Machine Learning
https://doi.org/10.1007/s10994-022-06177-w

Adversarial concept drift detection under poisoning attacks for robust data stream mining

Łukasz Korycki[1] · Bartosz Krawczyk[1]

Received: 19 September 2020 / Revised: 1 November 2021 / Accepted: 12 April 2022

Abstract

Continuous learning from streaming data is among the most challenging topics in the contemporary machine learning. In this domain, learning algorithms must not only be able to handle massive volume of rapidly arriving data, but also adapt themselves to potential emerging changes. The phenomenon of evolving nature of data streams is known as concept drift. While there is a plethora of methods designed for detecting its occurrence, all of them assume that the drift is connected with underlying changes in the source of data. However, one must consider the possibility of a malicious injection of false data that simulates a concept drift. This adversarial setting assumes a poisoning attack that may be conducted in order to damage the underlying classification system by forcing an adaptation to false data. Existing drift detectors are not capable of differentiating between real and adversarial concept drift. In this paper, we propose a framework for robust concept drift detection in the presence of adversarial and poisoning attacks. We introduce the taxonomy for two types of adversarial concept drifts, as well as a robust trainable drift detector. It is based on the augmented restricted Boltzmann machine with improved gradient computation and energy function. We also introduce Relative Loss of Robustness—a novel measure for evaluating the performance of concept drift detectors under poisoning attacks. Extensive computational experiments, conducted on both fully and sparsely labeled data streams, prove the high robustness and efficacy of the proposed drift detection framework in adversarial scenarios.

Keywords Data stream mining · Robust machine learning · Concept drift · Adversarial learning · Poisoning attacks · Boltzmann machine

Editors: Daniel Fremont, Mykel Kochenderfer, Alessio Lomuscio, Dragos Margineantu, Cheng Soon-Ong.

✉ Bartosz Krawczyk
bkrawczyk@vcu.edu

Łukasz Korycki
koryckil@vcu.edu

[1] Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA

Published online: 02 June 2022

1 Introduction

Modern machine learning algorithms should not assume that they will deal with finite, closed collections of data. Contemporary data sources generate new information constantly and at a high speed. This combination of velocity and volume gave birth to the notion of data streams that constantly expand and flood the computing system (Bifet et al., 2019). Data stream cannot be stored in memory and must be analyzed on the fly, without latency that may reduce the responsiveness and lead to a bottleneck. Such characteristics pose new challenges for machine learning algorithms. They need not only to display a high predictive accuracy, but also be capable of fast incorporation of new information, being computationally lightweight and responsive. The recent COVID-19 pandemic is an example of such a scenario, where models needed to be updated daily, whenever new information become available (Chatterjee et al., 2020). Due to the novelty of this disease and scarce access to ground truth, any new confirmed data was extremely valuable and updating the predictive model was of crucial importance. Efficient learning from data streams calls for such algorithms that can incorporate new data without a need for being retrained from scratch. Furthermore, algorithms dedicated to data stream mining problems must take into account the possibility of dealing with dynamic and non-stationary distributions (Ditzler et al., 2015). Properties of data may change over time, making previously trained model outdated and forcing constant adaptation to changes. The ever-changing nature of data streams is known as concept drift.

Concept drift can originate from the changes in underlying data generators (e.g., class distributions) or simply from having a lack of access to truly stable and well-defined collections of instances (Lu et al., 2019). The former is the most common case, where the problem under consideration is subject to non-stationary changes and shifts in its nature. Recommendation systems are excellent examples of such naturally occurring drifts, as the tastes of users may change over time. An example of more rapid changes would include a sensor network, where one of the sensors becomes suddenly damaged and the entire system needs to adapt to the new situation (Liu et al., 2017). The latter case is strongly connected with the problem of limited access to ground truth in data streams (Masud et al., 2011). As we deal with constantly arriving instances, it may be impossible to provide a label for every one of them. Therefore, this becomes a problem of managing budget (how many instances can we afford to analyze and label) and time (how quickly are we able to label selected instances) (Lughofer, 2017). Here, concept drift can be seen as a byproduct of the exploration-exploitation trade-off, where as we learn more about the underlying distributions, we need to update the previous models accordingly (Korycki and Krawczyk, 2017).

While there is a plethora of methods dedicated to explicit or implicit concept drift detection, they all assume that the occurrence of the drift originates purely in the underlying changes in data sources (Sethi and Kantardzic, 2018). But what would happen if we considered the potential presence of a malicious party, aiming at attacking our data stream mining system (Biggio and Roli, 2018)? Adversarial learning in the presence of malicious data and poisoning attacks recently gained a significant attention (Miller et al., 2020). However, this area of research focuses on dealing with corrupted training/testing sets (Biggio et al., 2012; Xiao et al., 2015) and handling potentially dangerous instances present there (Umer et al., 2019). Creating artificial adversarial instances (Gao et al., 2020) or using evasion (Mahloujifar et al., 2019) are considered as the most efficient approaches. The adversarial learning set-up has rarely been discussed in the data stream context, where adversarial instances may be injected into the stream at any point. One should consider the dangerous potential of introducing adversarial concept drift into the stream. Such a fake change may either lead to a premature and unnecessary adaptation to false changes, or slow down the adaptation to the real concept drift. Analyzing the presence of adversarial data and poisoning attack in the streaming setting is a challenging, yet important direction towards making modern machine learning systems truly robust.

Goal of the paper. To develop a concept drift detector capable of efficient change detection, while being robust to adversarial and poisoning attacks that inject fake concept drifts.

Summary of the content. In this paper, we propose a holistic analysis of the adversarial concept drift problem, together with a robust framework for handling poisoning attacks coming from data streams. We discuss the nature of adversarial concept drift and introduce a taxonomy of different types of possible attacks—based on poisoning instances or entire concepts. This allows us to understand the nature of the problem and gives a foundation for formulating a robust concept drift detector. We achieve this by introducing a novel and trainable drift detector based on Restricted Boltzmann Machine. It is capable of learning the compressed properties of the current state of stream and using a reconstruction error to detect the presence of concept drifts. In order to make it robust to adversarial and corrupted instances, we use a robust online gradient descent approach, together with a dedicated energy function used in our neural network. Finally, we introduce relative loss of robustness—a new measure for evaluating the robustness of concept drift detectors to varying levels of adversarial instances injected into the data stream.

Main contributions. This paper offers the following contributions to the field of learning from drifting data streams:

- *Taxonomy of adversarial concept drifts* we discuss the potential nature of poisoning attacks that may result in adversarial concept drift occurrence and their influence on drift detectors. We also formulate scenarios that can be used to evaluate the robustness of an algorithm dedicated to data stream mining.
- *Robust concept drift detector* we introduce a novel drift detector based on restricted Boltzmann machine that is augmented with robust online gradient procedure and dedicated energy function to alleviate the influence of poisoned instances on the detector.
- *Measure of robustness to adversarial drift* we propose relative loss of robustness, an aggregated measure used to analyze the effects of various levels of adversarial drifts injected into the stream.
- *Extensive experimental study:* we evaluate the robustness of the proposed and state-of-the-art drift detectors, using a carefully designed experimental test bed that involves fully and sparsely labeled data stream benchmarks.

