Artificial Intelligence Review (2021) 54:3725–3747
https://doi.org/10.1007/s10462-020-09939-x

Concept learning using one-class classifiers for implicit drift detection in evolving data streams

Ömer Gözüaçık[1] · **Fazli Can**[1]

Published online: 20 November 2020

Abstract

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 a near-limitless volume of data in temporal order. Such data are referred to as data streams, and are generally nonstationary as the characteristics of data evolves over time. This phenomenon is called concept drift, and is an issue of great importance in the literature, since it makes models obsolete by decreasing their predictive performance. In the presence of concept drift, it is necessary to adapt to change in data to build more robust and effective classifiers. Drift detectors are designed to run jointly with classification models, updating them when a significant change in data distribution is observed. In this paper, we present an implicit (unsupervised) algorithm called One-Class Drift Detector (OCDD), which uses a one-class learner with a sliding window to detect concept drift. We perform a comprehensive evaluation on mostly recent 17 prevalent concept drift detection methods and an adaptive classifier using 13 datasets. The results show that OCDD outperforms the other methods by producing models with better predictive performance on both real-world and synthetic datasets.

Keywords Concept drift · Data stream · Drift detection · Unlabeled data · Verification latency

This study is partially supported by Scientific and Technological Research Council of Turkey (TÜBİTAK) Grant No. 117E870.

✉ Fazli Can
canf@cs.bilkent.edu.tr

Ömer Gözüaçık
omer.gozuacik@bilkent.edu.tr

[1] Information Retrieval Group, Computer Engineering Department, Bilkent University, 06800 Ankara, Turkey

1 Introduction

Analyzing streaming data has become an important challenge in data mining as the amount of data being produced has increased over recent years. It is estimated that data produced is in the order of zetta-bytes, and it is growing at around 40% each year (Fan and Bifet 2013). Data streams are referred to as data arriving continuously with a large amount of samples. They are potential sources of valuable information, provided they can be analyzed at the right time (Wares et al. 2019). Data needs to be processed as it arrives, or it is lost due to the limitations of the streaming environment. Beforehand, data streams were generally studied in financial markets (Krawczyk and Woźniak 2015). However, they are now everywhere due to recent developments in personalized technologies (e.g, IoT), turning each individual a data-source (Pariser 2011).

There are various analytical approaches developed for solving problems in machine learning, one of them being classification following the idea that data can be generalized (Duda et al. 2012). A predictive function is modeled, mapping features to labels using training data later to be evaluated on test data. The main assumption for generalization is that data is stationary, where training and testing sets share the same characteristics. However, this assumption is not valid for most real-world streaming environments, since data shows change with time. This is known as concept drift (Gama et al. 2014). In such environments, the classification model becomes inconsistent due to changes in data distribution. The predictive function fails to generalize data properly, resulting in a decrease in prediction accuracy. Therefore, algorithms that are capable of dealing with the change under the restrictions of the streaming environments are needed.

The ubiquity of data stream applications has made the concept drift problem a hot topic. A "concept drift" Google Scholar exact match search on February 24 2020, 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 (Sethi and Kantardzic 2017). 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 (Masud et al. 2011). Žliobaite (2010) 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 drift detection is limited (Lu et al. 2018).

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 perpetually available (Lughofer et al. 2016). Only a limited number of them are accessible, or they arrive with delay in certain circumstances (Sethi and Kantardzic 2017). 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 (Sethi and Kantardzic 2017). This process requires a continuous workforce and funding, which may not be available. Furthermore, labeling will involve delay as they must be processed manually. These problems can be observed in many streaming environments. Therefore, streaming algorithms need to work with unlabeled or sparsely labeled data to be of any use in real-life scenarios (Žliobaite 2010).

Another motivational example for unsupervised concept drift detection is also available in our current research focus, which focuses on a multi-stream environment (Chandra et al. 2016). In such an 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 among 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 as labels are unavailable. A possible real-life application for this environment can be considered. 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 cards are issued. The target stream may be the transactions in a foreign country for customers from different cities (source data streams).

In this study, we propose 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. An approach similar to the proposed algorithm, D3, is used with a discriminative classifier for concept drift detection (Gözüaçık et al. 2019). The difference is that D3 monitors the separability of the old and new sample distributions, whereas our approach 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 also deal with non-linear change.

The main contributions of this paper are as follows. We:

- To the best of our knowledge, identify concept drift detection as the continuous form of the one-class classification process for the first time in the literature;
- 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;
- Present an effective and simple unsupervised concept drift detection algorithm that can be useful in environments when labels for new data items are not available or delayed, and make its implementation public for other researchers;
- Analyze the proposed algorithm on 13 datasets against mainly recent and most prevalent 17 concept drift detection methods, along with an adaptive classifier and an online classifier without any drift adaptation mechanism, and perform a comprehensive evaluation, showing that our method outperforms the other approaches in predictive performance;
- Identify the shortcomings of concept drift detection research based on our observations in our experiments.

