UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Simona Micevska

A Statistical Drift Detection Method

Master's Thesis (30 ECTS)

Supervisor: Sherif Sakr, PhD
Supervisor: Toivo Vajakas, MSc

Tartu 2019

A Statistical Drift Detection Method

Abstract:
Machine learning models assume that data is drawn from a stationary distribution. However, in practice, challenges are imposed on models that need to make sense of fast-evolving data streams, where the content of data is changing and evolving dynamically over time. This change between the underlying distributions of the training and test datasets is called concept drift.

The presence of concept drift may compromise the accuracy and reliability of prospective computational predictions. Therefore, handling concept drift is of great importance in the direction of diminishing its negative effects on a model's performance. In order to handle concept drift, one has to detect it first. Concept drift detectors have been used to accomplish this - reactive concept drift detectors try to detect drift as soon as it occurs by monitoring the performance of the underlying machine learning model. However, the importance of interpretability in machine learning indicates that it may prove useful to not only detect that drift is occurring in the data, but to also identify and analyze the causes of the drift.

In this thesis, the importance of interpretability in drift detection is highlighted and the Statistical Drift Detection Method (SDDM) is presented, which detects drifts in fast-evolving data streams with a smaller number of false positives and false negatives when compared to the state-of-the-art, and has the ability to interpret the cause of the concept drift. The effectiveness of the method is demonstrated by applying it on both synthetic and real-world datasets.

Keywords:
Concept drift, Concept drift detection, Distribution changes, Data streams

CERCS: P170 - Computer science, numerical analysis, systems, control

Statistiline triivi avastamise meetod

Lühikokkuvõte:

Masinõppemudelid eeldavad, et andmed pärinevad statsionaarsest jaotusest.

Praktikas on tihti vaja mudelitega tõlgendada andmeid, mis pärinevad kiiresti dünaamiliselt muutuvast andmevoost. Seda muutust õppe- ja testvalimis nimetatakse kontseptuaalseks triiviks (ingl k concept drift).

kontseptuaalsetriivi olemasolu võib kahjustada mudelennustuste täpsust ja usaldusväärsust. Seetõttu on kontseptuaalse triivi arvestamine väga oluline, et vähendada selle negatiivset mõju tulemustele. Kontseptuaalse triivi arvestamiseks tuleb see kõigepealt tuvastada. Kontseptuaalse triivi tuvastamiseks kasutatakse triivi detektoreid - reaktiivsed kontseptuaalse triivi detektorid püüavad tuvastada triivi niipea, kui see ilmneb, jälgides aluseks oleva masinõppe mudeli toimimist. Tõlgendatavus on masinõppes tähtis ja meetod võib osutuda kasulikuks mitte ainult triivi olemasolu tuvastamiseks andmekogumis, vaid ka triivi põhjuste tuvastamisel ja analüüsimisel.

Käesolevas töös rõhutatakse tõlgendatavuse tähtsust triivi tuvastamisel ja esitatakse statistilise triivi tuvastamise meetod (SDDM), mis tuvastab triivi kiiresti arenevates andmevoogudes, kusjuures võrdluses kaasaegsete meetoditega esineb vähem valepositiivseid ja valenegatiivsed tulemusi. Meetod annab ka kontseptuaalse triivi põhjuste tõlgenduse. Töös näidatakse meetodi tõhusust, rakendades seda nii sünteetilistele kui ka reaalsetele andmekogumitele.

Võtmesõnad:

Kontseptuaalne triiv, Kontseptuaalse triivi tuvastamine, Jaotuse muutus, Andmevood

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

In the traditional machine learning techniques, data is assumed to be drawn from stationary distributions. Thus, when a supervised classifier is trained, it is commonly assumed that the data in the training and testing sets follows the same distribution [1, 2]. However, in real-world applications, it is observed that processes are non-stationary and are characterized by a shifting nature, which means that the distribution of the features (the independent variables), or the concept being learned changes.

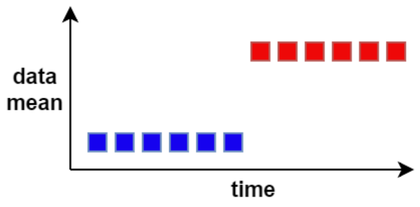
The scenario where the training set and test set follow different distributions is known as *concept drift* [3].

For instance, let us suppose that we have built a model for predicting the sales of a small ice cream store based on factors that have been deemed relevant, such as the current day of the week, previous sale records, weather season, etc. Let's assume that the model has determined peaks in ice cream sales from June until September and uses this for future predictions. However, if the main competitor of the ice cream store in the area has suddenly decided to close shop, this will lead to a high increase in the ice cream sales which our model doesn't take into account, meaning that the accuracy of our model will decrease significantly unless we adapt it to the changes that have occurred in the environment.

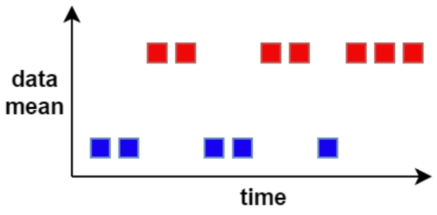
Therefore, when a supervised classifier has been trained with training data whose distribution differs from the distribution of the test data, the accuracy of the prospective computational predictions can be compromised. Thus, it is very important to detect these changes, and if significant changes between the data sets are detected, certain steps, such as training the classifier with new data or weighting the training data set and retraining, should be taken.

The action that should be taken for handling drift varies based on the pattern of the concept drift that occurs in the data. Some patterns of concept drift, illustrated in Figure 1, that can occur are: [3]:

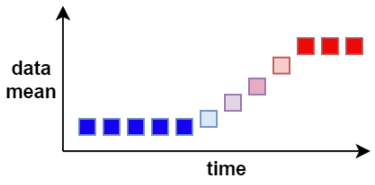
- *abrupt drift*: when the training data source is suddenly replaced by the test source (Figure 1a).
- *gradual drift*: when the change is not abrupt, but goes back to a previous pattern from a past time (Figure 1b).
- *incremental drift*: when, as time passes, the probability of sampling from the training set decreases and the probability of sampling from the test set increases (Figure 1c).
- *reoccurring drift*: when a previously active concept reappears after some time or when new concepts that have not been observed before appear (Figure 1d).



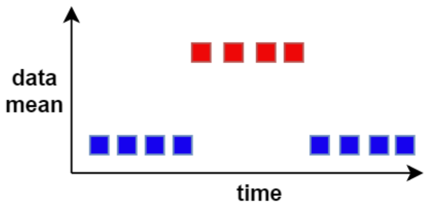
(a) Abrupt drift



(b) Gradual drift



(c) Incremental drift



(d) Reoccurring drift

Figure 1: Concept drift patterns

However, in addition to detecting whether drift has occurred, in practice it may also prove important to interpret and investigate the nature of the concept drift. Interpretability, in the context of machine learning systems, can be defined as *the ability to present results in terms that are understandable for a human* [4]. Introducing intepretability in concept drift detectors can help us understand how the data evolves over time. In addition, it may prove valuable in the context of predicting future drifts, meaning that it increases our capacity of handling drift. Therefore, an interpretable drift detection method, in addition to detecting drift, should also locate the source of the drift and characterize its nature. Consequently, such a method can be used not only for immediate response to drift, but will also provide the tools necessary for analyzing drift patterns and understanding how the data changes over time.

A lot of concept drift detection techniques have been developed that have been successful in detecting drifts in synthetic data streams, such as the Fast Hoeffding Drift Detection Method (FHDDM) [5] and the McDiarmid Drift Detection Methods (MDDMs) [6]. However, their applicability in the real world has not been properly evaluated. Furthermore, these methods operate in a black box manner, meaning that the methods don't provide any details of why a drift has occurred and what caused it. On the other side of the spectrum, techniques for describing the nature of the concept drift have emerged. The most relevant work in this group of techniques is Webb et al.'s method for quantifying the drift directly from data [7, 8]. That method has been applied to several real-world data streams by the authors in order to discover the causes of drift and drift patterns. This method, however, can not be directly applied to a wide range of inputs, which is observed in the numeric experiments performed in Section 4. This occurs because the method's ability to detect drift depends on parameters whose values the authors have fixed without any numeric justification. In addition, the method's ability to detect drifts has not been evaluated and compared against related work.

In this thesis, the drawbacks of both groups of related work are addressed in order to present the Statistical Drift Detection Method (SDDM). SDDM is built upon the work of Webb et al. [7, 8] and signals when a drift has occurred, which is useful in the direction of handling the drift, but it also outputs a comprehensive report of the nature of the drift, which is useful for long-term analysis of the drift patterns in real-world data streams. Therefore, the contributions of this thesis are the following:

- Identifying two equally important goals in the concept drift field: drift detection and interpreting the causes of the drift. Grouping the current related work in regards to the goals they achieve.
- Analysis of the advantages and disadvantages of both groups of related work.
- Thorough analysis of Webb et al.'s method [8] and the effect of the method's parameters on the output. Experiments on synthetic data streams with Webb et al.'s method.
- Designing a drift detection method that achieves both goals, and outperforms state-of-the-art methods, which is shown with experiments on synthetic and real-world data streams.

The rest of the thesis is organized as follows: Section 2 outlines the most relevant related work. In Section 3, the problem is described and the key symbols used in this thesis are summarized. Section 4 summarizes Webb et al.'s [8] work for quantifying the drift magnitude and analyzes its drawbacks. In section 5, the main contribution of this thesis is summarized by describing the Statistical Drift Detection method. In Section 6, the results from applying the drift detector on synthetic and real-world data are presented, and the effectiveness of the method is evaluated. Section 7 discusses the results, presents the conclusions and describes the possible directions for future research.

2 Related work

The related work for drift detection can be divided in two groups according to their goal: methods that detect drift for the purpose of responding to the drift when it occurs, and methods that quantify drift based on statistical properties of the data, and are mostly used for exploratory analysis and uncovering drift patterns in real-world data streams. In the following subsections, methods from both groups are described, and their advantages and disadvantages are discussed.

2.1 Performance-based concept drift detectors

In general, the concept drift detection methods in this group are based on the performance of the underlying machine learning model and they are used for immediate response to drift. They continually monitor its performance, and if it starts to degrade, it indicates that a drift has occurred. These methods can be classified into three general groups: sequential analysis methods, statistical methods, and window based methods [3]. The characteristics of each of these groups and the most relevant methods from each group are presented in the subsections that follow.

2.1.1 Sequential Analysis methods

The methods in this group sequentially evaluate the prediction results as soon as they are available, using the sequential probability ratio test [9]. When a user-defined threshold is reached, they alarm that a drift has occurred. The accuracy of these methods depends on the parameter values which affect the number of true positives and false positives. The Cumulative sum (CUSUM) method [10], and its variant, the Page-Hinkley test [10] are members of this group.

2.1.2 Statistical methods

These methods analyze statistical parameters such as the mean and the standard deviation of the predicted results in order to detect concept drifts.

1. ***Drift Detection Method (DDM)***: The Drift Detection Method by Gama et al. [1] is an online learning method that uses a classifier to detect drift in the data by monitoring the error rate of the model. It is based on the assumption that the error rate of the classifier decreases or stays constant as the number of instances increases. Thus, an increase in the error rate suggests the occurrence of drift. This method performs well for detecting abrupt changes and fast incremental changes, but slow incremental changes are detected too late or not detected at all.

2. ***Early Drift Detection Method (EDDM)***: The Early Drift Detection Method was developed by Baena-Garcia et al. [2] to address the issues of detecting incremental drift, which is detected too late with the Drift Detection method [1], while maintaining a good performance with abrupt concept drifts as well. The algorithm is based on the likelihood of drift occurring when the distance between errors is smaller, thus it calculates the average distance between two recent errors, and it detects a drift if a predefined threshold is met. EDDM achieves an early detection in presence of incremental changes, even when that change is slow, as it is more sensitive than the previous method. However, its sensitivity can be considered a drawback when the data contains a lot of noise.

3. ***Exponentially Weighted Moving Average (EWMA)***: This method by Ross et al. [11] computes a recent estimate of the error rate by progressively down-weighting older data. A change is detected when the mean of the error rate diverges significantly, and the significance of the divergence is controlled by a parameter.

4. ***Reactive Drift Detection Method (RDDM)***: The Reactive Drift Detection Method by Barros et al. [12] is based on DDM and it improves the final accuracy by discarding older instances of very long concepts, with the goal of detecting drifts earlier. The main idea behind RDDM is to shorten the number of instances of very long and stable concepts in order to handle a known performance loss problem of DDM. Whenever a concept becomes too long, which is measured by a predefined maximum number of instances, a softer concept drift is performed, which doesn't cause any modifications in the base learner, but it triggers the recalculation of the statistics used to detect the warning and drift levels using a smaller number of instances.

2.1.3 Window based methods

Window based methods use a fixed window that sums up past performance information and a sliding window summarizing the performance of most recent instances. Statistical tests are used to compare the distribution over the two windows, with the null hypothesis stating that the distributions are equal. A rejection of the null hypothesis indicates a significant difference between the distributions of these windows, which suggests the occurrence of a drift.

1. ***Adaptive windowing (ADWIN)***: The Adaptive Windowing method by Bifet et al. [13] uses sliding windows of variable size that are recalculated online according to the rate of observed change of data in these windows. The window is increased when there is no context change, and it's decreased

when a change has been detected. This method provides guarantees of its performance, in the form of limits on false positive rates and false negative rates. In addition, ADWIN doesn't make assumptions about the underlying data distribution. The drawback of this method is that it works only for one-dimensional data. When working with n-dimensional data, a separate window must be maintained for each window, thus handling the method is significantly more difficult.

2. ***Hoeffding Drift Detection Methods***: Frias-Blanco et al. [14] have proposed two methods for drift detection that use the Hoeffding's inequality as an upper bound of the difference between the averages. The $HDDM_{A-test}$ method compares the moving window averages to detect drift and is useful for detecting sudden drifts. The $HDDM_{W-test}$ uses a forgetting scheme to first weight the moving averages and then compare them in order to detect drift and is more appropriate in detecting incremental drifts.

3. ***Fast Hoeffding Drift Detection Method (FHDDM):*** The Fast Hoeffding Drift Detection method by Ali Pesaranghader and Herna L. Viktor [5] detects drift points using a sliding window and Hoeffding's inequality. Unlike the other window based methods that compare a window that contains historic information and another window that maintains the most recent information, this approach compares the current's classifier accuracy against the best accuracy observed so far using a single sliding window of size n.

