.5}$$

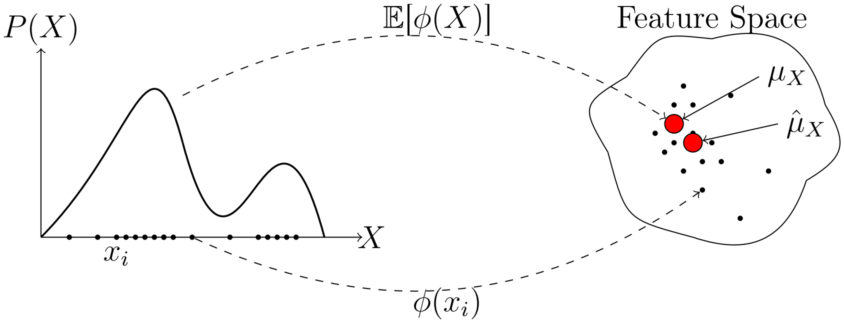
which is known as the *kernel mean embedding* of P. This can be interpreted as a distribution being mapped to its expected feature map in $\mathcal{H}$.

Because we are in a non-parametric setting, we cannot assume what $P(X)$ is. However, we can estimate the embedding using a finite sample average [55]. Suppose we have access to a sample $\{x_1, x_2, ..., x_m\}$ where x_i is drawn i.i.d from $P(X)$, the empirical kernel mean embedding is

$$\hat{\mu}_X = \frac{1}{m} \sum_{i=1}^{m} \phi(x_i). \tag{3.6}$$

See Figure 3 for a better understanding of the relationship between the distribution, the instances of the random variable and their embeddings into the new feature space.

Figure 3: Stylized depiction of a kernel mean embedding of $P(X)$ to a RKHS as μ_X. The empirical estimate, $\hat{\mu}_X$, is shown to be close in $\mathcal{H}$ to its theoretical counterpart.



Why bother mapping P to a RKHS? Because we can now compare two probability distributions in $\mathcal{H}$ as if they were vectors in an infinite-dimensional space. This means we can use vector operations such as dot products or norms with kernel mean embeddings.

While this new feature space representation is helpful, it is not necessarily unique. This a problem because P and Q may be mapped into a space where they are measurably similar when in fact they are not. Consequently, we need to ensure the kernel mapping is *injective*, i.e. that the mapping from $\mathcal{X}$ to $\mathcal{H}$ is one to one.

To ensure the kernel mapping is injective, we can make use of a special class of kernel functions known as *characteristic kernels* that are introduced in [14] and [56]. These papers show that if k is characteristic then the mapping of ϕ into the $\mathcal{H}$ is unique. This means if a characteristic kernel is used, then X and Y can be uniquely mapped to another feature space where the expected value in $\mathcal{H}$ is also unique.

With this last piece of information we can now map two different distributions into a new feature space and compute a distance-like metric between the two, which is the basis of the maximum mean discrepancy.

3.1.3 Maximum Mean Discrepancy

Suppose m observations from a sample $\{x_1, x_2, ..., x_m\}$ and n observations from a different sample $\{y_1, y_2, ..., y_n\}$ are derived from $\mathcal{X}$. Assume the observations from each sample are instantiations of random variables $X \sim P$ and $Y \sim Q$ respectively. Using ϕ as it is described in 3.1, the maximum mean discrepancy (MMD) is defined in [55] as:

$$\mathrm{MMD}(P,Q) = ||\mu_X - \mu_Y|| = ||\underset{X\sim P}{\mathbb{E}}[\phi(X)] - \underset{Y\sim Q}{\mathbb{E}}[\phi(Y)]||_{\mathcal{H}}, \tag{3.7}$$

where $||\cdot||_{\mathcal{H}}$ is the *norm* on $\mathcal{H}$. The MMD can be interpreted as the distance in $\mathcal{H}$ between the kernel mean embeddings of the features. As long as a kernel function can be defined on the given data structure, the MMD can be used. This is why it can be applied to non-numeric data such as strings, graphs, and other structured domains [24]. From theorem 5 of [17], the theoretical $\mathrm{MMD}(P,Q) = 0$ if and only if $P = Q$. This fact arises from the injective property defined earlier. While this is a nice property, in practice we use empirical estimates of the kernel mean embeddings, $\hat{\mu}_X$ and $\hat{\mu}_Y$, meaning an estimate of MMD will never be exactly zero even if our samples are from the same distribution. Nonetheless, the power of MMD lies in its ease of estimation and the fact that it can detect any type of parameter difference between P and Q.

Using X and Y, the unbiased estimate of the squared MMD is shown in [17] to be:

$$\begin{split}\widehat{\mathrm{MMD}}_u^2(X,Y) = \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j=1}^{m} k\left(x_i,x_j\right) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k\left(x_i,y_j\right) + \\ \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n} k\left(y_i,y_j\right).\end{split} \tag{3.8}$$

One caveat of equation 3.8 is that while it is an unbiased estimate of the square of 3.7, the calculation of $\sqrt{\widehat{\mathrm{MMD}}_u^2}$ is not an unbiased estimate of 3.7.

Similarly, the biased estimate of the squared MMD is

$$\widehat{\mathrm{MMD}}_b^2(X,Y) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k\left(x_i,x_j\right) + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} k\left(y_i,y_j\right) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k\left(x_i,y_j\right). \tag{3.9}$$

Because both 3.8 and 3.9 are computed in $\mathcal{O}((m+n)^2)$ time, they will be referred to as quadratic-time estimates of MMD. For high-dimensional datasets (large m or n), using a quadratic-time estimate of MMD is impractical.

An alternative is proposed in section 6 of [17] that is linear in computation time and still uses all available data. Letting X and Y have equal size m for notational simplicity, then

$$\widehat{\mathrm{MMD}}_l^2(X,Y) := \frac{1}{m_2}\sum_{i=1}^{m_2} h\left(\left(x_{2i-1},y_{2i-1}\right),\left(x_{2i},y_{2i}\right)\right) \tag{3.10}$$

is the linear-time estimate of MMD, where $m_2 = \frac{m}{2}$ and

$$h\left((x_i, x_j), (y_i, y_j)\right) := k\left(x_i, x_j\right) + k\left(y_i, y_j\right) - k\left(x_i, y_j\right) - k\left(x_j, y_i\right). \tag{3.11}$$

This estimation of the squared MMD is computed in $\mathcal{O}(m)$, which is significantly faster than the quadratic-time estimates. However, it pays for this speed with accuracy as the MMD_l estimate has higher variance than MMD_u.