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

2 Problem definition

Data stream classification is a supervised learning problem with restrictions on time and computational power. A data stream consists of data in temporal order, i.e. $\mathcal{D} = \{(X_0, y_0), (X_1, y_1), \dots (X_t, y_t), \dots\}$ where X_t represents input features, y_t classes at time t. Each data instance, X_t is first tested, then class labels, y_t, is revealed for evaluation. In this way, the model is always tested on instances that it has not seen.

The data generation process in streams is generally 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 (Tsymbal 2004). The cause of this change can depend on periodicity, change in habits, aging, etc. In such environments, concept drift is defined as the change of the joint distribution of the set of input variables, X, and the target variable, y, at times t_0 and t_1 (Gama et al. 2014).

$$p_{t_0}(X, y) \neq p_{t_1}(X, y) \tag{1}$$

Changes in data can be investigated as changes in the components of the relation (Gama et al. 2014). The equation can be expanded as:

$$p(X, y) = p(X)p(y|X) \tag{2}$$

Only changes that affect the prediction process require adaptation. Concept drift types are shown in Fig. 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 (Elwell and Polikar 2011).

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, true class labels may not be available immediately after classification. They can either be delayed or unavailable in some environments (Žliobaite 2010). 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 (Zhang et al. 2008).

Types of concept drift can be analyzed further by examining the rate of change when the drift occurs (Ditzler et al. 2015). Four patterns of concept drift are shown in Fig. 2. A concept drift may not be sudden and can last for some period of time. The term intermediate concept is introduced to describe the transformation from one concept to another (Lu et al. 2018). During the change from one concept to another, intermediate concepts may appear depending on the rate of change. Intermediate concepts can be

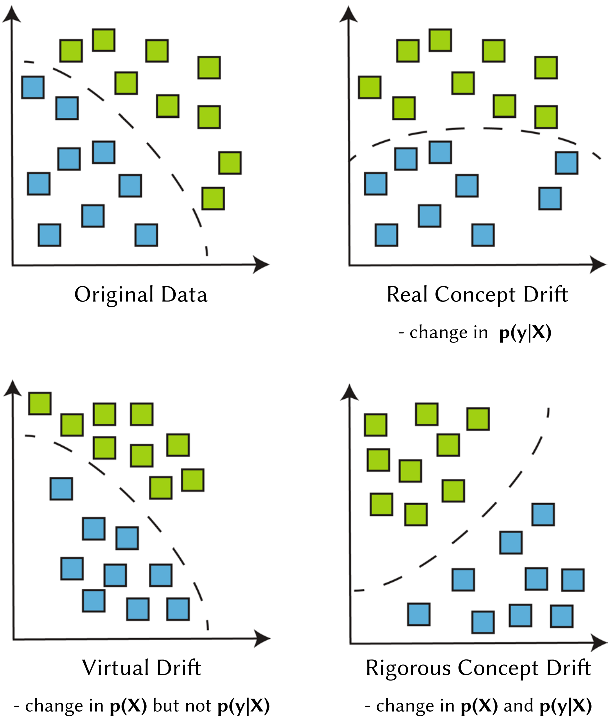


Fig. 1 Concept drift types: square objects are the instances; colors represent classes; the dashed line is the decision boundary. (Color figure online)

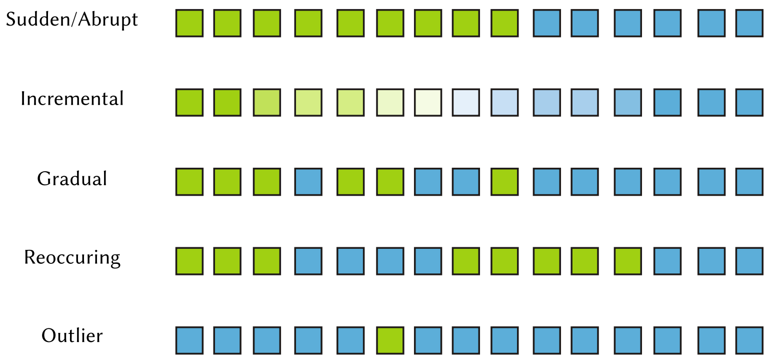


Fig. 2 Concept drift types with respect to rate of change patterns (excluding outlier which refers to noise in data). (Color figure online)

seen in incremental drift type in Fig. 2 as a mixture of two concepts: initial (green) and new (blue) concept. It should be noted that a data stream may have different patterns during different times.

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 (Faria et al. 2013). The model is updated accordingly, preventing it from being affected by the change, hence improving the predictive performance.

3 Related work

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

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 (Sethi and Kantardzic 2017; Hu et al. 2020).

3.1.1 Clustering-based methods

The methods in this group use distance or density measures to detect new concepts (Masud et al. 2011). OLINDDA (Spinosa et al. 2007) 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 (Faria et al. 2013) is an extension of it for multiclass problems. DETECTNOD (Hayat and Hashemi 2010) defines the boundaries of existing data by clustering. New samples that are outside 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 (Can 1993) identifies new cluster centroids and falsifies old ones as documents are being processed.