4. ***McDiarmid Drift Detection methods (MDMM)***: The McDiarmid Drift Detection methods by Ali Pesaranghader et al. [6] detect drift points using a sliding window and McDiarmid's inequality. This group of methods slides a window over prediction results and links the window entries with weights. The most recent entries are assigned higher weights, in order to emphasize their importance. The three variants that the authors propose: $MDDM_A$, $MDDM_G$ and $MDDM_E$ differ in the weight-increasing scheme that is used. The methods rely on the assumption that by weighting the predictions associated with the sliding window, and by emphasizing the most recent elements, concept drift could be detected faster and more efficiently. As new instances become available, the algorithm compares a weighted mean of elements inside the sliding window with the maximum weighted mean observed thus far. Drift is detected when a significant difference between the two weighted means is observed. Numeric experiments performed by the authors of the paper show that this method has shorter detection delays as well as lower false negative rates, while maintaining high classification accuracy [6], compared to previous related work.

The extensive research performed in this group of methods has resulted in techniques that detect drift with a short delay and a low false positive and false negative rate. However, using these measures as evaluation measures for the quality of drift detection has drawbacks. As it can be observed, one cannot evaluate the quality of drift detection in real-world datasets where the ground truth for the drift patterns is unavailable, as the evaluation needs to be performed against a known drift pattern. Thus, the methods from the group have been optimized and evaluated rigorously on several synthetic datasets, but their evaluation on real-world datasets is fuzzy and subjective. In addition, these methods often fail to distinguish between the different types of drifts, and in cases when gradual or incremental drift needs to be handled, they may be of limited use [8].

Most of the methods from this group, such as the state-of-the-art methods FHDDM [5] and MDDM [6], expect that a machine learning model has been trained on the data and capture drift magnitude when the model's performance degrades. Relying on the machine learning model's performance means that the ability of detecting drift will depend on the model's sensitivity to concept drifts. As a result, sensitive models will signal drifts more frequently, thus causing frequent retraining of the model [5]. On the other hand, models which have the ability to adapt to drift without formally detecting it, such as ensemble models [15], will signal drifts rarely or not at all. In terms of maintaining the model's accuracy, both of these edges of the spectrum are reasonable. However, they are unavailing in terms of interpretability and investigating drift patterns. Even if one were to train a model that captures the concept drifts with perfect precision, the output is still merely a flag that indicates whether there was a drift at a certain point of time, which is unrevealing in the direction of analyzing the nature of the drift.

2.2 Statistical concept drift detectors

The main drawbacks of the methods from the previous group can be overcome with a method that quantifies the drift directly from the data. This approach, in addition to drift detection, will enable interpretation of the causes of the drift, and distinguishing between different types of drift.

In the work done by Webb et al. [7, 8], the importance of interpretability in drift detection is brought to attention. They formalize the definitions of types of concept drift, propose quantitative drift measures based on distance measures between distributions and provide a thorough analysis of the nature of drift in several real-world datasets.

The main drawback of this method is that the quality of drift detection depends on the values of the method's parameters (which are elaborated in detail in Section 4), such as the number of bins used for discretization of the numeric features, yet the influence of the choices for these parameter values is not discussed. The authors' choices are not justified by numeric experiments and thus may not be appropriate for every kind of input.

In addition, the authors don't suggest how this method should be used for responding to drift - they don't propose how to translate the extensive output to a single Boolean value that indicates whether there is a concept drift at a certain point of time. Consequently, the method hasn't been validated on synthetic datasets against state-of-the-art related work.

The goal of this thesis is to address the issues that prevail in both groups of related work in order to develop the Statistical Drift Detection Method (SDDM) that can be used for responding to drift, as well as for analyzing the pattern of drift and understanding how the data changes over time. Quantifying the drift, as is done in Webb et al.'s work [8], ensures that the nature of the drift can be analyzed, and addresses the lack of interpretability and the inability to be tested on real-world data streams that is characteristic for the methods described in subsection 2.1. In order to address the drawbacks of Webb et al.'s method, it is first analyzed in Section 4, and then, a more general and optimized method is optimized in Section 5. The method is designed so that its output can be used to achieve two equally important goals:

1. Immediate drift handling. To achieve this goal, the method produces Boolean output that indicates whether a drift has occurred at some point of time.

2. Drift pattern analysis. To achieve this goal, the method produces extensive output that describes the nature and causes of the change.

3 Background

In this section, the reader is provided with the background information necessary for understanding the concepts that are discussed in the rest of the thesis.

3.1 Data streams

A data stream can be defined as a dataset in which the instances have time stamps with various granularity [7, 8]. In real-world applications, the system generally has access to data with time stamps prior to a certain time and models must be applied in order to make predictions for instances with subsequent time stamps.

The process that generates the stream can be considered to be a random variable X from which the objects $x \in domain(X)$ are drawn at random. In a classification learning context, a target (or class) variable $y \in domain(Y)$ is available, where Y denotes a random variable over the targets. Thus, the data stream is comprised of instances $(\mathbf{x_1}, y_1)$, $(\mathbf{x_2}, y_2)$,..., $(\mathbf{x_t}, y_t)$, where $(\mathbf{x_t}, y_t)$ represents an instance in time t, $\mathbf{x_t}$ represents the vector of feature values and y_t represents the target for that particular instance. Let's denote the function which maps the feature vector to a target with $y_t = f(\mathbf{x_t})$. Then, the learning algorithm must analyze the instances prior to a certain time (*training dataset*), learn an approximate function $y_t \approx f'(\mathbf{x_t})$ based on these instances, and apply this function for instances with subsequent time stamps (*test dataset*) in order to make predictions for their target labels.

In this thesis, classification learning context is assumed and the input is expected to be a data stream where batches of instances arrive in sequential time order. However, all methods trivially generalize to situations where there is no target attribute, and where the time order of the batches is not sequential or is unknown.

3.2 Concept drift

Most related work has adopted a probabilistic definition of the term concept [3, 6, 8, 16]. As stated in the Bayesian Decision theory [17], the process of classification can be described by the prior probabilities of the targets $P(y)$, and the target conditional probability density function $P(\mathbf{X}|y)$. The classification decision is made based on the posterior probabilities of the targets, which can be obtained from:

$$P(y|\mathbf{X}) = \frac{P(y) \cdot P(\mathbf{X}|y)}{P(\mathbf{X})} \tag{1}$$

Since $P(y)$ and $P(\mathbf{X}|y)$ uniquely determines the joint distribution $P(\mathbf{X}, y)$, concepts can be defined as the joint distribution $P(\mathbf{X}, y)$ [3]. A concept at point of time t will be denoted as $P_t(\mathbf{X}, y)$.

As it was stated in Section 1, concepts can change over time, resulting in concept drift. Based on the definition of concept, concept drift between data at point of time t and data at point of time u can formally be defined as a difference in the distributions of the data in these time points:

$$P_t(\mathbf{X}, y) \neq P_u(\mathbf{X}, y) \tag{2}$$

However, it can be observed from (1) that a concept drift can be caused by either a change in the distribution of the covariates $P(\mathbf{X})$ (and thus in the target conditional probability distribution $P(\mathbf{X}|y)$), a change in the prior probability distribution $P(y)$ or change in the posterior probability distribution $P(y|\mathbf{X})$.

Changes in the covariate distribution $P(\mathbf{X})$ that are not accompanied by changes in the posterior probability distribution $P(y|\mathbf{X})$ are referred to as ***"virtual concept drift"*** [3]:

$$P_t(\mathbf{X}) \neq P_u(\mathbf{X}) \wedge P_t(y|\mathbf{X}) = P_u(y|\mathbf{X}) \tag{3}$$

These changes do not affect the decision boundary of the model, thus handling of this type of drift is not required. It still may prove useful in practice to detect this drift for exploratory analysis of the data and better understanding of how the data changes over time.

Changes in the prior probability distribution $P(y)$ that are not accompanied by changes in the posterior probability distribution $P(y|\mathbf{X})$ are referred to as ***"prior probability drift"*** [18]. If a scoring learning model has been applied (a model that assigns scores for each of the target labels) and prior probability drift occurs, the accuracy can be improved by keeping the scoring part of the model and merely adjusting the threshold [19]. The appropriate threshold can be selected using a ROC analysis [19, 20], and the model doesn't have to be retrained.

Changes in the posterior probability distribution $P(y|\mathbf{X})$ are referred to as ***"real concept drift"*** [3]. These changes affect the decision boundary of the model and when it occurs, the prospective computational predictions may be compromised. Thus, it is very important to detect these types of drift and handle them by retraining the model.

For a given data stream as an input, a concept drift detector needs to detect the changes that occur in the stream, and raise an alarm if ***"real concept drift"*** has occurred. However, in addition, it should also detect the two other types of changes outlined above, and output appropriate descriptions for the nature and causes of the drift. Thus, the output produced by the detector, in addition to being used for automatically handling the drift that occurs, can be also used for investigating and interpreting the causes and patterns of drift in the data.

3.3 Concept drift magnitude

Webb et al. [7] propose quantitative measures of concept drift, one of which is *drift magnitude*, which measures the distance between concepts using standard distribution distance measures. Any distance measure between distributions can be employed, such as Kullback-Leibler divergence [21], Hellinger distance [22], Total Variation Distance etc. [23], and the choice of the distance measure varies depending on the domain and the characteristics of the stream.

In order to measure the distance between the distributions, the probability distribution has to be estimated from the samples first. Given that the numeric values are discretized, let (o_1, o_2,..., o_n) denote the counts for each of the n unique value combinations in the concept at time t. Based on these sample counts, one can estimate the probability distribution of the concept at time t:

$$P_t(z_i = o_i) = \frac{o_i}{\sum_{i=1}^{n} o_i}, i \in [1, n] \tag{4}$$

Thus, the probability distribution of a concept at time t can be denoted as ($P_t(z_1)$,$P_t(z_2)$,..., $P_t(z_n)$), where $z_i \in domain(Z)$ and Z is a vector of random variables over the feature values and class values. It can be observed from (4) that the sum of all values of the probability distribution ($P_t(z_1)$,$P_t(z_2)$,..., $P_t(z_n)$) equals to 1:

$$\sum_{i=1}^{n} P_t(z_i = o_i) = 1 \tag{5}$$

Then, the Total Variation distance [23] between the probability distribution of a concept at time t and the probability distribution of a concept at time u can be calculated as:

$$\sigma_{tvd}(P_t, P_u) = \frac{1}{2} \sum_{i=1}^{n} |P_t(z_i) - P_u(z_i)| \tag{6}$$

The Hellinger distance [22] between the probability distributions of a concept at time t and the probability distribution of a concept at time u is defined as:

$$\sigma_{hellinger}(P_t, P_u) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{n} (\sqrt{P_t(z_i)} - \sqrt{P_u(z_i)})^2} \tag{7}$$

Both the Total Variation Distance and Hellinger Distance are metrics and they are bounded between 0 and 1, where 0 indicates that the probability distributions are equivalent and 1 indicates that the probability distributions are completely disjoint.

The Kullback-Leibler divergence [21] from the probability distribution of a concept at time t to the probability distribution of a concept at time u is defined as:

$$\sigma_{kld}(P_t \parallel P_u) = \sum_{i=1}^{n} P_t(z_i) \cdot log\Big(\frac{P_t(z_i)}{P_u(z_i)}\Big) \tag{8}$$

The Kullback-Leibler divergence is an asymmetric measure, meaning that $\sigma_{kld}(P_t \parallel P_u) \neq \sigma_{kld}(P_u \parallel P_t)$. To obtain a symmetric measure, the sum of the Kullback-Leibler divergence from P_u to P_t and the Kullback-Leibler divergence from P_t to P_u can be employed [21]:

$$\sigma_{kld}(P_t, P_u) = \sigma_{kld}(P_t \parallel P_u) + \sigma_{kld}(P_u \parallel P_t) \tag{9}$$

The Kullback-Leibler divergence is bounded by 0 from below, and a value of 0 indicates that the distributions are equivalent. However, this measure is unbounded from above - an infinite value will be the result of the calculation if one distribution has zero density where the other does not, which can be observed from (8).

In practice, having an unbounded result is not particularly useful when calculating drift magnitude between two concepts. To avoid this, the calculations of the divergence could be either limited to the points of the probability distribution which have non-zero density in both distributions ($\forall i$, where $P_t(z_i) > 0 \wedge P_u(z_i) > 0$); or Laplace smoothing can be applied [24].

Laplace smoothing [24] is a technique for smoothing categorical data, most commonly used for dealing with zero distribution density. Laplace smoothing should be applied when estimating the probability distribution:

$$P_t(z_i = o_i) = \frac{o_i + \alpha}{\sum_{i=1}^{n}(o_i + \alpha)}, i \in [1, n], \tag{10}$$

where $\alpha > 0$ is the smoothing parameter and $\alpha = 0$ corresponds to no smoothing. Thus, in order to calculate the symmetric Kullback-Leibler divergence between two probability distributions, Laplace smoothing, as described in (10), can be applied. On the other hand, when $\alpha = 0$, points of the probability distribution which have non-zero density in both distributions should not be included in the calculations of the Kullback-Leibler divergence, in order to avoid unbounded results due to zero density.

After dismissing the zero-density points or applying Laplace smoothing, the Kullback-Leibler divergence remains unbounded from above - an infinite value can still be the result of the divergence calculation when one of the distributions has a much fatter tail than the other [21].

3.4 Methods for evaluating the performance of the drift detector

True positive, false positive, false negative rate and detection delay have been used to evaluate the performance of drift detectors in data streams where the ground truth is available [13, 25, 26]. If a detector reports drifts when there are none (has a high false positive rate), it will cause frequent retraining of the model, which increases the usage of resources unnecessarily [25, 26]. On the other hand, if the detection method fails to report drifts when they occur (has a high false negative rate), the error rate of the model increases, because it won't be retrained and thus it won't reflect the new distribution [25, 26]. Consequently, a drift detection method with high true positive rate, low false positive rate and low false negative rate is preferred.

Ali Pesaranghader and Herna L. Viktor [5] have introduced an approach for counting the number of true positives, false positives and false negatives by defining an acceptable delay length Δ, which determines how far can the detected drift be from the true location of the drift, to be accepted as a true positive. In regards to the acceptable delay length Δ, the evaluation measures are calculated as follows:

- *True positive*: A drift that occurs at time t is counted as a true positive if it has been reported by the method in the acceptable detection interval which is defined as $[t, t+\Delta]$. The *true positive rate* will be calculated as the number of drifts detected in the acceptable detection intervals.
- *False positive*: A drift that has been reported outside of the acceptable detection intervals is counted as a false positive. Consequently, the *false positive rate* is calculated as the overall number of drifts reported outside of the acceptable time ranges.
- *False negative*: If a drift has occurred at a time t, a false negative will be counted if the method hasn't reported a drift in the acceptable detection interval $[t, t + \Delta]$.

Figure 2 demonstrates how the true positive, false positive and false negative are counted. The first row displays the locations of the drifts in a data stream, as specified in the ground truth and the second row displays the results of the detection method. The last row shows how the detection method should be evaluated in each point of time. The green squares represent successfully detected drift points (true positives), the red squares represent the non-drift points that were mistaken as drift points by the method (false positives), whereas the yellow squares represent the drift points that were not detected by the detection method (false negatives).