Now that we have a suitable metric for computing a test statistic between samples X and Y, a hypothesis test can be constructed using the MMD statistic as shown in [17]. Similar to previous hypothesis tests, a threshold and kernel function is chosen to adequately balance between type-I and type-II error. The null hypothesis is that X and Y are equivalent in their kernel mean elements. If the MMD is greater than the threshold, then the null hypothesis is rejected and the alternative hypothesis is accepted. In this case, the alternative hypothesis would indicate X and Y differ in one or more of their statistical moments.

3.1.4 Choice of Kernel

Notice we have not yet explicitly chosen a kernel function for calculating the MMD statistic. Choosing an appropriate kernel is often done on a per case basis based on the type of application. Additionally, optimal kernel selection is a difficult problem that is still actively researched [13] [19]. We will only discuss one kernel function in this thesis, the Gaussian kernel. The Gaussian kernel is the most widely used kernel and it is the only kernel used in the kernel change point detection methods discussed in section 3.2. For other possible choices of kernel functions see table 3.1 in [45].

Given two d dimensional real vectors $x, y \in \mathbb{R}^d$, the Gaussian kernel is defined as:

$$k(x, y) = e^{-\frac{1}{2\sigma^2}||x-y||^2} \tag{3.12}$$

where $\sigma > 0$ and is called the *kernel bandwidth*. In [18], the authors recommend selecting the bandwidth based on the *median heuristic* using a sample of the data. The median is computed on the set of pairwise distances between all data points in the sample. Therefore, σ is computed by,

$$\sigma = \text{median}\{||x_i - x_j|| : i, j = 1, ..., n\}. \tag{3.13}$$

It is not clear where this heuristic was first introduced although many papers incorrectly cite the 2002 textbook, *Learning with kernels: support vector machines, regularization, optimization, and beyond* by Schölkopf, Smola, Bach et al [15]. Despite its unknown origins, it has been shown to be empirically suitable for many kernel applications. It seems as long as the bandwidth is chosen $\mathcal{O}(\sqrt{d})$, which the median heuristic often achieves, then test power is independent of the bandwidth. However, it has been shown to be sub-optimal in cases with very high-dimensions [44] or small sample size [50]. Because the median heuristic is critical to overall performance of the kernel two-sample test using the MMD, its online implementation will be explored in greater detail in section 3.3.

3.2 Related Work

3.2.1 Offline Kernel Change Point Detection

While offline change point detection techniques are not the focus of this thesis, we provide a short summary of some kernel approaches in this domain to highlight some important, recent work. Similar to online kernel methods, offline kernel methods are employed non-parametrically on data to provide more robust measures in situations where assumptions cannot be made about the data's distribution. Offline detection has the advantage of all data is fully available from the start, so the key problem is where to segment the time series so that the true and estimated change points align as closely as possible. In other words, predicting too many change points (over-segmentation) must be balanced with predicting too little change points (under-segmentation).

In [20], the authors approached the offline change point problem with a fixed number of change points using kernel change point detection. This was further extended to an unknown number of change points in [2]. The authors show their kernel change point detection procedure outputs an offline segmentation near optimal with high probability. The authors recommend choosing the kernel based on best possible signal to noise ratio. Therefore, they rely on prior knowledge of a reference or training set is necessary for calibrating the kernel.

More recently in [9], a kernel change point detection method is proposed that uses deep generative models to augment the test power of the kernel two sample test statistic from equation 3.7. They point out MMD's lack of test power when using small samples from the out of control distribution, which may easily leading to overfitting with kernels. Thus, they use a generative adversarial neural network (GAN), trained on historical samples of $X \sim P$ with noise injected into X. This surrogate distribution is then used in conjunction with possible change points to improve the test power of a modified MMD measure that makes use of compositional kernels. The method is compared to other prominent change point methods for offline change detection. All comparisons are done on synthetic data with piece-wise i.i.d. data. All methods are benchmarked using the AUC metric for classification performance and it is shown the KL-CPD method is competitive or better than the state of the art methods. Furthermore, the AUC performance is maintained as the dimensionality of the data is increased, making their kernel learning framework very interesting for future offline change point detection. It remains to be seen if this framework can be adopted in an online context where time to detection is a key constraint on practicality.

All sources cited above make use of the kernel mean embedding machinery presented in section 3.1.2. It is also interesting to note that all offline kernel change point methods cited above make use of the Gaussian kernel, which speaks to the ubiquitousness of this kernel function. See section 4.2.3 in [61] for a review of kernel-based methods used for offline detection.

3.2.2 Online Kernel Change Point Detection

One of the first papers to use the term kernel change point detection was in [11]. The authors present an online kernel change point detection model based on single class support vector machines (ν-SVMs). They train two single class support vectors, one on a past set and one on a future set. A ratio is then computed between the two sets that acts as the dissimilarity measure in a Hilbert space. If the sets are sufficiently dissimilar over some predetermined threshold, then a change point is assigned to the time step that splits the two sets of data. The authors argue that a dissimilarity measure between kernel mappings in a Hilbert space should estimate the *density supports* rather than estimate the probability distributions of each set of points. While this approach inspired a lot of interesting research that will be discussed below, it does not use the maximum mean discrepancy and has not been studied since.

The following methods are all inspired by the classic algorithms from section 2.3. They all make use of a variation of the maximum mean discrepancy. They are also the most recent online kernel techniques developed and will be used for experiments in later sections. Thus, they will be described in more detail than previous methods.

In [36], the authors use the B-test introduced in [65] and develop an offline and online change point detection algorithm called the Scan-B[1] algorithm. The B-test estimates the MMD by applying the MMD_u calculation from equation 3.8 on blocks of data of size B and averages them. It is meant to be a compromise between the linear-time estimation MMD_l and the quadratic-time estimation MMD_u.

