UNSUPERVISED CONCEPT DRIFT DETECTION USING SLIDING WINDOWS: TWO CONTRIBUTIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Ömer Gözüaçık
October 2020

Unsupervised Concept Drift Detection Using Sliding Windows: Two Contributions
By Ömer Gözüaçık
October 2020

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Fazlı Can(Advisor)

Özgür Ulusoy

İsmail Sengör Altıngövde

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

UNSUPERVISED CONCEPT DRIFT DETECTION USING SLIDING WINDOWS: TWO CONTRIBUTIONS

Ömer Gözüaçık
M.S. in Computer Engineering
Advisor: Fazlı Can
October 2020

Data stream mining has become an important research area over the past decade due to the increasing amount of data available today. Sources from various domains generate limitless volume of data in temporal order. Such data are referred to as data streams, and generally, they are nonstationary as the characteristics of the data evolve over time. This phenomenon is called concept drift, and it is an issue of great importance in the literature since it makes models outdated and decreases their predictive performance. In the presence of concept drift, adapting the change in data is necessary to have more robust and effective classifiers. Drift detectors are designed to run jointly with the classification models, updating them when a significant change in the data distribution is observed. In this study, we propose two unsupervised concept drift detection methods: D3 and OCDD. In D3, we use a discriminative classifier over a sliding window to monitor the change in the distribution of data. When the old and the new data are separable with the discriminative classifier, a drift is signaled. In OCDD, we use a one-class classifier over a sliding window. We monitor the number of outliers identified in the sliding window. We claim that the number of outliers are the signs of a new concept, and define concept drift detection as the continuous form of anomaly detection. A drift is signaled if the percentage of the outliers are over a pre-determined threshold. We perform a comprehensive evaluation on the latest and the most prevalent concept drift detectors using 13 datasets. The results show that OCDD outperforms the other methods by producing models with significantly better predictive performances on both real-world and synthetic datasets. D3 is on par with the other methods.

Keywords: data stream, concept drift.

ÖZET

KAYAN PENCERELER İLE GÜDÜMSÜZ KAVRAM SÜRÜKLENMESİNİN SAPTANMASI: İKİ YÖNTEM

Ömer Gözüaçık
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Fazlı Can
Ekim 2020

Veri akışı madenciliği, bugün mevcut veri miktarının artması nedeniyle son yıllarda önemli bir araştırma alanı haline gelmiştir. Veri akışları genellikle verilerin karakteristik özellikleri zaman içinde değiştiği için durağan değildir. Bu olguya kavram sürüklenmesi denir. Sınıflandırıcıları geçersiz hale getirdiği ve tahmin başarısını düşürdüğü için literatürde büyük öneme sahip bir konudur. Kavram sürüklenmesinin olduğu durumlarda, daha sağlam ve etkili sınıflandırıcılara sahip olmak için verilerdeki değişimin uyarlanması gerekir. Kavram sürüklenmesini saptayan yöntemler, sınıflandırma modelleriyle birlikte çalışacak ve veri dağılımında önemli bir değişiklik gözlemlendiğinde bunları güncelleyecek şekilde tasarlanmıştır. Bu çalışmada, D3 ve OCDD adlarında iki güdümsüz kavram sürüklenmesi tespit yöntemi sunulmaktadır. D3'te, veri dağıtımındaki değişikliği izlemek için kayan bir pencere üzerinde ayrıştırıcı sınıflandırıcı kullanılmaktadır. Eski ve yeni veriler kullanılan sınıflandırıcı ile başarılı olarak ayrıştırılabilirse kavram sürüklenmesi tespit edilmektedir. OCDD'de, kayan bir pencere üzerinde tek-sınıflı sınıflandırıcı kullanılmaktadır. Kayan pencerede belirlenen aykırı değerlerin sayısını izlenmektedir. Aykırı değerlerin sayısındaki değişimin yeni bir kavramın işaretleri olduğunu iddia edilmekte ve kavram sürüklenmesi tespitini anomali tespitinin sürekli formu olarak tanımlanmaktadır. Aykırı değerlerin yüzdesi önceden belirlenmiş bir eşiğin üzerindeyse kavram sürüklenmesi tespit edilmektedir. Çalışmada literatürde yaygın olarak kullanılan 13 veri setini kullanarak kavram sürüklenmesi tespiti yöntemleri üzerinde kapsamlı bir değerlendirme yapılmıştır. Sonuçlar, OCDD'nin hem gerçek hem de sentetik veri kümelerinde önemli ölçüde daha iyi tahmin performansına sahip modeller üreterek diğer yöntemlerden daha iyi tahmin sağladığını göstermektedir. D3 ise diğer yöntemlerle benzer sonuçlar vermektedir.

Anahtar sözcükler: veri-akışı, kavram sürüklenmesi.

Acknowledgement

First of all, I would like to thank my advisor Prof. Fazlı Can for his support and understanding over the past three years. His assistance and dedicated involvement in every step throughout the process pushed me to do better every day. I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK) 1001 program for supporting me in the 117E870 project. Besides my advisor, I would like to thank the rest of my thesis committee, Assoc. Prof. İ. Sengör Altıngövde, for reading my thesis, insightful comments, and difficult questions.

Getting through my dissertation required more than academic support, I would like to thank all my friends who supported me, specifically in Bilkent. Alper, Cihan, Çağlar, Ezgi, Giray, Gizem, Miray, Onur, Zülal and the rest of my fellow office-mates in EA507. You will be remembered with the good coffee and the quality time we've shared in Bilkent. I would like to thank Begüm for her love and support during stressful times. Having you by my side was very important for me. In addition, I would like pay my special regards to everyone in Bilkent University Computer Engineering Department specifically to our department sectary Ebru Ateş for everything in the past three years.

Most importantly, none of this could have been possible without my family. I must express my sincerest gratitude to my mother Özlem, my father Metin and my grandmother Leyla for their support throughout my entire life. I am forever grateful.

Beautiful is what we see, more beautiful is what we know, most beautiful by far is what we don't.

NICOLAS STENO

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An earlier version of this thesis is published as a conference paper [1] in ACM CIKM 2019. The current version is under review in *Artificial Intelligence Review* Journal [2].

1.1 Data Stream Characteristics

Data stream mining has became a major research topic due to the increasing amount of data generated by various sources such as social networks, online businesses, and military-financial applications [3]. According to latest records, 14 billion e-mails and 350 million tweets are sent every day [4]. Multinational retail companies record more than a million transactions every hour. These data are mostly wasted because of computational and memory-related limitations. Therefore, designing models that can deal with the data of enormous size with high speed is extremely important. Data streams are referred to as data which arrive continuously with infinite amount of samples. They need to be processed immediately otherwise they are lost due to the volatile nature of the streaming environment [5].

It is estimated that the data produced is in order of zeta bytes and it is

growing around 40% every year [3]. Internet of things (IoT) is a new phenomenon which is believed to further extend this problem. It is defined as the network of physical objects, having an embedded system allowing them to interact and sense or act with respect to internal and external conditions [3]. It consists multiple different networks of sensors and actuators working on multiple different areas: industrial automation, autonomous cars, home automation, financial prediction, recommender systems, heath care and etc. Beforehand, data streams were only studied in financial markets. However, they are now in everywhere due to the recent advancements personalized technologies (e.g: IoT) making every individual a source of data.

Algorithms and tools dealing with data of large size are different than the traditional approaches. Doug Laney introduce three main points for dealing with Big Data [6]. Bifet and Fan extends this idea to data streams and introduces 5V [3].

- Volume: the amount of data is growing in huge percentages, but the tools and algorithms are not prepared to deal with it. Most of the data cannot be stored due to its high volume. We need to use the resources in most efficient manner. We can only store samples or statistics of the data.
- Velocity: the data is arriving continuously with high speed in data stream form. We need to process this data as fast as possible and give results in real time.
- Variety: the data is produced in diferent types: images, videos, sequences, sensor data, texts, trees, graphs and we need algorithms that can deal with all.
- Variability: the distribution of the data can change over time. Therefore, algorithms that are adaptive to changes are needed.
- Value: organizations tackle their problems using algorithms for Big Data and make decisions accordingly. It gives them a compelling advantage in their decision making and product deployment, creating a business value

for the algorithms. Applicability of an algorithm to real-life scenarios is an important aspect as it directly influences its value.

1.2 Data Stream Mining

There are various analytical approaches developed for solving problems in machine learning and classification is one of them following the idea that data can be generalized [7]. A predictive function is modeled which maps features to labels using training data later to be evaluated on test data. The main assumption for generalization is that data is static which indicates both train and test environments are from the same distribution. However, this assumption is often violated as real-world applications are dynamic and evolve over time. This is known as concept drift [8]. In such cases, the classification model becomes obsolete as the data distribution changes and the predictive function can no longer correctly map features to labels, hence deteriorating the predictive performance of models. Therefore, we need algorithms that can deal with the change under restrictions of the streaming environment.

The ubiquity of data stream applications has made the concept drift problem a hot topic. A "concept drift" exact match search on February 24 2020, Google Scholar returns 1380, 1790, and 2030 matches for articles published in 2017, 2018, 2019, respectively; it returns 14,400 matches when no time restriction is given. Concept drift detection methods are of two types: explicit and implicit [9]. Explicit (supervised) methods track the prediction performance of the model and signal a drift if there is a significant decline. They need to verify the predictions of the classifier before continuing to the next data items. Therefore, they require the true labels of the data instances to be available right after classification. Otherwise, these algorithms fail to detect changes on time. This problem is referred to as verification latency [10]. [11] claims that explicit algorithms are not practically useful as most real-world data streams have verification latency. Most available techniques to cope with concept drift are explicit; work on implicit

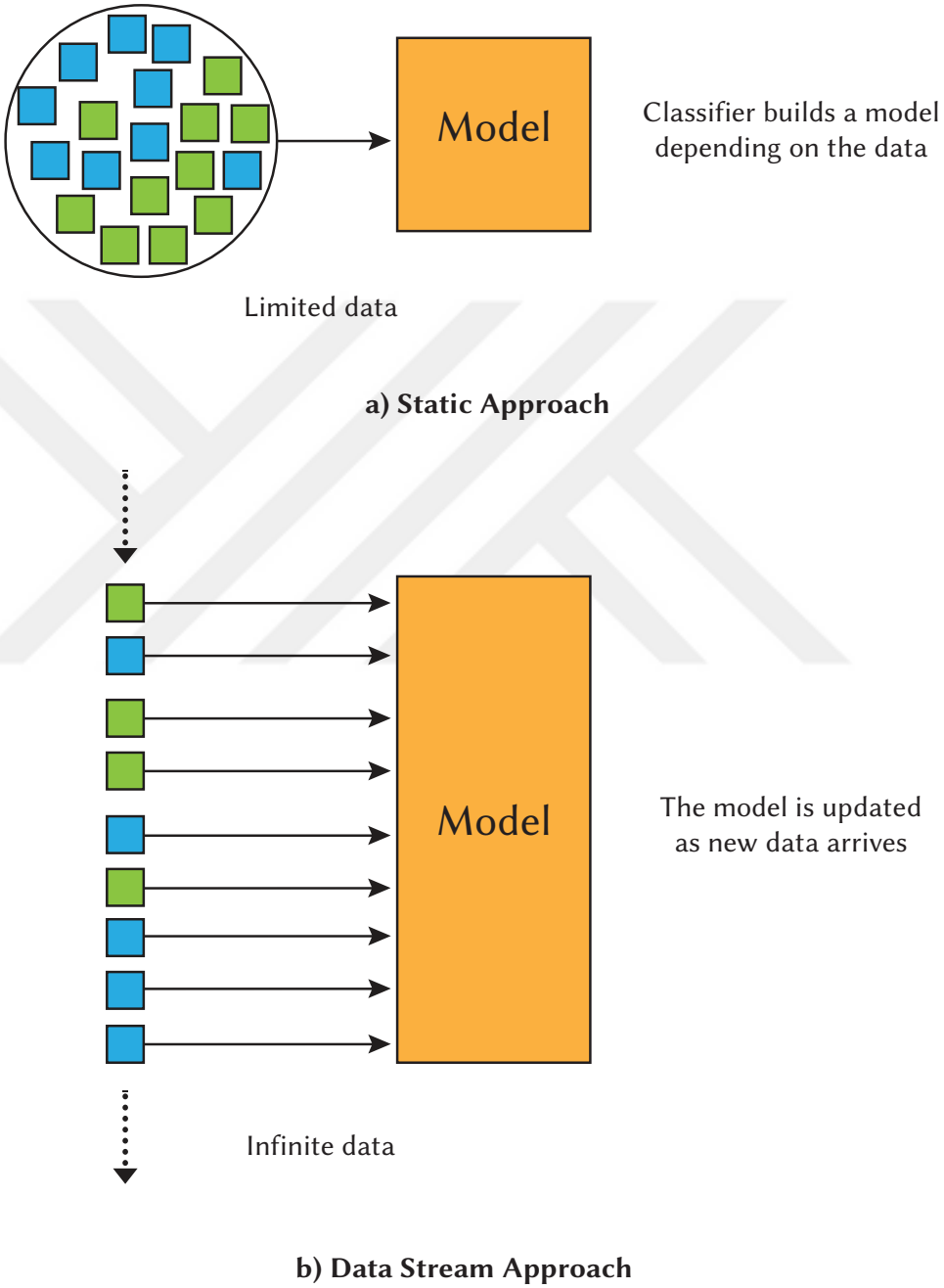

Figure 1.1: Data mining in static vs. streaming environments. Different colors represent different classes

drift detection is limited [12].