2 Data stream mining

Data stream is defined as a sequence $< S_1, S_2, ..., S_n, ... >$, where each element S_j is a new instance. In this paper, we assume the (partially) supervised learning scenario with classification task and thus we define each instance as $S_j \sim p_j(x^1, \cdots, x^d, y) = p_j(\mathbf{x}, y)$, where $p_j(\mathbf{x}, y)$ is a joint distribution of the j-th instance, defined by a d-dimensional feature space and assigned to class y. Each instance is independent and drawn randomly from a probability distribution $p_j(\mathbf{x}, y)$.

2.1 Concept drift

When all instances arriving over time originate from the same distribution, we deal with a stationary stream that requires only incremental learning and no adaptation. However, in real-world applications data very rarely falls under stationary assumptions (Masegosa et al., 2020). It is more likely to evolve over time and form temporary concepts, being subject to concept drift (Lu et al., 2019). This phenomenon affects various aspects of a data stream and thus can be analyzed from multiple perspectives. One cannot simply claim that a stream is subject to the drift. It needs to be analyzed and understood in order to be handled adequately to specific changes that occur (Goldenberg and Webb, 2019, 2020). More precise approaches may help us achieving faster and more accurate adaptation (Shaker and Hüllermeier, 2015). Let us now discuss the major aspects of concept drift and its characteristics.

Influence on decision boundaries. Firstly, we need to take into account how concept drift impacts the learned decision boundaries, distinguishing between real and virtual concept drifts (Oliveira et al., 2019). The former influences previously learned decision rules or classification boundaries, decreasing their relevance for newly incoming instances. Real drift affects posterior probabilities $p_j(y|\mathbf{x})$ and additionally may impact unconditional probability density functions. It must be tackled as soon as it appears, since it impacts negatively the underlying classifier. Virtual concept drift affects only the distribution of features $\mathbf{x}$ over time:

$$\hat{p}_j(\mathbf{x}) = \sum_{y \in Y} p_j(\mathbf{x}, y), \tag{1}$$

where Y is a set of possible values taken by S_j. While it seems less dangerous than real concept drift, it cannot be ignored. Despite the fact that only the values of features change, it may trigger false alarms and thus force unnecessary and costly adaptations.

Locality of changes. It is important to distinguish between global and local concept drifts (Gama and Castillo, 2006). The former one affects the entire stream, while the latter one affects only certain parts of it (e.g., regions of the feature space, individual clusters of instances, or subsets of classes). Determining the locality of changes is of high importance, as rebuilding the entire classification model may not be necessary. Instead, one may update only certain parts of the model or sub-models, leading to a more efficient adaptation.

Speed of changes. Here we distinguish between sudden, gradual, and incremental concept drifts (Lu et al., 2019).

- *Sudden concept drift* is a case when instance distribution abruptly changes with t-th example arriving from the stream:

$$p_j(\mathbf{x}, y) = \begin{cases} D_0(\mathbf{x}, y), & \text{if } j < t \\ D_1(\mathbf{x}, y), & \text{if } j \geq t. \end{cases} \tag{2}$$

- *Incremental concept drift* is a case when we have a continuous progression from one concept to another (thus consisting of multiple intermediate concepts in between), such that the distance from the old concept is increasing, while the distance to the new concept is decreasing:

$$p_j(\mathbf{x}, y) = \begin{cases} D_0(\mathbf{x}, y), & \text{if } j < t_1 \\ (1 - \alpha_j)D_0(\mathbf{x}, y) + \alpha_j D_1(\mathbf{x}, y), & \text{if } t_1 \le j < t_2 \\ D_1(\mathbf{x}, y), & \text{if } t_2 \le j \end{cases} \tag{3}$$

where

$$\alpha_j = \frac{j - t_1}{t_2 - t_1}. \tag{4}$$

- *Gradual concept drift* is a case where instances arriving from the stream oscillate between two distributions during the duration of the drift, with the old concept appearing with decreasing frequency:

$$p_j(\mathbf{x}, y) = \begin{cases} D_0(\mathbf{x}, y), & \text{if } j < t_1 \\ D_0(\mathbf{x}, y), & \text{if } t_1 \le j < t_2 \wedge \delta > \alpha_j \\ D_1(\mathbf{x}, y), & \text{if } t_1 \le j < t_2 \wedge \delta \le \alpha_j \\ D_1(\mathbf{x}, y), & \text{if } t_2 \le j, \end{cases} \tag{5}$$

where $\delta \in [0, 1]$ is a random variable.

Recurrence. In many scenarios it is possible that a previously seen concept from k-th iteration may reappear $\mathrm{D}_{j+1} = \mathrm{D}_{j-k}$ over time (Sobolewski and Wozniak, 2017). One may store models specialized in previously seen concepts in order to speed up recovery rates after a known concept re-emerges (Guzy and Wozniak, 2020).

Presence of noise. Apart from concept drift, one may encounter other types of changes in data. They are connected with the potential appearance of incorrect information in the stream and known as blips or noise. The former stands for singular random changes in a stream that should be ignored and not mistaken for a concept drift. The latter stands for significant corruption in the feature values or class labels and must be filtered out in order to avoid feeding false (Krawczyk and Cano, 2018) or even adversarial information to the classifier (Sethi and Kantardzic, 2018).

Feature drift. This is a type of change that happens when a subset of features becomes, or stops to be, relevant to the learning task (Barddal et al. ,2017). This can be directly related to real concept drift, as such changes will affect decision boundaries. Additionally, new features may emerge (thus extending the feature space), while the old ones may cease to arrive.

2.2 Drift detectors

In order to be able to adapt to evolving data streams, classifiers must either have explicit information on when to update their model, or use continuous learning to follow the progression of a stream. Concept drift detectors are external tools that can be paired with any classifier and used to monitor a state of the stream (de Barros and de Carvalho Santos, 2018). Usually, this is based on tracking the error of the classifier (Pinage et al., 2020) or measuring the statistical properties of data (Korycki and Krawczyk, 2019). Drift detectors emit two-level signals. The warning signal means that changes start to appear in the stream and recent instances should be stored in a dedicated buffer. This buffer is used to train a new classifier in the background. The drift signal means that changes are significant enough to require adaptation and the old classifier must be replaced with the new one trained on the most recent buffer of data. This reduces the cost of adaptation by lowering the number of times when we train the new classifier, but may be subject to costly false alarms or missing changes appearing locally or on a smaller magnitude.