At each time-step t, the online Scan-B algorithm samples new data from a window of size B_0 and computes a B-test statistic with N past samples that are kept as reference samples.

$$Z_{B_0,t} := \frac{1}{N}\sum_{i=1}^{N} \text{MMD}_u^2 \left(X_i^{(B_0,t)}, Y^{(B_0,t)} \right) \tag{3.14}$$

The resulting test statistic is then normalized by $Z_{B_0,t}/\sqrt{Var[Z_{B_0}]}$ where the authors provide a theoretical calculation of $\text{Var}[Z_B]$. If the normalized test statistic exceeds some predefined threshold then a change point is declared. Because the statistic is calculated each time and only the value of the last calculation has any weight, it is essentially memoryless. This is similar to a Shewart control chart that calculates a z-score at each iteration. Adjusting the size of the window, B_0, results in the usual trade-off of performance in online change point detection. A smaller block size will have a smaller computational cost and a smaller detection delay but will result in higher type II error. A larger block size will have better type II error but will take longer to compute. Unfortunately, no matter what block size is chosen, the computation time for the Scan-B algorithm is the longest relative to the other kernel change point methods discussed in this section.

From here, theoretical bounds are developed for the time to false alarm and expected detection delay. They run several experiments on synthetic data including change in mean, change in variance, change from Gaussian to Gaussian mixture, and change from Gaussian to Laplace distribution. Experiments are also done on real-data sets including a speech dataset and the Human Activity Sensing Consortium

[1] This algorithm was originally called MStats but was subsequently re-named in a later version.

(HASC) dataset where the performance was better than the relative density-ratio (RDR) algorithm described in [38].

A modified, "no-prior-knowledge" exponentially-weighted moving average called NEWMA is introduced in [31]. Based on the standard exponentially weighted moving average shown in 2.3.3, NEWMA computes two EWMA statistics of different weights. If the difference between the two EWMA statistics exceeds a predefined threshold then a changepoint is declared at that time step. The reason for using two EWMA statistics is to set one to have a larger forgetting factor. Any recent changes in a distribution will weigh heavily on one statistic, resulting in a sudden, large difference between the two statistics.

Since a standard EWMA is a parametric method, the authors apply a kernel mapping function, Ψ, to the data prior to applying the exponential weights. This provides a memoryless, non-parametric, online change point detection method that does not need to constantly store all previously streamed data. Once the statistics are updated at each iteration, the raw data may be discarded. This characteristic makes it especially useful in applications where security is a concern.

While kernel mean embeddings could be used for approximating, Ψ, as is the case for standard implementations of MMD, this would require the storage of past examples of data. Because the authors aim to reduce run-time cost and storage cost, they use a *random fourier features* (RFF) approach for estimating Ψ. Because Ψ is estimated using RFF, this is the only method discussed that does not use the kernel function for an implicit representation of the data in another feature. Instead, the RFF explicitly maps the data to a lower dimension Euclidean product space using a randomized feature map $\mathbf{z} : \mathbb{R}^d \to \mathbb{R}^m$ such that:

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}} \approx \mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{y}) \tag{3.15}$$

where $\mathbf{z}(\mathbf{x}) := \mathbf{W}^\top \mathbf{x}$. Each element, w_{ij}, is sampled from a distribution that is the Fourier transform of a translation invariant kernel [49].

There are several approaches available for calculating RFF from the literature and the authors use three common ones for comparison. They use the standard RFF implementation from [49], the FastFood implementation introduced in [34], and Optical Processing Unit implementation from [52]. All three are used to create three variants of their NEWMA algorithm.

The NEWMA variants are compared to the Scan-B algorithm by running empirical experiments on synthetic and real datasets. The synthetic datasets use streaming data that is generated from different Gaussian mixture models. They use an audio dataset for testing on real data. The variants of NEWMA are similar, if not better than Scan-B in terms of missed detection percentage. In terms of average detection delay and false alarm trade-off, the NEWMA algorithm and its variants appear to be mildly better as well. The largest advantage of the NEWMA variants over the Scan-B method is in the execution time. Scan-B's execution scales linearly with window size, while NEWMA's execution time does not depend on window size.

Finally, in a recent, preprint paper [12], a kernel CUSUM (KCUSUM) algorithm is proposed, where the classic CUSUM algorithm from 2.3.2 is adapted using the faster MMD_l statistic for online detection. The algorithm functions as follows, every two observations, the MMD_l is calculated using newly observed data points and data points sampled from some reference distribution that is known at the outset. The calculated MMD_l acts as the update term to the cumulative sum statistic. If this kernel cumulative sum statistic exceeds some predefined threshold, then a change point is flagged. Interestingly, unlike the previous methods that set the Gaussian kernel's bandwidth using the median heuristic, the authors chose to fix the bandwidth to one. Furthermore, because it relies on a reference distribution that is outside the data stream for comparison, KCUSUM is more of a semi-supervised algorithm rather than a completely unsupervised one.

Besides its speed of computation, an additional benefit of MMD_l is that it is normally distributed under the null distribution unlike the quadratic-time estimate. This facilitates analysis and provides statistical guarantees for worst-case detection delays and the expected time to false alarm. While this non-parametric approach can detect any change in the distribution of a sequence, it does struggle with more complicated distributional changes such as variance changes of a single dimension and changes beyond first and second-order moments. It was also not benchmarked against other kernel methods, which we cover in this thesis for completeness.

Given how important the bandwidth hyperparameter is for the Gaussian kernel function, it is noteworthy then to see how the above online change point models tune this hyperparameter. In the context of an infinite data stream, most methods simply take a small initial sample and use that to set the bandwidth once at the beginning and never change it again, like the Scan-B algorithm. But how long is an initial bandwidth valid in a continuous data stream? If the data changes significantly in orders of magnitude then this bandwidth selection will become stale and will have to be re-computed. One option is to set it to a fixed value rather than use the median heuristic like KCUSUM does. However, this is somewhat arbitrary and may lead to poor performance in certain cases. Another possibility would be to simply run an expanding median on all incoming data. But this would require storing all the observations ever observed which is obviously intractable. To our knowledge, these issues surrounding online bandwidth selection have not been addressed anywhere in the kernel change point detection literature. This is why the topic of successive median calculation is a focus in section 3.3.