Implicit (unsupervised) methods do not require labels. They monitor the data distribution and detect drift in case of significant change; they are more suitable for real-life scenarios. In stream settings, labels are not available perpetually [13]. Only a limited number of them are accessible, or they arrive with delay in certain circumstances [9]. On Twitter, 500M tweets are produced every day. Training an online and supervised model for tasks like sentiment analysis in such environment is very challenging due to the size of the data. Labeling just 1% (of tweets) can cost over $100K using crowd sourcing websites like Amazon's Mechanical Turk, with a worker being paid $1 per 50 tweets [9]. This process requires a continuous workforce and funding, which may not be available. Furthermore, labeling will have a delay as they will be processed manually. In many streaming environments, these problems can be observed. Therefore, streaming algorithms need to work with unlabeled or sparsely labeled data to be of any use in real-life scenarios [11].

Another motivational example for unsupervised concept drift detection is also available in our current research agenda, which focuses on a multi-stream environment [14]. In this environment there are separate source data streams with labels. There is a separate ensemble classifier for each source data stream. Furthermore, there is a data stream which is referred to as the target data stream that classifies unlabeled data items. The ensemble classifier of the target data stream is generated by selecting from the components of the source data stream ensembles, or one if others are unavailable. However, the target data stream does not have labels and its ensemble is updated when a concept drift is detected in the target data stream. In order to detect concept drift in the target data stream, using an unsupervised method is the only option when the labels are unavailable. A possible real-life application for this environment can be defined as follows. Consider a credit card application where customer transactions are classified as safe and unsafe. In this environment each source data stream may be transactions of safe customers in the separate cities of the country where the cards are issued. The target stream may be the transactions in a foreign country for customers from different cities (source data streams).

1.3 Work Done and Contributions

In this thesis, we propose two unsupervised concept drift detection methods: D3 and OCDD. D3 (Discriminative Drift Detector) is an unsupervised method using a discriminative classifier over a sliding window to monitor the change in the distribution of data. A classifier is trained and tested periodically aiming to distinguish whether the new samples are from a similar distribution as the old. In a batch setting, a related problem is *covariate shift adaptation* [15] where training and test distributions differ. A method that uses a discriminative classifier similar to the proposed approach is introduced to detect and correct covariate shift [16]. In a stream setting, to the best of our knowledge, this is the first method that utilizes a discriminative classifier for the *concept drift detection* task.

OCDD (One-Class Drift Detector) is an implicit concept drift detection algorithm using a one-class classifier over a sliding time window. A one-class classifier is trained to distinguish whether new samples differ from the old. We signal a drift depending on the percentage of the outliers detected in the sliding window. The difference is that D3 monitors the separability of the old and the new sample distributions, OCDD learns the current distribution and detects drift if new samples are from another distribution, classified as an outlier with the one-class learner. Furthermore, D3 is limited to detect drifts that show a linear pattern on the feature space. Our approach can deal with non-linear change as well.

The summary of main contributions of this study are as follows. We:

- Present an effective and simple two unsupervised concept drift detection algorithms that can be useful in environments when labels for new data items are not available or delayed, and make their implementation public for other researchers;
- Discuss the similarities of concept drift detection and novelty, anomaly or outlier detection, and demonstrate how methods for these tasks can also be used for drift detection;

- Analyze the proposed algorithms on 13 datasets against mainly recent and most prevalent 16 concept drift detection methods, an adaptive classifier and an online classifier without any drift adaptation mechanism, and perform a comprehensive evaluation, showing that OCDD outperforms the other approaches in predictive performance.

In Chapter 2, we define the concept drift detection problem formally and an inclusive review of concept drift detection approaches under the categories of implicit and explicit is presented. We describe our approaches in Chapter 3-4. Chapter 5 introduces the datasets and the experimental setup. In Chapter 6, we present the experimental results, and provide a discussion accompanied with some recommendations on how to use our methods in various situations. We conclude our paper and provide some future research directions in Chapter 7.

Chapter 2

Background and Related Work

In this chapter, we introduce concept drift and its detection formally. We discuss the methods for learning under concept drift in the literature. We divide the concept drift detectors with respect to their characteristics: supervised and unsupervised, and give an overall perspective to the problem.

2.1 Problem Definition

Stream classification is a supervised learning problem that takes place in data streams, under time and memory constraints [17]. A data stream consists of data instances that arrive in time order, i.e. $\mathcal{D} = \{(X_0, y_0), (X_1, y_1), \dots (X_t, y_t), \dots\}$ where X_t represents features, and y_t classes associated with the instance at time t. Class information for a data instance, y_t, is available only after testing, i.e. predicting from X_t. By this way, the model is always tested on instances that it has not seen.

In general, the data generation process in streams is considered to be stationary. The data is drawn from a fixed probability distribution $p(X, y)$ which can be referred to as a concept. However, in real world applications, the concept can depend on some hidden context which is not defined explicitly in the features,

changing the process of data generation [18]. The cause of this change can depend on periodicity, change in habits, aging and etc. In such an environment, concept drift detection is a task of determining whether the joint distribution of inputs X and targets y differ between times t_0 and t_1 [8].

$$p_{t_0}(X, y) \neq p_{t_1}(X, y) \tag{2.1}$$

In concept drift detection, the main objective is to design an efficient method that works simultaneously with the classification model, signaling drift or novelty when there is a significant change in data characteristics [19]. The model is updated accordingly, preventing it from being affected by the change, hence improving the predictive performance.

2.2 Learning Under Concept Drift

While making analytical analysis on streams, it is often assumed that the process that generates data is static. The data is drawn from a fixed distribution that does not change during the process. However, this is not possible as the data generation process in real-world applications is dynamic where the probabilistic properties of the data evolves over time. In such cases, a model which is not adaptive to changes would become obsolete when the distribution of the data differs from the time the model is trained. This is known as *concept drift*. The prevalence of it in research and common applications is on rise [8]. Efficient and effective algorithms for learning in dynamic environments are required. They are referred to as adaptive algorithms and studied under two branches: active and passive approaches [20].

They differ on how they react to the change. Active approaches monitor the data for changes and activate the adaptation mechanism when a drift is detected. (E.g: using SVM with a drift detector, retrain the model when a drift is detected) Passive approaches update the model regularly as new data comes. (E.g: using

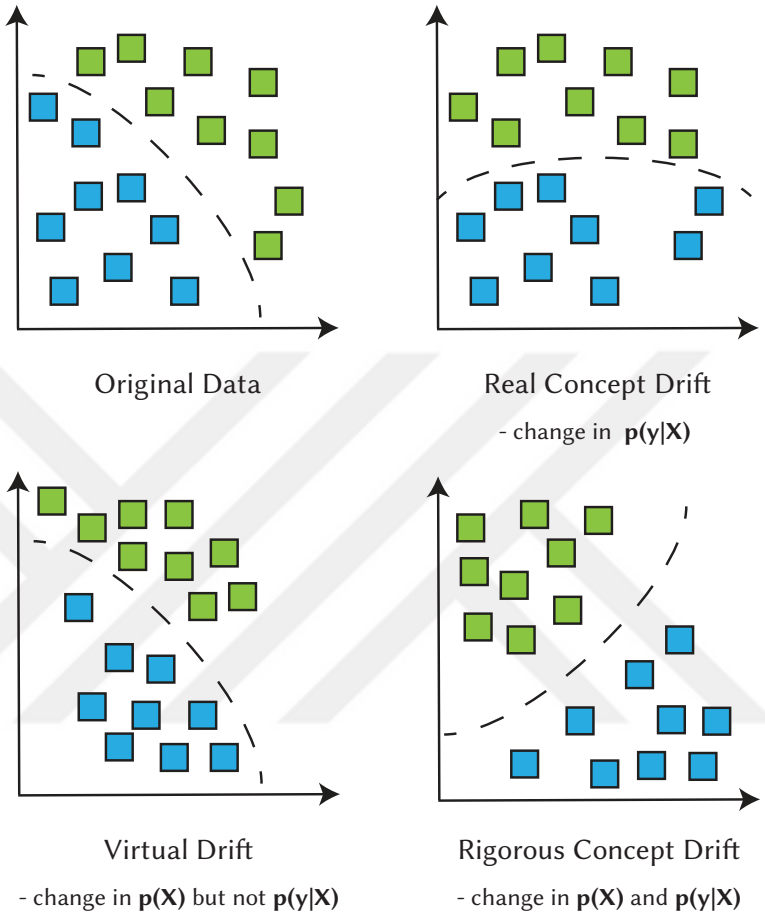

Figure 2.1: Concept drift types: square objects are the instances; colors represent classes; the dashed line is the decision boundary.

Hoeffding Tree and updating it with every sample) There are also some hybrid models as well that both update the model constantly and track the possible drifts. They make major updates when they detect a drift. (E.g: using Hoeffding Tree with a drift detector, reset the tree when a drift is detected)

Changes in data can be investigated as changes in the components of the relation (Equation 2.1) [8]. The equation can be expanded as:

$$p(X, y) = P(y|X)P(X) \tag{2.2}$$

$$p(X, y) = P(X|y)P(y) \tag{2.3}$$

Only changes that affect the prediction process require adaptation. Concept drift types are shown in Figure 2.1. Virtual concept drift can be defined as a change in $p(X)$ only. Real concept drift refers to the changes in $p(y|X)$, but a change in $p(X)$ can also be present. The main difference is that under real concept drift, the old knowledge (concept) becomes irrelevant, and replacement learning (restructuring the learning model) is required. Whereas under virtual drift, the old knowledge is extended with additional data from the same environment, and supplementary learning (tuning) is needed [21].

In classification tasks, $p(y|X)$ is estimated by training a model on data. The changes in $p(y|X)$ are highly important as they directly affect the classifiers' performance. However, the true class labels may not be available right after classification. They can either be delayed or unavailable in some environments [11]. Therefore, it is also necessary to monitor whether changes on the distribution of features, $p(X)$, affect predictive performance. Most of the implicit concept drift detection methods assume that changes in $p(X)$ lead to changes in $p(y|X)$. In the literature, cases where both the posterior probability, $p(y|X)$, and the marginal distribution of data, $p(X)$, change are identified as rigorous concept drift [22].

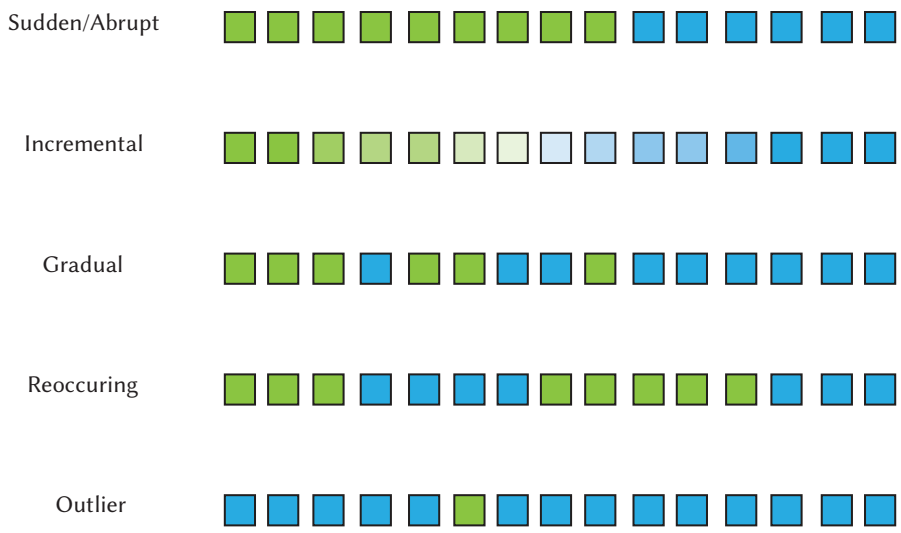

Figure 2.2: Types of changes in data (outlier is not considered as a change)

- **Sudden/Abrupt**, changing from a concept to the other suddenly. (e.g., replacing a sensor with another one that has different initial settings)
- **Incremental**, having intermediate concepts while changing from a concept to the other. (e.g., A sensor getting old and not giving accurate results in time)
- **Gradual**, changing from a concept to the other slowly. For some period, the old concept is present with the new (e.g., when a new road is constructed which is a shortcut going from A to B, people tend to use the old road along with the new due to their habits but eventually use the new afterwards)
- **Reoccurring**, changing from a concept to another which was observed before. (e.g., thermal sensor in a room getting sunlight gathering data both in daytime and night. Concepts in day and night will differ but they will recur every day)

Active approaches follow the idea of detecting and reacting [20]. They use a drift detector along with the model and update it when they detect a drift. They tend to perform better when the drift is abrupt [20]. In contrast, they are not robust to gradual and abrupt drift types. On the other hand, passive approaches are much better in these type of drifts. Passive models get adapted to changes that happen slowly as they get updated frequently. However, they fail to adapt sudden changes because they do not discard the older concepts info directly when a drift is detected. Both active and passive approaches are inconsistent when the drift is reoccurring.

2.3 Related Work: Concept Drift Detection

In this section, we present concept drift detection methods under two categories: implicit and explicit.

2.3.1 Implicit Concept Drift Detection Methods

There are various approaches specialized for implicit drift detection using clustering, distribution monitoring-based methods, model-dependent methods, and learner monitoring-based methods [9, 23].

2.3.1.1 Clustering-Based Methods

The methods in this group use distance or density measures to detect new concepts [10]. OLINDDA [24] uses K-means for clustering the data. When an unknown sample arrives it is either added to an existing cluster or to a new profile. MINAS [19] is an extension of it for multiclass problems. DETECTNOD [25] defines the boundaries of existing data by clustering. New samples that are out of the defined region are first clustered and then determined to be drift, depending on their similarity to existing clusters. Similar ideas are available in information retrieval in the form of incremental clustering. C2ICM [26] identifies new cluster centroids and falsifies old ones as documents are being processed.