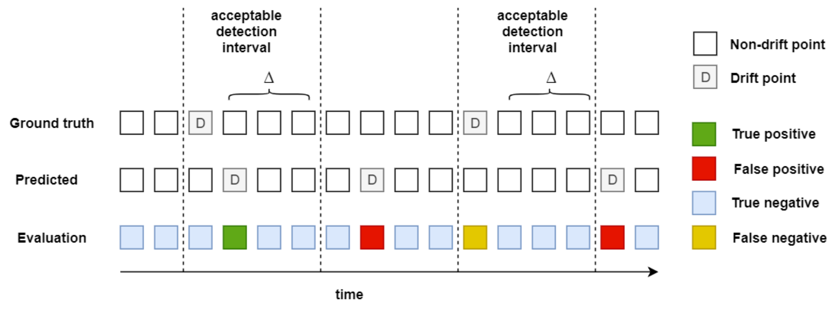


Figure 2: Illustration of counting true positives, false positives and false negatives.

A method that has a high true positive rate, low false positive rate and a low false negative rate calculated with a low detection delay is preferred, as it implies that the method is able to locate the exact point of time when the drift has occurred. A low detection delay has a positive effect on both the accuracy of the model, and the ability to interpret the causes of drift.

In addition, in this thesis, a new evaluation measure will be defined, which measures whether the method is able to locate the true source of the drift by comparing the predicted source of the drift against the ground truth, where the source of the drift refers to the subset of features which caused the drift. It needs to be noted that with this measure, only SDDM can be evaluated, as the other methods that will be used for comparison don't indicate the source of the drift.

These five evaluation measures (true positive rate, false positive rate, false negative rate, detection delay and source of drift) can only be used for evaluating synthetic data streams where the ground truth for drifts is available.

It has been attempted to compare methods in real-world data streams using the classification accuracy [5, 6, 27]. However, the consensus is that classification accuracy is not a valid evaluation measure for comparing methods, as higher classification accuracy doesn't necessarily imply precise drift detection and it depends on other factors [6, 27].

In order to evaluate the method in real-world data streams, Webb et al.[8] have been analyzing the drift pattern and drift causes, and determining whether the results comply with known changes in that specific real-world domain. This approach for evaluating drift detectors in real-world datasets will be also applied in this thesis.

3.5 Data streams with concept drift

Several synthetic and real-world datasets with drift have been widely used for evaluating drift detectors in the literature [1, 5, 6, 8, 11, 12, 28, 29]. In the following subsections, the data streams and their characteristics are described.

1. *Synthetic data streams*

   Four synthetic data streams `Sine1`, `Mixed`, `Circles` and `LED` were generated, which are described in related work [1, 2, 5, 6, 11, 12, 13, 28, 29]. Each of these data streams consists of $100,000$ instances, and contains a 10% class noise, thus the robustness of the drift detectors against noisy data can be evaluated. A description of each stream follows.

   - `Sine1` (with abrupt drift): This data stream consists of two features x and y, uniformly distributed between 0 and 1, and a binary target. The classification function is $y = sin(x)$. The instances are classified as positive if they are under the curve and as negative otherwise. Drift is simulated by reversing the class labels at every $20,000$ instances.
   - `Mixed` (with abrupt drift): This data stream consists of two numeric features x and y which are uniformly distributed between 0 and 1, two Boolean features v and w, and a binary target. An instance is classified as positive if at least two of the three following conditions are satisfied: $v, w, y < 0.5 + 0.3 \cdot sin(3\pi x)$. Drift is simulated by reversing the class labels at every $20,000$ instances.
   - `Circles` (with gradual drift): This data stream has two numeric features x and y that are uniformly distributed between 0 and 1, and a binary target. The classification function is the circle equation $(x - x_c)^2 + (y - y_c)^2 = r_c^2$ where (x_c, y_c) is the center of the circle and r_c is its radius. Instances inside the circle are classified as positive. Drift is simulated by changing the classification function, that is, the circle equation with these parameters $< (0.2, 0.5), 0.15 >$, $< (0.4, 0.5), 0.2 >$, $< (0.6, 0.5), 0.25 >$ and $< (0.8, 0.5), 0.3 >$, appropriately. The drift points are at every $25,000$ instances.
   - `LED` (with gradual drift): This data stream has 7 features that are related to the target and 17 irrelevant features, and a target with 10 class labels. The goal is to predict the digit on a seven-segment display, where each digit has a 10% chance of being displayed. Concept drift is simulated by interchanging relevant attributes at every $25,000$ instances.

   In all synthetic data streams, the sigmoid function is used to simulate abrupt and incremental concept drifts [6, 28]. The function determines the probability of belonging in a new context during a transition between concepts. The transition length allows to simulate different types of drifts and it is set to 50 for abrupt concept drifts, and to 500 for incremental concept drifts.

To sum up, drift occurs at every $20,000$ instances in `Sine1` and `MIXED` with transition length of 50, and at every $25,000$ instances in `Circles` and `LED` with transition length of 500.

2. *Real-world data streams*

   The `Airlines` and `Electricity` real-world data streams were considered, which are extensively used in drift literature [1, 2, 5, 6, 8, 13, 28]. There is no consensus in the literature on when and where drift occurs in these datasets. Both streams are described below.

   - `Airlines`: This data stream consists of instances that represent a flight. It contains $539,383$ instances with features *airline*, *flight*, *airportFrom*, *airportTo*, *dayOfWeek*, *time* and *length*. The binary target indicates whether the flight has arrived on time. Concept drift could appear in the stream as the result of changes in the flights schedules.
   - `Electricity`: This data stream describes the electricity pricing in South-East Australia. It consists of $45,312$ instances with the numeric features which record the price and demand in the states of New South Wales and Victoria and the amount of power transferred between the states. The features have been normalized to the interval [0,1]. The binary target indicates whether the transfer price has increased or decreased relative to the average of the previous 24 hours. Instances have been generated for every 30 minute period from 7 May 1996 to 5 December 1998. Drift may occur due to changes in consumption habits, unexpected events and seasonality.

Table 1 summarizes the characteristics of each of the data streams described in the two previous subsections.

Type	Data stream	Number of features	Number of instances	Type of drift	Number of drift points	Drift length (in instances)
Synthetic	Sine1	2	$100,000$	Abrupt	4	50
	Mixed	4	$100,000$	Abrupt	4	50
	Circles	2	$100,000$	Incremental	3	500
	LED	24	$100,000$	Incremental	3	500
Real-world	Airlines	7	$539,383$	Unknown	Unknown	Unknown
	Electricity	6	$45,312$	Unknown	Unknown	Unknown

Table 1: Characteristics of the synthetic and real-world data streams that contain drift

4 A method for understanding concept drift

As aforementioned, interpretable drift detection methods can be developed by quantifying the concept drift between two data streams. Quantifying the drift consists of calculating statistics that capture the similarity between two sets of multivariate data, which is addressed in Webb et al.'s work [7] and described in Subsection 3.3.

Some issues that prevail when measuring the distance between distributions in practice are highlighted in a later work by the same authors [8]:

1. The techniques for measuring distance between distributions are limited to discrete-valued data. The techniques for computing distance for continuous data drawn from specific distributions require strong assumptions about the form of the distribution, thus they are avoided. For this reason, the authors discretize all numeric features using five bin equal frequency discretization of each attribute across all time periods.

2. The authors provide a proof of the distance measures being monotonic as the dimensionality of data increases. From a practical view point, this means that these measures are likely to be close to their maximum through accumulation of small differences across many dimensions, which significantly reduces their capacity to distinguish between drift and non-drift points. In consequence, in addition to a single drift magnitude calculated based on the whole set of features, the authors also calculate drift magnitudes in the marginal distributions defined over different combinations of features.

Having these issues in consideration, and given as input a concept at time t (X_t, y_t) and a concept at time u (X_u, y_u), Webb et al. [8] propose a method for obtaining the **concept drift magnitude** between two concepts, which is measured as a distance between the probability distributions $P(X_t, y_t)$ and $P(X_u, y_u)$. In addition, they compute:

- **covariate drift magnitude**: distance between the covariate probability distributions $P(X_t)$ and $P(X_u)$.

- **class drift magnitude**: distance between the class probability distributions $P(y_y)$ and $P(y_u)$

- **conditioned covariate drift magnitude**: distance between the conditional covariate probability distributions $P(X_t|y_t)$ and $P(X_u|y_u)$

- **posterior drift magnitude**: distance between the posterior probability distributions $P(y_t|X_t)$ and $P(y_u|X_u)$

Computing these distances will point out to the source of the drift and might reveal different aspects of a complex drift.

The pseudocode for the algorithm that Webb et al. employ can be seen in Algorithm 1. In Algorithm 1, DISCRETIZE is a function that discretizes the numeric attributes using five bin equal frequency discretization, and leaves the categorical attributes unaltered. The procedure TVD calculates the Total Variation Distance [23] between the probability distributions given as input.

Algorithm 1 Method for measuring drift magnitude by Webb et al. [8]

```
    **Input**: Concept at time $t$ $(X_t, y_t)$, concept at time $u$ $(X_u, y_u)$
    **Output**: Drift magnitude between the concepts
1: procedure CALCULATEDRIFTMAGNITUDE
2:     $X'_t, y'_t, X'_u, y'_u \leftarrow$ DISCRETIZE$(X_t, y_t, X_u, y_u, bins = 5)$
3:     estimate probabilities $P(X'_t), P(y'_t), P(X'_t, y'_t), P(X'_t|y'_t), P(y'_t|X'_t)$
4:     estimate probabilities $P(X'_u), P(y'_u), P(X'_u, y'_u), P(X'_u|y'_u), P(y'_u|X'_u)$
5:     $covariate\_drift \leftarrow \text{TVD}(P(X'_t), P(X'_u))$
6:     $class\_drift \leftarrow \text{TVD}(P(y'_t), P(y'_u))$
7:     $concept\_drift \leftarrow \text{TVD}(P(X'_t, y'_t), P(X'_u, y'_u))$
8:     $conditional\_marginal\_covariate\_drift \leftarrow \text{TVD}(P(X'_t|y'_t), P(X'_u|y'_u))$
9:     $posterior\_drift \leftarrow \text{TVD}(P(y'_t|X'_t), P(y'_u|X'_u))$
```

In addition, Webb et al. calculate the drift magnitude in the marginal distributions defined over different combinations of attributes, which is useful for both quantifying drift in high-dimensional data (where the total drift magnitude will likely be close to its maximum), as well as for locating the exact source of the drift. To accomplish this, the same procedure described in Algorithm 1 is utilized. However, in this case, the inputs X_t and X_u don't represent the whole covariate set, but instead consist of a subset of features.

The work done by Webb et al. [8] is state-of-the-art when it comes to measuring and analyzing concept drift in real-world datasets. However, their work lacks numeric justification of the parameter choices - there is no reasoning behind the choice of parameters, such as the number of bins for discretizing continuous data, the distance measure or the chosen number of instances for comparison. In addition, even though they investigate the effect of a single drift measure against drift magnitudes defined over marginal distributions, they haven't evaluated these measures on datasets with ground truth, thus they can't make observations on when one should use which type of measure. Furthermore, despite the statement that the method can be utilized to manage response to drift, the authors don't propose how to flag whether a drift has occurred at a given point of time based on the reported drift magnitude.

In order to improve upon this method and develop a method that can be used to respond to drift and interpret the drift on a wider range of inputs, Webb's method has to be analyzed. In order to do so, the method was implemented as described in Algorithm 1. It was determined that the method was successfully reproduced after running it on the same data streams that the authors have evaluated it on, and confirming that the results are identical.

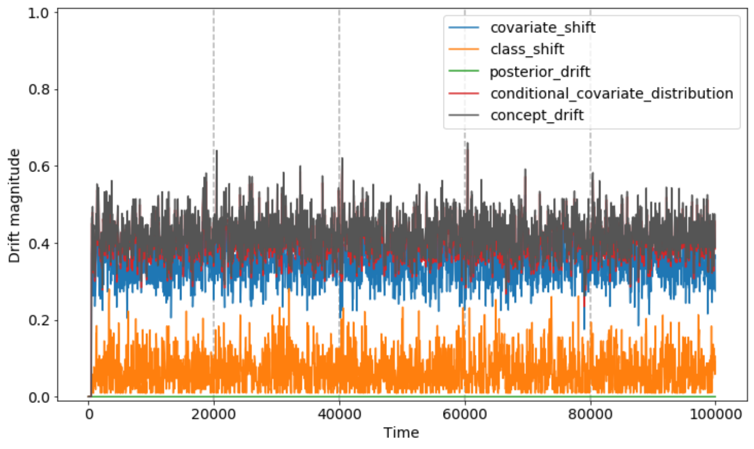
In the rest of this section, Webb et al.'s method [8] is analyzed by evaluating it on data streams where ground truth is available.

4.1 Evaluation of the method on synthetic data streams

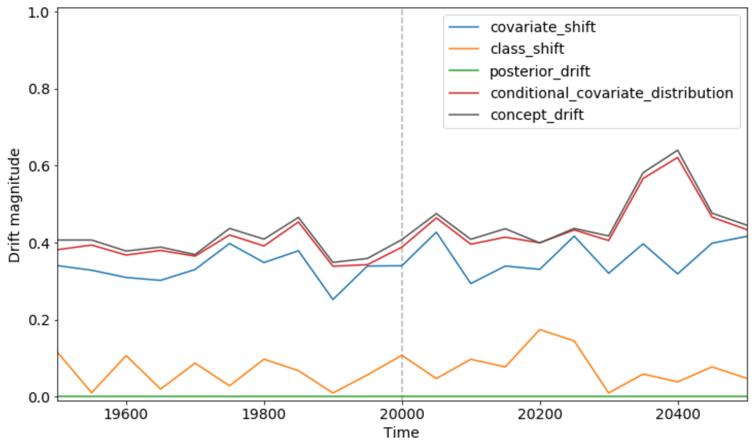
Webb et al.'s method was run on two of the synthetic data streams: SINE1 and CIRCLES, that were described in Subsection 3.5. These two data streams were selected because the first simulates abrupt drift, and the latter simulates incremental drift. Thus, the response of the method for different types of drift can be evaluated.

Webb et al.'s method assumes that the data arrives in batches of instances at a time. Thus, the data was split in batches of equal size p, and the drift magnitude was calculated between the distribution of each batch with respect to the distribution of the previous t instances. In their work, the batch sizes and the number of instances for comparison are selected arbitrarily and they vary between the different data streams they evaluate. Thus, the batch size was initially set to the known length of the drift for the data stream, that is $p = 50$ for SINE1 and $p = 500$ for CIRCLES. The number of instances for comparison was initially set to be twice the batch size, that is, $t = 100$ for SINE1, and to $t = 1000$ for CIRCLES; these values were selected without any numeric justification.