3.3 Our Approach

As stated in the related work, all kernel change point detection methods that use a Gaussian kernel use the median heuristic to determine the bandwidth. While this method has proved to be sufficient for most applications, it has not been thoroughly tested against any alternative bandwidth selection procedures. This is ironic given how much impact the choice of bandwidth can have on performance. We present a natural alternative for calculating the median heuristic in real-time for online change point detection. This method is based on the Binapprox algorithm shown below [60].

Algorithm 1: Binapprox algorithm

```
Input: $x_i, .., x_n, B$
Output: Approx. median
Calculate $\bar{x}$ and $s_x$
Form $B$ bins across $[\bar{x} - s_x, \bar{x} + s_x]$
Map $x_i, .., x_n$ to bins
Find the bin $b$ that contains the median
Return the midpoint of bin b
```

Several characteristics make Binapprox particularly appealing for online, non-parametric data streaming. Firstly, the runtime of Binapprox does not depend on a data's distribution. Secondly, the algorithm requires $\mathcal{O}(1)$ storage space, and doesn't perturb the input data, so rearranging is never a problem. Thirdly, in the worst-case scenario the algorithm has $\mathcal{O}(n)$ computational complexity. In practice, it consistently runs faster than the classic quickselect algorithm and `Binmedian`, a more exhaustive algorithm that is presented in the same paper. Lastly, it can handle newly acquired data very quickly to provide an updated approximation of the median.

The main drawback of the algorithm is hinted in its name. It is an approximation of the median and will be at most σ/B away from the true median. Therefore, if the standard deviation of the data is extremely large, the approximation could be significantly different from the actual median. To combat this, the author recommends setting $B = 1000$, which yields sufficient approximations in most empirical cases.

Naturally, this fits in exactly to what we want to accomplish with online change point detection. We have incoming data that we want to roll into the calculation of the median heuristic but we also want a fast way to update the median while being accurate. The `Binapprox` algorithm checks all of these boxes and is simple to incorporate into any online change point detection work flow.

As new data arrives, we recompute the bandwidth using an online median for the kernel function. As a new samples arrive, we store them in a rolling window that pops out the oldest data. Using this updated window, the median heuristic is then recalculated and used for the kernel change point test statistic. Using the binapprox algorithm, the new data is allocated to the existing bins, and the new median bin is found if it is already in an existing bin.

For this setup, there are two hyperparameters that must be chosen that will affect the estimated bandwidth. The size of the window of past observations and how often we update this window of observations. On one extreme we could keep track of all past data and run an expanding median that is updated as soon as we observe a new observation. This would be cumbersome as the amount of data stored would grow linearly over time, which is not sustainable. Therefore, we must be careful not to store more past data than we need for storage efficiency sake. This must be balanced with how often we add new samples and remove the oldest samples simultaneously. Ideally, we want to update the bandwidth as often as possible so it does not become stale over time. However, simply removing the oldest data point and adding the newest data point to our median window may result in unstable median estimations. Therefore, this must also be tuned to the exact dataset to get a stable rolling result over time.

Chapter 4

Experiments on Synthetic Data

4.1 Setup

4.1.1 Initialization

An important factor for online change point methods is determining how to initialize the algorithm. Because we are running an unsupervised model for classifying change points, we do not spend time training the model or tuning any of the hyperparameters. This is a double-edged sword because on one hand, we can drop in a change point algorithm onto a data stream and let it start running without much interference. On the other hand, if it does not have any prior distribution to compare to then it will need to be adjusted to know what's an appropriate reference or in-control distribution. This is usually referred to as the initialization phase for online algorithms.

There are many ways to get through the initialization step and create an appropriate reference sample. One way is to initially compare the oncoming data stream with some zero valued data. As data comes in, the algorithm can replace the reference data with real observations, creating a reference distribution. While this does assume that the data is initially in-control, it does provide a simple way to create a reference distribution on the fly without prior knowledge. When analysing the results, the practitioner must exclude the initial phase when comparisons were done with the initial zero values because the test statistic calculated from them will be insignificant.

Another method is to construct the reference distribution with past data that do not contain change points if it is available as done in [36] and [12]. This method benefits from not having to spend time initializing the algorithm since it can immediately provide an informed statistical comparison. If the data stream is not very long or costly to acquire then this method is beneficial because the entire signal is retained. On the other hand, prior work needs to be done to construct this sample and this may not always be feasible in practice. Another possible issue is what if the regime of the in-control distribution shifts to a new normal? Then the old reference distribution is no longer applicable and must be recreated again, which may be costly.

4.1.2 Structuring the Data Stream

In practice, there is no universal way for evaluating online change point models given the nature of the problem. There are two main reasons for this. First, because change point detection originated in the statistical literature, most algorithms or models have strong theoretical results but have not being applied to synthetic or real datasets. Being able to handle high dimensional data streams is a relatively new benefit that has not been so research up until the mid-90's was largely theoretical in nature. Second, it's not obvious how an infinite data stream should be simulated to evaluate the efficacy of an algorithm. How long does synthetic data stream have to be to objectively evaluate an algorithm? How can you actually know where change points occurred in a real dataset that has no obvious ground truth? It is not obvious how to answer these questions, nor does there exist a universal answer. Like most things, the answer depends on the context and what a practitioner is aiming to test with their particular algorithm.

For this thesis, we wish to compare several methods not just one novel method, therefore data consistency and reproducibility will be important for drawing conclusions about performance. Synthetic datasets are a good way to replicate different distributions over and over. Real change points can also be exactly identifiable and compared to with synthetic data. Take for example a data stream where a single random variable is sampled from a normal distribution $\mathcal{N}(0, 1)$ that incurs a change point and becomes a random variable from a normal $\mathcal{N}(0, 2)$. I can control both of these distributions and know exactly where the change point occurred, but how should I determine how many observations should be observed before switching distributions? How many observations after the change point should I draw from the new distribution before realizing the algorithm will never detect this change point? Again, decisions like this are arbitrary and nuanced for evaluating online change point models because there is no universal method.