Woo (Ryu et al. 2012), ECSMiner (Masud et al. 2011), GC3 (Sethi et al. 2016) 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 (Haque et al. 2016) 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 a new region in the space, they are ineffective.

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 old data chunks, and is compared with 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 (Lee and Magoules 2012). PLOVER uses statistical moments and the power spectrum (de Mello et al. 2019).

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. Kuncheva and Faithfull (2014) state that monitoring the principal components with top eigenvalues is enough to detect drifts whereas Qahtan et al. (2015) 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.

3.1.3 Model-dependent methods

There are methods that implicitly track concept drift without assuming the changes in $P(X)$ will lead to 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 (Dries and Rückert 2009).

There are other methods that track the confidence of the classifiers by monitoring how well the classes are separated with the classifier (Dredze et al. 2010). They flag a drift depending on the changes of the classification margin among classes. They hold a reference margin and compare it with 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 (Lindstrom et al. 2013). 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.

3.1.4 Learner monitoring-based methods

The methods in this group track predictions of the learning model. MD3 (Sethi and Kantardzic 2017) 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 (Harel et al. 2014) 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 (Demšar and Bosnić 2018) 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.

D3 (Gözüaçık et al. 2019) monitors changes in the feature space using a discriminative classifier and signals a drift if new data is separable from the old. Song et al. (2007) 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 (Pinto et al. 2019) 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.

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 (Pesaranghader et al. 2018a). 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 (Page 1954) 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 (Gama et al. 2004), EDDM (Baena-García et al. 2006), RDDM (Barros et al. 2017) and EWMA (Ross et al. 2012) are representatives of these type of methods.

Window-based methods hold a reference of past results and compare them to the initial state. 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 (Bifet and Gavalda 2007); Seq2D (Pears et al. 2014; MDDM_A, MDDM_E, MDDM_G (Pesaranghader et al. 2018b); HDDM_A, HDDM_W (Frías-Blanco et al. 2014); FHDDM (Pesaranghader and Viktor 2016; FHDDMS, FHDDMS_A (Pesaranghader et al. 2018a) 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.

4 Proposed approach: OCDD

We propose OCDD (One-Class Drift Detector), 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. We define start as the time when the sliding window is full. 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.

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 (Faria et al. 2016). Data of only one class is available during training. In an earlier work, one-class classification is identified as concept learning (Tax et al. 2001). 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, (Fig. 3). 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.

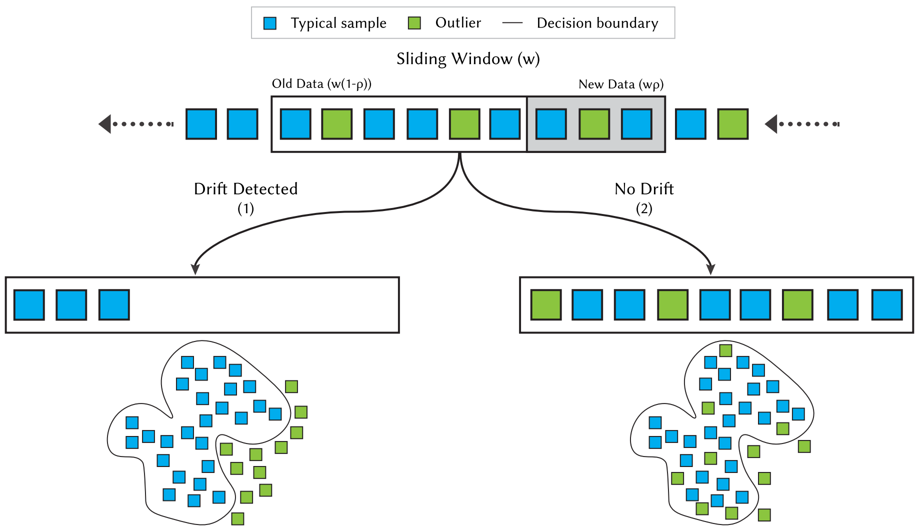


Fig. 3 Drift detection workflow: (1): Drift detected. The percentage of outliers exceed the threshold (ρ). There is a change in the distribution of the data. Samples from the old portion are discarded and are partially filled with samples from the new data window. (2): No drift. There is no change in the data distribution. The oldest sample is removed and the window is shifted to the left, filling the empty space

Algorithm 1 OCDD: One-Class Drift Detector

procedure OCDD($\mathcal{D}, w, \rho$):
  Initialize windows W, O where $|W| = |O| = w$
  One-class classifier C ▷*e.g: One-Class SVM*
  Train C ▷*with the available samples*
  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*
      $T = C(X)$ ▷*classify the new sample with C*
      $O \leftarrow O \cup T$ ▷$T = 1$ *for outliers and* 0 *otherwise*
    else
      $\alpha = \sum_{i=1}^{w} O[i]/w$ ▷*measure percentage of outliers*
      if $\alpha \geq \rho$ **then**
        drift = True ▷*drift detected*
        Drop $w(1-\rho)$ elements from the tail of W
        Retrain C ▷*with the samples available in W*
      else
        drift = False ▷*no drift*
        Drop 1 element from the tail of W

There are two possible results as illustrated in Fig. 3. (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.