The results from running Webb's method on the SINE1 and the CIRCLES data streams can be seen on Figures 3 and 4. On these figures, the X axis represents the point of time in which the batch of p instances has arrived, and the Y axis represents the drift magnitude calculated between the distribution of the current batch of p instances and the distribution of the previous t instances. The grey dashed lines represent the points in time when drift has occurred, as specified in the ground truth.

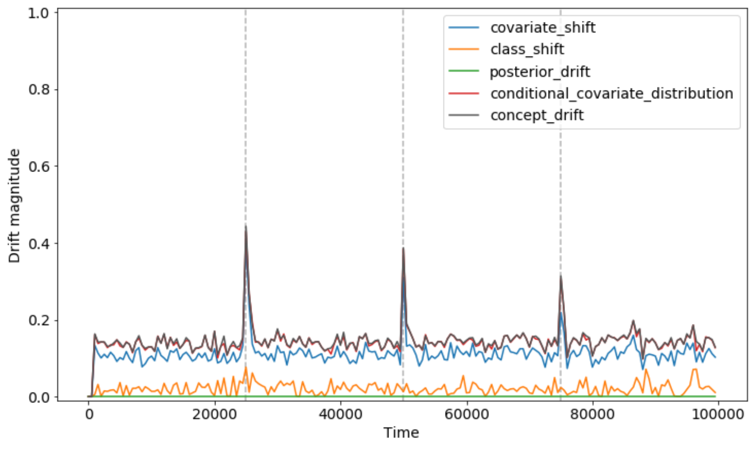


(a) `Sine1` dataset

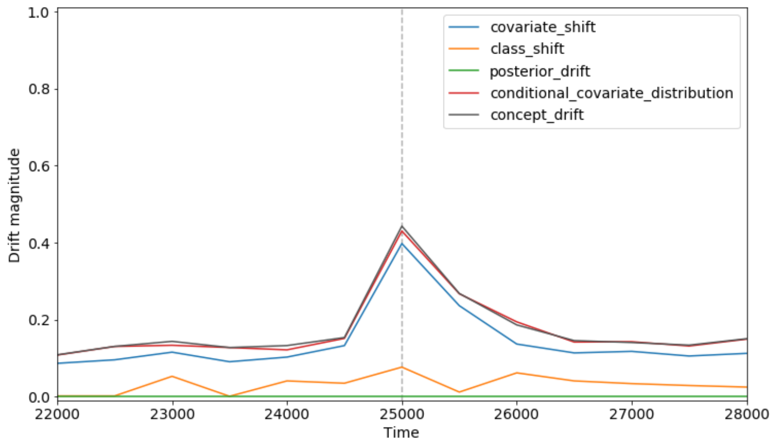


(b) `Sine1` dataset (x axis zoomed to first drift point)

Figure 3: Concept drift magnitude, covariate drift magnitude, class drift magnitude, conditioned covariate drift magnitude and posterior drift magnitude on the Sine1 dataset (All magnitudes shown on Figure 3a, and the magnitudes zoomed to the first drift point are shown on Figure 3b).



(a) `Circles` dataset



(b) `Circles` dataset (x axis zoomed to first drift point)

Figure 4: Concept drift magnitude, covariate drift magnitude, class drift magnitude, conditioned covariate drift magnitude and posterior drift magnitude on the Circles dataset (All magnitudes shown on Figure 4a, and the magnitudes zoomed to the first drift point are shown on Figure 4b).

From Figures 3a and 3b, it can be observed that Webb's method is able to capture the drift that occurs in the SINE1 data stream, with a slight delay. However, the drift magnitude for the drift points is not significantly higher than the drift magnitude for the points of time when there is no drift. This means that if this method needs to be used for responding to drift, it may be hard to find a suitable threshold for distinguishing between drift and non-drift points for a wider range of inputs, without analyzing the inputs individually. The drift magnitude exhibits a noisy behaviour for this data stream, which may be due to the low number of instances for comparison, or as an effect of the other parameters, such as unsuitable number of bins or unsuitable distance measure.

The values of the concept drift magnitude on Figures 4a and 4b indicate that Webb's method is able to capture the drift in the CIRCLES data stream without delay, as there is a significant increase in the drift magnitude in the points of time with drift. In addition, the method locates the source of the drift - it indicates that the drift was caused by changes in the conditional covariate drift, which corresponds to the actual source of the drift, per the description of the data stream in 3.5.

In order to be able to respond to drift, and additionally, to calculate the true positives, false positives and false negatives as described in 3.4, a threshold needs to be defined, so that all points of time where the drift magnitude is over the threshold will be labelled as drift points. Webb et al. [8] don't propose a value for such threshold. Furthermore, different thresholds need to be defined in order to distinguish between abrupt drifts and incremental drifts.

By merely analyzing Figure 3 and Figure 4, it is clear that choosing a threshold value is quite challenging when the drift magnitude is unrevealing. Visual inspection may deem it proper to choose a threshold of 0.6 for abrupt drifts (based on the peaks from Figure 3a) and a threshold of 0.3 for incremental drifts (based on the peaks from Figure 4a). The results from applying these thresholds and calculating the number of true positives, false positives and false negatives for the SINE1 and CIRCLES data streams are shown on Table 2. For calculating these evaluation measures, the acceptable detection delay was set to $\Delta = 250$ for the SINE data stream and to $\Delta = 1000$ for the CIRCLES data stream. These values were chosen as they are recommended and used by the authors of FHDDM [5] and MDDM [6].

	SINE1	CIRCLES
True positives	0	3
False positives	4	0
False negatives	4	0

Table 2: Number of true positives, false positives and false negatives for the SINE1 and CIRCLES data streams

From the table, it can be seen that by choosing a threshold of 0.3 for gradual drifts, the method is able to capture all three gradual drifts in the CIRCLES data stream. On the other hand, choosing a threshold of 0.6 is unsuitable for detecting the abrupt drifts in the SINE1 data stream. There are four points of time when the drift magnitude is above 0.6, however, these occur after the actual drift point (as can be seen on Figure 3a and Figure 3b), thus they are counted as false positives.

To conclude, in this Subsection, it was shown that Webb's method cannot be directly applied to every kind of input. In order to be able to generalize, one needs to investigate the effect of the parameter choices on the reported drift magnitude. Observing how the drift magnitude changes in respect to the parameter values, means that recommendations on the appropriate parameter choices depending on the stream characteristics can be made. This is addressed in the following Subsection.

4.2 Analysis of the effect of the parameter value choices

In this Subsection, an analysis and investigation is performed on the effect of the parameter value choices on the drift magnitude reported by Algorithm 1.

The parameters that need to be analyzed are:

1. **The feature set** - whether the magnitude should be reported from calculations performed on the whole feature set (total drift magnitude) or it should be reported from the per-feature drift magnitudes
2. **The distance measure** for measuring distance between the distributions.
3. **The number of bins** for discretizing the numeric attributes
4. **The number of instances for comparison**, which is the number of instances against which the current batch's drift will be quantified
5. **The smoothing parameter**, in cases when the distance measure is Kullback-Leibler divergence [21]. As it was described in subsection 3.3, this parameter is used to handle the zero density issue that causes an infinite value to be the result of the Kullback-Leibler divergence measure.

To analyze these parameters, the implementation of the method described in Algorithm 1 was modified; instead of fixing the values for *number of bins*, *distance measure* and *number of instances for comparison*, they were defined as parameters to the method. Thus, the values for these parameters can be changed between different runs and their effect on the reported drift magnitude can be investigated.

The three distance measures: symmetric Kullback-Leibler divergence [21] with Laplace smoothing [24], Hellinger distance [22] and Total Variation distance [23] were implemented, as described in subsection 3.3.

For the same reasons outlined in the previous subsection, the investigation of parameter effects was performed on the two synthetic data streams: SINE1 and CIRCLES. Coordinately, the data streams were split in batches of equal size p and the batch size was set to the known length of the drift for the data stream, that is $p = 50$ for SINE1 and $p = 500$ for CIRCLES. However, these two data streams have the same dimensionality and similar characteristics. In order to investigate the effect of parameters in high-dimensional data with different characteristics, it was decided that for this experiment, the LED data stream would be considered as well. The LED data stream consists of 24 features, and a target with 10 class labels, thus its characteristics differ from those of the Sine1 and Circles data streams. The length of the drift for LED is 500, thus the batch size was set to $p = 500$ for this data stream.

In the experiment setup for investigating the effect of the parameters, the values for the parameters were initially fixed to replicate Webb et al.'s [8] setting. Thus, the distance measure was selected to be Total Variation distance [23], the number of bins was set to 5 and the concept drift magnitude was reported based on the whole feature set. However, as mentioned in the previous subsection, in Webb et al.'s setting, the number of instances for comparison is selected arbitrarily and it varies between different data streams. Preliminary experiments and inspections showed that an appropriate number of instances for comparison for the Sine1 data stream is $t = 500$, $t = 1000$ for the Circles data stream, and $t = 1000$ for the LED data stream.

Then, in order to investigate the effects of each parameter on the drift magnitude, the following procedure was employed. One parameter was chosen at a time so that its values were modified, while the values of the other parameters remained fixed to the values of Webb's default setting, described above. This way, results for the per-parameter effect were obtained for each parameter.

In all the plots in this subsection, the X axis represents the point of time in which the batch of p instances has arrived, and the Y axis represents the concept drift magnitude calculated between the distribution of the current batch of p instances and the distribution of the previous t instances. The grey dashed lines represent the points in time when drift has truly occurred, as specified in the ground truth.

The results along with an analysis of the effects of each parameter on the drift magnitude are presented as follows:

A. Effect of the chosen feature set

Figure 5 displays the concept drift magnitude obtained from all features (marked as *all* on the plots) against the drift magnitude obtained from the individual features.

In the case of the Sine1 data stream (Figure 5a) and the Circles data stream (Figure 5b), the drift points are captured by both the drift magnitude calculated based on all features, and by the drift magnitude based on the x feature. The high drift magnitude based on the x feature indicates that the drift has occurred due to changes in the x feature.

On the other hand, for the LED data stream, the drift magnitude calculated based on all features is extremely uninformative, as it is near or equal to the maximum value for Total Variation distance in all points of time, independent of whether there's a drift or not. This behaviour is expected because the distance measures are monotonic as the dimensionality of data increases, as previously discussed. The drift in the LED data stream can be successfully captured from the drift magnitudes calculated from the individual features. For this reason, in the following subsections for investigating the effect of other parameters on the LED data stream, the effect will be evaluated on the per-feature concept drift magnitude instead of on the total concept drift magnitude. To obtain a single value for the drift magnitude at a time step t based on all per-feature drift magnitudes, the maximum of all the per-feature drift magnitudes will be calculated.

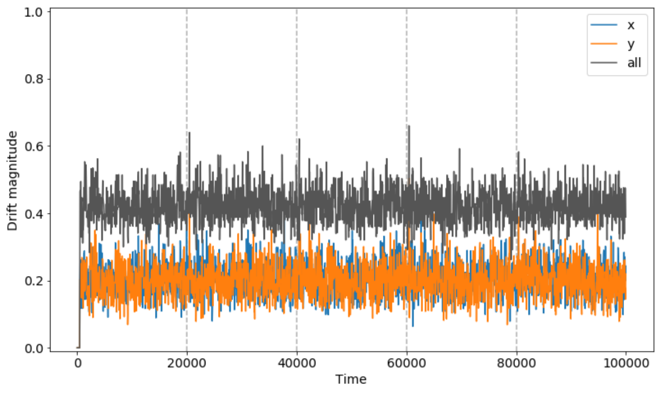
B. Effect of the distance measure

Figure 6 displays the concept drift magnitude obtained from the three distance measures: Kullback-Leibler divergence, Hellinger distance and Total Variation distance, for all three data streams. For the Sine1 (Figure 6a) and Circles (Figure 6b) data streams, each line represents the total concept drift magnitude over time, calculated with one of the three different distance measures. For the LED data stream (Figure 6c), each line represents the maximum of all per-feature concept drift magnitudes over time, calculated with one of the three distance measures.

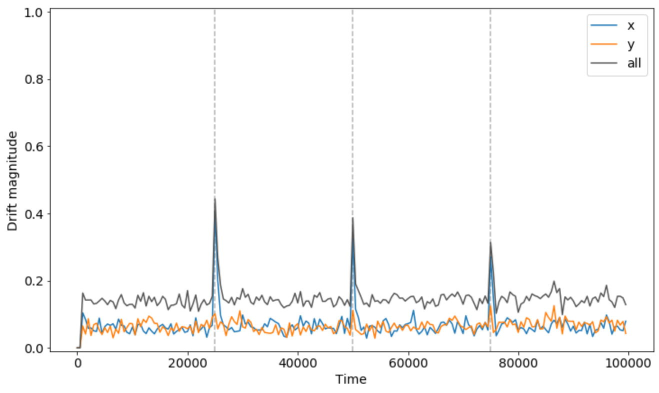
It can be observed that with all three distance measures, the same pattern is discovered, making all three distance measures appropriate if the goal is drift pattern analysis. However, for the needs of responding to drift, where a threshold needs to be applied in order to obtain a single Boolean value that indicates whether a drift has occurred at some point of time, the Kullback-Leibler divergence measure may be the most suitable. Figure 6 shows that the difference in the drift magnitude obtained from the Kullback-Leibler divergence is significantly higher between points of time with no drift and drift points, compared to the other two distance measures, most notably on the Sine1 data stream. From this experiment, we can conclude that in order to make an interpretable drift detector that responds to drift across a range of inputs, it may be deemed appropriate to consider the Kullback-Leibler divergence, instead of the Total Variation distance, as proposed by Webb et al. [8].