One of the first and most popular drift detectors is drift detection method (DDM) (Gama et al., 2004) that analyzes the standard deviation of errors coming from the underlying classifier. DDM assumes that the increase in error rates directly corresponds to changes in incoming data stream and thus can be used to signal the presence of drift. This concept was extended by Early Drift Detection Method (EDDM) (Baena-García et al., 2006) by replacing the standard error deviation with a distance between two consecutive errors. This makes EDDM more reactive to slower, gradual changes in the stream, at the cost of losing sensitivity to sudden drifts. Reactive Drift Detection Method (RDDM) (de Barros et al., 2017) is an improvement upon DDM that allows detecting sudden and local changes under access to a reduced number of instances. RDDM offers better sensitivity than DDM by implementing a pruning mechanism for discarding outdated instances. Adaptive Windowing (ADWIN) (Bifet and Gavaldà, 2007) is based on a dynamic sliding window that adjusts its size according to the size of the stable concepts in the stream. ADWIN stores two sub-windows for old and new concepts, detecting a drift when mean values in these sub-windows differ more than a given threshold. Statistical Test of Equal Proportions (STEPD) (Nishida and Yamauchi 2007) also keeps two sub-window, but uses a statistical test with continuity correction to determine if instances in both windows originate from similar or different distributions. Page-Hinkley test (PHT) (Sebastião and Fernandes, 2017) measures the current accuracy, as well as mean accuracy over a window of older instances. PHT computes cumulative and minimum differences between those two values and compares them to a predefined threshold, assuming that higher values of cumulative differences indicate increasing presence of concept drift. Exponentially Weighted moving average for concept drift detection (ECDD) (Ross et al., 2012) detects changes in the mean of sequences of instances realized as random variables, without the need for a prior knowledge about their mean and standard deviation. Sequential drift (SEQDRIFT) (Pears et al., 2014) can be seen as an improved version of ADWIN, where the sub-window of instances from the old concept is obtained by a reservoir sampling with a single-pass approach. Additionally, the comparison between two windows is done using the Bernstein bound, alleviating the need for a user-specified threshold. SEED Drift Detector (SEED) (Huang et al., 2014) is another example of ADWIN extension, where the Hoeffding's inequality with Bonferroni correction is used as the threshold bound for comparing two sub-windows. Furthermore, SEED compresses its sub-windows by eliminating redundant instances from homogeneous parts of its windows. Drift detection methods based on the Hoeffding's bounds (HDDM) (Blanco et al., 2015) uses the identical bound as SEED, but drops the idea of sub-windows and focuses on measuring both false positive and false negative rates. Fast hoeffding drift detection method (FHDDM) (Pesaranghader and Viktor, 2016) is yet another drift detector utilizing the popular Hoeffding's inequality, but its novelty lies in measuring the probability of correct decisions returned by the underlying classifier. Fisher Test Drift Detector (FTDD) (de Lima Cabral and de Barros, 2018) is an extension of STEPD that addressees the issue of sub-windows being of insufficient size or holding imbalanced distributions. Wilcoxon rank sum test drift detector (WSTD) (de Barros et al., 2018) is another extension of STEPD that uses the Wilcoxon rank-sum statistical test for comparing distributions in sub-windows. Diversity measure as drift detection method (DMDDM) (Mahdi et al., 2020) uses a combination of a pairwise diversity measure and PHT to offer an aggregated measure of difference between two concepts.

Recently, we can see the emergence of the ensemble learning paradigm applied to drift detectors (Krawczyk et al., 2017). Dynamic Classifier Selection with local accuracy and drift detector (DCS-LA+DDM) (Pinage et al., 2020) uses an ensemble of base classifiers together with a single drift detector. However, while based on ensemble idea, one cannot consider it a true ensemble of drift detectors. Drift Detection Ensemble (DDE) (Maciel et al. 2015) uses a combination of three independent drift detectors, which can be seen as a heterogeneous ensemble drift detector. Stacking Fast Hoeffding Drift Detection Method (FHDDMS) (Pesaranghader et al., 2018) uses a sequential combination of FHDDM detectors. Ensemble drift detection with feature subspaces (EDFS) (Korycki and Krawczyk, 2019) uses a combination of incremental Kolmogorov–Smirnov detectors (dos Reis et al. 2016) deployed on a set of diverse feature subspaces. Every time drift is detected, the subspaces are reconstructed, allowing EDFS to capture the new data characteristics. EDFS can be seen as a homogeneous ensemble drift detector. Monitoring changes in data partitions detected by *k*-means clustering to capture local data shifts was proposed in (Liu et al., 2021).

2.3 Classifiers for drifting data streams

Alternative approaches assume using classifiers that are capable of learning in an incremental or online manner. Sliding windows storing only the most recent instances are very popular, allowing for natural forgetting of older instances (Ramírez-Gallego et al., 2017; Roseberry et al., 2019). The size of the window is an important parameter and adapting it over time seems to yield the best results (Bifet and Gavaldà, 2007). Online learners are capable of learning instance by instance, discarding data after it passed the training procedure. They are very efficient on their own, but need to be equipped with a forgetting mechanism in order not to endlessly grow their complexity (Yu and Webb, 2019). Adaptive Hoeffding Trees (Bifet and Gavaldà, 2009) and gradient-based methods (Jothimurugesan et al., 2018) are among the most popular solutions.

2.4 Ensemble approaches

Combining multiple classifiers is a very popular and powerful approach for standard learning problems (Wozniak et al., 2014). The technique transferred seamlessly to data stream mining scenarios, where ensemble approaches have displayed a great efficacy (Krawczyk et al., 2017). They not only offer improved predictive power, robustness, and reduction of variance, but also can easily handle concept drift and use it as a natural way of maintaining diversity. By encapsulating new knowledge in the ensemble pool and removing outdated models, one can assure that the base classifiers are continuously mutually complementary, while adapting to changes in the stream. Popular solutions are based on the usage of online versions of bagging (Bifet et al., 2010b), boosting (Oza and Russell, 2001), random forest (Gomes et al., 2017), or instance-based clustering (Korycki and Krawczyk, 2018), as well as dedicated architectures such as Accuracy Updated Ensemble (Brzezinski and Stefanowski, 2014) or Kappa Updated Ensemble (Cano and Krawczyk, 2020).

3 Adversarial concept drift and poisoning attacks in streaming scenarios

In this section, we will discuss: the unique characteristic of adversarial learning scenario in the context of drifting data streams, the inadequacy of existing detectors for handling adversarial instances, why adversarial drift cannot be simply disregarded as a noise, how it impacts the learning from data streams, and what type of poisoning attacks we may expect.