5 Empirical evaluation

5.1 Datasets

We perform a comprehensive evaluation of our approach on 13 commonly used real-world and synthetic datasets. Their properties are presented in Table 1. The datasets are chosen from various application domains, containing a wide range and number of features and classes.

Table 1 Datasets we use for evaluation[a]

	Name (References)	#Features	#Classes	#Samples
Real	ELEC (Harries and Wales 1999)	6	2	45,312
	COVTYPE (Blackard et al. 1998)	54	7	581,012
	Poker Hand (Dua and Graff 2017)	10	10	829,201
	Outdoor (Losing et al. 2016)	21	40	4,000
	Rialto (Losing et al. 2016)	27	10	82,250
	Airlines (Expo 2009)	7	2	539,383
	Spam (Sethi and Kantardzic 2017)	499	2	6,213
	Phissing (Sethi and Kantardzic 2017)	46	2	11,055
Synthetic	Rotating Hyperplane (Losing et al. 2016)	10	2	200,000
	Moving squares (Losing et al. 2016)	2	4	200,000
	Moving RBF (Losing et al. 2016)	10	5	200,000
	Interchanging RBF (Losing et al. 2016)	2	15	200,000
	Mixed (Losing et al. 2016)	2	15	600,000

[a]Datasets are available on: https://github.com/ogozuacik/concept-drift-datasets-scikit-multiflow

5.2 Experimental setup

The experiments are implemented in Python using the libraries: Scikit-learn (Pedregosa et al. 2011), Scikit-multiflow (Montiel et al. 2018) and Tornado (Pesaranghader et al. 2018a). One-class SVM is used for one-class classification; however, any one-class method can also be used. Stream classification is done using a Hoeffding Tree (HT) (Domingos and Hulten 2000). Similarly, any online method that does not have a built-in concept drift adaptation mechanism can be employed. In a recent review, HT and Naïve Bayes were used to evaluate the performance of multiple drift detectors (Barros and Santos 2018). 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 and One-class SVM due to their recognition and effectiveness, as reported in the literature. They are operated with default parameters. If a drift is detected, the Hoeffding Tree 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 their efficiency results.

For evaluation, we use the Interleaved Test-Then-Train approach, which is utilized extensively in streaming environments (Gama et al. 2014). 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 OCDD against 17 drift detection methods, and an adaptive classifier, the Hoeffding Adaptive Tree (HAT). The methods are presented in Table 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 OCDD's performance with as many well-established methods as possible.

Table 2 Concept drift detection methods we use for evaluation (excluding HAT which is the drift adaptive version of the HT)

Method	References	Method	References
D3	Gözüaçık et al. (2019)	HAT	Bifet and Gavaldà (2009)
ADWIN	Bifet and Gavalda (2007)	HDDM_A	Frías-Blanco et al. (2014)
CUSUM	Page (1954)	HDDM_W	Frías-Blanco et al. (2014)
DDM	Gama et al. (2004)	RDDM	Barros et al. (2017)
EDDM	Baena-García et al. (2006)	MDDM_A	Pesaranghader et al. (2018b)
EWMA	Ross et al. (2012)	MDDM_E	Pesaranghader et al. (2018b)
FHDDM	Pesaranghader and Viktor (2016)	MDDM_G	Pesaranghader et al. (2018b)
FHDDMS	Pesaranghader et al. (2018a)	Page-Hinckley	Page (1954)
FHDDMS_A	Pesaranghader et al. (2018a)	Seq2D	Pears et al. (2014)

All methods are used with default parameters. Apart from D3, they are explicit methods. 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 publicly available[1].

6 Experimental results and analysis

6.1 Setting the parameters of OCDD

The overall accuracy of OCDD with different hyperparameter settings is presented in Table 3. 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.

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

[1] The source code is available on: https://github.com/ogozuacik/one-class-drift-detection.

Table 3 Overall accuracy of OCDD with multiple parameter settings for each dataset with the best scores highlighted in bold

Datasets	Parameters of OCDD (w, ρ)								
	100 0.1	100 0.3	100 0.5	250 0.1	250 0.3	250 0.5	1000 0.1	1000 0.3	1000 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

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 (Elwell and Polikar 2011; Bonab and Can 2018).

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 parameters starting at 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 Comparative evaluation and discussion

The overall accuracies for all methods are presented in Table 4. 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 true class labels. OCDD performs notably 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 predictive performance to drop. This can be observed in Fig. 4, 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 (false positives) 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 (false negatives) in which changes happen only on $P(y|X)$. Even with these weaknesses, our results confirm that OCDD performs well, outperforming other methods overall, achieving 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 into account the possible differences in the testing environments and 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 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.