C. Effect of the number of bins

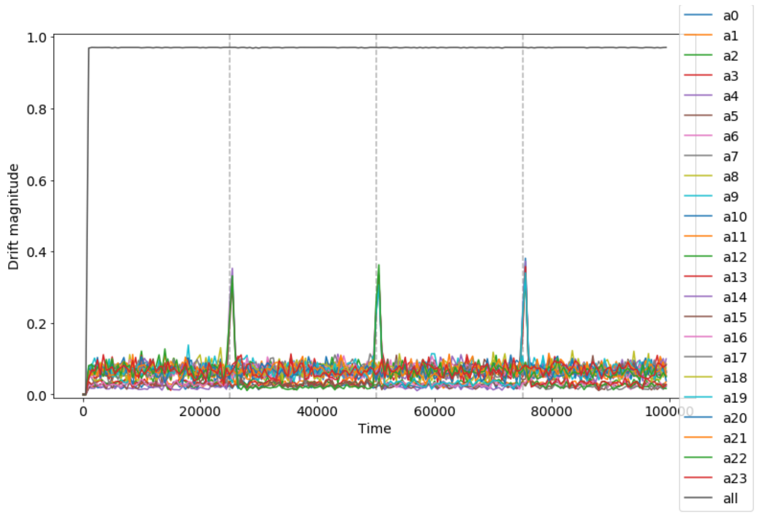
Figure 7 shows the concept drift magnitude obtained from data that has been discretized in n bins, where $n \in [2, 10]$, as indicated in the legends of the plots. This experiment was not performed for the LED data stream, because LED



(a) `Sine1` dataset

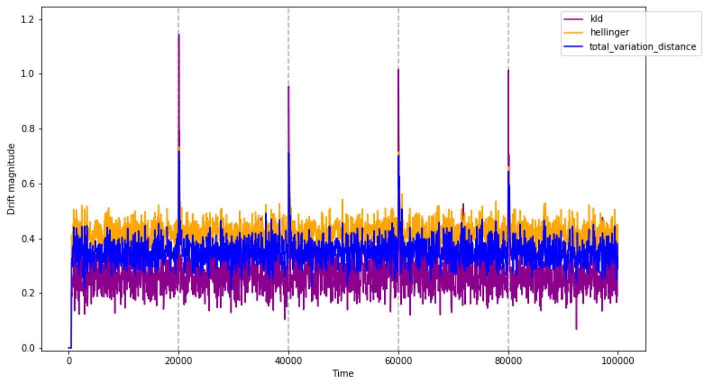


(b) `Circles` dataset

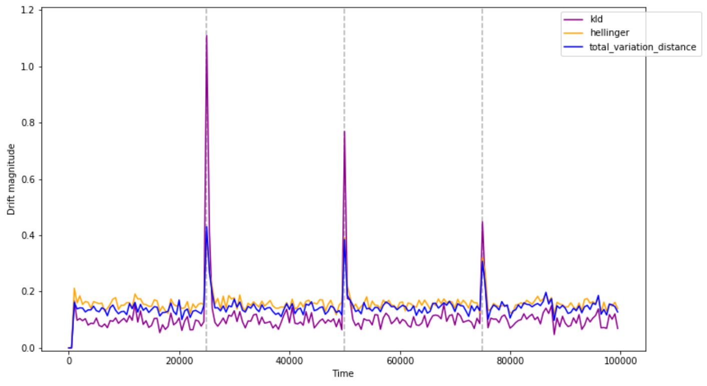


(c) `LED` dataset

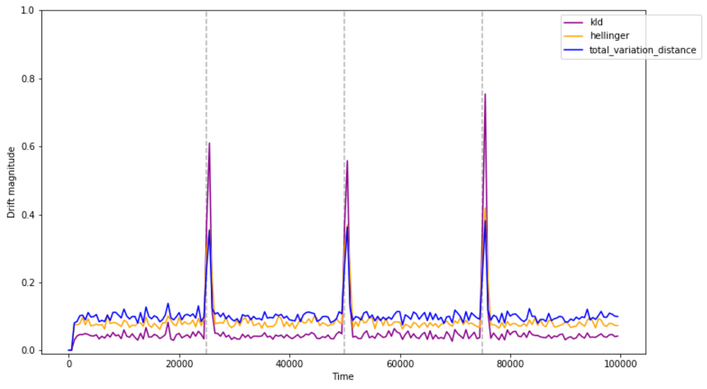
Figure 5: Effect of the selected feature set on the concept drift magnitude for the three synthetic data streams. The different lines in the plot indicate which feature was used for calculating the drift - "all" (the dark grey line in all three plots) means that the drift magnitude was calculated using the whole feature set and the other labels represent the name of the feature based on which the drift magnitude was calculated.



(a) `Sine1` dataset (zoomed in to first drift point)



(b) `Circles` dataset (zoomed in to first drift point)



(c) `LED` dataset

Figure 6: Effect of the selected distance measure on the concept drift magnitude for the three synthetic data streams.

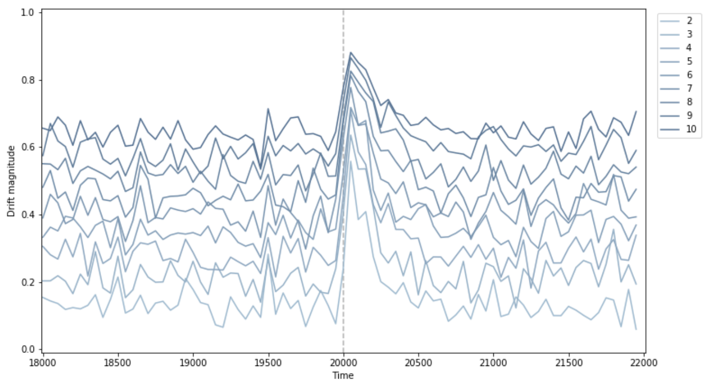
doesn't contain continuous numeric features, thus the effect of the number of bins can not be observed. The plots for both data streams have been zoomed in around the first drift point, for better visualization of the differences in the drift magnitude.

For both the SINE1 (Figure 7a) and CIRCLES (Figure 7b) data streams, each line represents the total concept drift magnitude over time calculated with the numeric data discretized in the number of bins specified in the legends of the plots.

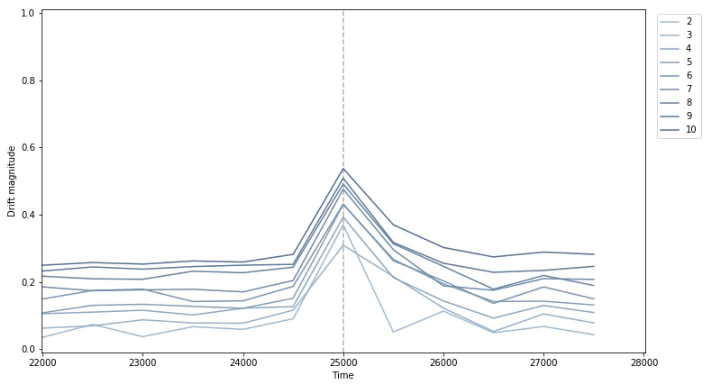
From Figure 7a, it can be observed that increasing the number of bins for discretization in the SINE1 data stream reduces the capacity of the method to distinguish between points with no drift and drift points. In the CIRCLES, the effect of the number of bins is milder - by increasing the number of bins, the overall drift magnitude is increased in every point, but the drift pattern can still be discovered, which can be observed from Figure 7b.

Certainly, this effect is connected to the batch size that is used for obtaining an estimate of the probability distribution. When the batch size is small, as in the case of the SINE1 data stream where $p = 50$, discretizing the numeric values in a large number of bins means that it is likely that the distribution will differ when compared to the distribution of a previous sample, simply due to the random noise between them. The effect of the noise is reduced when small batches are discretized in a smaller number of bins, as can be seen from Figure 7a.

From this experiment, it can be concluded that the appropriate number of bins for discretization needs to be chosen by taking the batch size in account. This observation is further supported by the results from the experiments performed in Section 6.



(a) `Sine1` dataset



(b) `Circles` dataset

Figure 7: Effect of the number of bins for discretizing the numeric values on the concept drift magnitude for the two synthetic data streams.

D. Effect of the number of instances for comparison

Figure 8 displays the concept drift magnitude obtained by comparing the instances from the current batch against the previous t instances, where $t \in 500, 1000, 1500, 2000$, for all three data streams. Smaller values for t were also considered in the experiments, however they are not presented in this thesis as they were extremely uninformative - the small sample sizes couldn't get rid of the effect of noise and made the concept drift magnitude quite unrevealing.

For the Sine1 (Figure 8a) and Circles (Figure 8b) data streams, each line in the plots represents the total concept drift magnitude over time calculated against the number of instances as specified in the legends of the plots. For the LED data stream (Figure 8c), each line represents the maximum of all per-feature concept drift magnitudes over time, calculated against the number of instances as specified in the legend of the plot.

From Figure 8 it can be observed that increasing the number of instances for comparison reduces the noise's strength before the point of drift and increases the difference between the average drift magnitude of non-drift points compared to drift points. However, it can also be observed that comparing against a large number of instances means that the drift is propagated for a long time after it occurred, resulting in high drift magnitude after the actual drift point. In the case when the goal of the method is to respond to drift, this propagation means that the method will report a large number of false positives and cause frequent retraining of the model, which is inefficient.

In order to avoid propagating the drift for a long time, smaller samples for comparison can be considered. However, the capacity to deal with noise is reduced in smaller samples, which can be observed from these synthetic data streams that contain a small amount of noise. Consequently, in real-world data streams that typically contain more noise, comparing against small samples may significantly reduce the ability to distinguish between real drift points and non-drift points. Thus, it is more appropriate to compare against a larger number of instances and deal with the propagation in a different manner. Some ways to deal with the propagation of drift are described in Section 5.

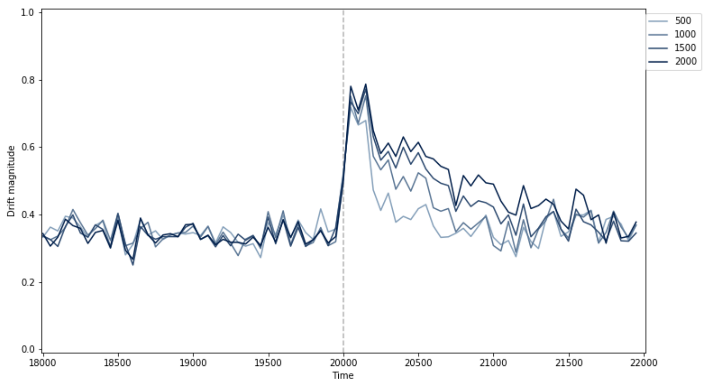
E. Effect of the Laplace smoothing parameter

Figure 9 displays the concept drift magnitude calculated with Kullback-Leibler divergence between the probability distributions obtained by applying Laplace smoothing with $\alpha \in \{0, 0.5, 1\}$.

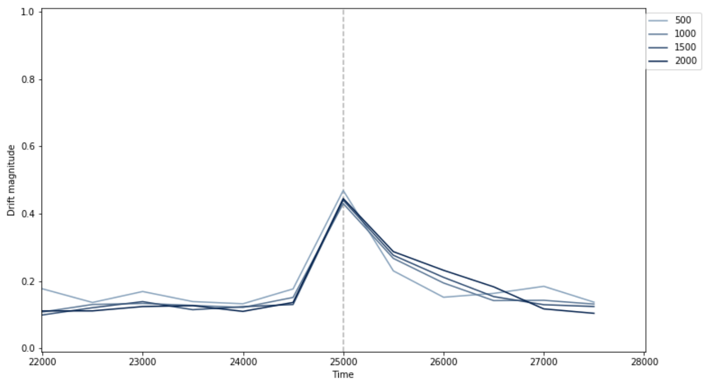
For the Sine1 (Figure 9a) and Circles (Figure 9b) data streams, each line represents the total concept drift magnitude of the data where the smoothing parameter was applied with value specified in the legends of the plots. For the LED data stream (Figure 9c), each line represents the maximum of all per-feature concept drift magnitudes over time, on the data where the smoothing parameter was applied with value specified in the legend of the plot.

From Figure 9c, it can be seen that the smoothing parameter doesn't have an effect on the calculated magnitudes for the LED data stream, as this stream doesn't contain any points with zero density in any of the distributions.

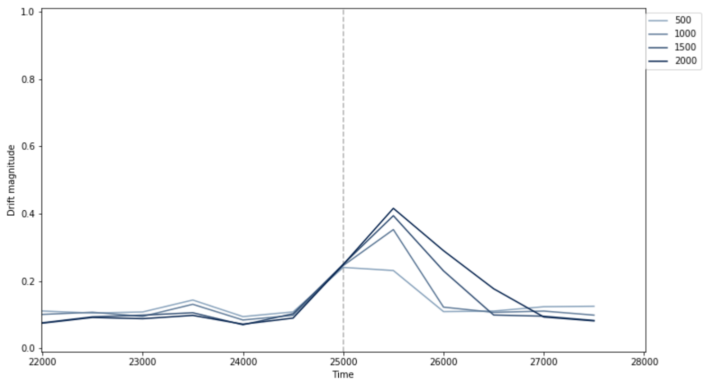
From Figure 9a and Figure 9b, it can be observed that the effect of the smoothing parameter is almost insignificant, as it slightly affects the total drift magnitude, but overall the drift pattern is the same for both data streams. It can be concluded that the value of the smoothing parameter is irrelevant for both exploratory data



(a) `Sine1` dataset



(b) `Circles` dataset



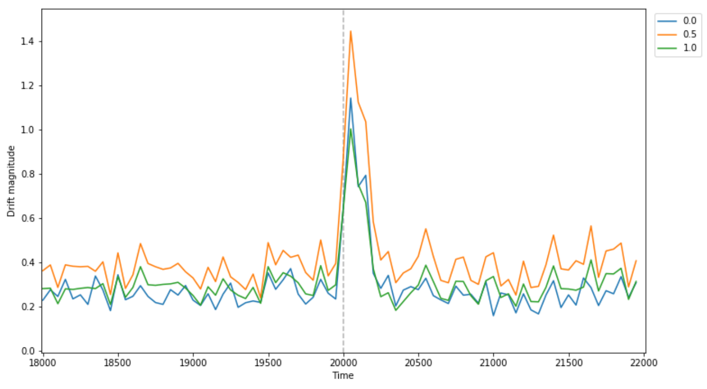
(c) `LED` dataset

Figure 8: Effect of the number of instances for comparison on the concept drift magnitude for the three synthetic data streams.

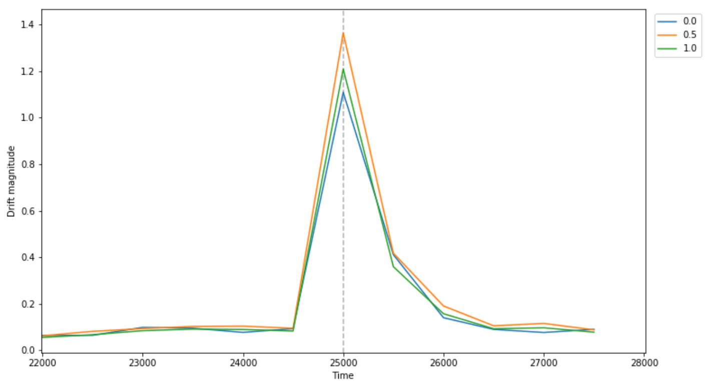
analysis of the drift patterns, as well as for responding to drift. Thus, it can be fixed to a certain for value, for example $\alpha = 1$, which is commonly used in literature [24] and is also called *add-one smoothing*.

Overall, from the analysis of Webb's method and the parameters' effect, it can be concluded that the choices for *chosen feature set*, *number of bins*, *number of instances for comparison* should depend on the stream characteristics. In addition, one needs to find a way to not propagate the old distribution after a drift has occurred.