We choose to

4.1.3 Synthetic Datasets

A common difficulty in change point detection is evaluating the performance of an algorithm with datasets that aren't overly simplistic and difficult enough to ascertain some real world use.

Unlike fields like image recognition where standardized datasets like MNIST provide a common benchmark, there are no standard datasets that are widely used across the change point detection literature for evaluating new methods. Most papers propose experiments that are relevant for the specific problem they are trying to solve but lack examples or explanations of when their method would not be applicable. Furthermore, because change point evolved out of the statistics literature, many papers focus on theoretical results and provide minor experimental results, if any.

Given the empirical focus of this thesis, we attempt to put together the most comprehensive experiments using synthetic data. To the best of our knowledge, no change point detection paper covers as many variations as presented in this thesis. While synthetic datasets are idealistic in their formulation, they provide a good starting point for comparing different methods because the exact location of the change points can be controlled for. This is a luxury that is often not available with real world datasets, making it difficult to ascertain performance on them. Therefore, to compare several kernel change point detection methods, they will be evaluated across several synthetic datasets.

Inspired by recent papers [9] and [12] that attempt to bridge the gap between the statistics and machine-learning literature, we put together various challenging changes in distribution that may be encountered in a data stream. They are the following: change in mean, scaling variance, alternating between a Gaussian distribution and a Gaussian mixture, alternating between Gaussian mixtures, and alternating between Gaussian distribution and Laplace distribution. It is truly hard to properly generalize all the possible situations a non-parametric algorithm may be used in, but the synthetic cases presented in this thesis cover a range of applications. The following paragraphs describe how each one is constructed in detail.

For a change in mean, a change point is inserted in the time series at some random time where the mean is shifted either positively or negatively. There are two variants to this scenario. In the first, the mean change is in all dimensions simultaneously. This is the most common experiment run by non-parametric change point detection models (add citation for this). In the second variation, the mean change is in only one dimension making it harder to detect.

For a change in variance, the distribution alternates between a Gaussian with $\mathcal{N}(0, 1)$ to a Gaussian where the variance is scaled by a factor of 2 giving $\mathcal{N}(0, 2)$.

For a change between Gaussian mixtures, the data stream is setup identically to the synthetic tests ran in the NEWMA paper [31]. One million samples are drawn from Gaussian Mixture Models (GMM) in dimension $d = 20$ with $k = 10$ components. Every 2000 samples the GMM changes, i.e. k new vector means according to a centred Gaussian distribution, k new covariance matrices from an inverse-Wishart distribution, and k new mixing weights from a Dirichlet distribution. This results in 500 changes to detect in a row.

In the last scenario, the time series alternates from a Gaussian with $\mathcal{N}(0, 1)$ to a Laplace distribution with zero mean and unit variance., i.e. the location is $\mu = 0$ and the scale is $b = \sqrt{0.5}$. The idea for this scenario is detecting changes when a data stream is consistent in its mean and variance, but not in its kurtosis and is same as one of the an experiments from the Scan-B paper.

For each experiment above, a synthetic time series is created with 500 change points that will be fed into each of the algorithms. Once the detection statistics are calculated, the evaluation metrics from 4.1.4 are computed at various thresholds. A Monte-Carlo approach is used to estimate false alarm rates and average detection delay at various thresholds. See table 1 for a summary of each synthetic dataset.

It should be noted this testing setup is nearly identical to the testing procedure used in the NEWMA except we use longer time series with more change points and more variations of possible distribution changes. The other two kernel algorithms, Scan-B and KCUSUM, did not use long time series with several change points but rather a repeated test approach with a single change point.

Table 1: Synthetic Datasets Summary

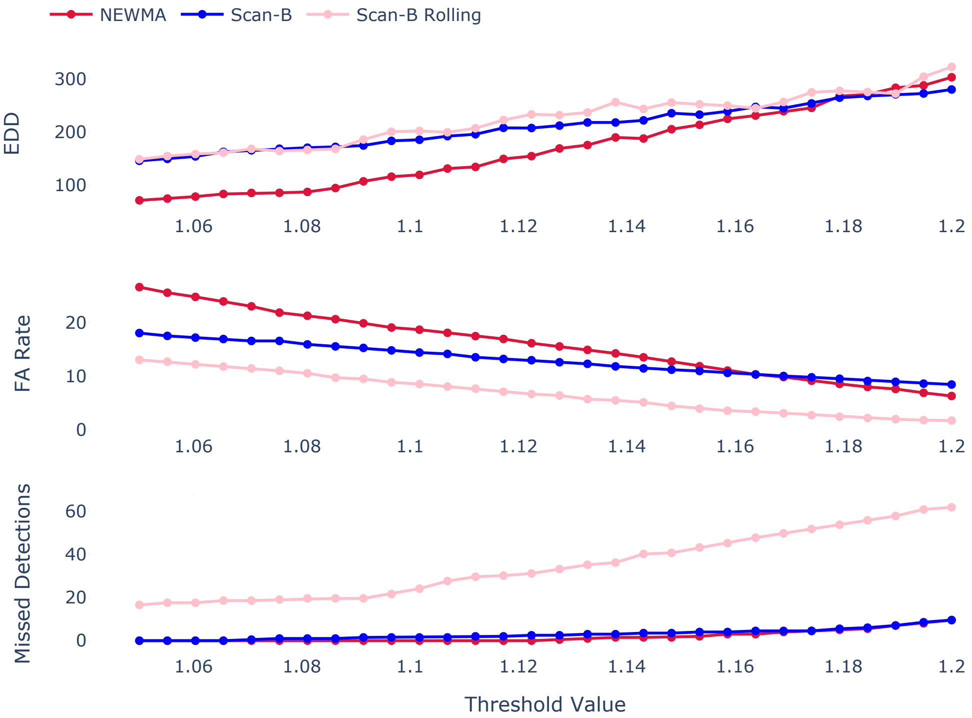
Type of Change	No. of Dimensions	Length	No. of Changepoints
Mean (all dimensions)	20	1M	500
Mean (single dimension)	20	1M	500
Variance	20	1M	500
Blobs	20	1M	500
GMMs	20	1M	500
Normal to GMM	20	1M	500
Normal to Laplace	20	1M	500

4.1.4 Evaluation

Performance of each change point method will be evaluated using the measures introduced in section XXX. Specifically, we compare the estimated expected detection delay, the number of false alarms, and the missed detection rate. These metrics are evaluated at a range of thresholds and the results are plotted below.