The Self Parameter Tuning (SPT) algorithm is introduced to solve this issue by dynamically updating the model parameters while the stream is being processed (Veloso et al. 2018). 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 a new one, an abrupt drift may be observed. These type of changes frequently happen in real-life scenarios. 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. SPT and similar methods 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.

6.3 Statistical comparison of the methods

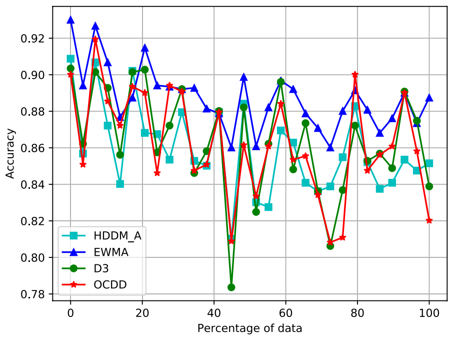
The *Friedman Test with Nemenyi post-hoc analysis* is applied to check the statistical significance of predictive performance differences. The test is applied with $\alpha = 0.05$. The null hypothesis is that there is no significant difference between the measurements. In other

Table 4 Overall accuracy and the average rankings of the methods for each stream with the best scores highlighted in bold

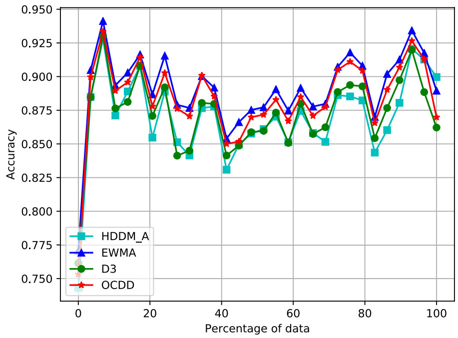
Datasets	Accuracy									
	OCDD	D3	ADWIN	CUSUM	DDM	EDDM	EWMA	FHDDM	FHDDMS	FHDDMS_A
ELEC	86.22	86.69	81.33	81.48	79.3	78.25	**88.76**	82.91	83.91	82.89
COVTYPE	88.36	87.17	80.48	82.67	83.36	82.76	**89.09**	83.78	84.65	83.78
Poker Hand	75.29	75.99	66.96	71.63	73.42	71.31	**76.83**	74.60	75.64	74.68
Outdoor	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
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

Table 4 (continued)

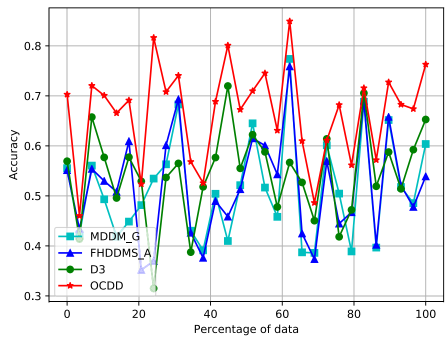
Datasets	HAT	HDDM_A	HDDM_W	RDDM	MDDM_A	MDDM_E	MDDM_G	Page-Hinckley	Seq2D	NONE
Moving RBF	36.36	49.78	**52.54**	50.99	51.26	51.28	51.29	35.46	46.95	33.92
Interchanging RBF	62.15	98.67	98.53	**98.79**	98.28	98.36	98.36	85.56	89.70	25.80
Mixed	**49.07**	48.48	48.73	48.15	48.24	48.04	48.05	42.28	45.92	44.97
Average rank	14.00	6.69	6.08	6.54	8.85	7.46	7.31	14.08	12.31	15.54



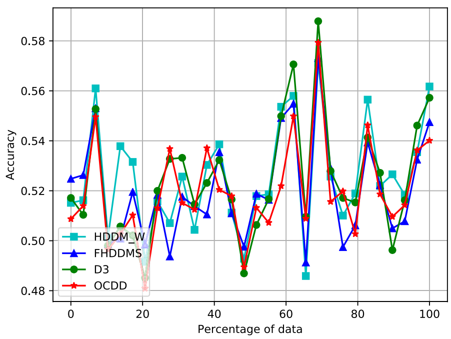
(a) ELEC



(b) COVTYPE



(c) Rialto



(d) Moving RBF

Fig. 4 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

words, the results share the same distribution. If it fails, the Nemenyi post-hoc analysis is applied in order to see which method is statistically significantly better. First, the methods are ranked on each dataset individually, then the average rank for each method is measured. Models that have better predictive performances have lower averaged ranks. The critical distance for Nemenyi Significance is calculated according to the values specific to our setting, using the *Critical Values Table for the Two Tailed Nemenyi Test* (Demšar 2006). We calculate $CD = 8.22$ (see Fig. 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.