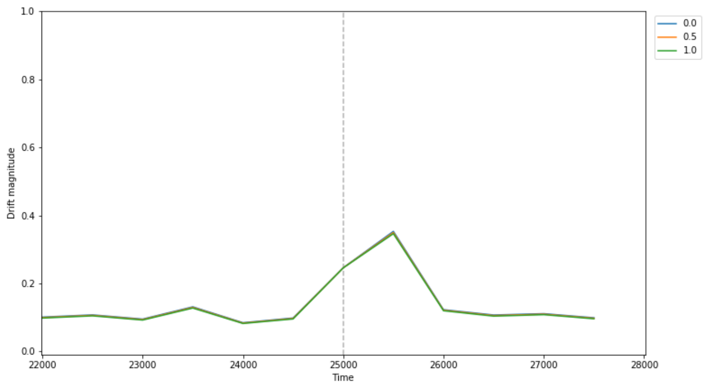
These issues are addressed in the following section, where the Statistical Drift Detection Method is described.



(a) `Sine1` dataset



(b) `Circles` dataset



(c) `LED` dataset

Figure 9: Effect of the Laplace smoothing parameter on the concept drift magnitude for the three synthetic data streams.

5 Statistical drift detection method

This section describes the Statistical Drift Detection Method (SDDM), that can be used for both responding to drift and for discovering drift patterns, and can be applied across a range of inputs. The method builds upon the work done by Webb et al. [8], as described in Section 4.

In this section, it is assumed that Webb et al.'s method [8] is implemented as described in the previous section, and that it outputs concept drift magnitude, covariate drift magnitude, class drift magnitude, conditioned covariate drift magnitude and posterior drift magnitude calculated based on the whole feature set X, and based on the subsets consisted of a single feature. The drift magnitudes calculated on the subsets consisted of a single feature will also be referred to as *per-feature drift magnitudes*. Thus, the overall number of outputs from the method is $5 \cdot (k+1)$ (where k is the number of features) - five drift magnitudes calculated per each feature, and five drift magnitudes calculated based on the distribution of all features. The pairwise marginal distributions were not considered, as it is not feasible to examine every one of them in high-dimensional data streams. However, in practice, the pairwise marginal distributions may provide useful insight [8], and this is discussed in more details in Section 7.

The subsections that follow highlight the modifications that were made on Webb et al.'s method 4, in order to address its drawbacks. More specifically, the following issues are addressed: how to obtain a single Boolean output for responding for drift from the $5 \cdot (k+1)$ current outputs (in order to make the method suitable for handling drift), how to reduce the number of false positives after a drift has been detected, and how to optimize the method to make it more time-efficient.

5.1 Obtaining a flag for responding to drift

Current adaptive data stream learning methods handle drift by incorporating a drift detection method that signals whether a drift has occurred at some point of time [3]. If the drift detection method has signalled that a drift has occurred, the learning method is retrained to reflect the new distribution.

By quantifying the drift as described in Section 4, many outputs are produced. One of the outputs is the overall *posterior drift magnitude*, which quantifies the change between the posterior distributions at time t $P_t(y|X)$ and at time u $P_u(y|X)$. As it was discussed in Section 3.2, the changes between the posterior distributions are referred to as *actual concept drift* and they affect the decision boundary of the model. The adaptive stream learning methods are designed to handle *actual concept drift* by retraining the model [3], thus, in order to provide a flag for these learners that indicates whether actual drift has occurred, the *poste-*

rior drift magnitude should be considered. In Webb et al.'s work [8], the *concept drift magnitude* has been used to analyze and evaluate the drift. However, as it was discussed in 3.2, changes in the concept drift magnitude can be caused by changes that don't affect the decision boundary and that don't need to be handled. Thus, the *posterior drift magnitude* is more appropriate for this task, as it refers to the real concept drift.

The posterior drift magnitude is a real number between $[0, 1]$ in the cases when Total-Variation distance or Hellinger distance are employed; and a real number in the range $[0, \infty)$ when the Kullback-Leibler divergence is calculated. In order to be incorporated in the adaptive stream learning methods, this output needs to be mapped to a single Boolean value that designates whether a drift has occurred.

When the posterior drift magnitude is calculated on a low-dimensional data stream, a flag that indicates whether a drift has occurred at a point of time can be trivially obtained by applying a threshold, so that all points of time where the posterior drift magnitude is above the threshold are flagged as drift points. Selecting a proper value for the threshold, however, is not a straightforward task because it depends on the type of drift that occurs, and it is discussed in detail in Subsection 6.1.

On the other hand, when the posterior drift magnitude is calculated on a high-dimensional data stream, the magnitude value is invariable and independent of whether a drift has truly occurred or not, due to the monotonicity of the distance measures. This effect was detected in the work of Webb et al. [8], and was observed in the analysis of the high-dimensional LED data stream performed in Section 4 (Figure 5c).

Consequently, in the case of high-dimensional data streams, the per-feature posterior drift magnitudes are more appropriate. From the per-feature posterior distributions, k magnitude values are obtained, where k is the number of features. To obtain a single magnitude from the per-feature magnitudes, the maximum of all per-feature posterior magnitudes can be calculated:

$$max(\sigma(P_t(y|X_i), P_u(y|X_i))), \forall i \in [1, k],$$

Then, a threshold should be applied on this magnitude, just as for the total posterior drift magnitude, in order to obtain flags that indicate whether a drift has occurred.

By using the maximum of all per-feature posterior drift magnitudes, it is certain that the learning method will respond to changes in any of the features. However, not every feature is equally important in the decision boundary, meaning that large changes in the distribution of one feature may not affect the overall decision boundary significantly. Consequently, using the maximum of the per-feature posterior drift magnitudes for responding to drift may cause a lot of false positives, which in return will cause frequent retraining of the learning model that is a resource-expensive operation. In order to overcome this issue, instead of the maximum of the per-feature drift magnitudes, one could obtain a weighted average of the per-feature magnitudes, where the weights would represent the features' importance. Calculating feature's importance is outside of the scope of this thesis, thus, in the rest of the thesis the posterior drift magnitude for high-dimensional data will be calculated based on the maximum of all per-feature posterior drift magnitudes, which assumes that all features have the same importance.

Most importantly, in addition to the flag for responding to drift, all drift magnitudes, including concept drift magnitude, covariate drift magnitude, class drift magnitude, conditioned covariate drift magnitude and posterior drift magnitude, based on all features, and per-feature magnitudes, are outputted by the method so they can be used for interpreting the causes of drift and long-term drift pattern analysis.

5.2 Reducing the propagation of old drifts

In Section 4, it was observed that choosing a higher number of instances for comparison t increases the method's robustness against noise. However, it also enables propagation of the drift that occurred at time t_{drift} to the next t instances, due to the propagation of instances from the old distribution. Thus, the method reports a high drift magnitude in the time period $[t_{drift}, t_{drift} + \frac{t}{p}]$, where p is the number of instances in a single batch.

Let's consider that the batches of instances arriving at times $[0, t_{drift})$ have a similar distribution, thus the method reports a low drift magnitude at these times. If at time t_{drift} a batch of p instances arrives with a distribution significantly different compared to the distribution of the previous t instances, the method will report a high drift magnitude which indicates that the data has started drifting. In cases of real drift, it is expected that the batch of p instances that arrives at $t_{drift} + 1$ will have a distribution similar to the one at point t_{drift}. At the point of time $t_{drift+1}$, the model doesn't need to be retrained, as it was already retrained in the previous point of time to reflect the new distribution. However, because we compare the distribution of the current p instances against the previous t instances, when $t >> p$, the majority of t instances for comparison will reflect the old distribution and thus the method will report a high drift magnitude. In practice, this means that false positives will be reported by the method after the actual drift point, which will cause the model to be retrained unnecessarily.

On the other hand, as it was observed in the previous section, choosing a lower number of instances for comparison makes the method susceptible to noise and decreases its capacity to distinguish between drift and non-drift points. Thus, lowering the number of instances for comparison is not desirable.

In order to preserve the method's robustness to noise and simultaneously decrease the propagation of the old distribution prior to the drift, two approaches were considered.

With the first approach, the t instances previous to the current batch of p instances were split into m batches of n instances, so that $n \geq p$. Then, the drift magnitude was calculated between the distribution of the current p instances and the distribution of each of the m batches separately. This resulted in m different drift magnitudes, and to obtain an overall drift magnitude, the minimum of the m drift magnitudes was chosen. Considering the minimum ensures that the effect of noise is reduced and that when drift occurs, the old distribution doesn't have a significant effect on the reported drift magnitude for the batch that arrives after the real drift point. Numeric experiments showed that this approach was successful in lowering the number of false positives while simultaneously detecting abrupt and incremental drifts on time. However, it can be observed that this approach cannot detect a reoccurring drift if it reoccurs anywhere between the last drift point and the $t - p$ instances after it $(t_{drift}, t_{drift} + \frac{t-p}{p})$. When reoccurring drift occurs in that interval (meaning that a previously active concept reappears), the minimum of all drift magnitudes will be lower than the threshold, because the distribution of the current batch will be similar to the distribution of some of the older batches, and thus this type of drift won't be detected.

In order to successfully detect reoccurring drifts as well, a new approach was tailored. With this approach, the drift magnitude is calculated between the distribution of the current p instances and the previous t instances. However, if a drift occurs at a point of time t_{drift}, then the drift magnitude for the next $\frac{t}{p}$ batches will be calculated only with instances that arrived after the drift has occurred. More specifically, the drift magnitude for a batch of instances that has arrived at a point of time t_n will be calculated against the previous $min(t, t_n - t_{last\_drift\_point})$ instances. This "resetting" approach ensures that the old distribution won't be propagated after the drift point, as the instances from the old distribution are not used in the calculations. Furthermore, because the instances from the old distribution are not used in the calculations that follow, reoccurring drifts won't be dismissed. Finally, with this approach, most of the calculations will be performed with a high number of instances for comparison t, thus the robustness to noise will be preserved. Initial numeric experiments on synthetic data streams showed that this approach is as good as the previous one in detecting abrupt and incremental drifts, while simultaneously being vastly superior in detecting reoccurring drifts. More details about the experimental results are given in Section 6.

5.3 Optimizing the method

In the method thus far, many time-consuming operations are repeated multiple times, such as discretizing the numeric features and aggregating the data by values to obtain value counts.

If the data consists of k features and n instances, the time complexity of the discretization is $\mathcal{O}(k \cdot n)$, assuming that all k features are numeric. The time complexity of aggregation is $\mathcal{O}(n)$ when the total drift magnitude is computed based on all features, and $\mathcal{O}(k \cdot n)$ when calculating drift magnitude in the marginal distributions consisted of a single covariate.

Let's consider a data stream where the data comes in batches of n instances, and we need to detect the drift of the latest batch with respect to the previous m batches, where $m \geq 1$ and $m \in \mathbb{N}$. This means that for each batch, the discretization of features and obtaining value counts would be repeated $m + 1$ times: once when the batch arrives, and m times for the next m batches it would be compared against. Thus, the time complexity of discretization of a batch of n instances and k features would increase to $\mathcal{O}((m + 1) \cdot k \cdot n)$, and the time complexity for computing the per-covariate drift magnitude would also become $\mathcal{O}((m + 1) \cdot k \cdot n)$. For high-dimensional data, this increase in time complexity makes the algorithm inefficient.

Since all the operations related to measuring drift magnitude are performed on the discretized and aggregated data, in order to avoid performing the same operations multiple times, the discretized and aggregated data can be stored. With this optimization, the repetitive operations will be performed only once, when the batch of instances arrives. After the batch is discretized and aggregated, the data will be stored, so when a new batch arrives, instead of discretizing and aggregating the previous m batches, this information can be retrieved from storage. This technique reduces the time complexity of both discretization and aggregation to $\mathcal{O}(k \cdot n)$, however, it requires additional storage space.

If all the features are numeric, the necessary storage for a single batch of instances would be $k \cdot b$, where b is the number of bins and $b \geq 2, b \ll n, b \in \mathbb{N}$. However, if all features are categorical and consist of practically unique values, the required storage would be close to $k \cdot n$, that is, as much space as it takes to store the original batch, meaning that the storage requirements for the system would increase twofold. To avoid using too much additional storage, all discretized and aggregated data files prior to the last m can be deleted from storage as a new batch arrives. This additional optimization means that storing the discretized and aggregated batches would require at most $m \cdot k \cdot n$ additional storage.

In general, before applying this optimization, one needs to consider the time complexity vs storage trade-off. If the environment has limited storage resources and/or the computational time is not critical, then applying this optimization is unnecessary. However, in the case when storage resources are not constrained, the data is high-dimensional and consists of mostly non-unique values, and/or quick running time is crucial, applying this optimization is beneficial and highly recommended. During the work on this thesis, the storage resources were not constrained, thus, all experiments described in the rest of the thesis were performed with the optimization included.

6 Experiments and results

This section describes the experiments that were performed in order to determine the appropriate values for the method parameters, and the experiments performed for evaluating the effectiveness of the proposed method.

As it was observed in Section 4, the choice of values for the method parameters influences the reported drift magnitude, and they should be chosen depending on the data stream characteristics. In order to determine the proper choice of parameter values, experiments on different synthetic data streams with various characteristics are performed.

The effectiveness of the method on the synthetic data streams is evaluated by its ability to capture the drift pattern with a low detection delay, high true positive rate and low false positive and low false negative rate, as well as the method's ability to point to the source of the drift, as specified in the ground truth. This evaluation is performed to determine how well the statistical drift detection method can detect drifts, and how does it compare against current state-of-the-art methods.

On the real-world data streams, where the ground-truth is not available, the effectiveness of the method is evaluated by analyzing the drift pattern and drift causes, and determining whether the results comply with known changes in that specific real-world domain.

6.1 Choosing appropriate parameter values based on the characteristics of the data stream

As it was concluded in Section 4, the values for the parameters *distance measure*, *chosen feature set*, *number of bins* and *number of instances for comparison* should be chosen depending on the data stream characteristics. Choosing the appropriate values for these parameters is of great importance as they affect the reported drift magnitude and it depends on them whether the drift patterns are going to be discovered and whether a threshold can be applied to distinguish between drift and non-drift points. However, it can be quite challenging to determine these values in a way that would be applicable to a wider range of inputs. The four synthetic data streams that were described in Subsection 3.5 are not enough to draw general conclusions for the parameter values. For one, the number of data streams is too low. In addition, three of the four data streams: SINE1, MIXED and CIRCLES have very similar data stream characteristics as they all have the same number of instances and similar number of features.

In order to be able to make recommendations for the parameter values based on the stream characteristics, many data streams with varying characteristics are necessary. However, none of the related work on the topic [1, 2, 5, 6, 11, 12, 8, 28, 29] has described or utilized such data streams.

Therefore, for the purpose of choosing the appropriate parameter values, it was decided that new synthetic data streams with various characteristics will be generated. An additional advantage to creating new synthetic data streams and tuning the parameters solely on these data streams, is that one can test how well the parameter values were chosen by applying them on the synthetic data streams that are widely used in the literature and described in Subsection 3.5.