Woo [27], ECSMiner [10], GC3 [28] uses micro-clusters. They first cluster data and then assign each a classifier. Samples falling out of the clustered region are monitored continuously. If their density increases, it is identified as a new concept. In such cases, data is clustered again and the classifiers are reset. SAND [29] uses an ensemble of classifiers each trained on different data. The ensemble is used to create clusters, and these clusters map the current data regions. If a new region is clustered, a drift is detected. These methods only work when the drift is clusterable. If the drift does not have a pattern and occupies new region in space, they are ineffective.

2.3.1.2 Multivariate Distribution Monitoring-Based Methods

The methods in this group identify each feature in data as a stream and individually track any changes. A reference is held, representing the properties of

the old data chunks, and it is compared with the new data chunks. If there is a significant change from the average, a drift is detected. For measuring differences between chunks, Hellinger distance, KL-divergence and correlation are generally used [30]. PLOVER uses statistical moments and the power spectrum [31].

These methods are costly in high dimensional data streams as each feature is monitored. PCA based methods are proposed to reduce the number of features to be tracked. However, the results are not in agreement. [32] state that monitoring the principal components with top eigenvalues is enough to detect drifts whereas [33] claim the opposite. Furthermore, all features have equal weight regardless of their importance for classification. Therefore, they are prone to false alarms and signal a drift even if the change in the drifting feature is insignificant.

2.3.1.3 Model-Dependent Methods

There are methods that implicitly track concept drift without assuming the changes in $P(X)$ will lead changes in $P(y|X)$. They track the posterior probability estimates of the classifier. For that reason, they require probabilistic classifiers that give $P(y|X)$ of the classes before the final prediction. The Kolmogrov-Smirnov test and Wilcoxon rank-sum test are used for detecting changes in the estimate [34].

There are other methods that track the confidence of the classifiers by monitoring how well the classes are separated with the classifier [35]. They flag a drift depending on the changes of the classification margin among classes. They hold a reference margin and compare it with the upcoming cases. The reference margin is continuously updated. With a similar methodology, KL divergence is used on posterior probability estimates in another drift detection method [36]. Depending on how the estimate differs from the reference case, a drift is detected. With these methods, the size of the problem is reduced to much smaller dimensions as the number of values to be tracked is limited by the class count. However, they depend on classifier selection and require a probabilistic model to be used.

2.3.1.4 Learner Monitoring-Based Methods

The methods in this group track predictions of the learning model. MD3 [9] monitors the density of the samples in the margin learned by the model. The margin is the boundary for the classes, being referred to as the ambiguous region of the model. If the density of the data in this region exceeds a certain threshold, a drift is detected. PERM [37] compares the empirical risks on the ordered stream data and its random permutations. In a window, they split data into train and test sets according to their temporal order, where newer samples are put into the test set. They train a model and calculate its empirical risk. They claim that the shuffled version of the data should have a similar risk compared to the ordered data if concept drift is not present. A drift is signaled if there is a significant difference between the risks calculated with the ordered and permuted data. ExStream [38] is based on observing changes in model explanation. It continuously measures the explanation of the online learner, and calculates dissimilarities in the stream explanations. Then, these dissimilarities are fed to a supervised drift detection algorithm to detect a drift.

[39] define a statistical test called the density test by applying kernel density to check if the new data is sampled from the same distribution as the reference set. SAMM [40] uses Jensen-Shannon divergence to measure dissimilarity of model scores of the target data and the reference continuously, flagging a drift if the dissimilarity measure exceeds the threshold. These methods are dependent to the choice of the classifier similar to the model-dependent methods.

2.3.2 Explicit Concept Drift Detection Methods

The majority of the concept drift detectors are explicit and evaluate the predictive performance of models. They can be classified into three different groups: sequential, statistical, and window-based methods [41]. Sequential approaches track the results of the model, signaling a drift when a pre-defined threshold is exceeded. The Page-Hinckley test and the CUSUM test [42] are members of this

group. Statistical approaches evaluate properties of the results, such as mean and standard deviation. They detect drift if there is substantial change. DDM [43], EDDM [44], RDDM [45] and EWMA [46] are representatives of these type of methods.

Window-based methods hold a reference of past results and compare them to the initial. A sliding window is used to capture the most recent statistical properties of the data. They signal a drift when there is a significant difference between the reference and the current window. ADWIN [47]; Seq2D [48]; MDDM_A, MDDM_E, MDDM_G [49]; HDDM_A, HDDM_W [50]; FHDDM [51]; FHDDMS, FHDDMS_A [41] are examples of such methods. Explicit methods depend on the true class labels and do not work properly when they are not present. It is one of the main weaknesses of these drift detectors.

Chapter 3

Proposed Approach 1: D3

We propose D3 (**D**iscriminative **D**rift **D**etector), an unsupervised drift detection method which uses a discriminative classifier that can be used with any online algorithm without a built-in drift detector. We hold a fixed size sliding window of the latest data having two sets: the old and the new. A simple classifier is trained to distinguish these sets. We detect a drift with respect to classifier performance. This process is done repeatedly as long as there is new data. It is a simple and practical method that can executed with few lines of code. In the following sections, D3 is analyzed in depth, explaining the motivations, design and methodology. A pseudocode is presented to explain the steps of the algorithm. We summarize the drift detection workflow for two cases in Figure 3.1.

3.1 Motivation

Ideally, the shift between two sets can be observed by estimating their distributions and measuring the change between them (e.g. using KL Divergence). However, it is costly in streaming environments as the estimations need to be done repeatedly and we want instant results. What we want is to observe whether two sets differ continuously, not to estimate their distributions. In our intuition, it may be sufficient to learn the divergence between distributions, to be able to detect concept drifts implicitly.

According to Vapnik's principle, when there is limited information for solving some problem, the problem can be solved directly without solving a more general problem as an intermediate stage [52].The available information can be sufficient for a direct solution but not enough for a more general intermediate problem. In our case, it is similar, but due to the complexity of the problem. By-passing learning the distributions and focusing only on learning the divergence between them, concept drifts can be detected implicitly.

3.2 Design and Methodology

We hold a sliding window: W of the latest data with size $w(1+\rho)$ (Alg. line 2). w is the size of the old data. ρ is the percentage of new data with respect to old. The size for the new data is calculated as $w\rho$. We store the samples without breaking their time order. The leftmost side (tail) has the older samples whereas the other side (head) has the newer, which can be seen in Figure 3.1. In the initialization phase, when the whole window is empty, we wait until it gets full (Alg. line 5). Afterwards, we start with the first check. A new slack variable s is introduced. The oldest members of size w are labeled as old, and given value 0 (Alg. line 9). The remaining are labeled new, and given value 1 (Alg. line 10). Later, a logistic regression model is trained as a discriminative classifier to distinguish old and new with s as labels (Alg. line 11).

Algorithm 1 D3: Discriminative Drift Detector

procedure D3($\mathcal{D}, w, \rho, \tau$):
    Initialize window W where $|W| = w(1 + \rho)$
    Discriminative classifier C ▷*e.g: Logistic Regression*
    for (X, y) in $\mathcal{D}$ **do** ▷*class label (y) is not used*
        if W is not full **then**
            $W \leftarrow W \cup X$ ▷*i.e., add X to the head of W*
        else
            S is vector of s, $|W| = |S|$ ▷*s is a slack variable*
            $s = 0$ for $W[1, w]$ ▷*label for old* (0) *and new* (1)
            $s = 1$ for $W[w + 1, end]$
            Train $C(W, S)$ ▷*train C with S as labels of W*
            if $AUC(C, S) \geq \tau$ **then** ▷*measure AUC score*
                drift = 1 ▷*drift detected*
                Drop w elements from the tail of W
            else
                drift = 0 ▷*no drift*
                Drop $w\rho$ elements from the tail of W

We use AUC as a measure of separability. It expresses to what degree a model can distinguish two classes [53]. The AUC of a perfect model is near 1 indicating that the model can discriminate the classes successfully. A poor model has an AUC near 0.5 when the distribution of the classes overlap. Depending on the divergence of the classes, we detect a drift. We expect the class distributions to overlap or have slight differences when there is no drift. We set a threshold (τ) for AUC to measure how much the classes (old and new) are separable. If it is over the threshold, we signal a drift (Alg. line 12-17).

There are two possible outcomes of the discriminative classifier, as it is illustrated in Figure 3.1. **(1)**: AUC is greater than or equal to τ. A drift is detected, the classifier's performance is high, and the old and the new data are separable in the feature space. In that case, we discard the samples from the old data part and replace them with the ones from the new data. **(2)**: AUC is less than τ. The classifier's performance is poor. This indicates that the predictive function fails to separate the two intertwined distributions. Then, we remove the oldest $w\rho$ samples and shift the window to left where the samples from the new fill the recently freed space. For both cases, we wait for new samples to come and check again for drift when the window is filled. This work-flow can go on, as long as the stream generates data.

Even though the sizes of the old (w) and new data ($w\rho$) are hyper parameters of our model, we set the old to be always larger. We need the old portion of the data to be descriptive enough for the current distribution and span as much area in the feature space as possible. However, it should not be very large, otherwise it may contain multiple concepts. Since the old is compared with a relatively smaller portion (new data) to detect changes, using a metric that works well in the presence of class imbalance is highly important. For this reason, we use AUC to measure the performance of the discriminative classifier, as it works well in such cases [53].

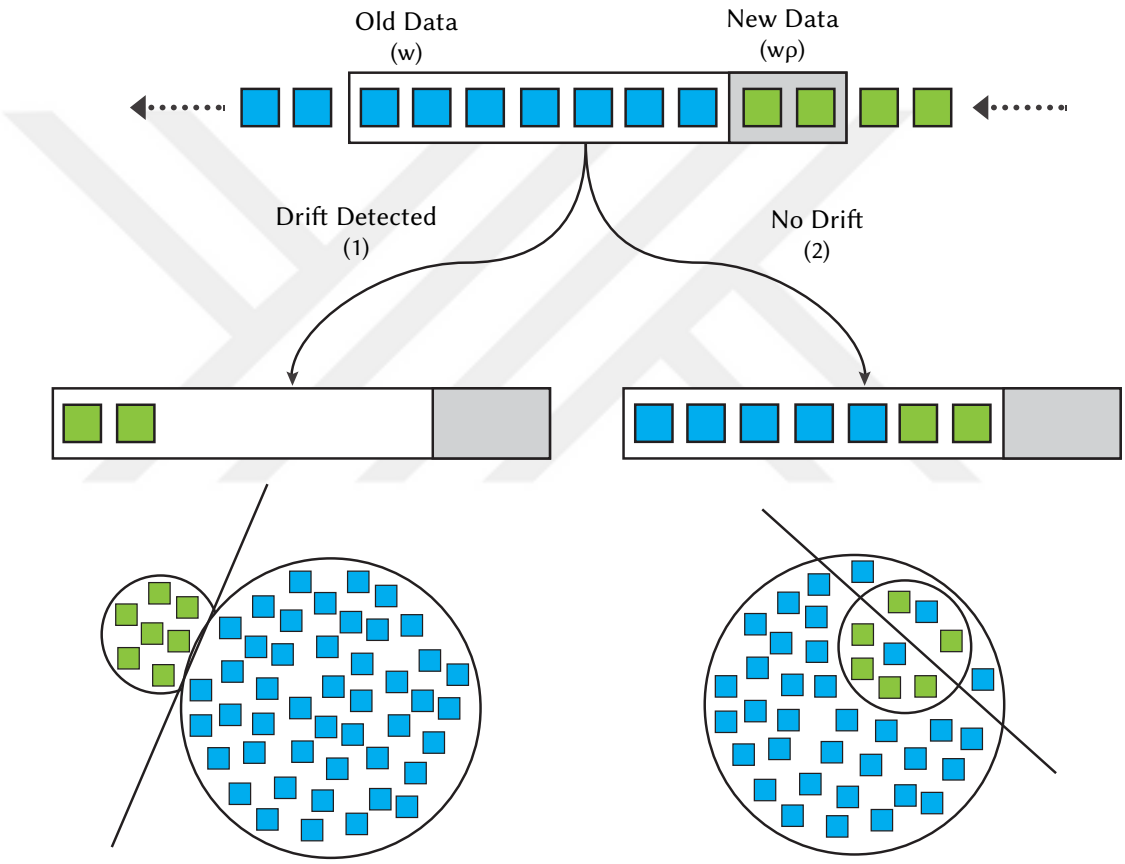

Figure 3.1: Drift detection workflow: **(1)**: drift detected. The old and the new data are separable. Samples from the old portion are discarded and partially filled with the samples from the new. **(2)**: no drift. These sets are nested. The oldest $w\rho$ samples are removed and the window is shifted to left where the samples from the new fill the space that becomes empty.

Chapter 4

Proposed Approach 2: OCDD

We propose OCDD (**O**ne-**C**lass **D**rift **D**etector), an implicit concept drift detector which uses a one-class classifier with a sliding window. It can be used with any existing online classifier that does not intrinsically have a drift-adapting mechanism. A one-class classifier is trained at the start, with the data in the sliding window. The one-class classifier is used to estimate the distribution of the new concept, classifying whether new samples are from the current concept or are outliers. Samples that are classified as outliers are identified as data from the new concept. Depending on the percentage of the outliers detected in the sliding window, we signal a drift. We do this process continuously until there is no new data. In the following sections, the intuition behind the design of OCDD is shown, including the similarities of concept drift detection and one-class classification. A pseudocode is given to explain the steps of the algorithm.

4.1 Similarities of Concept Drift Detection and One-Class Classification

One-class classification is studied under novelty, anomaly or outlier detection. It aims to detect if test data differs from the data used in training [54]. Data of only one class is available during training. In an earlier work, one-class classification is identified as concept learning [55]. Concept in this context represents the distribution of data similarity, as in drift detection. According to the data available in training, a decision boundary that spans the current concept in the feature space is estimated. In the testing phase, a sample is identified as being either typical or an outlier, depending on where it lies in the feature space.