6.4 Lessons learned: shortcomings in the literature of concept drift detectors

While researching 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) (Vapnik 1999) provides a theoretical framework for supervised machine learning algorithms, ensuring that the constructed models

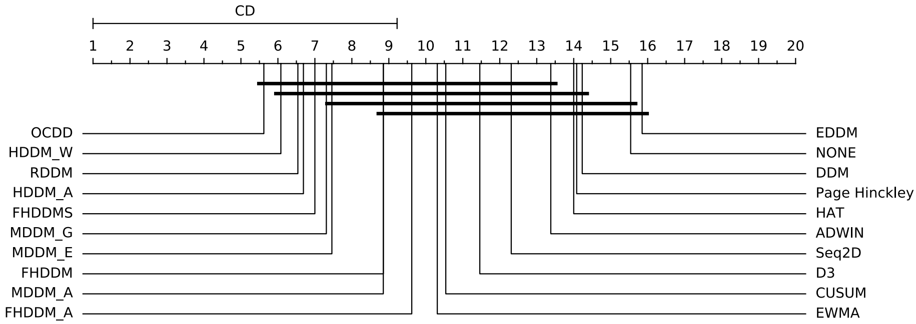


Fig. 5 Critical distance diagram for the overall accuracy using the data provided on Table 4. (CD = 8.22)

generalize on the train data. SLT can only be applied if labels are present, and therefore cannot be used in unsupervised learning. Empirical Risk Minimization is used to give theoretical bounds on the model performance, guaranteeing generalization (Bousquet et al. 2003).

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 (Rendón et al. 2011). 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 unsupervised task, the definition of structure and well can be different. In clustering, the compactness of clusters and the distance between them 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 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 (de Mello et al. 2019).

In unsupervised concept drift detection, most of the algorithms do not provide theoretical learning guarantees (de Mello et al. 2019). The methods are evaluated indirectly, according to the accuracy of the predictive model, without relying on any theoretical foundation to assure their performance. The concept of Algorithmic Stability (Bousquet and Elisseeff 2002) is suggested to solve this problem by checking the probabilistic convergence of a function and its expected value (de Mello et al. 2019). By this way, the 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 (Bifet 2017). The predictive accuracy of this setting is considered as a good measure of the quality of the drift detector. However, Bifet (2017) claims this approach is risky as there may be temporal dependencies in the data. 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, the 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 (Zliobaite 2013). 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 and FHDDMS do not perform statistically significantly different compared to NONE, i.e., using no concept drift detector (see Fig. 5).

Bifet (2017) proposes other evaluation techniques, but they are not applicable for most datasets, as they require the true location of the drifts. 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 (Barros and Santos 2018).

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 difficult for researchers to compare performances.

Computing today mainly focuses on efficiency which is defined as the optimal adaptation to an existing environment (Vardi 2020). 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 manipulation, such as "Google bombing" (Bar-Ilan 2007); for this reason, the research field of search-engine optimization (SEO) has developed protections for inorganic behavior.

In a data stream classification environment, which is well-balanced in terms of the efficiency vs. resiliency trade-off, a classifier avoids unnecessary computations and remains satisfactory under various types of concept drift. The goal of concept drift detection is to make streaming algorithms (classification, clustering, etc.) more resilient to changes in the environment. However, due to unexpected concept drifts, the resilience of data stream 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 vs. resilience trade-off may be a promising research area. In this regard, our work: "less is more" tries to achieve a sustainable (more resilient) performance with a lesser number of ensemble components (more efficiently) (Bonab and Can 2019).

7 Conclusion

In this paper, we introduce OCDD, an implicit algorithm for concept drift detection. 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 application 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 methods. We make our implementation publicly available along with the datasets used in our experimental analysis.

Future research possibilities include studying incremental one-class classification and adapting currently available methods to OCDD (Krawczyk and Woźniak 2015). During the experiments, we train a 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 (Liu et al. 2008) or Local Outlier Factor (Kriegel et al. 2009). We use the default parameters for OCDD, with which it performs 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 OCDD according to datasets characteristics can significantly boost prediction accuracy. Ensemble use of different combinations of supervised and unsupervised concept drift detection methods is another research possibility.

Acknowledgements We would like to thank two anonymous referees and Alper Can for their valuable comments and pointers on this article.

Code availability The source code is available on: https://github.com/ogozuacik/one-class-drift-detection.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Availability of data and material Datasets are available on: https://github.com/ogozuacik/concept-drift-datasets-scikit-multiflow.

References

Baena-García M, del Campo-Ávila J, Fidalgo R, Bifet A, Gavalda R, Morales-Bueno R (2006) Early drift detection method. Fourth Int Workshop Knowl Discov Data Streams 6:77–86

Bar-Ilan J (2007) Google bombing from a time perspective. J Comput Mediat Commun 12(3):910–938

Barros RS, Cabral DR, Gonçalves PM Jr, Santos SG (2017) RDDM: reactive drift detection method. Expert Syst Appl 90:344–355

Barros RSM, Santos SGTC (2018) A large-scale comparison of concept drift detectors. Inf Sci 451:348–370

Bifet A (2017) Classifier concept drift detection and the illusion of progress. In: International conference on artificial intelligence and soft computing. Springer, pp 715–725

Bifet A, Gavalda R (2007) Learning from time-changing data with adaptive windowing. In: Proc of the 2007 SIAM SDM, SIAM, pp 443–448