3.1 Adversarial learning for static data

With the advent of deep learning and numerous success stories of its application in real-life problems, researchers started to notice that deep models can be easily affected by corrupted or noisy information (Elsayed et al., 2018). Multiple studies reported that even small perturbations in the training data may have devastating effects on the efficacy of the neural model (Su et al., 2019). While learning in the presence of noisy instances [either in a form of feature noise (Adeli et al., 2019) or label noise (Frénay and Verleysen, 2014)] has been studied thoroughly in machine learning, the specifics of deep learning gave a rise to novel challenges (Wang et al., 2020). As deep learning usually deals with complex representations of data, such as images (Dong et al., 2020) or text (Wallace et al., 2019), noise can be introduced in various new forms and affect multiple types of outputs generated by deep models: predictions, extracted features, embeddings, or generated instances (Choi et al., 2018). Adversarial learning is nowadays used in the broad context of preparing models to deal with noisy and corrupted data (Zhang et al.. 2016), ether by evasion (Li and Li. 2020) or poisoning attack (Bojchevski and Günnemann. 2019) training schemes. It is important to note that in the literature the term adversarial learning is used for both scenarios with a malicious party attacking the model (Madry et al.. 2018) and scenarios where corrupted information is the effect of environmental factors (Kaneko and Harada. 2020). As for the effect on data, it is either assumed that the training set is already corrupted (Cohen et al.. 2020), or that the corruption may appear during the prediction phase and thus the training set must be enriched in order to prepare the model (Xiao et al., 2018).

3.2 Unique characteristics of adversarial learning in data streams

Works in adversarial learning assume the static nature of data and the fact that the true nature of classes is known beforehand. Therefore, one may generate instances that differ from the training data in order to predict the nature of noisy or corrupted instances. In this work, we propose to extend the concept of adversarial learning into the data stream scenario with concept drift presence. This poses massive new challenges for the learning algorithms that cannot be tackled by the existing adversarial models. Let us now discuss the unique challenges that are present in the adversarial set-up of learning from data streams.

- *The true state of classes is subject to change over time* Data streams are subject to change over time. Therefore, what could be considered an adversarial case, may become a valid instance from a drifted distribution. This prevents us from simply enhancing the training set with artificial adversarial instances, as we cannot know a priori if any changes in the stream originate from the actual drift presence or from an adversarial attack.

- *Adversarial concept drift should be treated as malicious* In order to inject an adversarial concept drift into data, an attacker must be aware of the nature of data. Through this work, we will consider a scenario with a malicious party providing corrupted, poisoned data with a clear aim of damaging and harming our learning system. Similarities to insider attacks (Tuor et al., 2017) can be drawn, as in both cases poisoning data is prepared in such a way that eludes clear early detection as outliers and damages the learning system over time.
- *Need to differentiate between valid and adversarial concept drift* We must always assume that the analyzed stream may be subject to a valid, non-malicious concept drift. This poses a very interesting challenge–how can we differentiate between the changes that we want to follow and changes that may be of adversarial nature? Here, we should assume that concept drift will become more and more present as the stream progresses, while adversarial attacks may have a periodic nature and usually constitute only a small portion of the incoming instances. It is highly unlikely that we will have equal proportions of valid and adversarial instances in the stream.
- *Robustness to adversarial data cannot hinder the adaptation process* While designing robust machine learning algorithms for adversarial concept drift, we cannot follow the standard procedure that treats all data different from the training set as adversarial ones. A robust learner that treats all new data distributions as corrupted will not be able to adapt to new concepts. We need to design novel algorithms capable of differentiating between valid and adversarial drifts, which in turn will be used to make autonomous and on-the-fly decisions whether the learner should adapt to new data or not. Therefore, we find creating robust concept drift detectors particularly promising direction.

3.3 Limitations of existing drift detectors

State-of-the-art drift detectors, discussed in details in Sect. 2, share common principles. They all monitor some characteristics of newly arriving instances from the data stream and use various thresholds to decide if the new data are different enough to signal concept drift presence. It is easy to notice that the used measures and statistical tests are very sensitive to any perturbations in data and offer no robustness to poisoning attacks. Existing drift detectors concentrate on checking the level of difference between two distributions, without actually analyzing the content of the newly arriving instances. Furthermore, they are realized as simple thresholding modules, not being able to adapt themselves to data at hand. This calls for new drift detectors that have enhanced robustness and can learn properties of data, instead of just measuring some simple statistics.

3.4 Noisy data streams vs adversarial concept drift

It is important to offer a clear distinction between noisy data streams and adversarial concept drift. The former cases assume that either some features or labels have been incorrectly measured or provided by annotators. They do not have any underlying characteristics and usually come from additive random distributions or human errors. They may misguide the training procedure, but there is no malicious intent associated with them. Adversarial concept drift and associated poisoning attacks assume that there is a malicious party actively working against our machine learning system. The injected poisoned data was designed and crafted in such a way that will purposefully damage the classifier while avoiding easy detection. Adversarial drift must be seen as a coordinated attempt to disturb or completely shut down the system under attack. Negative impacts of poisoning attacks will be discussed next.

3.5 Negative impact of adversarial concept drift

In order to offer a complex and holistic view on the problem of adversarial concept drift, let us now discuss the major difficulties imposed by such a phenomenon on the underlying machine learning system:

- *Forcing false and unnecessary adaptation* Data stream mining algorithms, primarily detectors or adaptive classifiers, actively monitor the characteristics of the stream and trigger changing the learning model when newly arriving data differ from the previous concepts. Therefore, by injecting an adversarial drift during the stable period of a data stream (when no changes in data distributions are taking place), a false alarm will be raised. The learning algorithm will be cheated into thinking that a drift is taking place and will adapt to poisoned instances provided by the attacker. This can be used by a malicious party to damage the accuracy of the system at will, e.g., when two competitors are analyzing the same stream of data. Unnecessary adaptation not only will reduce the accuracy of the classifier, but also will consume precious computational resources and time, slowing the system down (Zliobaite et al., 2015) or even paralyzing it completely during the recovery after such an attack (Shaker and Hüllermeier, 2015).
- *Impairing the adaptation process* Adversarial concept drift may also be injected during the occurrence of a valid concept drift. If poisoning instances are well-crafted, they will offer information contrary to the actual changes in data. This will confuse any adaptive learning system, forcing it to adapt to two completely opposing concepts. If the algorithm deals with a high number of poisoned instances, this may lead to a complete nullification of the adaptation process. Even under the assumption that the adversarial concept is much smaller than the valid concept, it will still significantly slow down the adaptation process. The speed of adaptation to new data is of crucial importance and any tampering with it may result in massive financial losses or staying behind the competition.
- *Hindering label query process in partially labeled streams* In real-life scenarios, one does not have access to fully labeled streams. As it is impossible to obtain the ground truth for each newly arriving instance, active learning is being used to select the most useful instances for the label query (Lughofer, 2017). Obtaining labels is connected with budget management (i.e., the monetary cost for paying the annotator or domain expert) and time (i.e., how quickly the expert can label instance under the consideration). Adversarial concept drift will produce a number of instances that will pretend to be useful for the classifier (as they seem to originate from a novel concept). Even if the domain expert will correctly identify them as adversarial instances, the budget and time have already been spent. Therefore, adversarial concept drift is particularly dangerous for partially labeled streams, where it may force misuse of already scarce resources (Zliobaite et al., 2015).