4.1.5 Synthetic Results

Figure 4: Evaluation Graphs for Gaussian to Laplace change points, where expected detection delay (**top**), false alarm rate (**middle**), and missed detection rate (**bottom**) are plotted over a threshold region between $[1, 1.2]$.

NEWMA
Scan-B
Scan-B Rolling
EDD
300
200
100
1.06
1.08
1.1
1.12
1.14
1.16
1.18
1.2
FA Rate
20
10
0
1.06
1.08
1.1
1.12
1.14
1.16
1.18
1.2
Missed Detections
60
40
20
0
1.06
1.08
1.1
1.12
1.14
1.16
1.18
1.2
Threshold Value

Chapter 5

Application to Market Liquidity

In finance, *market liquidity* describes how quickly a market participant may buy or sell an asset without causing significant fluctuations in the price. In liquid markets, there is minimal impact to the price by quickly buying (selling) it. In illiquid markets, a purchase (sale) of the asset will cause the price to rise (fall) resulting in adverse price selection if the buyer (seller) is making a large transaction. In both cases, market liquidity is in constant fluctuation. It is this fluctuation that is the focus of this thesis. By detecting changes in market liquidity a market participant can identify periods of when trading may be more or less favourable. For example, an institutional investor wishing to unwind a large position over the course of several days may want to quickly detect rapid changes in a liquidity that could negatively impact their execution strategy. The following description of the limit order book is based on the notation used in [16].

5.1 Properties of the Limit Order Book

All modern financial markets are set-up as a double-sided auction between buyers (bid side) and the sellers (ask side) called limit order books (LOBs). A market participant may post orders at specific prices on either side of the book. A limit order, denoted by x, is defined as a tuple containing a price, p_x, and a size, w_x where $|w_x| > 0$:

$$x = \langle p_x, w_x \rangle. \tag{5.1}$$

A limit order's size indicates how many units of an asset a buyer (seller) is willing to trade at p_x. The prices for which an order can be submitted at are discrete prices.

The LOB at time, t, is denoted as $\mathcal{L}(t)$, and it represents the set of all orders currently active at time t. All bid orders are at the best bid price or lower. The best bid price at time, t, is denoted as:

$$b(t) = \max_{\{x \in \mathcal{L}(t) | w_x < 0\}} p_x \tag{5.2}$$

All ask orders are placed at the best ask price or higher. The best ask price at time, t, is denoted as:

$$a(t) = \min_{\{x \in \mathcal{L}(t) | w_x > 0\}} p_x \tag{5.3}$$

The *bid-ask spread* or simply the spread at a time, t, is defined as:

$$s(t) = a(t) - b(t) \tag{5.4}$$

The normalized spread, $s_n(t) \geq 0$, can be defined by:

$$s_n(t) = \frac{s(t)}{\pi} - 1 \tag{5.5}$$

There are many types of orders that can be placed in the LOB. In this context, we assume all orders in the LOB at a given time are basic limit orders. That is, they are orders placed at a specific price with a specific size. Therefore, at a price, p, and a time, t, the total number of orders is:

$$n^b(p,t) = \sum_{\{x \in \mathcal{L}(t) | w_x < 0, p_x = p\}} |w_x| \tag{5.6}$$

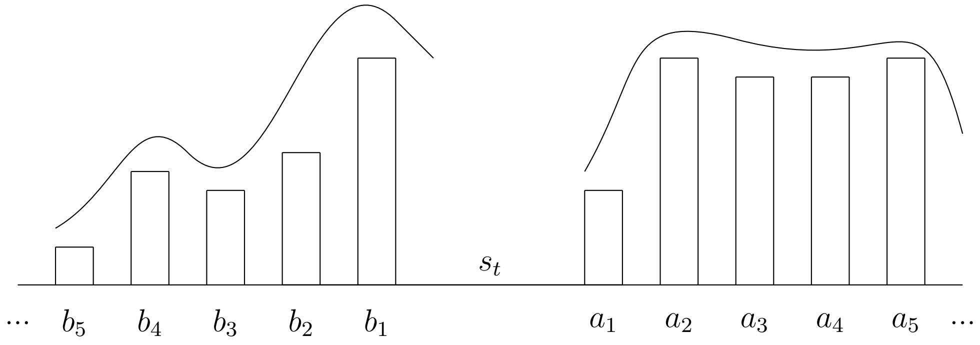
On the ask side, it is similarly defined as:

$$n^a(p,t) = \sum_{\{x \in \mathcal{L}(t) | w_x > 0, p_x = p\}} w_x \tag{5.7}$$

Everytime a trade occurs or a resting limit order is cancelled at that price, size is removed from the LOB at that price. If the size at the $b(t)$ reaches zero then the spread increases the $b(t)$ decreases.

Every LOB has two configuration parameters: the tick size, $\pi > 0$, which is the smallest possible price increment between orders at different prices and lot size, which dictates the smallest amount that can be traded. As mentioned in [16], LOB is essentially a one-dimensional lattice where the dimension is price and every point on the price axis is a multiple [1] of π.

There are two possible distributional changes that are of interest. The first is how the entire distribution of liquidity changes over time. These changes are often associated with events happening in the market such as the start of U.S. trading hours, the release of economic data or a planned speech by the head of the federal reserve. There may also be unplanned events that affect the overall liquidity in the book such as a the 9/11 terrorist attack or a virus outbreak like the Corona virus.



The second is how the distribution of the bid orders differs from the distribution of the ask orders. Typically, the distribution of liquidity on each side of the book is more or less symmetric. However, there have been studies that have shown times when there is an asymmetry between the two sides. This asymmetry has been shown to correlate to price changes.