One-class classifiers have similarities with concept drift detectors as both aim to classify whether new samples share similar characteristics to old samples. The flow of data is of little importance, rather they check if samples are from the same concept. However, drift detectors monitor the flow of data, and signal a drift if there is a significant change. To the best of our knowledge, we identify concept drift detection as the continuous form of one-class classification for the first time in the literature, as listed in the main contributions. If a one-class classifier is trained on the streaming data with concept drift, it will classify the new data as outliers when they form a new concept. By using this observation, we can detect drifts using one-class classifiers depending on the amount of new data being classified as an outlier, without explicitly estimating the distributions.

4.2 Implementation Details of OCDD

Pseudocode of OCDD is given in Algorithm 1. We hold two sliding windows, W to store the latest data, and O, to store the predictions of the one-class classifier with size, w. The samples are stored without breaking their temporal order. In both sliding windows, the left-hand side has older samples and the other side has newer ones, (Figure 4.1). For simplicity, we illustrate the method with only one

sliding window, as O stores the results for the predictions of the data in W, which can be either 1 (typical) or 0 (outlier). In the initialization phase, we train the one-class classifier with the initial samples. We set the size of initial samples to w, similar to the sliding windows, but it can be changed depending on the data available before the stream starts generating data. After initialization, we start processing data. We wait for the sliding windows to become fully populated: W, with the new data and O, with the results of the one-class classification. When the windows are full, we do the first test. We calculate the percentage of outliers (α) detected in the window. If α is over the threshold, ρ, we signal a drift.

Algorithm 2 OCDD: One-Class Drift Detector

```
1:  procedure OCDD(D, w, ρ):
2:      Initialize windows W, O where |W| = |O| = w
3:      One-class classifier C                          ▷e.g: One-Class SVM
4:      Train C                                         ▷with the available samples
5:      for (X, y) in D do                              ▷class label (y) is not used
6:          if W is not full then
7:              W ← W ∪ X                               ▷i.e., add X to the head of W
8:              T = C(X)                                ▷classify the new sample with C
9:              O ← O ∪ T                               ▷T = 1 for outliers and 0 otherwise
10:         else
11:             α = Σ_{i=1}^{w} O[i]/w                  ▷measure percentage of outliers
12:             if α ≥ ρ then
13:                 drift = True                        ▷drift detected
14:                 Drop w(1 − ρ) elements from the tail of W
15:                 Retrain C                           ▷with the samples available in W
16:             else
17:                 drift = False                       ▷no drift
18:                 Drop 1 element from the tail of W
```

There are two possible results as illustrated in Figure 4.1. (1) The percentage of outliers, α is higher than the threshold: ρ as we indicated above. In this case, we signal a drift. A significant amount of new data is from a concept different

from the old, as the one-class classifier detects them as outliers. The samples from the sliding windows except for the latest, $w\rho$, are discarded. The remaining data is shifted left, where they fill the freed space. The one-class classifier is retrained with the available data in the window in order to learn the new concept. (2) The value of α is lower than ρ. There is no drift in this case. The desired amount of the new samples are from the same concept as the old one since the one-class classifier detects them as typical. In such circumstance, we remove the data of the oldest sample and shift the windows left. In both cases, we wait for the windows to get full and check repeatedly for the drift. This process continues until there is no more data.

We use ρ for both the threshold of the percentage of the outliers and the percentage of new data. They can be set to different parameters individually. Claiming that ρ percentage of the data is enough to detect a drift, we also think it can be enough to retrain the one-class classifier, and thus learn the new concept. The new data section, $w\rho$, needs to be expressive enough to represent the new concept properly, spanning most of the feature space, depending on the properties of the data. Therefore, the size of the sliding window, w should be set properly. If it is too small, the data may not represent a concept. Otherwise, when it is too large, it may have multiple concepts.

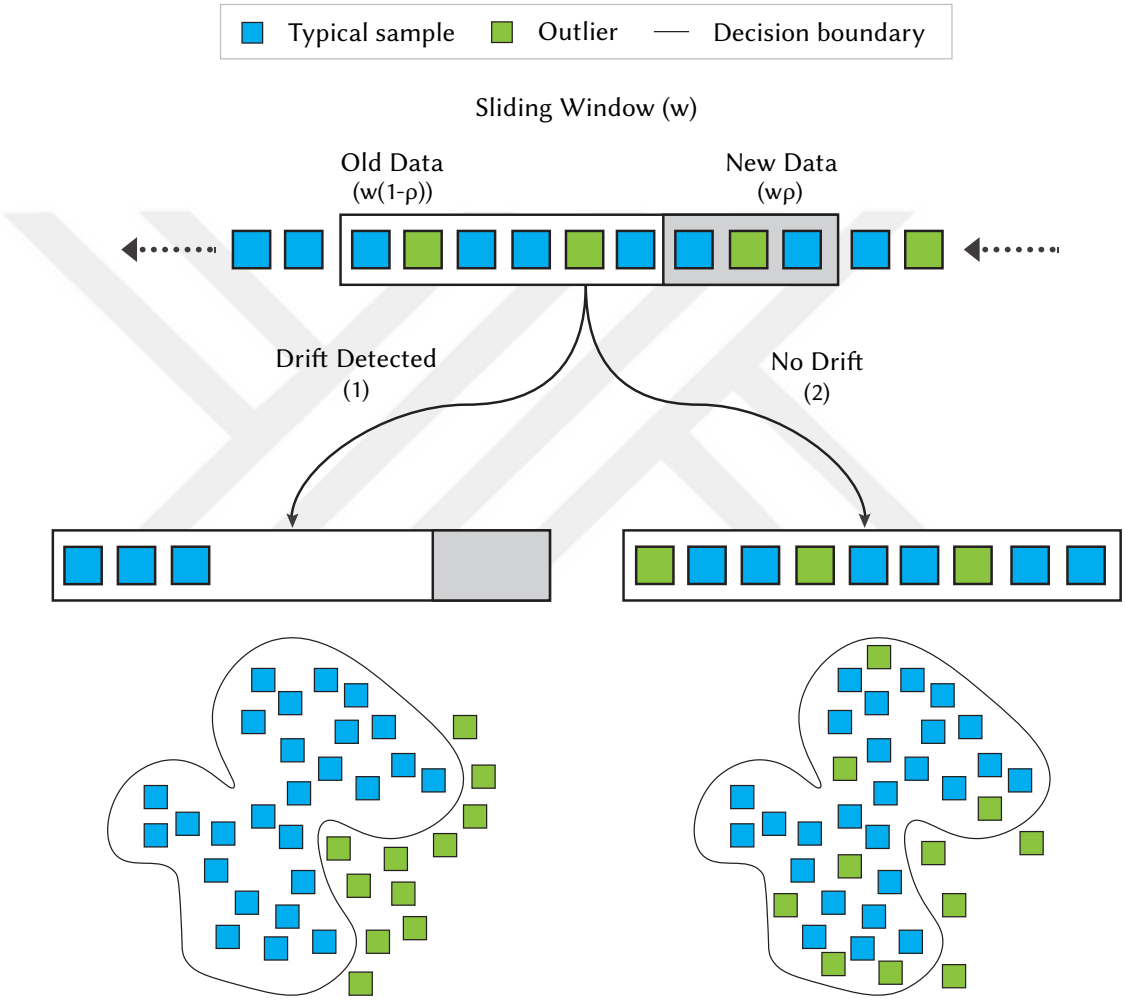

Figure 4.1: Drift detection workflow: **(1)**: Drift detected. The percentage of outliers exceed the threshold (ρ). There is a change in the distribution of the data. Samples from the old portion are discarded and partially filled with samples from the new data window. **(2)**: No drift. There is no change in the data distribution. The oldest sample is removed and the window is shifted to the left, filling the empty space.

Chapter 5

Empirical Evaluation

In this chapter, we introduce the datasets that we use and our experimental setup for testing D3 and OCDD. We use the terms dataset and data stream equivalently. We follow the main points of dealing with data streams given by Fan and Bifet while making the evaluation [3]. Variety is one of these points. While making comparison of one method to another, it is very important to evaluate them under different scenarios. Volume and velocity are another important aspects. Datasets should be large enough to represent a streaming environment which refers to data of infinite amount. Furthermore, stream algorithms need to be fast enough to process high amount of data in real-time by using the resources efficiently. In order to satisfy these requirements, we choose datasets of large size from different contexts and evaluate the predictive performance of our models. The details are given in the following sections.

5.1 Datasets of Streams

We test our approach both on 13 well-known real-world and synthetic datasets which have concept drift in them. We introduce every one of them with a small description along with their specifications in the following sections. The summary of datasets is shown in Table 5.1. They can be accessed from the GitHub link provided in the Appendix.

Table 5.1: Datasets we use for evaluation

	Name (Reference)	**#Features**	**#Classes**	**#Samples**
Real	ELEC [56]	6	2	45,312
	COVTYPE [57]	54	7	581,012
	Poker Hand [58]	10	10	829,201
	Outdoor [59]	21	40	4,000
	Rialto [59]	27	10	82,250
	Airlines [60]	7	2	539,383
	Spam [9]	499	2	6,213
	Phishing [9]	46	2	11,055
Synthetic	Rotating Hyperplane [59]	10	2	200,000
	Moving Squares [59]	2	4	200,000
	Moving RBF [59]	10	5	200,000
	Interchanging RBF [59]	2	15	200,000
	Mixed [59]	2	15	600,000

5.1.1 Real-World Datasets

These datasets are large in size and variety. They are known to have multiple drifts. The drift types, number of drifts, drift speed, noise percentages are unknown. They are used extensively for research in learning under concept drift.

Some of them aren't naturally streams. Due to their large sample size, they are used as if they are a stream by adding time series properties.

ELEC. Initially it was introduced by Harris [56] as an open source data stream in 1999. Since then, it is used as a baseline in many data stream concept drift classification tasks. It consists data about the Australian New South Wales Electricity Market. The prices are affected by the supply and demand. Each sample in the data stream has 10 features such as date of the week, hour, demand and etc. They represent a period of 30 minutes. There are two classes: increase and decrease in prices. Unlike the features, the classes are determined with respect to the average of the last 24 hours. In total, there are 45,312 samples.

COVTYPE. This data is gathered in a research done by United States Forest Service (USFS) and has the data about the forest cover types of the 30x30 meter-squared areas. The research is done in the Roosevelt National Park which is located in North Colorado. The research area is chosen among the regions with minimum possible human intervention. Therefore, the acquired data is only the result of the natural process. It is used a baseline just like ELEC in multiple data stream applications [61]. Even though, it is not a data stream on its own, it is used as a stream due to the fact that it has a lot of samples. It has 54 features such as soil type, wilderness type and etc. There are 7 classes and 581,012 samples in total.

Poker Hand. It is formed by researchers in Carleton University by randomly drawing 5 cards from a regular poker deck. For every card, its suit (spades, diamonds...) and rank (Ace, King, Queen...) is used as a feature. There are 10 features in total. Depending on to the cards in the hand (for one sample), the class is given for example: one pair, two pair, flush and etc. There are 10 classes. In its original version, there is no concept drift as there is no change while assigning class labels. In our research, we use the version in Bifet's paper [61] which adds concept drift virtually by ordering cards with respect to suit and rank. There are 829,201 samples in total.

Rialto. This data is gathered with a steadicam from the famous Rialto Bridge

in Venice by monitoring 10 buildings in that area for 20 days [59]. The images are encoded with 27 dimensinal RGB histogram and normalized. The recordings are done for 20 days in May to June 2016. During this period, the changes in the weather and the lighting affected the recordings and naturally created a concept drift. Each building represents a class. There are 82,250 samples in total.

Airlines. It is prepared from the data from the Data Expo competition in 2009. It consists records of all commercial flights within USA starting from October 1987 to April 2008. Each sample represents a flight, having features: AirportFrom, AirportTo, time and length of the flight and etc. There are two classes: delayed and on-time. In our research, we use the version given in MOA [62]. In total, there are 539,383 samples.

Spam. This data is formed with the emails in the Spam Assassin Collection. In the original dataset, there are 4 classes: spam, spam2, ham and easy ham. Samples from easy ham class are legitimate messages which can be easily recognized [63]. 20% of the data is spam. Boolean bag-of-words approach is used for text vectorization. There are 500 features (words). The characteristics of spam messages gradually change as time passes [63]. In our research, we use the version in [9]. It has only 2 classes: spam and ham. In total, there are 6,213 samples.

Phishing. It is formed as a combination of malicious web pages and nsl-kdd dataset [64]. The main purpose for its generation is to build an intrusion detector, which can classify attacks (bad) and normal (good) connections. It has a potential to be used to filter malicious network traffic. There are two classes: attack and normal. We use the version in [9]. There are 46 features and 11,055 samples in total.

5.1.2 Synthetic Datasets

These datasets share same properties similarly as the real-world ones in terms of size and variety. Since they are generated by the researches in the field, the properties of the stream: drift types, number of drifts, drift speed, noise percentage

are known. They are used frequently in the concept drift research.

Rotating Hyperplane. It is created with Random Hyperplane Generator in MOA by using the parameters given in [61]. In a 10 dimensional space, an hyperplane is formed. Its rotation and orientation is constantly changed with a speed of 0.001. There are two classes. In total, there are 200,000 samples.

Moving Squares. Two equidistant, seperated squared uniform distributions are moving horrizontally with constant speed. The distribuiton in the front is reversed when it reaches a pre-defined limit which is defined as 120 in [59]. This prevents older samples to overlap. There are 2 classes and 200,000 samples in total.

Moving RBF. It is generated with the Random RBF generator in MOA using the parameters given in [61]. Gaussian distributions with randomly initialized starting positions, weights and standard deviations are moved with a constant speed in a 10 dimensional space. In total, there are 5 classes and 200,000 samples.