3.6 Taxonomy of adversarial concept drift

We have analyzed and understood the unique nature of adversarial concept drift, the challenges connected with its differentiation from a valid concept drift, and the negative impacts it may have on the learning algorithms. Let us now propose the first taxonomy of poisoning attacks in the streaming scenario:

- *Adversarial concept drift with instance-based poisoning attacks* The first type assumes that the malicious party injects singular corrupted instances into the stream. They may be corrupted original instances with flipped labels or with modified feature values. Instance-based attacks are common when the attacker wants to test the robustness of the system and still needs to learn about the distribution of data in the stream. Additionally, supplying independent poisoned instances, especially when done from multiple sources, may be harder to detect that injecting a large number of instances at once. Such attacks, at lower rates, may be picked up by noise/outlier detection techniques. However, these attacks will appear more frequently than natural anomalies and will be crafted with malicious intent, requiring dedicated methods to filter them out. Instance-based poisoning attacks will not cause a false drift detection, but may significantly impair the adaptation to the actual concept drift. Figure 1 depicts the correct adaptation to a valid concept drift, while Fig. 2 depicts the same scenario with a hindered or exaggerated adaptation affected by the instance-based poisoning attacks. Figure 2 shows singular adversarial examples injected into the incoming data stream that may lead to one of the two possible poisoning situations: (i) slower adaptation; or (ii) overfitting. The hindrance of adaptation speed (see Fig. 2b) will take place when adversarial examples come from the previous concept and are purposefully injected to falsely inform the system that the old concept is still valid. This will negatively affect both drift detectors (that will underestimate the magnitude of change) and adaptive classifiers (that will impact the quality of decision boundary estimation). The second situation, overfitting (see Fig. 2c), can be caused by purposefully injecting adversarial instances to increase the presence of the noise and cause the streaming classifier to fit too closely to the fake state of the stream. This will reduce the generalization capabilities of the classifier, which in turn disables its adaptation capabilities to new, unseen data after the real drift.
- *Adversarial concept drift with concept-based poisoning attacks* The second type assumes that the malicious party have crafted poisoned instances that form a coherent concept. This can be seen as injecting an adversarial distribution of data that fulfills the cluster and smoothness assumptions. Therefore, now we must handle a difficult attack that will elude any outlier/noise/novelty detection methods. With the concept-based poisoning attack, we may assume that the malicious party poses a significant knowledge about the real data distributions and is able to craft such concepts that are going to directly cause false alarms and conflicts with valid concept drift. The effects of concept-based poisoning attacks are much more significant that its instance-based counterparts and may result, if undetected, in significant harm to the learning system and increased recovery times for rebuilding the model. Such attacks can both cause false drift detection and hinder, critically misguide or even completely nullify, the adaptation of the learning algorithm. Figure 1 depicts the correct adaptation to a valid concept drift, while Fig. 3 depicts the same scenario with an incorrect adaptation thwarted by the concept-based poisoning attacks. Figure 3 shows structured adversarial examples that form a (sub)concept injected into the incoming data stream that may lead to one of the two possible poisoning situations: (i) false adaptation; or (ii) lack of adaptation. The false adaptation (see Fig. 3b) will take place when data stream has been poisoned by a collection of adversarial instances forming a structured concept. In such a case, the underlying classifier will adapt to this false concept, treating it as an actual change in distributions. This is especially dangerous, as this adversarial concept may either com-

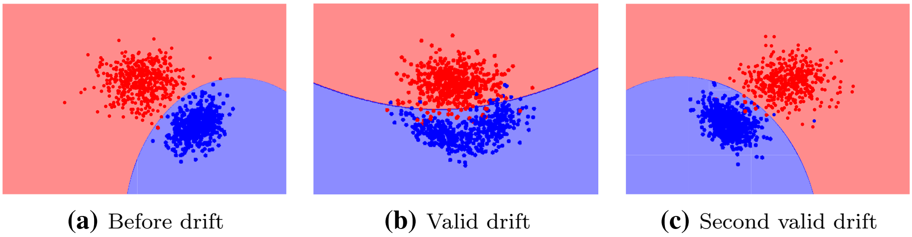

Fig. 1 Accurate adaptation to valid concept drift

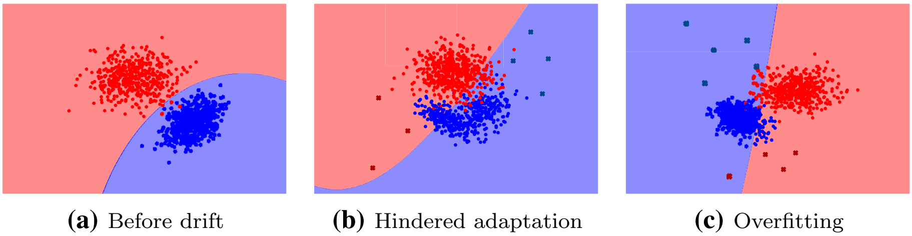

Fig. 2 Adversarial drift via instance-based poisoning attacks hinders (**b**) or exaggerates (**c**) the adaptation process

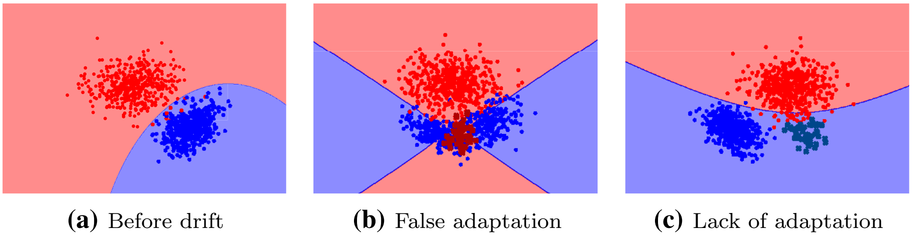

Fig. 3 Adversarial drift via concept-based poisoning attacks critically misguide (**b**) or completely nullifies (**c**) the adaptation process

pete with a real one (mistaking drift detectors as to which distribution the classifier should adapt), or force a false adaptation where no actual drift takes place (rendering the classifier useless and impairing the entire classification system). The second situation, lack of adaptation (see Fig. 3c), can be caused by purposefully injecting adversarial concepts that reinforce the old concept. This will mask the appearance of the real concept drift and inhibit any drift detection or adaptation mechanisms. This will prohibit the classifier from following the changes in data stream and adapting to novel information.