5.2 Dataset Construction

While many types of markets operate using a basic order book, this thesis will be focused on futures markets found in the Chicago Mercantile Exchange (CME). The reasoning for this is two-fold. The first, unlike bond markets or currency pair markets, liquidity for many futures products is concentrated on a single exchange. This simplifies analysis of liquidity by not having to aggregate data across several exchanges into a synthetic ladder. The second advantage is, unlike equity markets where stocks and exchange traded funds (ETFs) are traded, there are no dark pools available for

[1] In most cases these multiples of π are strictly positive but in particular markets, such as ED packs and bundles, there may be negative integers of π, i.e. a price axis with positive and negative prices.

the futures markets under consideration in this thesis. Again this simplifies where the liquidity data for an instrument may appear.

The futures that will be used in our dataset are based on a combination of selecting the most traded contracts on the CME and selecting futures that represent different asset classes. They are summarized in the following table:

Table 2: Summary of Futures Studied

Name	Asset Class	CME Symbol	Tick Size	Other
E-mini S&P 500	Equity	ES	0.25	1.402
10 Year T-Notes	Fixed Income	ZN	3.442	0.299
Crude Oil	Commodity	CL	3.442	0.299
Gold	Commodity	GC	3.442	0.299
Euro FX	Currency	6E	3.442	0.299

For each future contract in table 2, the state of the order book is sampled every 60 seconds from 7 AM to 4 PM EST for a period of 45 days. The characteristics of the order book that are sampled at every time step, t_I, are: 1) Quantity, q, at the top N levels of the book, $q_1 ... q_N$ 2) Spread, $s : [0, \mathbb{Z}^+]$, normalized by tick size, where spread of zero indicates there is no spread.

While spread is not indicative of liquidity but rather an absence of liquidity, we believe it is an important feature to include because all other things being equal, a change in spread is an important change in liquidity distribution for market participants.

Chapter 6

Conclusion

6.1 Summary of the thesis

Add description of thesis, summarizing the key points from each chapter and what this thesis attempted to add to the literature.

6.2 Discussion and Future Work

One focus of this thesis was using the MMD measure for two-sample testing and, by extension, for change point detection. It would interesting to explore other statistical distances that could compare two distributions non-parametrically. It would be interesting to re-use the same approaches explored in this thesis, but with the MMD swapped for a different distance measure.

Recently, interesting research has been made on re-purposing binary classifiers as two-sample tests. In [23], random forests are compared to different implementations of MMD. The authors run many tests comparing the test power across the different two-sample tests and their random forest. The results are interesting because while the random forest classifier is not better in every situation, it is better on hard two-sample tests such as the blobs dataset. Therefore, it remains to be seen whether these classifier approaches to two-sample testing can be adapted to online change point detection in an efficient manner.

- lopez paper

Bibliography

[1] Samaneh Aminikhanghahi and Diane J Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.

[2] Sylvain Arlot, Alain Celisse, and Zaıd Harchaoui. Kernel change-point detection. *arXiv preprint arXiv:1202.3878*, 6, 2012.

[3] Sayantan Banerjee and Kousik Guhathakurta. Change-point analysis in financial networks. *arXiv preprint arXiv:1911.05952*, 2019.

[4] MICHELE Basseville. Edge detection using sequential methods for change in level–part ii: Sequential detection of change in mean. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(1):32–50, 1981.

[5] Sotiris Bersimis, Stelios Psarakis, and John Panaretos. Multivariate statistical process control charts: an overview. *Quality and Reliability engineering international*, 23(5):517–543, 2007.

[6] Marcel Bosc, Fabrice Heitz, Jean-Paul Armspach, Izzie Namer, Daniel Gounot, and Lucien Rumbach. Automatic change detection in multimodal serial mri: application to multiple sclerosis lesion evolution. *NeuroImage*, 20(2):643–656, 2003.

[7] E Brodsky and Boris S Darkhovsky. *Nonparametric methods in change point problems*, volume 243. Springer Science & Business Media, 2013.

[8] George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[9] Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. Kernel change-point detection with auxiliary deep generative models. *arXiv preprint arXiv:1901.06077*, 2019.

[10] Jie Chen and Arjun K Gupta. *Parametric statistical change point analysis: with applications to genetics, medicine, and finance.* Springer Science & Business Media, 2011.

[11] Frédéric Desobry, Manuel Davy, and Christian Doncarli. An online kernel change detection algorithm. *IEEE Trans. Signal Processing*, 53(8-2):2961–2974, 2005.

[12] Thomas Flynn and Shinjae Yoo. Change detection with the kernel cumulative sum algorithm. *arXiv preprint arXiv:1903.01661*, 2019.

[13] Kenji Fukumizu, Arthur Gretton, Gert R Lanckriet, Bernhard Schölkopf, and Bharath K Sriperumbudur. Kernel choice and classifiability for rkhs embeddings of probability distributions. In *Advances in neural information processing systems*, pages 1750–1758, 2009.

[14] Kenji Fukumizu, Arthur Gretton, Xiaohai Sun, and Bernhard Schölkopf. Kernel measures of conditional dependence. In *Advances in neural information processing systems*, pages 489–496, 2008.

[15] Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

[16] Martin D Gould and Julius Bonart. Queue imbalance as a one-tick-ahead price predictor in a limit order book. *Market Microstructure and Liquidity*, 2(02):1650006, 2016.

[17] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[18] Arthur Gretton, Ralf Herbrich, Alexander Smola, Olivier Bousquet, and Bernhard Schölkopf. Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6(Dec):2075–2129, 2005.

[19] Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems*, pages 1205–1213, 2012.

[20] Zaid Harchaoui and Olivier Cappé. Retrospective mutiple change-point estimation with kernels. In *2007 IEEE/SP 14th Workshop on Statistical Signal Processing*, pages 768–772. IEEE, 2007.

[21] Salah Haridy, Ahmed Maged, Saleh Kaytbay, and Sherif Araby. Effect of sample size on the performance of shewhart control charts. *The International Journal of Advanced Manufacturing Technology*, 90(1-4):1177–1185, 2017.