The data streams that were generated were partially inspired by the LED data stream, described in Subsection 3.5. However, in order to be able to choose the appropriate number of bins, the features were generated to consist of numeric values, instead of categorical values. The following stream characteristics were varied in order to create contrasting data streams:

1. **Number of features**: {2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100}
2. **Batch size**: [100, 10100] with step 100
3. **Number of classes**: {2, 10}
4. **Type of drift**: {abrupt, incremental}.

All the combinations of characteristics were considered, therefore the total number of data streams that were generated is $22 \cdot 100 \cdot 2 \cdot 2 = 8,800$.

The feature values were generated to be real values uniformly distributed between 0 and 1.

Similarly to the original LED data stream, the instances were classified according to the values in the "relevant" features. The number of relevant features for a data stream was chosen randomly as a number between 1 and half the total number of features of that particular data stream - $[1, \lfloor k/2 \rfloor]$.

In the case with two classes, an instance was classified as positive if the average value of the relevant features is greater than 0.5, and as negative otherwise:

$$y = \begin{cases} 1, & \text{if } \frac{\sum\limits_{x' \in X'} x'}{|X'|} > 0.5. \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

,

where X' represents the set of relevant features for that data stream.

In the case of ten classes, the classification function is the following:

$$y = k, \text{if } \frac{k}{10} \leq \frac{\sum\limits_{x' \in X'} x'}{|X'|} < \frac{k+1}{10}, \text{ where } k \in [0, 9] \tag{12}$$

Each data stream consists of 100 batches. The drift is simulated by interchanging the relevant attributes at every twenty batches with a transition length of a single batch, which corresponds to four drift points per data stream. For incremental drift, the probability of belonging in a new context during the start of the transition is low and slowly gets higher during the transition. For abrupt drift, the probability of belonging in a new context is high from the beginning of the transition period. To evaluate the robustness of the method against noisy data, 10% class noise was added to each data stream.

After the $8,800$ data streams were generated, the method was run on each data stream with different parameter value combinations for the:

- Number of bins: $[2, 10]$
- Number of instances for comparison: $[b, 20 \cdot b]$ with step b, where b is the batch size for the particular data stream
- Distance measure: Hellinger distance, Total Variation distance or Kullback-Leibler divergence
- Chosen feature set: drift magnitude obtained from all features or maximum of per-feature magnitudes.

Consequently, each data stream was run $9 \cdot 19 \cdot 3 \cdot 1 = 513$ times, meaning that a total of $8,800 \cdot 513 = 4,514,400$ runs were performed. In average, each run took approximately 40 milliseconds to complete, thus the overall running time of the whole experiment was 50 hours.

After the experiments for each data stream have completed, the true positive rate, false positive rate and false negative rate were calculated for each run, as discussed in Subsection 3.4. However, as aforementioned, when calculating these evaluation measures, one needs to define a threshold so that the drift magnitude can be translated into a Boolean value that indicates whether a drift has occurred or not. Since the previous related work [8] doesn't specify a value for such threshold, the evaluation measures were calculated with 9 different values for the threshold: $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. This way, the appropriate threshold could be chosen as well.

The Δ parameter for evaluation was set to a single batch, meaning that a very small detection delay was considered acceptable. After calculating the evaluation measures, a run was labelled as positive if the true positive rate corresponded to the actual number of drift points and the false positive and false negative rate were equal to zero; otherwise, the run was labelled as negative. From all the positive runs, the results were summarized according to the data stream characteristics and the summary of the results can be seen in Table 3 .

Table 3 represents a recommendation for the parameter values based on the number of features and batch size of the stream, as well as the type of drift that is present in the stream. From the table, several things can be observed. First, for all data streams, independent of their characteristics, the Kullback-Leibler divergence was the most appropriate distance measure that is able to discover the drift patterns as specified in the ground truth. Second, the appropriate number of bins for discretization and the number of instances for comparison depend on the size of the batch, a pattern that was already observed in the experiments performed in Section 4. Furthermore, for low-dimensional data streams (in this case, data streams with less than ten features), it is most well-suited to calculate the drift magnitude based on all features, whereas for higher-dimensional data streams, it's more suitable to calculate the drift magnitude as the maximum of all per-feature drift magnitudes; something that was also previously observed in Section 4. Finally, it can be concluded that for these data streams, a threshold of 0.6 can distinguish between non-drift points and abrupt drift points, and a threshold of 0.3 can distinguish between non-drift points and incremental drift points.

One needs to note that the appropriate parameter values listed in Table 3 represent a rule of thumb, and not absolute ground-truth for choosing the parameter values. The reason for this is that these values were chosen based on a finite set of data streams that doesn't cover all the possible data stream characteristics. This is discussed in more details in Section 7.

Data stream characteristics			Parameter choices				
Number of features	Batch size	Type of drift	Distance measure	Number of bins	Number of instances for comparison	Chosen feature set	Threshold
< 10	< 500	Abrupt	Kullback-Leibler divergence	[2,5]	$[15, 20] \cdot b$	Whole feature set	0.6
		Gradual	Kullback-Leibler divergence	[2,5]	$[15, 20] \cdot b$	Whole feature set	0.3
	≥ 500	Abrupt	Kullback-Leibler divergence	[2,10]	$[2, 10] \cdot b$	Whole feature set	0.6
		Gradual	Kullback-Leibler divergence	[2,10]	$[2, 10] \cdot b$	Whole feature set	0.3
≥ 10	< 500	Abrupt	Kullback-Leibler divergence	[2,5]	$[15, 20] \cdot b$	Maximum of per-feature magnitudes	0.6
		Gradual	Kullback-Leibler divergence	[2,5]	$[15, 20] \cdot b$	Maximum of per-feature magnitudes	0.3
	≥ 500	Abrupt	Kullback-Leibler divergence	[2,10]	$[2, 10] \cdot b$	Maximum of per-feature magnitudes	0.6
		Gradual	Kullback-Leibler divergence	[2,10]	$[2, 10] \cdot b$	Maximum of per-feature magnitudes	0.3

Table 3: Recommendation for the parameter choices based on the data stream characteristics

6.2 Evaluation of the method on synthetic datasets

The experiments in this section were performed on the four synthetic data streams Sine1, Mixed, Circles and LED, that were described in Subsection 3.5 and are extensively used in related work [1, 2, 5, 6, 11, 12, 13, 28, 29] for evaluating the performance of concept drift detectors.

It was decided that the experimental results will be compared against results obtained from the FHDDM [5] and MDDM [6] methods, which are currently considered state-of-the-art, as they outperform all other methods that were described in Section 2.

The experimental settings for running the Statistical Drift Detection Method (SDDM) proposed in this thesis were set as follows. For each of the four synthetic data streams, the recommended parameter values from Table 3 were considered. In this table, the recommended values for *number of bins* and *number of instances for comparison* are not single values, but instead ranges of values. Therefore, it was decided that all the values from the range for the appropriate data stream characteristics will be considered and combined, and then the results will be averaged. Thus, SDDM was run $4 \cdot 6 = 24$ times for the SINE1 and Mixed data streams, and $9 \cdot 8 = 81$ times for the CIRCLES and LED data streams.

The FHDDM and MDDM methods were run using the MOA framework [30], which is a framework for data stream mining. The methods' parameters were set as specified in the original papers [5, 6]; that is, the Hoeffding Tree (HT) [31] and Naive Bayes (NB) were used as the incremental classifiers, and the instances were processed prequentially, which means that they are first tested and then used for training. In addition, the experiment for each data stream with the FHDDM and the MDDM method on each data stream was repeated 100 times, as specified in the original papers [5, 6]. For the MDDM method, all three variants of the method were evaluated - $MDDM_A$, $MDDM_G$ and $MDDM_E$.

After running the SDDM method proposed in this thesis and the FHDDM and MDDM methods, the true positive rates, false positive rates and false negative rates were calculated and then averaged over all runs for each "method + data stream" combination.

For the SDDM method, the rates were calculated by first applying a threshold of 0.6 for the SINE1 and MIXED data streams, and a threshold of 0.3 for the CIRCLES and LED data streams, as per the recommendations in Table 3. The acceptable drift detection delay length Δ was set to 250 instances for the SINE1 and MIXED data streams, and to 1000 instances for the CIRCLES and LED data streams, values that are suggested by the authors of FHDDM [5] and MDDM [6].

First, the method's ability to capture abrupt drifts is presented.

Table 4 presents the results on the data streams with abrupt drift - the SINE1 and MIXED data streams. In the table, the (HT) and (NB) that follow the method names refer to which incremental classifier was used - the Hoeffding Tree (HT) or Naive Bayes (NB).

Method	Sine1			Mixed		
	TPR	FPR	FNR	TPR	FPR	FNR
SDDM	**4**	**0**	**0**	**4**	**0.16**	**0**
FHDDM (HT)	4	0.1	0	4	0.65	0
FHDDM (NB)	4	0.04	0	4	0.25	0
MDDD$_A$ (HT)	4	0.21	0	4	1.11	0
MDDD$_A$ (NB)	4	0.13	0	4	0.69	0
MDDD$_G$ (HT)	4	0.2	0	4	1.19	0
MDDD$_G$ (NB)	4	0.14	0	4	0.7	0
MDDD$_E$ (HT)	4	0.2	0	4	1.19	0
MDDD$_E$ (NB)	4	0.14	0	4	0.7	0

Table 4: Average true positive rate (TPR), false positive rate (FPR) and false negative rate (FNR) produced by each method on the Sine1 and Mixed data streams

The number of drifts in both the Sine1 and Mixed data streams is four, as described in Subsection 3.5. From Table 4 it can be observed that all methods, including the SDDM method that was proposed in this thesis, have managed to capture all drift points, as the true positive rate for all methods is equal to four. From the table it can also be observed that the SDDM method has the lowest false positive rate compared to all other current state-of-the-art methods.

Next, the method's ability to capture incremental drifts is presented. Table 5 summarizes the results on the data streams with incremental drift - the Circles and LED data stream.

Method	Circles			LED		
	TPR	FPR	FNR	TPR	FPR	FNR
SDDM	**3**	**0**	**0**	**3**	**0**	**0**
FHDDM (HT)	3	0.15	0	2.97	0.03	0.03
FHDDM (NB)	2.96	0.39	0.04	2.97	0.03	0.03
MDDD$_A$ (HT)	3	0.24	0	2.98	0.03	0.02
MDDD$_A$ (NB)	2.95	0.58	0.05	2.98	0.03	0.02
MDDD$_G$ (HT)	3	0.33	0	2.98	0.03	0.02
MDDD$_G$ (NB)	2.94	0.8	0.06	2.98	0.03	0.02
MDDD$_E$ (HT)	3	0.31	0	2.98	0.03	0.02
MDDD$_E$ (NB)	2.94	0.77	0.06	2.98	0.03	0.02

Table 5: Average true positive rate (TPR), false positive rate (FPR) and false negative rate (FNR) produced by each method on the Circles and LED data streams

The number of drifts in both the Circles and LED data streams is three, as described in Subsection 3.5. From Table 5 it can be observed that while all methods are able to able to capture the drift patterns in these synthetic data streams with incremental drift, the SDDM method captures the drifts with lowest false positive and lowest false negative rates.

From the results presented in Table 4 and Table 5, two conclusions can be made. First, the high true positive rate and low false positive rate indicate that the parameter value recommendations from Table 3 were proper choices for these data streams. Second, the SDDM outperforms current state-of-the-art methods in detecting both abrupt and incremental drifts.

However, the increase in detection performance when compared to related work isn't the biggest advantage of SDDM. As it was previously mentioned, methods such as FHDDM and MDDM merely flag whether a drift has occurred at a point of time, while SDDM produces an extensive output that can be used to locate the exact source of drift. For all four synthetic data streams, the output of all method runs was studied by analyzing the per-feature drift magnitudes and the drift magnitude obtained from the whole feature set. The analysis included finding all drift magnitudes in each point of time that were higher than the specified threshold and evaluating whether the feature(s) from which the magnitude was obtained corresponded to the true source of the drift, as specified in the ground truth and described in 3.5. For all runs and all data streams, the predicted source of drift corresponded to the actual source of the drift.

The experimental results in this subsection show that in addition to SDDM outperforming the current state-of-the-art methods, its extensive output accurately describes the exact nature of the drift. These results indicate that the current approach of designing concept drift detectors can be substituted with statistical methods that have an additional benefit of interpreting the cause of the drift, without compromising the performance of drift handling.

6.3 Evaluation of the method on real-world datasets

The experiments in this section were performed on the two real-world data streams *Airlines* and *Electricity* that were described in Section 3.5. As it was previously discussed, the location and duration of the drift points is not known for these data streams, therefore, the true positive rates, false positive rates and false negative rates can't be determined. This means that one can not evaluate the method on these data streams and compare it against the performance of other methods.

Instead, the method can be run on these data streams and its output can be analyzed to discover drift patterns and determine whether the patterns can be explained by events that have occurred in the specific real-world domain. This approach is also applied by Webb et al. [8] when they evaluate their method on real-world data streams.

For both data streams, the sizes of the batches in which the instances arrive in are not specified, therefore, as in the work of Webb et al. [8], it was assumed that the data comes in batches on a daily basis.

First, the experiment settings and the results for the Electricity data stream are described.

The Electricity data stream, as discussed in Section 3.5, consists of six features which record the price and demand of electricity in two Australian states, as well as the amount of power that is transferred between the states. The target indicates whether the transfer price has increased or decreased relative to the average of the previous day. If it's assumed that the instances arrive in batches once a day, the number of instances per batch in this data stream is 48 - the instances are generated every half hour. According to this stream's characteristics, which has less than 10 features and less than 500 instances per batch and per the recommendations in Table 3, the number of bins that should be used is between 2 and 5, and the chosen number of instances for comparison should be between $[15, 20] \cdot b$, where b is the batch size. As in the previous subsection, the method was run with all parameter value combinations, that is, the method was run for a total of $4 \cdot 6 = 24$ times for the Electricity data stream. The standard deviation of the posterior drift magnitude between the different runs was less than 0.02 in all points of time, meaning that all different parameter settings captured approximately the same pattern. Next, instead of taking the average drift magnitude for each point of time from all method runs, it was decided to present the posterior drift magnitudes as reported by one of the combinations of parameter values computed above, to replicate real-world settings where it's likely that the method will be run once with one set of parameter values.

Figure 10 shows the per-feature posterior drift magnitudes and the total posterior drift magnitude (obtained from all features), when the method was run with 5 discretization bins and $20 \cdot b$ as the number of instances for comparison. The X axis on the figure represents the day from the data stream. The first day in the data stream is 7 May 1996, however, on the plot, the first day is 27 May 1996, which is due to the chosen number of instances for comparison - 27 May 1996 is the first day in the stream that can be evaluated against 20 previous batches. The Y axis on the figure represents the value of the per-feature posterior drift magnitudes and the total posterior drift magnitude.