3.7 Differences between two types of poisoning attacks

After introducing the taxonomy of adversarial concept drift, let us shortly discuss the differences in impact of those two type of poisoning attacks. Instance-based attacks can be seen as having degenerative influence on the drift detection and underlying classifier. They will impact the speed or quality of adaptation, slowing down those processes and deteriorating the performance of stream mining. Concept-based attacks have a much more extreme impact on the rift detection and underlying classifier, either forcing a false adaptation or completely stopping any adaptation. Therefore, from a real-world standpoint they are much more dangerous and can pose a significant threat to any company that becomes a subject of such an attack. Therefore, proposing a drift detector displaying robustness to both instance-based and concept-based poisoning attacks is of high importance.

4 Robust restricted Boltzmann machine for adversarial drift detection

In this section, we describe in detail the proposed concept drift detector, realized as a trainable Restricted Boltzmann Machine with enhanced robustness to adversarial instances.

Overview of the proposed method We introduce a novel concept drift detector that is characterized by an increased robustness to adversarial concept drift, while maintaining high sensitivity to valid concept drift. It is realized as a Restricted Boltzmann Machine with leveraged robustness via improved online gradient calculation and extended energy function. It is a fully trainable drift detector, capable of autonomous adaptation to the current state of the stream and not relying on user-defined thresholds or statistical tests.

4.1 Restricted Boltzmann machine

4.1.1 Neural network architecture

Restricted Boltzmann machines (RBMs) are generative two-layered neural networks constructed using the $\mathbf{v}$ layer of V visible neurons and the $\mathbf{h}$ layer of H hidden neurons:

$$\begin{aligned}\mathbf{v} &= [v_1, \cdots, v_V] \in \{0, 1\}^V, \\ \mathbf{h} &= [h_1, \cdots, h_H] \in \{0, 1\}^H\end{aligned} \tag{6}$$

As we deal with the task of supervised learning from data streams (as defined in Sect. 2), we need to add a third (final) $\mathbf{z}$ layer for responsible for class representation. One can implement it as a 'one-hot' encoding, meaning that only a single neuron in $\mathbf{z}$ may activate at a given moment (i.e., have value set to 1, while every other one has value set to 0). By $\mathbf{1}_z$ we denote the vector of RBM outputs with 1 returned by z-th neuron and 0 returned by all other neurons. This allows to define $\mathbf{z}$, known also as the class layer or the softmax layer:

$$\mathbf{z} = [z_1, \cdots, z_Z] \in \{\mathbf{1}_1, \cdots, \mathbf{1}_Z\}. \tag{7}$$

This third layer uses the softmax function to estimate the probabilities of activation of each neuron in $\mathbf{z}$.

In RBM models no connection between units in the same layer (which holds for $\mathbf{v}$, $\mathbf{h}$, and $\mathbf{z}$) are assumed. However, there are connections between layers. Neurons in the visible layer $\mathbf{v}$ are connected with neurons in the hidden layer $\mathbf{h}$, and neurons in $\mathbf{h}$ are connected with those in the class layer $\mathbf{z}$. The weight assigned to a connection between the i-th visible neuron v_i and the j-th hidden neuron h_j is denoted as w_{ij}, while the weight assigned to a connection between the j-th hidden neuron h_j and the k-th class neuron z_k

is denoted as u_{jk}. With this RBM energy function is defined as (Salakhutdinov and Hinton, 2009):

$$E(\mathbf{v},\mathbf{h},\mathbf{z}) = -\sum_{i=1}^{V} v_i a_i - \sum_{j=1}^{H} h_j b_j - \sum_{k=1}^{Z} z_k c_k - \sum_{i=1}^{V}\sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{j=1}^{H}\sum_{k=1}^{Z} h_j z_k u_{jk}, \tag{8}$$

where a_i, b_j, and c_k stand for biases introduced to $\mathbf{v}, \mathbf{h}$, and $\mathbf{z}$ respectively. Energy formula $E(\cdot)$ for state $[\mathbf{v}, \mathbf{h}, \mathbf{z}]$ can be used to calculate the probability of RBM entering a given state with use of the Boltzmann distribution:

$$P(\mathbf{v},\mathbf{h},\mathbf{z}) = \frac{\exp\left(-E(\mathbf{v},\mathbf{h},\mathbf{z})\right)}{F}, \tag{9}$$

where F is a partition function normalizing the probability $P(\mathbf{v},\mathbf{h},\mathbf{z})$ to 1.

RBM assumes that its hidden neurons in $\mathbf{h}$ are independent and use variables (or features in the considered case of supervised learning) given by the visible layer $\mathbf{v}$. The activation probability of the j-th given neuron h_j can be calculated as follows:

$$\begin{aligned} P(h_j|\mathbf{v},\mathbf{z}) &= \frac{1}{1+\exp\left(-b_j - \sum_{i=1}^{V} v_i w_{ij} - \sum_{k=1}^{Z} z_k u_{jk}\right)} \\ &= \sigma\left(b_j + \sum_{i=1}^{V} v_i w_{ij} + \sum_{k=1}^{Z} z_k u_{jk}\right), \end{aligned} \tag{10}$$

where $\sigma(\cdot) = 1/(1+\exp(-\cdot))$ denotes a sigmoid function.

The same assumption holds for visible layer $\mathbf{v}$ neurons for known values of neurons in the hidden layer $\mathbf{h}$. With this the activation probability of the i-th visible neuron can be calculated as:

$$P(v_i|\mathbf{h}) = \frac{1}{1+\exp\left(-a_i - \sum_{j=1}^{H} h_j w_{ij}\right)} = \sigma\left(a_i + \sum_{j=1}^{H} h_j w_{ij}\right), \tag{11}$$

where given $\mathbf{h}$, the activation probability of neurons in $\mathbf{v}$ is not dependent on $\mathbf{z}$. The activation probability of class layer (i.e., decision which class the object should be assigned to) is calculated using the softmax function:

$$P(\mathbf{z}=\mathbf{1}_k|\mathbf{h}) = \frac{\exp\left(-c_k - \sum_{j=1}^{H} h_j u_{jk}\right)}{\sum_{l=1}^{Z} \exp\left(-c_l - \sum_{j=1}^{H} h_j u_{jl}\right)}, \tag{12}$$

where $k \in [1,\cdots,Z]$ and $k \neq l$.