Interchanging RBF. Two dimensional Gaussian distributions with 15 random covariance matrices are replaced with each other in every 3000 samples. The number of interchanging distributions increase till a point when every one of them interchanges. The authors argue that this data stream is good for evaluating streams with abrupt drift of increasing strength [59]. There are 15 classes and 200,000 samples in total.

Mixed. It is generated as a mixture of the datasets: Interchanging RBF, Moving Squares and Transient Chessboard (this dataset is not used in the evaluation single-handedly). Samples from these datasets are consecutively put one after the other. The individual characteristics of each dataset is present in the stream. According to its authors, incremental, abrupt and virtual drift are present simultaneously, and local adaptation is required [59]. There are 600,000 samples.

5.2 Experimental Setup

Earlier version of this research is published as a conference paper (D3) [1] and a journal paper (OCDD) [2]. D3 is our first work on concept drift detection and OCDD is the extension of it with a new approach. While evaluating OCDD, we use D3 as a baseline. Therefore, the evaluation for OCDD also encapsulates the results for D3 as well. Experimental setups of D3 and OCDD are almost equal with minor differences. D3 is evaluated with less methods (3) and datasets (8). Furthermore, the methods that they use in their workflow are different. In OCDD, one-class SVM is used for one-class classification. Logistic regression is used as a discriminative classifier in D3. For both methods any one-class method and discriminative classifier can also be used.

The experiments are implemented in Python using the libraries: Scikit-learn [65], Scikit-multiflow [66] and Tornado [41]. Stream classification is done using a Hoeffding Tree (HT) [67]. Any online method that does not have a built-in concept drift adaptation mechanism can be employed as well. In a recent review, HT and Naïve Bayes were used to evaluate the performance of multiple drift detectors [68]. HT is chosen specifically as our goal is to focus on drift detection. We set stream classifier selection as a control variable in the experiments.

We use HT because of its recognition and effectiveness, as reported in the literature. It is operated with default parameters. If a drift is detected, the HT is reset and retrained with the latest samples available in the new data section of the sliding window for all drift detection methods. The time and memory requirements of drift detectors are negligible compared to training and updating the classifiers. Therefore, we do not provide any efficiency results for them.

For evaluation, we use the Interleaved Test-Then-Train approach, which is utilized extensively in streaming environments [8]. Whenever a new sample arrives, it is used by the classification model first for prediction, and an evaluation metric is stored; then it is used to update the model.

We compare D3 with 3 baseline methods: ADWIN, DDM and EDDM. We

Table 5.2: Concept drift detection methods we use for evaluation (excluding HAT which is the drift adaptive version of the HT)

Method	**Reference**	**Method**	**Reference**
D3	[1]	HAT	[69]
ADWIN	[47]	HDDM_A	[50]
CUSUM	[42]	HDDM_W	[50]
DDM	[43]	RDDM	[45]
EDDM	[44]	MDDM_A	[49]
EWMA	[46]	MDDM_E	[49]
FHDDM	[51]	MDDM_G	[49]
FHDDMS	[41]	Page-Hinckley	[42]
FHDDMS_A	[41]	Seq2D	[48]

compare OCDD against 17 drift detection methods, and an adaptive classifier, the Hoeffding Adaptive Tree (HAT). The methods are presented in Table 5.2. HAT is a modified version of the Hoeffding Tree that extends the performance of HT under concept drift. It constructs alternative branches, and switches them if their predictive accuracy is better. As we are using HT as a base classifier, we add HAT to the evaluation in order to observe how the concept adaptive version of HT performs compared to a using concept drift detector with the classifier. We choose the presented methods specifically as they are available open-source, mainly recent, and prevalent in the literature. Our goal is to compare D3 and OCCD's performances with as many well-established methods as possible.

All methods are used with default parameters. D3 is tuned with multiple parameters of $w = [100, 250, 500, 1000, 2500]$, $\rho = [0.1, 0.2, 0.3, 0.4, 0.5]$ and $\tau = [0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90]$. OCDD is tested with different choices of hyperparameters: $w = [100, 250, 500, 1000, 2500]$ and $\rho = [0.1, 0.2, 0.3, 0.4, 0.5]$. We make our implementation for both methods publicly available. They can be accessed from the GitHub link provided in the Appendix.

Chapter 6

Results and Discussion

In this chapter, we present the experimental results of D3 and OCDD separately. As OCDD is the extension of D3 using a different methodology, its evaluation includes the results of D3 as well. For both experimental setups, we evaluate the effects of different hyperparameter selections. We introduce *default parameters* for which D3 and OCDD perform well in all datasets. This makes D3 and OCDD to be used in high performance without an exhaustive hyperparameter search when testing it on a new dataset. Finally, we make a comparative analysis of D3 and OCDD with default parameters against the baselines specific to the experimental setup. We finalize our analysis with a statistical test for both settings.

6.1 D3

In this section, we present the experimental results of D3 with 3 concept drift detectors: ADWIN, DDM and EDDM under different scenarios. Only 8 of the datasets given in Table 5.1 are used.

6.1.1 Hyperparameter Selection

The hyperparameter choice is important for D3's performance. It determines on how much change is needed for D3 to detect the drifts. Every hyperparameter has a different effect on the algorithm. Their effect depends on to the properties of the dataset.

We tuned D3 with different hyperparameters on all datasets as it is described in the previous chapter. Three of the best results for every dataset is recorded along the hyperparameter settings and they can be seen in Tables 6.1 and 6.2.

Table 6.1: Top three parameters that give best results for real-world datasets

Real-World	**Accuracy**	**Parameters**		
Datasets	**D3***	w	ρ	τ
ELEC	86.72	100	0.1	0.55
	86.69	100	0.1	0.65
	86.69	100	0.1	0.70
COVTYPE	87.22	100	0.1	0.60
	87.21	100	0.1	0.55
	87.17	100	0.1	0.70
Poker Hand	76.22	100	0.1	0.90
	76.11	100	0.1	0.65
	76.09	100	0.1	0.60
Rialto	66.54	100	0.3	0.55
	63.50	100	0.3	0.60
	62.56	100	0.1	0.55

6.1.1.1 Old Data Window Size (w)

As it can be seen from the tables, the results are best when $w = 100$ for most cases. The effect of w is very high on overall accuracy. The results are consistent. When we use larger w's, the accuracy drops. This can be caused by multiple concepts filling one window. As window size increases, the variety of data in the window increases as well. This makes the current data to span more area in the feature space. As a result, the discriminative classifier would become less alert to changes as data is intertwined. There are more than one distribution in the window and the model learns the averaged version of them. It cannot detect drifts that originate from a similar feature space as the current window has.

Even though, D3 performs well with smaller windows, there is a limit for it. Data in the window should be large enough to represent a certain distribution.

Table 6.2: Top three parameters that give best results for synthetic datasets

Synthetic Datasets	**Accuracy**	**Parameters**		
	D3*	w	ρ	τ
Rotating Hyperplane	87.43	100	0.3	0.65
	87.41	100	0.1	0.80
	87.01	100	0.3	0.70
Moving Squares	96.27	100	0.1	0.55
	93.52	100	0.1	0.60
	91.20	100	0.3	0.55
Moving RBF	53.32	500	0.1	0.55
	53.21	250	0.1	0.55
	52.56	100	0.3	0.55
Interchanging RBF	93.62	250	0.1	0.55
	92.98	100	0.3	0.55
	90.33	500	0.1	0.55

When w is very small, D3 can not learn the distribution accurately and give false alarms even when there is no drift. Our experimental results show that the window size should be at least 100. We also observe that if the window is larger than 1000, the overall performance decreases.

6.1.1.2 New Data Percentage (ρ)

According to our results, the new data percentage is the least affecting parameter of D3. When it is set to 0.1 or 0.3, the model performs good for most cases. The results for ρ are consistent as well as w compared to τ. We observe that smaller ρ's are better performing. As we are aiming to detect a change in the new data, smaller windows for storing new samples are more robust. The new data region spans the feature space more as the size increases. D3 detects less drifts with ρ being large because, the likelihood of new data region coinciding with the

old increases. In such cases, the model needs more discrimination to detect a drift even when there is one. It affects the performance of the D3 adversely. In addition, small ρ's are ineffective because we need the new portion to represent a certain distribution. In such cases, D3 may consider outliers as a new concept if they are abundant in the window and detect a drift unnecessarily.

6.1.1.3 AUC threshold (τ)

According to the results, AUC threshold is the most important parameter with w. D3 performs best when $\tau \pm 0.6$. However, it performs better for datasets: Poker Hand and Rotating Hyperplane when τ is larger.

AUC is the measure of discrimination. It determines how much the model can separate two classes. For our case, the classes are old and new. Due to the properties of AUC, τ is in between 0.5 and 1. An AUC of 0.5 means that the classes are inseparable with the model as their distributions overlap. If it is equal to 1, it shows that the classes can be separated perfectly with no overlapping regions in their distributions. By setting a threshold to the AUC, we determine how much separation is required to detect a drift.

Therefore, using lower τ's increase the number of detected drifts. Except the datasets mentioned earlier, the others benefit from this condition. Setting the threshold to 0.5 would result detecting a drift every time when a test is made. Even though, the performance of the model is best when $tau = 0.6$ in general, the results show that AUC threshold is the most inconsistent parameter of our model. Small changes on it have huge impact on the final accuracy.

6.1.2 Comperative Evaluation

After evaluating it with multiple parameters, we found that it works best overall when $w = 100$, $\rho = 0.1$ and $\tau = 0.70$. They are set as the default parameters of our model. Parameters may be optimized for a selected dataset to achieve better

individual scores.

We analyze the results of D3 with default parameters. The overall accuracy for every method on each dataset is given in Table 6.3. The best scores are highlighted in bold. D3 does not utilize class labels unlike the other concept drift detectors. With less information, it outperforms them in most cases. When the accuracies for each dataset are ranked from best to worst, D3 comes first with an average rank of 1.38.

The other methods track the performance of the classifier and signal a drift when there is a significant drop. They wait for the new concept to affect the performance of the classifier adversely and act afterwards. According to the results, we see that D3 acts quicker by signaling a drift when there is a change in $P(X)$. In Figure 6.1, we give the prequential accuracy scores for 4 different datasets. It shows that the locations of the drifts (declines in accuracy) are in similar areas, but the classifier adapts to it faster with D3.

Table 6.3: Averaged prequential accuracy and the rankings of the methods for each dataset

Datasets	**Accuracy (%)**			
	D3	ADWIN	DDM	EDDM
ELEC	**86.69**	81.33	79.30	78.25
COVTYPE	**87.17**	80.48	83.36	82.76
Poker Hand	**75.59**	66.96	73.42	71.31
Rialto	**52.39**	46.64	38.44	51.01
Rotating Hyperplane	85.29	**87.28**	84.01	80.27
Moving Squares	66.28	**67.89**	45.13	33.68
Moving RBF	**51.59**	40.21	35.04	35.33
Interchanging RBF	82.81	**88.65**	41.23	60.06
Average Rank	**1.38**	2.25	3.13	3.25

However, this does not apply for all cases. D3 can not detect concept drifts

caused by changes only in $P(y|X)$ which are referred to as *real concept drift*. In such cases, $P(X)$ stays the same; therefore, our method cannot detect the change. Furthermore, it detects drifts unnecessarily when the change in $P(X)$ does not affect $P(y|X)$ *(virtual drift)*. D3's performance on real-world datasets is better compared to synthetics. This can be caused by the properties of the datasets. In synthetic datasets, real and virtual concept drifts can be more dominant. Even under these drawbacks, the results presented in Table 6.3 and Figure 6.1 confirm that D3 works well compared to the other methods.

6.2 OCDD

In this section, we present the experimental results of OCDD against all drift detectors given in Table 5.1 and methods in Table 5.2.

6.2.1 Setting the Parameters of OCDD

The overall accuracy of OCDD with different hyperparameter settings is presented in Table 6.4. For brevity, we only show some of the parameter settings we experimented with during our analysis. We observe that both parameters influence predictive accuracy. Setting w is important as it determines the number of samples that represent the concept at a time. If it is set very small, the data may not span the area for a concept. Then, one-class SVM would detect new samples originally from the concept as outliers, resulting in inaccurately detected drifts. On the other hand, setting w too large may cause multiple concepts to appear inside the sliding window. This degrades the performance of the one-class classifier, resulting poor performance on outlier detection, and drift detection. The parameter ρ is the threshold for the percentage of outliers needed for drift detection, and if it is set low, OCDD detects drifts needlessly. Even small changes in the data that do not require the classification model to be retrained are also identified as a drift. However, when ρ is set high, OCDD is more conservative while signaling drifts, causing it to ignore drifts which are not abrupt.