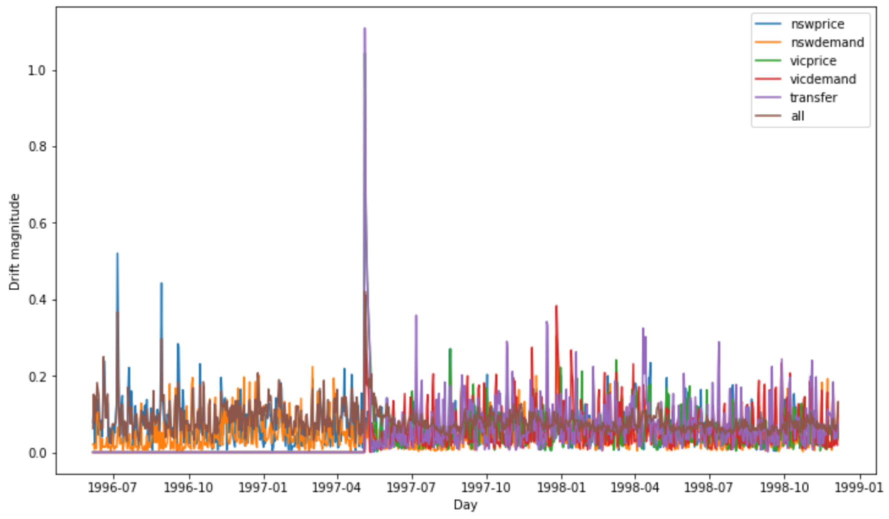


Figure 10: Per-feature posterior drift magnitudes and total posterior drift magnitude on the ELECTRICITY data stream

From Figure 10 it can be observed that a very significant drift has occurred on 2 May 1997, caused by large changes in the $transfer$, *vicdemand* and *vicprice* features, as the high drift magnitude for these features suggests. Descriptive statistics of the data prior to the drift and after the drift shows that the values for these three features are constant before 2 May 1997 and start to vary after this date. Research shows that on 2 May 1997, the national electricity market was introduced, which allowed wholesale electricity sales between the Australian states [8, 32], and explains the abrupt drift in this point of time. Webb et al. [8] have discovered the same abrupt drift in their experiments, however, the values for the drift magnitude long after the abrupt drift point are also significantly high and may indicate drift, which is not supported by the data or by reports from the specific real-world domain [32]. The reason for the observed behaviour in Webb et al.'s work is due to several factors, such as allowing the drift in a past point to propagate in future points when using a high number of instances for comparison, as well as different values for the parameters, for instance, using different distance measure.

The experiment on the ELECTRICITY data stream shows that the method can locate drifts in real-world data streams which can be linked to actual changes in that specific domain. Furthermore, the method can point to the exact source of the drift, which in this case was changes in the $transfer$, *vicdemand* and *vicprice* features. In addition, the method seems to be less sensitive to noise when compared to Webb et al.'s method, because the different combinations of parameters have lead to the same pattern being found, but also because the changes that were observed by SDDM were supported by actual changes in the real-world domain, while there is no support for the high drift magnitude in the points after 2 May

1997 which was reported in Webb et al.'s work.

Next, the experiment settings and the results for the Airlines data stream will be described.

The Airlines data stream, as discussed previously, consists of seven features that describe a flight and a target variable that indicates whether the flight has arrived on time. If it's assumed that the instances arrive in batches on a daily basis, the average number of instances per batch in this data stream is 17399.5. The number of instances fluctuates between different days according to the number of flights, with a standard deviation of 1353.2. According to this stream's characteristics, which has less than 10 features and more than 500 instances per batch per the recommendations in Table 3, the number of bins that should be used is between 2 and 10, and the chosen number of instances for comparison should be between $[2, 10] \cdot b$, where b is the batch size. As in the previous subsection, the method was run with all parameter value combinations, that is, the method was run for a total of $9 \cdot 9 = 81$ times for the Airlines data stream. The standard deviation of the posterior drift magnitude between the different runs was less than 0.01 in all points of time, meaning that all different parameter settings captured approximately the same pattern. As discussed before, instead of taking the average drift magnitude for each point of time from all method runs, it was decided to present the drift magnitudes as reported by one combination of parameter values.

Figure 11 shows the per-feature posterior drift magnitudes and the total posterior drift magnitude (obtained from all features), when the method was run with 10 as the number of bins and $2 \cdot b$ as the number of instances for comparison. The X axis on the figure represents the sequential number of the day in the data - the exact dates from which the data is collected in unknown and instead the days were labelled with integers starting from 0. It is also known from the data that the first day (day 0) is a Tuesday . In the figure, the first point represents the day labelled with 2 (third day) - as this is the first day for which the magnitude can be calculated with the selected granularity.

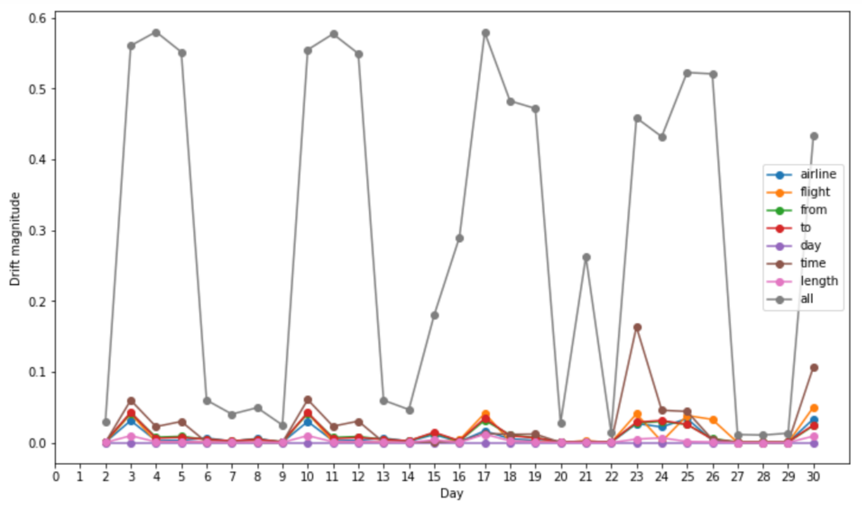


Figure 11: Per-feature posterior drift magnitudes and total posterior drift magnitude on the AIRLINES data stream

From Figure 11, a cyclical pattern of the drift magnitude can be observed, most notably in the first two weeks. In these weeks, it can be seen that a very significant drift occurs on Friday (day 3 and day 10), on Saturday (day 4 and day 11) and on Sunday (day 5 and day 12), while the distribution is consistent during the weekdays. During the next two weeks, the pattern slightly changes, however, significant peaks can still be observed around the weekends. Another observation that can be made is that the drift occurs as the result of a change in the joint posterior distribution, because not a single one of the per-feature drift magnitudes is large enough to be considered the source of the drift. Unlike for the ELECTRICITY data stream, the exact dates of the data are not known, meaning that one can not link the observed drift pattern to actual changes that may have occurred in the specific real-world domain. However, this experiment shows the power of the statistical drift detection method in discovering drift patterns, that can prove useful in handling and detecting future drifts, and which are not possible with the performance-based methods such as FHDDM [5] and MDDM [6]. For example, in this particular instance, if such cyclical pattern is observed in the data regularly, it may point out to the need of using multiple machine learning models to model the data - one model that would be trained on data from weekdays and three additional ones - one for each day of the weekend.

7 Conclusion and future research

The environment is constantly evolving and changing, and machine learning models need to adapt to these changes in order to maintain their performance. In the real world, the capacity of handling concept drifts by merely detecting them can be further expanded by analyzing the nature and causes of the underlying changes. Understanding the factors that have caused drifts to occur may point out to when future drifts will occur, meaning that the machine learning algorithm can be adjusted for these scenarios.

In this thesis, the Statistical Drift Detection Method (SDDM) for detecting and describing concept drifts was introduced and described. It was shown how the SDDM can be used in order to address two challenges in the concept drift field: concept drift detection and interpreting the causes of the concept drift. The effectiveness of the solution was evaluated on synthetic and real-world data streams. The strong performance of SDDM when compared to state-of-the-art methods indicates that this approach can substitute the standard approach of designing drift detection methods which is based on monitoring the performance of the underlying model. Statistical methods, such as the SDDM, that derive and quantify the drift directly from the data, offer interpretability of the causes of drift, and are thus more useful in the real-world compared to methods that only indicate whether a drift has occurred.

However, the proposed method leaves some scope for refinement:

- The method can only be applied when the data arrives in batches. When the data arrives instance by instance, the probability distribution can not be estimated. In such cases, a different approach needs to be considered, such as goodness-of-fit tests that test whether the newly arrived instance comes from the distribution of the previously arrived instances.
- The recommendations for appropriate parameter values have been obtained by analyzing a finite set of data streams, thus they may not be relevant for every input. Possible future research may include developing an approach to generalize the process of selecting parameter values to any kind of input.
- In this work, only the distribution of the whole feature set, and the distributions of the individual features were considered. However, one needs to consider other marginal distributions in order to provide richer insights into the nature of the cause of the drift. In high dimensional data, the number of marginal distributions is significantly high, thus it may not be efficient and/or feasible to consider every marginal distribution. Therefore, techniques that identify which marginal distributions are most relevant are necessary.

8 Acknowledgments

I would like to express gratitude to my supervisors Sherif Sakr and Toivo Vajakas for their continuous support and extensive feedback during the work on my thesis.

References

[1] Gama, Joao, et al. *"Learning with drift detection."* Brazilian symposium on artificial intelligence. Springer, Berlin, Heidelberg, 2004.

[2] Baena-Garcıa, Manuel, et al. *"Early drift detection method."* Fourth international workshop on knowledge discovery from data streams. Vol. 6. 2006.

[3] Gama, João, et al. *"A survey on concept drift adaptation."* ACM computing surveys (CSUR) 46.4 (2014): 44.

[4] Doshi-Velez, Finale, and Been Kim. *"Towards a rigorous science of interpretable machine learning."* arXiv preprint arXiv:1702.08608 (2017).

[5] Pesaranghader, Ali, and Herna L. Viktor. *"Fast hoeffding drift detection method for evolving data streams."* Joint European conference on machine learning and knowledge discovery in databases. Springer, Cham, 2016.

[6] Pesaranghader, Ali, Herna L. Viktor, and Eric Paquet. *"McDiarmid drift detection methods for evolving data streams."* 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018.

[7] Webb, Geoffrey I., et al. *"Characterizing concept drift."* Data Mining and Knowledge Discovery 30.4 (2016): 964-994.

[8] Webb, Geoffrey I., et al. *"Understanding Concept Drift."* arXiv preprint arXiv:1704.00362 (2017).

[9] WALD, A. 1947. *Sequential Analysis.* John Wiley and Sons, Inc.

[10] Page, Ewan S. *"Continuous inspection schemes."* Biometrika 41.1/2 (1954): 100-115.

[11] Ross, Gordon J., et al. *"Exponentially weighted moving average charts for detecting concept drift."* Pattern recognition letters 33.2 (2012): 191-198.

[12] R. S. Barros, D. R. Cabral, S. G. Santos et al., *"Rddm: Reactive drift detection method,"* Expert Systems with Applications, 2017.

[13] Bifet, Albert, and Ricard Gavalda. *"Learning from time-changing data with adaptive windowing."* Proceedings of the 2007 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2007.

[14] Frías-Blanco, Isvani, et al. *"Online and non-parametric drift detection methods based on Hoeffding's bounds."* IEEE Transactions on Knowledge and Data Engineering 27.3 (2015): 810-823.

[15] Wang, Haixun, et al. *"Mining concept-drifting data streams using ensemble classifiers."* Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. AcM, 2003.

[16] Kuncheva, Ludmila I. *"Classifier ensembles for changing environments."* International Workshop on Multiple Classifier Systems. Springer, Berlin, Heidelberg, 2004.

[17] Duda, Richard O., Peter E. Hart, and David G. Stork. *"Pattern Classification Wiley."* New York 680 (2001).

[18] Storkey, Amos. *"When training and test sets are different: characterizing learning transfer."* Dataset shift in machine learning (2009): 3-28.

[19] Kull, Meelis, and Peter Flach. *"Patterns of dataset shift."* First International Workshop on Learning over Multiple Contexts (LMCE) at ECML-PKDD. 2014.

[20] Flach, Peter A. *"ROC analysis."* Encyclopedia of machine learning (2010): 869-875.

[21] Kullback, Solomon, and Richard A. Leibler. *"On information and sufficiency."* The annals of mathematical statistics 22.1 (1951): 79-86.

[22] Hoens, T. Ryan, Nitesh V. Chawla, and Robi Polikar. *"Heuristic updatable weighted random subspaces for non-stationary environments."* 2011 IEEE 11th International Conference on Data Mining. IEEE, 2011.

[23] Levin, David A., and Yuval Peres. *Markov chains and mixing times.* Vol. 107. American Mathematical Soc., 2017.

[24] Manning, Christopher, Prabhakar Raghavan, and Hinrich Schütze. *"Introduction to information retrieval."* Natural Language Engineering 16.1 (2010): 100-103.

[25] Huang, David Tse Jung, et al. *"Drift detection using stream volatility."* Joint European conference on machine learning and knowledge discovery in databases. Springer, Cham, 2015.

[26] Žliobaitė, Indrė, Marcin Budka, and Frederic Stahl. *"Towards cost-sensitive adaptation: When is it worth updating your predictive model?."* Neurocomputing 150 (2015): 240-249.

[27] Bifet, Albert. *"Classifier concept drift detection and the illusion of progress."* International Conference on Artificial Intelligence and Soft Computing. Springer, Cham, 2017.

[28] Bifet, Albert, et al. *"New ensemble methods for evolving data streams."* Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009.

[29] Kubat, Miroslav, and Gerhard Widmer. *"Adapting to drift in continuous domains."* European conference on machine learning. Springer, Berlin, Heidelberg, 1995.

[30] Bifet, Albert, et al. *"Moa: Massive online analysis."* Journal of Machine Learning Research 11.May (2010): 1601-1604.

[31] Olorunnimbe, M. Kehinde, Herna L. Viktor, and Eric Paquet. *"Dynamic adaptation of online ensembles for drifting data streams."* Journal of Intelligent Information Systems 50.2 (2018): 291-313.

[32] Roarty M (1998) Electricity industry restructuring: The state of play. Research Paper 14, Science, Technology, Environment and Resources Group, URL http://www.aph.gov.au/About_Parliament/ Parliamentary_Departments/Parliamentary_Library/pubs/rp/RP9798/98rp14

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Simona Micevska**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   A Statistical Drift Detection Method,

   supervised by Sherif Sakr and Toivo Vajakas.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Simona Micevska
16/05/2019