Bifet A, Gavaldà R (2009) Adaptive learning from evolving data streams. In: International symposium on intelligent data analysis. Springer, pp 249–260

Blackard JA, Dean DJ, Anderson CW (1998) The forest covertype dataset. UCI Machine Learning Repository

Bonab H, Can F (2019) Less is more: a comprehensive framework for the number of components of ensemble classifiers. IEEE Trans Neural Netw Learn Syst 30(9):2735–2745

Bonab HR, Can F (2018) GOOWE: geometrically optimum and online-weighted ensemble classifier for evolving data streams. ACM Trans Knowl Discov Data TKDD 12(2):1–33

Bousquet O, Elisseeff A (2002) Stability and generalization. J Mach Learn Res 2(Mar):499–526

Bousquet O, Boucheron S, Lugosi G (2003) Introduction to statistical learning theory. In: Summer school on machine learning. Springer, pp 169–207

Can F (1993) Incremental clustering for dynamic information processing. ACM Trans Inform Syst TOIS 11(2):143–164

Chandra S, Haque A, Khan L, Aggarwal C (2016) An adaptive framework for multistream classification. In: Proceedings of the 25th ACM international conference on information and knowledge management. ACM, pp 1181–1190
Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7(Jan):1–30
Demšar J, Bosnić Z (2018) Detecting concept drift in data streams using model explanation. Expert Syst Appl 92:546–559
de Mello RF, Vaz Y, Grossi CH, Bifet A (2019) On learning guarantees to unsupervised concept drift detection on data streams. Expert Syst Appl 117:90–102
Ditzler G, Roveri M, Alippi C, Polikar R (2015) Learning in nonstationary environments: a survey. IEEE Comput Intell Mag 10(4):12–25
Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining, pp 71–80
Dredze M, Oates T, Piatko C (2010) We're not in Kansas anymore: Detecting domain changes in streams. In: Proceedings of the 2010 conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 585–595
Dries A, Rückert U (2009) Adaptive concept drift detection. Stat Anal Data Min ASA Data Sci J 2(5–6):311–327
Dua D, Graff C (2017) The Pokerhand dataset. UCI Machine Learning Repository
Duda RO, Hart PE, Stork DG (2012) Pattern classification. Wiley, Hoboken
Elwell R, Polikar R (2011) Incremental learning of concept drift in nonstationary environments. IEEE Trans Neural Netw 22(10):1517–1531
Expo AD (2009) Airline on-time performance, ASA section on: statistical computing statistical graphics. http://stat-computing.org/dataexpo/2009
Fan W, Bifet A (2013) Mining big data: current status, and forecast to the future. ACM SIGKDD Explor Newslett 14(2):1–5
Faria ER, Gama J, Carvalho AC (2013) Novelty detection algorithm for data streams multi-class problems. In: Proc of the 28th annual ACM symposium on applied computing. ACM, pp 795–800
Faria ER, Gonçalves IJCR, de Carvalho ACPLF, Gama J (2016) Novelty detection in data streams. Artif Intell Rev 45(2):235–269
Frías-Blanco I, del Campo-Ávila J, Ramos-Jimenez G, Morales-Bueno R, Ortiz-Díaz A, Caballero-Mota Y (2014) Online and non-parametric drift detection methods based on hoeffding's bounds. IEEE Trans Knowl Data Eng 27(3):810–823
Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. In: Brazilian symposium on artificial intelligence. Springer, pp 286–295
Gama J, Zliobaite I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Comput Surv 46(4):44:1–44:37
Gözüaçık Ö, Büyükçakır A, Bonab H, Can F (2019) Unsupervised concept drift detection with a discriminative classifier. In: Proceedings of the 28th ACM international conference on information and knowledge management. ACM, pp 2365–2368
Haque A, Khan L, Baron M (2016) Sand: semi-supervised adaptive novel class detection and classification over data stream. In: 30th AAAI conference on artificial intelligence
Harel M, Mannor S, El-Yaniv R, Crammer K (2014) Concept drift detection through resampling. In: International conference on machine learning, pp 1009–1017
Harries M, Wales NS (1999) Splice-2 comparative evaluation: electricity pricing
Hayat MZ, Hashemi MR (2010) A DCT based approach for detecting novelty and concept drift in data streams. In: 2010 international conference of soft computing and pattern recognition. IEEE, pp 373–378
Hu H, Kantardzic M, Sethi TS (2020) No free lunch theorem for concept drift detection in streaming data classification: a review. Wiley Interdiscip Rev Data Min Knowl Discov 10(2):1327–1351
Krawczyk B, Woźniak M (2015) One-class classifiers with incremental learning and forgetting for data streams with concept drift. Soft Comput 19(12):3387–3400
Kriegel HP, Kröger P, Schubert E, Zimek A (2009) Loop: local outlier probabilities. In: Proceedings of the 18th ACM conference on information and knowledge management, pp 1649–1652
Kuncheva LI, Faithfull WJ (2014) Pca feature extraction for change detection in multidimensional unlabeled data. IEEE Trans Neural Netw Learn Syst 25(1):69–80
Lee J, Magoules F (2012) Detection of concept drift for learning from stream data. In: 2012 IEEE 14th HPCC & 2012 IEEE 9th ICESS, IEEE, pp 241–245
Lindstrom P, Mac Namee B, Delany SJ (2013) Drift detection using uncertainty distribution divergence. Evol Syst 4(1):13–25