[22] Douglas M Hawkins and KD Zamba. A change-point model for a shift in variance. *Journal of Quality Technology*, 37(1):21–31, 2005.

[23] Simon Hediger, Loris Michel, and Jeffrey Näf. On the use of random forest for two-sample testing. *arXiv preprint arXiv:1903.06287*, 2019.

[24] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[25] Harold Hotelling. The generalization of student's ratio. In *Breakthroughs in statistics*, pages 54–65. Springer, 1992.

[26] J Stuart Hunter. The exponentially weighted moving average. *Journal of quality technology*, 18(4):203–210, 1986.

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

[28] Venkata Jandhyala, Stergios Fotopoulos, Ian MacNeill, and Pengyu Liu. Inference for single and multiple change-points in time series. *Journal of Time Series Analysis*, 34(4):423–446, 2013.

[29] P Johnson, J Moriarty, and G Peskir. Detecting changes in real-time data: a user's guide to optimal detection. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2100):20160298, 2017.

[30] Steven M Kay. *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.

[31] Nicolas Keriven, Damien Garreau, and Iacopo Poli. Newma: a new method for scalable model-free online change-point detection. *arXiv preprint arXiv:1805.08061*, 2018.

[32] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada, 2004.

[33] Marc Lavielle and Gilles Teyssiere. Adaptive detection of multiple change-points in asset price volatility. In *Long memory in economics*, pages 129–156. Springer, 2007.

[34] Quoc Viet Le, Tamás Sarlós, and Alexander Johannes Smola. Fastfood: Approximate kernel expansions in loglinear time. *arXiv preprint arXiv:1408.3060*, 2014.

[35] Tze-San Lee. Change-point problems: bibliography and review. *Journal of Statistical Theory and Practice*, 4(4):643–662, 2010.

[36] Shuang Li, Yao Xie, Hanjun Dai, and Le Song. M-statistic for kernel change-point detection. In *Advances in Neural Information Processing Systems*, pages 3366–3374, 2015.

[37] Matthew D Lieberman and William A Cunningham. Type i and type ii error concerns in fmri research: re-balancing the scale. *Social cognitive and affective neuroscience*, 4(4):423–428, 2009.

[38] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[39] James M Lucas. Combined shewhart-cusum quality control schemes. *Journal of Quality Technology*, 14(2):51–59, 1982.

[40] James M Lucas and Ronald B Crosier. Fast initial response for cusum quality-control schemes: give your cusum a head start. *Technometrics*, 24(3):199–205, 1982.

[41] Rakesh Malladi, Giridhar P Kalamangalam, and Behnaam Aazhang. Online bayesian change point detection algorithms for segmentation of epileptic activity. In *2013 Asilomar Conference on Signals, Systems and Computers*, pages 1833–1837. IEEE, 2013.

[42] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[43] DS Moore and GP McCabe. Introduction to the practice of statistics, 2nd. *Edition. WH Freedom and Company, New York, New York*, 1993.

[44] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Arthur Gretton, and Bernhard Schölkopf. Kernel mean estimation and stein effect. In *International Conference on Machine Learning*, pages 10–18, 2014.

[45] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

[46] Yue S Niu, Ning Hao, Heping Zhang, et al. Multiple change-point detection: A selective overview. *Statistical Science*, 31(4):611–623, 2016.

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

[48] Andrey Pepelyshev and Aleksey S Polunchenko. Real-time financial surveillance via quickest change-point detection methods. *arXiv preprint arXiv:1509.01570*, 2015.

[49] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[50] Aaditya Ramdas, Sashank Jakkam Reddi, Barnabás Póczos, Aarti Singh, and Larry Wasserman. On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[51] SW Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 1(3):239–250, 1959.

[52] Alaa Saade, Francesco Caltagirone, Igor Carron, Laurent Daudet, Angélique Drémeau, Sylvain Gigan, and Florent Krzakala. Random projections through multiple optical scattering: Approximating kernels at the speed of light. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6215–6219. IEEE, 2016.

[53] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[54] Walter Andrew Shewhart. *Economic control of quality of manufactured product*. ASQ Quality Press, 1931.

[55] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

[56] Bharath K Sriperumbudur, Kenji Fukumizu, and Gert RG Lanckriet. Universality, characteristic kernels and rkhs embedding of measures. *Journal of Machine Learning Research*, 12(Jul):2389–2410, 2011.

[57] M Staudacher, S Telser, A Amann, H Hinterhuber, and M Ritsch-Marte. A new method for change-point detection developed for on-line analysis of the heart beat variability during sleep. *Physica A: Statistical Mechanics and its Applications*, 349(3-4):582–596, 2005.

[58] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[59] Alexander Tartakovsky, Igor Nikiforov, and Michele Basseville. *Sequential analysis: Hypothesis testing and changepoint detection*. Chapman and Hall/CRC, 2014.

[60] Ryan J Tibshirani. Fast computation of the median by successive binning. *arXiv preprint arXiv:0806.3301*, 2008.

[61] Charles Truong, Laurent Oudre, and Nicolas Vayatis. A review of change point detection methods. *arXiv preprint arXiv:1801.00718*, 2018.

[62] JO Westgard, T Groth, T Aronsson, and CH De Verdier. Combined shewhart-cusum control chart for improved quality control in clinical chemistry. *Clinical Chemistry*, 23(10):1881–1887, 1977.

[63] Ping Yang, Guy Dumont, and John Mark Ansermino. Adaptive change detection in heart rate trend monitoring in anesthetized children. *IEEE transactions on biomedical engineering*, 53(11):2211–2219, 2006.

[64] Emmanuel Yashchin. On the analysis and design of cusum-shewhart control schemes. *IBM Journal of Research and Development*, 29(4):377–391, 1985.

[65] Wojciech Zaremba, Arthur Gretton, and Matthew Blaschko. B-test: A non-parametric, low variance kernel two-sample test. In *Advances in neural information processing systems*, pages 755–763, 2013.

[66] Xiaoqian Zhu, Yongjia Xie, Jianping Li, and Dengsheng Wu. Change point detection for subprime crisis in american banking: From the perspective of risk dependence. *International Review of Economics & Finance*, 38:18–28, 2015.