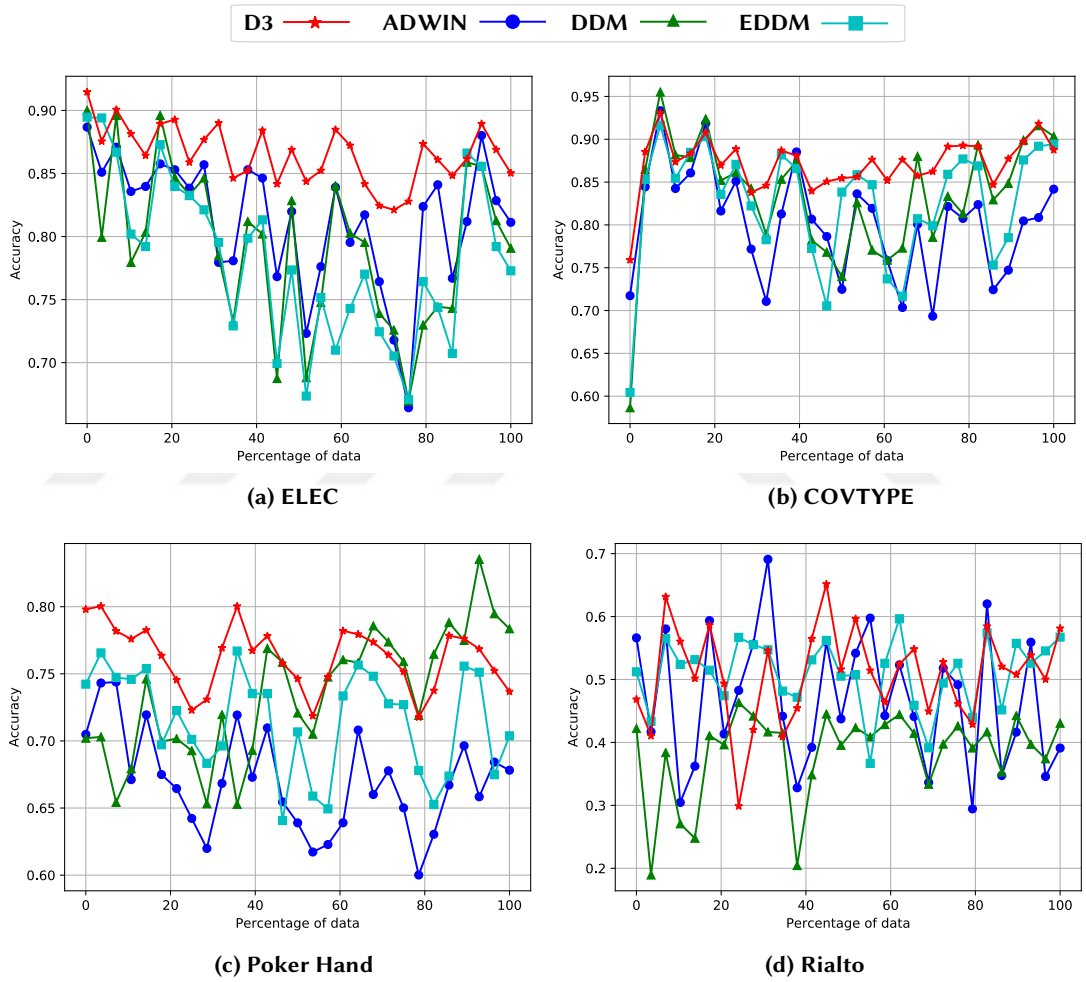

Figure 6.1: Prequential accuracy of the methods for the real-world datasets. Each dataset is divided into 30 chunks and the results are the averaged prequential accuracies within them.

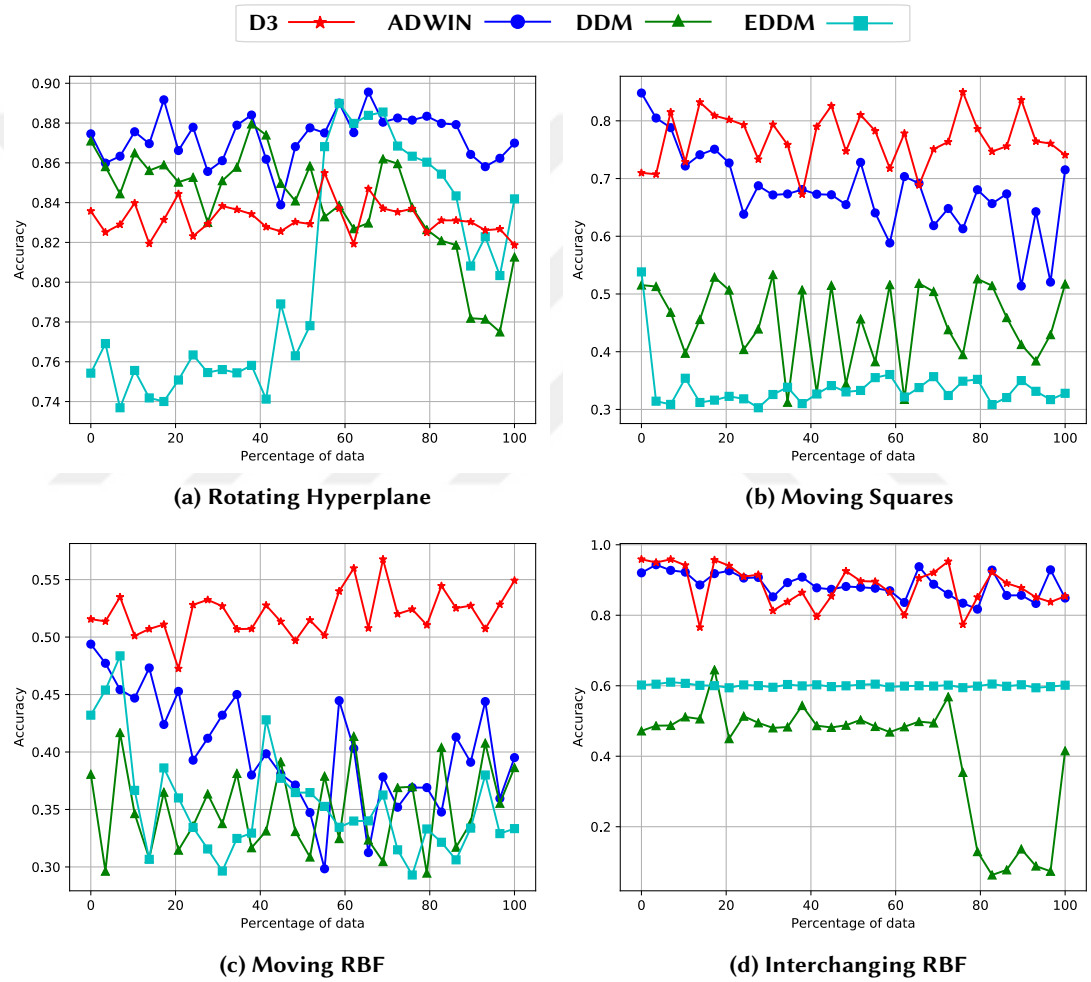

Figure 6.2: Prequential accuracy of the methods for the synthetic datasets. Each dataset is divided into 30 chunks and the results are the averaged prequential accuracies within them.

Table 6.4: Overall accuracy of OCDD with multiple parameter settings for each dataset with the best scores highlighted in bold

	Parameters of OCDD (w, ρ)								
	100	100	100	250	250	250	1000	1000	1000
Datasets	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5
ELEC	85.56	87.33	82.07	**88.49**	86.22	79.77	82.04	80.91	78.32
COVTYPE	88.31	88.12	86.35	**88.59**	88.36	82.45	83.15	81.29	81.75
Poker Hand	**78.07**	76.37	74.01	76.79	75.29	73.48	73.90	70.01	67.87
Outdoor	58.71	58.59	60.15	58.85	**62.24**	59.95	59.83	60.63	59.73
Rialto	13.18	**68.09**	60.41	62.56	66.68	52.91	63.23	49.66	38.72
Airlines	59.95	60.31	62.68	60.02	**63.16**	62.71	61.35	62.01	62.71
Spam	78.27	80.71	85.80	82.97	87.02	**87.93**	86.67	87.26	87.78
Phissing	69.91	85.32	89.65	83.28	90.56	89.68	**90.81**	90.27	90.22
Rotating Hyperplane	62.37	84.12	82.60	74.41	86.01	84.09	84.10	**87.61**	84.85
Moving Squares	87.02	**99.36**	83.21	99.15	95.89	78.94	90.18	82.05	74.65
Moving RBF	33.04	45.42	46.01	43.48	**61.95**	40.01	53.31	55.52	38.68
Interchanging RBF	24.95	75.37	30.49	66.51	**97.02**	25.51	97.46	93.17	44.89
Mixed	35.15	41.62	44.75	38.89	**46.15**	38.45	43.91	47.58	40.72

For most datasets, setting both w and ρ low or high yields poor predictive performance, as they both affect the number of drifts being detected. We recommend that there should be a balance between the two, and both should not be set to too low or high values at the same time. If there is a gradual or an incremental change in the data (in an application area such as sensor monitoring, where the sensor gets old and not giving accurate results in time), we recommend using low w or ρ, making OCDD to be more sensitive to small changes. When the change is abrupt (monitoring daily trends in social media, where the trend changes suddenly), it is better to use high w or ρ, resulting OCDD to focus less on small changes in the data. It should be mentioned that multiple independent learners and multiple concept drift detectors can be active at the same time reflecting different concerns for the same data stream. Furthermore, an ensemble of classifiers can be used to address different worries as well [21, 70].

After analyzing OCDD with multiple parameters, we find that it performs best for majority of the datasets when $w = 250$ and $\rho = 0.3$; and we set these values as the default parameters. The other methods have default parameters similar to OCDD. In order to have a better predictive accuracy for a dataset, parameters may be optimized. We recommend users to tune the parameters starting from the default. If the user has extra information on the dataset, specifically on the drift pattern, parameters can be changed to match the nature of that specific data stream.

6.2.2 Comparative Evaluation

The overall accuracies for all methods are presented in Table 6.5. The best scores are highlighted in bold for each dataset. The results show that OCDD outperforms other methods, having the highest average rank when the overall accuracies on each stream are ranked from the best to worst. Apart from D3, the other methods are explicit, utilizing more information by using true class labels. They have a significant advantage over OCDD as they detect drifts by verifying the predictions of the classifier. However, they are not useful for detecting the

drifts in cases of verification latency, due to their dependence on the true class labels. OCDD notably performs better on detecting concept drifts with less information.

Explicit methods track the predictive performance and signal a drift in case of a considerable decrease in accuracy. They first need to observe a drop in predictive performance, then they detect change. OCDD can react faster to drifts since it monitors the changes in data characteristics without waiting for the predictive performance to drop. This can be observed in Figure 6.3, where we present the prequential accuracies for 4 datasets. Particularly in plot c, the location of the drifts which we identify as the places where accuracy drops are in similar points for all methods. However, we can observe that OCDD is quicker in adapting in these situations and the performance of the classifier is better.

The main weakness of OCDD is, it detects drifts redundantly when there are virtual drifts in which $P(X)$ changes, but $P(y|X)$ does not. The classifier is unnecessarily modified and the model loses necessary information. Moreover, it cannot detect concept drifts in which changes happen only on $P(y|X)$. Even with these weaknesses, our results confirm that OCDD performs well, outperforming other methods overall, with the highest rank. Classification without using a drift detector, NONE, has the second lowest average rank. This shows that drift detection is necessary to achieve better performing models in evolving data streams. HAT has a better ranking on overall accuracy to NONE and 3 drift detectors. Even though, it can create and replace branches as data changes to adapt to the new concept, it only has a slightly better predictive performance compared to NONE. Consequently, we observe that retraining the base classifier (HT) results in better predictive performances than modifying it for most cases. Fifteen drift detectors have better rankings on overall accuracy compared to HAT.

Concept drift detectors are dependent on human expertise for solving complex tasks like parameter tuning. Most drift detectors have default parameters which they perform best on overall or specific to a certain scenario (drift type). We also follow a similar approach and present default parameters of our model along with the recommendations to tune it under different conditions. This approach

has major drawbacks as it does not take account the possible differences in the testing environments and the real-life scenarios. Even though OCDD is evaluated on datasets from various application domains and a default parameter setting is determined, its performance might not be a good in some real-life use cases. In such circumstances, tuning the parameters might be necessary according to the user responses. All drift detectors suffer from this problem.

Self Parameter Tuning (SPT) is introduced to solve this issue by dynamically updating the model parameters while the stream is being processed [71]. This can help the drift detectors when the characteristics of the drift change over time. A data stream may have more than one drift type in different time intervals. There can be gradual drifts in the start and then the change can become abrupt. As an example, a sensor getting old and not giving accurate results in time can be considered as a gradual drift. When this sensor is replaced with another one , an abrupt drift may be observed. These type of changes happen in real-life scenarios a lot. In such cases, the drift detectors performance might not be at its best as they are tuned for the setting (drift type) in the start of the data stream. Algorithms like SPT can be useful to solve this issue by dynamically changing the drift detectors parameters when the stream is running. However, these approaches are fairly new and not applied in practice for concept drift detection. They are used mainly used for regression tasks. Evaluating SPT with drift detectors, and seeing its effects on the performance can be good research agenda in future works.