4.1.2 RBM training procedure

As RBM is a neural network model, we train it using a minimization of a cost function $C(\cdot)$ using any gradient descent method. RBM usually apply the negative log-likelihood of both external layers $\mathbf{v}$ and $\mathbf{z}$:

$$C(\mathbf{v},\mathbf{z}) = -\log(P(\mathbf{v},\mathbf{z})). \tag{13}$$

Gradient of the cost function using each independent weight w_{ij} can be calculated as:

$$\nabla C(w_{ij}) = \frac{\delta C(\mathbf{v},\mathbf{z})}{\delta w_{ij}} = \sum_{\mathbf{v},\mathbf{h},\mathbf{z}} P(\mathbf{v},\mathbf{h},\mathbf{z}) v_i h_j - \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v},\mathbf{z}) v_i h_j. \tag{14}$$

This calculates the cost function gradient for a single instance. However, concept drift cannot be detected by analyzing each individual instance independently. If we would base our change detection on variations induced by a single new instance, we would be highly sensitive to even the smallest noise ratio. Therefore, to achieve stability of detection the properties of the stream should be analyzed over a batch of most recent instances. The proposed RBM model uses mini-batches of instances for its learning procedure. For a mini-batch of n instances arriving in t time $\mathbf{M}_t = x_1^t, \cdots, x_n^t$, we can rewrite the gradient from Eq. 14 using expected values:

$$\frac{\delta C(\mathbf{M}_t)}{\delta w_{ij}} = E_{\text{model}}[v_i h_j] - E_{\text{data}}[v_i h_j], \tag{15}$$

where E_{data} is the expected value over the current mini-batch of instances and E_{model} is the expected value from the current state of RBM. Of course, we cannot measure directly the value of E_{model} over time, therefore we must approximate it using Contrastive Divergence with k Gibbs sampling steps to reconstruct the input data (CD-k):

$$\frac{\delta C(\mathbf{M}_t)}{\delta w_{ij}} \approx E_{\text{recon}}[v_i h_j] - E_{\text{data}}[v_i h_j]. \tag{16}$$

After processing the t-th mini-batch $\mathbf{M}_t$, we update the RBM weights using any gradient descent method as follows:

$$w_{ij}^{t+1} = w_{ij}^{t} - \eta\big(E_{\text{recon}}[v_i h_j] - E_{\text{data}}[v_i h_j]\big), \tag{17}$$

where η is the learning rate. The way to update the a_i, b_j, and c_k biases, as well as weights u_{jk} is analogous to Eq. 17 and is defined as:

$$a_i^{t+1} = a_i^t - \eta\big(E_{\text{recon}}[v_i] - E_{\text{data}}[v_i]\big), \tag{18}$$

$$b_j^{t+1} = b_j^t - \eta\big(E_{\text{recon}}[h_j] - E_{\text{data}}[h_j]\big), \tag{19}$$

$$c_k^{t+1} = c_k^t - \eta\big(E_{\text{recon}}[z_k] - E_{\text{data}}[z_k]\big), \tag{20}$$

$$u_{jk}^{t+1} = u_{jk}^t - \eta\big(E_{\text{recon}}[h_j z_k] - E_{\text{data}}[h_j z_k]\big). \tag{21}$$

4.2 Drift detection with robust RBM

RBM, being a generative neural network model, can be used as an explicit drift detector (Jaworski et al., 2021). This is possible as RBM model stores compressed characteristics of the training data distributions. Using any similarity measure between the information stored in RBM and properties of newly arrived instances, one may evaluate if any changes in distribution took place. This allows us to use RBM as a drift detector, forming a base for our restricted Boltzmann machine for drift detection (RBM-DD). RBM-DD model uses similarity measure for monitoring the state of the stream and the level to which the newly arrived instances differ from the previous concepts. RBM-DD is a fully trainable drift detector, capable not only of capturing the trends in a single evaluation measure, but also of learning and adapting to the current state of the stream. This makes it a highly attractive approach for handling difficult and rapidly changing streams where non-trainable drift detectors fail.

4.2.1 Measuring data similarity

To evaluate the similarity between newly arrived instances and old concepts stored in RBM-DD, we will use the reconstruction error measure. We calculate it in an online fashion, by inputting a newly arrived d-dimensional instance $S_n = [x_1^n, \cdots, x_d^n, y^n]$ to the $\mathbf{v}$ layer of RBM. Then values of neurons in $\mathbf{v}$ are calculated to reconstruct the feature values. Finally, class layer $\mathbf{z}$ is activated and used to reconstruct the class label. We can denote the reconstructed vector as:

$$\tilde{S}_n = [\tilde{x}_1^n, \cdots, \tilde{x}_d^n, \tilde{y}_1^n, \cdots, \tilde{y}_Z^n], \tag{22}$$

where the reconstructed vector features and labels are given by probabilities calculated using the hidden layer:

$$\tilde{x}_i^n = P(v_i|h), \tag{23}$$

$$\tilde{y}_k^n = P(z_k|h). \tag{24}$$

The $\mathbf{h}$ layer is taken from the conditional probability, in which the $\mathbf{v}$ layer is identical to the input instance:

$$\mathbf{h} \sim P(\mathbf{h}|\mathbf{v} = x^n, \mathbf{z} = \mathbf{1}_{y_n}). \tag{25}$$

This allows us to formulate the reconstruction error as the mean squared error between the original and reconstructed instance:

$$R(S_n) = \sqrt{\sum_{i=1}^{d}(x_i^n - \tilde{x}_i^n)^2 + \sum_{k=1}^{Z}(\mathbf{1}_k^{y_n} - \tilde{y}_k^n)^2}. \tag{26}$$

For the purpose of a stable concept drift detector, we do not look for a change in distribution over a single instance. Therefore, we need to calculate the average reconstruction error over the recent mini-batch of data:

$$R(\mathbf{M}_t) = \frac{1}{n}\sum_{m=1}^{n} R(x_m^t). \tag{27}$$

4.2.2 Adapting reconstruction error to drift detection

To make the reconstruction error a practical measure for detecting the presence of concept drift, we monitor the evolution of this measure (i.e., its trends) over arriving mini-batches of instances. To achieve this, we use the well-known sliding window approach that will move over the arriving mini-batches. Let us denote the trend of reconstruction error over time as $Q_r(t)$ and calculate it using the following equation:

$$Q_r(t) = \frac{\bar{n}_t \bar{TR}_t - \bar{T}_t \bar{R}_t}{\bar{n}_t \bar{T}^2{}_t - (\bar{T}_t)^2}. \tag{28}$$