Liu FT, Ting KM, Zhou ZH (2008) Isolation forest. In: 2008 Eighth IEEE international conference on data mining. IEEE, pp 413–422

Losing V, Hammer B, Wersing H (2016) KNN classifier with self adjusting memory for heterogeneous concept drift. In: 2016 IEEE 16th ICDM. IEEE, pp 291–300

Lu J, Liu A, Dong F, Gu F, Gama J, Zhang G (2018) Learning under concept drift: a review. IEEE Trans Knowl Data Eng 31(12):2346–2363

Lughofer E, Weigl E, Heidl W, Eitzinger C, Radauer T (2016) Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances. Inf Sci 355:127–151

Masud M, Gao J, Khan L, Han J, Thuraisingham BM (2011) Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Trans Knowl Data Eng 23(6):859–874

Montiel J, Read J, Bifet A, Abdessalem T (2018) Scikit-multiflow: a multi-output streaming framework. J Mach Learn Res 19(1):2914–2915

Page ES (1954) Continuous inspection schemes. Biometrika 41(1/2):100–115

Pariser E (2011) The filter bubble: what the internet is hiding from you. Penguin UK

Pears R, Sakthithasan S, Koh YS (2014) Detecting concept change in dynamic data streams. Mach Learn 97(3):259–293

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830

Pesaranghader A, Viktor HL (2016) Fast Hoeffding drift detection method for evolving data streams. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 96–111

Pesaranghader A, Viktor H, Paquet E (2018a) Reservoir of diverse adaptive learners and stacking fast Hoeffding drift detection methods for evolving data streams. Mach Learn 107(11):1711–1743

Pesaranghader A, Viktor HL, Paquet E (2018b) Mcdiarmid drift detection methods for evolving data streams. In: 2018 international joint conference on neural networks (IJCNN). IEEE, pp 1–9

Pinto F, Sampaio MO, Bizarro P (2019) Automatic model monitoring for data streams. arXiv preprint arXiv:190804240

Qahtan AA, Alharbi B, Wang S, Zhang X (2015) A PCA-based change detection framework for multidimensional data streams: change detection in multidimensional data streams. In: Proc of the 21th ACM SIGKDD. ACM, pp 935–944

Rendón E, Abundez I, Arizmendi A, Quiroz EM (2011) Internal versus external cluster validation indexes. Int J Comput Commun 5(1):27–34

Ross GJ, Adams NM, Tasoulis DK, Hand DJ (2012) Exponentially weighted moving average charts for detecting concept drift. Pattern Recogn Lett 33(2):191–198

Ryu JW, Kantardzic MM, Kim MW, Khil AR (2012) An efficient method of building an ensemble of classifiers in streaming data. In: International conference on big data analytics. Springer, pp 122–133

Sethi TS, Kantardzic M (2017) On the reliable detection of concept drift from streaming unlabeled data. Expert Syst Appl 82:77–99

Sethi TS, Kantardzic M, Hu H (2016) A grid density based framework for classifying streaming data in the presence of concept drift. J Intell Inform Syst 46(1):179–211

Song X, Wu M, Jermaine C, Ranka S (2007) 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

Spinosa EJ, de Leon F de Carvalho AP, Gama J (2007) OLINDDA: a cluster-based approach for detecting novelty and concept drift in data streams. In: Proc of the 2007 ACM symposium on applied computing. ACM, pp 448–452

Tax DMJ et al (2001) One-class classification, concept learning in the absence of counter example. Delft University of Technology

Tsymbal A (2004) The problem of concept drift: definitions and related work. Tech Rep Department of Computer Science, Trinity College, Dublin

Vapnik VN (1999) An overview of statistical learning theory. IEEE Trans Neural Netw 10(5):988–999

Vardi MY (2020) Efficiency vs. resilience: what COVID-19 teaches computing. Commun ACM 63(5):9

Veloso B, Gama J, Malheiro B (2018) Self hyper-parameter tuning for data streams. In: International conference on discovery science. Springer, pp 241–255

Wares S, Isaacs J, Elyan E (2019) Data stream mining: methods and challenges for handling concept drift. SN Appl Sci 1(11):1412

Zhang P, Zhu X, Shi Y (2008) Categorizing and mining concept drifting data streams. In: Proc of the 14th ACM SIGKDD. ACM, pp 812–820

Žliobaite I (2010) Change with delayed labeling: when is it detectable? In: 2010 IEEE international conference on data mining workshops. IEEE, pp 843–850

Zliobaite I (2013) How good is the electricity benchmark for evaluating concept drift adaptation. arXiv preprint arXiv:13013524

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.