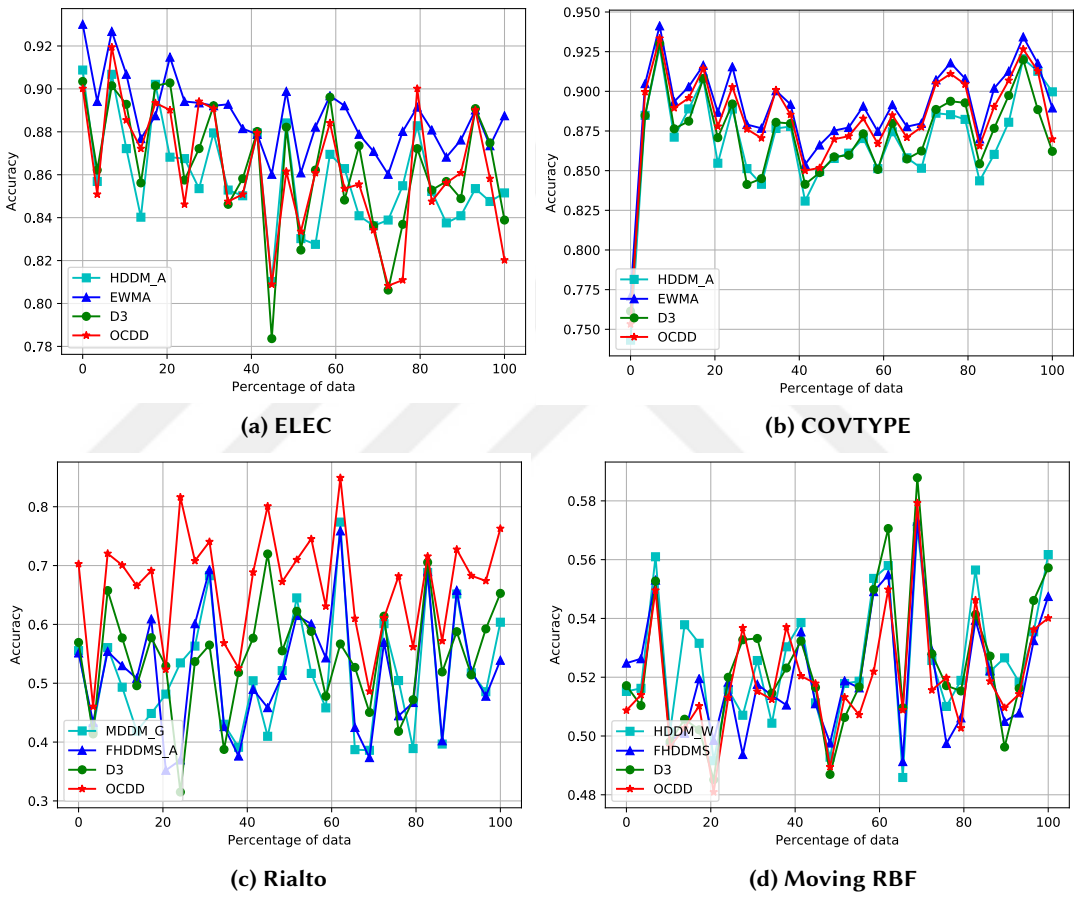

Figure 6.3: Prequential accuracy of the methods for the selected datasets. Each dataset is divided into 30 chunks and the results are the averaged prequential accuracies within each chunk. OCDD, D3 and the remaining top two methods are presented for each dataset. D3 is the only implicit method other than OCDD; therefore, it is added to the plots regardless of its performance.

Table 6.5: Overall accuracy and the average rankings of the methods for each stream with the best scores highlighted in bold (continues on the next page)

	Accuracy									
Datasets	**OCDD**	**D3**	**ADWIN**	**CUSUM**	**DDM**	**EDDM**	**EWMA**	**FHDDM**	**FHDDMS**	**FHDDMS_A**
ELEC	86.22	86.69	81.33	81.48	79.3	78.25	**88.76**	82.91	83.91	82.89
COVTYPE	88.36	87.17	80.48	82.67	83.36	82.76	**89.09**	83.78	84.65	83.78
Poker Hand	75.29	75.99	66.96	71.63	73.42	71.31	**76.83**	74.60	75.64	74.68
Outdoor	62.24	60.00	**63.66**	58.55	62.01	59.12	61.05	60.78	63.43	60.78
Rialto	**66.68**	52.39	46.64	45.28	38.44	51.01	33.03	50.91	49.42	51.72
Airlines	63.16	60.50	62.80	63.72	64.67	**64.80**	61.15	62.39	61.62	62.25
Spam	87.02	82.42	84.72	87.70	80.96	82.12	85.51	87.59	86.97	86.70
Phissing	**90.56**	86.88	89.22	90.70	89.98	80.82	89.39	89.98	89.98	89.98
Rotating Hyperplane	86.01	85.29	87.28	87.63	84.01	80.27	86.28	86.41	86.41	85.87
Moving Squares	**95.89**	66.28	67.89	83.96	45.13	33.68	95.88	87.47	87.52	86.82
Moving RBF	51.95	51.59	40.21	50.14	35.04	35.33	33.92	51.20	52.04	50.62
Interchanging RBF	97.02	82.81	88.65	97.52	41.23	60.06	98.55	98.08	98.48	97.91
Mixed	46.15	44.87	44.83	47.92	36.85	39.9	40.32	48.17	48.18	48.49
Average Rank	5.62	11.46	13.38	10.54	14.23	15.85	10.31	8.85	7.00	9.62

	Accuracy									
Datasets	**HAT**	**HDDM_A**	**HDDM_W**	**RDDM**	**MDDM_A**	**MDDM_E**	**MDDM_G**	**Page-Hinckley**	**Seq2D**	**NONE**
ELEC	81.71	85.95	84.65	84.94	83.35	83.59	83.59	79.47	81.53	77.96
COVTYPE	80.98	86.99	85.61	86.08	84.05	84.15	84.16	80.66	82.41	82.49
Poker Hand	64.65	75.78	75.86	75.44	74.94	75.10	75.09	67.39	70.83	73.39
Outdoor	59.52	62.03	62.48	61.10	60.83	60.85	60.85	61.28	59.07	59.65
Rialto	31.65	46.97	48.93	46.39	50.14	51.19	51.23	33.56	50.49	33.03
Airlines	62.64	62.69	61.45	63.30	62.26	62.24	62.24	62.91	62.90	63.90
Spam	87.27	86.31	**88.05**	87.23	86.22	87.04	87.04	85.58	85.15	86.69
Phissing	88.73	89.98	89.98	89.52	89.98	89.98	89.98	89.59	89.98	89.96
Rotating Hyperplane	85.37	86.91	85.89	**87.97**	86.51	86.53	86.53	86.83	86.57	84.09
Moving Squares	86.22	86.61	87.58	94.97	87.83	87.92	87.90	47.10	84.65	33.66
Moving RBF	36.36	49.78	**52.54**	50.99	51.26	51.28	51.29	35.46	46.95	33.92
Interchanging RBF	62.15	98.67	98.53	**98.79**	98.28	98.36	98.36	85.56	89.70	25.80
Mixed	**49.07**	48.48	48.73	48.15	48.24	48.04	48.05	42.28	45.92	44.97
Average Rank	14.00	6.69	6.08	6.54	8.85	7.46	7.31	14.08	12.31	15.54

6.3 Statistical Analysis

We used the *Friedman Test with Nemenyi post-hoc analysis* to check the statistical significance of the used methods. The test is applied with $\alpha = 0.05$. The null hypothesis is that there is no significant difference between the measurements. In other words, the results share the same distribution. If it fails, Nemenyi post-hoc analysis is applied in order to see which method is statistically significantly better compared to the others.

Firstly, the methods are ranked on every task individually and then their average is taken. Models that are better have lower averaged ranks. The critical distance for Nemenyi Significance is calculated by plugging in values specific for the experimental setting from the *Critical Values Table for Two Tailed Nemenyi Test* [72].

$$CD = q_{\alpha,m}\sqrt{\frac{m(m+1)}{6|\mathcal{D}|}} \tag{6.1}$$

If the averaged rank of a method is not within the critical distance of an other, it indicates that the one with the lower rank is statistically significantly better compared to the other. We show the results in *Critical Distance Diagram* where the averaged ranks are sorted on a number line and the methods which are in the range of critical distance of each other are linked with a line.

For the first setting focusing on D3 (Table 6.3), $m = 4$ (the number of models) and $|\mathcal{D}| = 8$ (the number of datasets). We find $CD = 1.66$ (see Figure 6.4 resulting D3 to be statistically significantly better than DDM and EDDM.

For the second setting focusing on OCDD (Table 6.5), $m = 20$ (the number of models) and $|\mathcal{D}| = 13$ (the number of datasets). We calculate $CD = 8.22$ (see Figure 6.5 showing OCDD to be statistically significantly better than three prominent drift detection methods: DDM, EDDM and the Page-Hinckley test, and one adaptive classifier: HAT. OCDD is on par with the remaining ones.

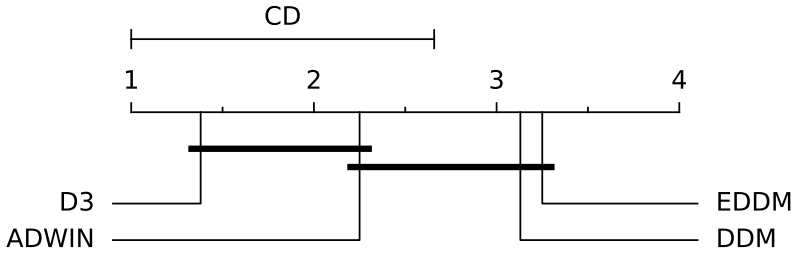

Figure 6.4: Critical Distance Diagram for the overall accuracy using the data provided in Table 6.3. (CD=1.66)

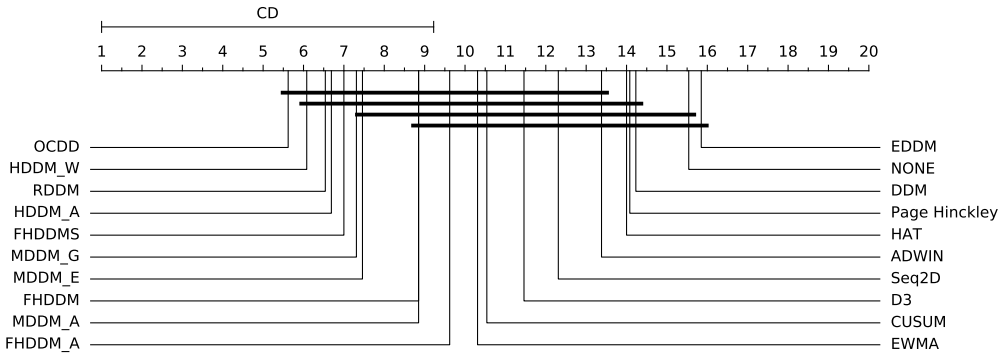

Figure 6.5: Critical distance diagram for the overall accuracy using the data provided on Table 6.5. (CD=8.22)

6.3.1 Shortcomings in the Literature of Concept Drift Detectors

While doing research on unsupervised concept drift detection, we faced certain problems and weak points in the literature that need to be addressed. Lack of theoretical foundation, issues with the evaluation methodologies, and lack of open-source implementations are a few of the major problems that we have encountered.

The Statistical Learning Theory (SLT) [52] provides a theoretical framework for supervised machine learning algorithms, ensuring that the constructed models generalize on the train data. SLT can only be applied if labels are present, and

therefore it cannot be used in unsupervised learning. Empirical Risk Minimization is used to give theoretical bounds on the model performance, guaranteeing generalization [73].

On the other hand, unsupervised learning in general does not rely on a theoretical framework that ensures the generalization of the data. The results are mostly evaluated according to internal or external measures [74]. Internal measures do not need a priori information from dataset. They are calculated according to the output of the trained model, checking if the resulting structure is formed well. Depending on the type of the unsupervised task, the definition of structure and well can be different. In clustering, compactness of the clusters and distance between the clusters can be a measure to assess performance. External measures require a priori information to be available for the dataset: The results are evaluated according to a prespecified structure (labels), and a measure is calculated. However, this process is against the nature of the unsupervised learning as it is assumed that labels are unavailable. In conditions when there are no labels to verify the performance of the model externally, unsupervised learning lacks theoretical learning guarantees, and the results may be obtained by chance [31].

In unsupervised concept drift detection, most of the algorithms do not provide theoretical learning guarantees [31]. The methods are evaluated indirectly, according to the accuracy of the predictive model, without relying any theoretical foundation to assure their performance. The concept of Algorithmic Stability [75] is suggested to solve this problem by checking the probabilistic convergence of a function and its expected value [31]. By this way, stability of an unsupervised algorithm can be verified, however it requires a suitable function (internal measure) to be selected depending on the application domain. A theoretical framework that can be applied to all unsupervised concept drift detection tasks is unavailable.

Typically, a new supervised concept drift detector is evaluated with a classifier where the classifier is modified when a drift is detected [76]. The predictive accuracy of this setting is considered as a good measure of the quality of the drift detector. However, Bifet claims this approach is risky as there may be temporal dependencies in the data [76]. In his work, he experiments with two datasets,

and compares three drift detectors: ADWIN, DDM and EDDM with a pseudo drift detector that signals a drift every 60 samples. For both datasets, 60-sample detector outperforms the others, showing that evaluating a drift detector based on classification accuracy is not enough. In another study, ELEC dataset is evaluated to see whether it is a good benchmark for concept drift research [77]. The results show that a random drift detector that arbitrarily signals drift without using any data from the data stream is on par with the state-of-the-art approaches in terms of predictive accuracy. In this work, we observe that 15 concept drift detection methods other than OCDD, HDDM_W. RDDM, HDDM_A, FHDDMS does perform statistically significantly different from NONE, ie., using no concept drift detector (see Figure 6.4.

Bifet proposes other evaluation techniques, but they are not applicable for most datasets, as they require the true location of the drifts [76]. Only the drift locations of the artificial datasets are known. Therefore, even the latest works still evaluate the quality of the concept drift detector based on the predictive performance [68].

Another observation is that if a researcher selects an evaluation methodology, one of the main problems while comparing unsupervised concept drift detectors is the lack of open-source implementations. Although there are several works on the topic, almost none of them have publicly available implementations, making it hard for researchers to compare their own methods performance to the others.

Computing today mainly focuses on efficiency which is defined as the optimal adaptation to an existing environment [78]. Resilience, the capacity to adapt to disruptive changes, is usually ignored. Vardi discusses the importance of resilient algorithms and points out that there is a trade-off between efficiency and resilience. Google's PageRank algorithm is given as an example of an efficient algorithm which lacks resilience. Therefore, it is prone to manipulations, and for this reason the research field of search-engine optimization is introduced.

In a data stream classification environment, which is well-balanced in terms of efficiency and resilience trade-off, a classifier avoids unnecessary computations

and remains desirably satisfactory under various types of concept drift. The goal of concept drift detection is to make streaming algorithms (classification, clustering, etc.) more resilient to the changes in the environment. However, the resilience of concept drift algorithms is still an open question. Ensemble learning is applied in many machine learning areas to increase the resilience of an algorithm at the expense of its efficiency. Studying ensemble approaches within the framework of efficiency and resilience trade-off can be a good research area. In this regard, our work: "less is more" tries to achieve a sustainable (more resilient) performance with lesser number of ensemble components (more efficient) [79].

Chapter 7

Conclusion and Future Work

In this thesis, we present two unsupervised methods for concept drift detection, D3 and OCDD. In D3, we use a discriminative classifier over a sliding window of latest elements to monitor whether the old and the new samples vary in their distributions. We evaluate D3 using 8 datasets against 3 drift detectors. The experimental results show that D3 performs better overall, compared to the most extensively used methods for drift detection: ADWIN, DDM and EDDM.

In OCDD, we use a one-class classifier with a sliding time window to monitor whether new data is generated from a concept different to the current one. We evaluate OCDD using 13 datasets of a wide variety from different appliaction areas against 17 drift detection methods and an adaptive classifier. Other than D3, the methods are all supervised. The results demonstrate that OCDD has the highest average rank in predictive accuracy. Even without utilizing class labels, it is on the same level with most of the drift detectors statistically, while significantly outperforming three of them and one adaptive classifier (HAT). One of the main issues when conducting a research on implicit drift detection is the lack of open-source implementations of the methods. We make our implementation publicly available along with the datasets used in our experimental analysis.

As a future work, we plan to investigate the kernelized version of D3. Throughout the experiments, we use Logistic Regression, which limits us to detect drifts that show a linear pattern on $P(X)$. OCDD can react to non-linear changes and its performance is better compared to D3. Therefore, using a kernel or different classifier may boost its drift detection performance. We want to test our approach with different types of classifiers, including non-linear ones, such as: SVMs and Decision Trees.

Future research possibilities include studying incremental one-class classification and adapting currently available methods to OCDD [4]. During the experiments, we train one-class classifier in batch form, but it can be improved by using an incremental method which updates itself while the stream is being processed. Several interesting aspects may be explored further by using different types of one-class methods such as: Isolation Forests [80] or Local Outlier Factor [81].

We use the default parameters for D3 and OCDD, with which they perform well for the majority of datasets. However, there are better parameter settings for individual datasets, where performance is higher. Therefore, an adaptive parameterization method that can dynamically change the parameters of D3 or OCDD according to datasets characteristics can significantly boost prediction accuracy.

Bibliography

[1] Ö. Gözüaçık, A. Büyükçakır, H. Bonab, and F. Can, "Unsupervised concept drift detection with a discriminative classifier," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 2365–2368, ACM, 2019.

[2] Ö. Gözüaçık and F. Can, "Concept learning using one-class classifiers for implicit drift detection in evolving data streams," *Artificial Intelligence Review*, 2020 (under review).

[3] W. Fan and A. Bifet, "Mining big data: current status, and forecast to the future," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 1–5, 2013.

[4] B. Krawczyk and M. Woźniak, "One-class classifiers with incremental learning and forgetting for data streams with concept drift," *Soft Computing*, vol. 19, no. 12, pp. 3387–3400, 2015.

[5] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, *et al.*, "Open challenges for data stream mining research," *ACM SIGKDD Explorations Newsletter*, vol. 16, no. 1, pp. 1–10, 2014.

[6] D. Laney, "3d data management: Controlling data volume, velocity and variety," *META group research note*, vol. 6, no. 70, p. 1, 2001.

[7] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[8] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[9] T. S. Sethi and M. Kantardzic, "On the reliable detection of concept drift from streaming unlabeled data," *Expert Systems with Applications*, vol. 82, pp. 77–99, 2017.

[10] M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 6, pp. 859–874, 2011.

[11] I. Žliobaite, "Change with delayed labeling: When is it detectable?," in *2010 IEEE International Conference on Data Mining Workshops*, pp. 843–850, IEEE, 2010.

[12] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[13] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer, "Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances," *Information Sciences*, vol. 355, pp. 127–151, 2016.

[14] S. Chandra, A. Haque, L. Khan, and C. Aggarwal, "An adaptive framework for multistream classification," in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, pp. 1181–1190, ACM, 2016.

[15] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. The MIT Press, 2009.

[16] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning for differing training and test distributions," in *Proc. of the 24th ICML*, pp. 81–88, ACM, 2007.

[17] C. C. Aggarwal, *Data streams: models and algorithms*, vol. 31. Springer Science & Business Media, 2007.

[18] A. Tsymbal, "The problem of concept drift: Definitions and related work," *Tech. rep. Department of Computer Science, Trinity College, Dublin*, 2004.

[19] E. R. Faria, J. Gama, and A. C. Carvalho, "Novelty detection algorithm for data streams multi-class problems," in *Proc. of the 28th Annual ACM Symposium on Applied Computing*, pp. 795–800, ACM, 2013.

[20] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[21] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[22] P. Zhang, X. Zhu, and Y. Shi, "Categorizing and mining concept drifting data streams," in *Proc. of the 14th ACM SIGKDD*, pp. 812–820, ACM, 2008.

[23] H. Hu, M. Kantardzic, and T. S. Sethi, "No free lunch theorem for concept drift detection in streaming data classification: A review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 2, pp. 1327–1351, 2020.

[24] E. J. Spinosa, A. P. de Leon F de Carvalho, and J. Gama, "OLINDDA: A cluster-based approach for detecting novelty and concept drift in data streams," in *Proc. of the 2007 ACM Symposium on Applied Computing*, pp. 448–452, ACM, 2007.

[25] M. Z. Hayat and M. R. Hashemi, "A DCT based approach for detecting novelty and concept drift in data streams," in *2010 International Conference of Soft Computing and Pattern Recognition*, pp. 373–378, IEEE, 2010.

[26] F. Can, "Incremental clustering for dynamic information processing," *ACM Transactions on Information Systems (TOIS)*, vol. 11, no. 2, pp. 143–164, 1993.

[27] J. W. Ryu, M. M. Kantardzic, M.-W. Kim, and A. R. Khil, "An efficient method of building an ensemble of classifiers in streaming data," in *International Conference on Big Data Analytics*, pp. 122–133, Springer, 2012.

[28] T. S. Sethi, M. Kantardzic, and H. Hu, "A grid density based framework for classifying streaming data in the presence of concept drift," *Journal of Intelligent Information Systems*, vol. 46, no. 1, pp. 179–211, 2016.

[29] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream," in *30th AAAI Conference on Artificial Intelligence*, 2016.

[30] J. Lee and F. Magoules, "Detection of concept drift for learning from stream data," in *2012 IEEE 14th HPCC & 2012 IEEE 9th ICESS*, pp. 241–245, IEEE, 2012.

[31] R. F. de Mello, Y. Vaz, C. H. Grossi, and A. Bifet, "On learning guarantees to unsupervised concept drift detection on data streams," *Expert Systems with Applications*, vol. 117, pp. 90–102, 2019.

[32] L. I. Kuncheva and W. J. Faithfull, "Pca feature extraction for change detection in multidimensional unlabeled data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 69–80, 2014.

[33] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A PCA-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams," in *Proc. of the 21th ACM SIGKDD*, pp. 935–944, ACM, 2015.

[34] A. Dries and U. Rückert, "Adaptive concept drift detection," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 2, no. 5-6, pp. 311–327, 2009.

[35] M. Dredze, T. Oates, and C. Piatko, "We're not in Kansas anymore: Detecting domain changes in streams," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 585–595, Association for Computational Linguistics, 2010.

[36] P. Lindstrom, B. Mac Namee, and S. J. Delany, "Drift detection using uncertainty distribution divergence," *Evolving Systems*, vol. 4, no. 1, pp. 13–25, 2013.

[37] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer, "Concept drift detection through resampling," in *International Conference on Machine Learning*, pp. 1009–1017, 2014.

[38] J. Demšar and Z. Bosnić, "Detecting concept drift in data streams using model explanation," *Expert Systems with Applications*, vol. 92, pp. 546–559, 2018.

[39] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Statistical change detection for multi-dimensional data," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 667–676, 2007.

[40] F. Pinto, M. O. Sampaio, and P. Bizarro, "Automatic model monitoring for data streams," *arXiv preprint arXiv:1908.04240*, 2019.

[41] A. Pesaranghader, H. Viktor, and E. Paquet, "Reservoir of diverse adaptive learners and stacking fast Hoeffding drift detection methods for evolving data streams," *Machine Learning*, vol. 107, no. 11, pp. 1711–1743, 2018.

[42] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[43] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian Symposium on Artificial Intelligence*, pp. 286–295, Springer, 2004.

[44] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Fourth International*

Workshop on Knowledge Discovery from Data Streams, vol. 6, pp. 77–86, 2006.

[45] R. S. Barros, D. R. Cabral, P. M. Gonçalves Jr, and S. G. Santos, "RDDM: Reactive drift detection method," *Expert Systems with Applications*, vol. 90, pp. 344–355, 2017.

[46] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognition Letters*, vol. 33, no. 2, pp. 191–198, 2012.

[47] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proc. of the 2007 SIAM SDM*, pp. 443–448, SIAM, 2007.

[48] R. Pears, S. Sakthithasan, and Y. S. Koh, "Detecting concept change in dynamic data streams," *Machine Learning*, vol. 97, no. 3, pp. 259–293, 2014.

[49] A. Pesaranghader, H. L. Viktor, and E. Paquet, "Mcdiarmid drift detection methods for evolving data streams," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, IEEE, 2018.

[50] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2014.

[51] A. Pesaranghader and H. L. Viktor, "Fast Hoeffding drift detection method for evolving data streams," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 96–111, Springer, 2016.

[52] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.

[53] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[54] E. R. Faria, I. J. C. R. Gonçalves, A. C. P. L. F. de Carvalho, and J. Gama, "Novelty detection in data streams," *Artificial Intelligence Review*, vol. 45, no. 2, pp. 235–269, 2016.

[55] D. M. J. Tax *et al.*, "One-class classification, concept learning in the absence of counter example," *Delft University of Technology*, 2001.

[56] M. Harries and N. S. Wales, "Splice-2 comparative evaluation: Electricity pricing," 1999.

[57] J. A. Blackard, D. J. Dean, and C. W. Anderson, "The Forest Covertype dataset," *UCI Machine Learning Repository*, 1998.

[58] D. Dua and C. Graff, "The Pokerhand dataset," *UCI Machine Learning Repository*, 2017.

[59] V. Losing, B. Hammer, and H. Wersing, "KNN classifier with self adjusting memory for heterogeneous concept drift," in *2016 IEEE 16th ICDM*, pp. 291–300, IEEE, 2016.

[60] A. D. Expo, "Airline on-time performance, asa section on: Statistical computing statistical graphics," 2009.

[61] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *Proc. of the 28th Annual ACM Symposium on Applied Computing*, pp. 801–806, ACM, 2013.

[62] A. Bifet, G. Holmes, B. Pfahringer, J. Read, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: a real-time analytics open source framework," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 617–620, Springer, 2011.

[63] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: an application to email filtering," *Knowledge and Information Systems*, vol. 22, no. 3, pp. 371–391, 2010.

[64] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, IEEE, 2009.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,

D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[66] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: A multi-output streaming framework," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2915–2914, 2018.

[67] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 71–80, 2000.

[68] R. S. M. Barros and S. G. T. C. Santos, "A large-scale comparison of concept drift detectors," *Information Sciences*, vol. 451, pp. 348–370, 2018.

[69] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *International Symposium on Intelligent Data Analysis*, pp. 249–260, Springer, 2009.

[70] H. R. Bonab and F. Can, "GOOWE: Geometrically optimum and online-weighted ensemble classifier for evolving data streams," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 2, pp. 1–33, 2018.

[71] B. Veloso, J. Gama, and B. Malheiro, "Self hyper-parameter tuning for data streams," in *International Conference on Discovery Science*, pp. 241–255, Springer, 2018.

[72] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1–30, 2006.

[73] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," in *Summer School on Machine Learning*, pp. 169–207, Springer, 2003.

[74] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz, "Internal versus external cluster validation indexes," *International Journal of Computers and Communications*, vol. 5, no. 1, pp. 27–34, 2011.

[75] O. Bousquet and A. Elisseeff, "Stability and generalization," *Journal of Machine Learning Research*, vol. 2, no. Mar, pp. 499–526, 2002.

[76] A. Bifet, "Classifier concept drift detection and the illusion of progress," in *International Conference on Artificial Intelligence and Soft Computing*, pp. 715–725, Springer, 2017.

[77] I. Zliobaite, "How good is the electricity benchmark for evaluating concept drift adaptation," *arXiv preprint arXiv:1301.3524*, 2013.

[78] M. Y. Vardi, "Efficiency vs. resilience: What COVID-19 teaches computing," *Communications of the ACM*, vol. 63, no. 5, pp. 9–9, 2020.

[79] H. Bonab and F. Can, "Less is more: A comprehensive framework for the number of components of ensemble classifiers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2735–2745, 2019.

[80] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, IEEE, 2008.

[81] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Loop: local outlier probabilities," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pp. 1649–1652, 2009.

Appendix A

Code and Data

The experiments are implemented with Scikit-multiflow [66] which is an open source data stream library in Python. They can be reproduced with the code available in:

- D3: https://github.com/ogozuacik/d3-discriminative-drift-detector-concept-drift.
- OCDD: https://github.com/ogozuacik/one-class-drift-detection.

Instructions on how to run the code and reproduce the experiments are provided in the given link.

Datasets used in the experiments can be found in:

- https://github.com/ogozuacik/concept-drift-datasets-scikit-multiflow

There are other dataset with concept drift which are edited to work directly with Scikit-multiflow.