The trend over time can be computed using a standard linear regression, with the terms in Eq. 28 being simply sums over time as follows:

$$\bar{TR}_t = \bar{TR}_{t-1} + tR(\mathbf{M}_t), \tag{29}$$

$$\bar{T}_t = \bar{T}_{t-1} + t, \tag{30}$$

$$\bar{R}_t = \bar{R}_{t-1} + R(\mathbf{M}_t), \tag{31}$$

$$\bar{T}^2{}_t = \bar{T}^2{}_{t-1} + t^2, \tag{32}$$

where $\bar{TR}_0 = 0$, $\bar{T}_0 = 0$, $\bar{R}_0 = 0$, and $\bar{T}^2{}_0 = 0$. We capture those statistics using a sliding window of size W. We use a self-adaptive window size (Bifet and Gavaldà, 2007) to find the best size of the sliding window for each moment. To allow flexible learning from various sizes of mini-batches, we must consider a case where $t > W$. Here, we must compute the terms for the trend regression using the following equations:

$$\bar{TR}_t = \bar{TR}_{t-1} + tR(\mathbf{M}_t) - (t-w)R(\mathbf{M}_{t-w}), \tag{33}$$

$$\bar{T}_t = \bar{T}_{t-1} + t - (t-w), \tag{34}$$

$$\bar{R}_t = \bar{R}_{t-1} + R(\mathbf{M}_t) - R(\mathbf{M}_{t-w}), \tag{35}$$

$$\bar{T}^2{}_t = \bar{T}^2{}_{t-1} + t^2 - (t-w)^2. \tag{36}$$

The required number of instances $\bar{n}_t$ to compute the trend of $Q_r(t)$ as time t is given as follows:

$$\bar{n}_t = \begin{cases} t & \text{if } t \leq w \\ w & \text{if } t > w \end{cases} \tag{37}$$

4.2.3 Drift detection

Discussed Eq. 28is used to compute the trends for every analyzed mini-batch of data. To detect the presence of drift we require a capability of checking if the new mini-batch differs significantly from the previous one. Our RBM-DD achieves this by using Granger causality test (Sun, 2008) on trends from subsequent mini-batches of data $Q_r(\mathbf{M}_t)$ and $Q_r\mathbf{M}_{t+1})$. It is a statistical test that determines whether one trend is useful in forecasting another. As we deal with non-stationary processes we perform the variation of Granger causality test based on first differences (Mahjoub et al., 2020). Accepted hypothesis means that it is assumed that there exist Granger causality relationship between $Q_r(\mathbf{M}_t)$ and $Q_r\mathbf{M}_{t+1})$, which means there is no concept drift. If the hypothesis is rejected, RBM-DD signals the presence of concept drift.

4.3 Introducing robustness to RBM

The previous section introduced our RBM-DD – a drift detection method based on restricted Boltzmann machine. While this novel trainable drift detector may offer a significant drift detection capabilities and adaptation to the current state of the data stream, it offers no robustness to adversarial concept drift. In this section, we discuss how to introduce robustness into RBM-DD, leading to robust restricted Boltzmann machine for drift detection (RRBM-DD) that is capable of handling adversarial concept drift, while still displaying sensitiveness to the emergence of valid drifts.

4.3.1 Robust gradient descent

The first weak point of the RBM-DD in the adversarial scenario lies in its training phase. Adversarial instances may affect the update of our trainable drift detector and thus make it less robust to poisoning attacks over time. To avoid this, we propose to create RRBM-DD by replacing the original online cost gradient calculation in RBM-DD (see Eq. 14) used to update weights (see Eq. 17) by the robust gradient descent introduced in (Holland and Ikeda, 2019). It postulates to rescale the instance values by using a soft truncation of potentially adversarial instances. It achieves this by using a class of M-estimators of location and scale (Koltchinskii, 1997). It introduces a robust truncation factor $\hat{\theta}$ to control the influence of adversarial instances during the weight update:

$$w_{ij}^{t+1} = w_{ij}^{t} - \eta\left(E_{\text{recon}}[v_i h_j] - \hat{\theta}_i E_{\text{data}}[v_i h_j]\right), \tag{38}$$

where $\hat{\theta}_i$ stands for the truncation factor for the i-th neuron in $\mathbf{v}$ (and thus for the i-th input feature) that is calculated as follows:

$$\hat{\theta}_i \in \arg\min_{\theta\in\mathbb{R}} \sum_{a=1}^{n} \rho\left(\frac{L_i(y_a;\mathbf{z}) - \theta}{s_i}\right), \tag{39}$$

where ρ is a convex, even function and $L_i(y_a;\mathbf{z})$ is a loss function between true and predicted class labels (0–1 loss function is commonly used here). The authors of the robust gradient descent (Holland and Ikeda, 2019) postulate that for $\rho(\cdot) = \cdot^2$ the estimated truncation factor $\hat{\theta}_i$ is reduced to the sample mean of the loss function, thus alleviating the impact of extreme (in our case adversarial) instances. Therefore, they recommend to take $\rho(\cdot) = o(\cdot^2)$ for the function argument → ±∞.

The parameter s_i is a scaling factor used to ensure that consistent estimates take place irrespective of the order of magnitude of the observations. It is calculated as:

$$s_i = \hat{\sigma}_i \sqrt{n/\log(2\delta^{-1})}, \tag{40}$$

where $\delta \in (0, 1)$ is the confidence level and $\hat{\sigma}_i$ stands for an dispersion estimate of the instances in the mini-batch:

$$\hat{\sigma}_i \in \left\{ \sigma > 0 : \sum_{a=1}^{n} \chi\left(\frac{L_i(y_a;\mathbf{z}) - \gamma_i}{\sigma} \right) = 0 \right\}, \tag{41}$$

where $\chi : \mathbb{R} \rightarrow \mathbb{R}$ is an even function that satisfies $\chi(0) < 0$ and $\chi(\cdot) > 0$ for the function argument $\rightarrow \pm\infty$. This ensures that $\hat{\sigma}_i$ is an adequate measure of dispersion of the loss function about a pivot point $\gamma_i = \sum_{a=1}^{n} L_i(y_a;\mathbf{z})/n$ (Holland and Ikeda, 2019).

Analogously to Eq. 38, we use the robust truncation factor $\hat{\theta}$ to update remaining weights and parameters of RRBM-DD given by Eq. 18–21.

4.3.2 Robust energy function

Another weak spot of the RBM-DD lies in its energy function (see Eq. 8). It is a crucial part for calculating the probability of RBM-DD entering a given state and is used as the basis for the RBM-DD training procedure. In our case, as we use RBM as a trainable drift detector, energy function plays a crucial role in updating the detector with new instances and adapting it to the current state of the stream. RBM-DD energy function assumes working on clean (i.e., non-adversarial) data and thus can easily be corrupted by any poisoning attacks. Therefore, we propose to improve it for our RRBM-DD to alleviate its sensitivity to corrupted instances. We achieve this by adding a gating step to the visible layer $\mathbf{v}$, allowing for switching on and off neurons that may be strongly affected by poisoning attack (i.e., coming from adversarial sources). We add two new sets of variables: $\tilde{\mathbf{v}} = [\tilde{v}_1, \cdots, \tilde{v}_V]$ that stands for a Gaussian noise model for each neuron in the visible layer and $\mathbf{g} = [g_1, \cdots, g_V]$ that stands for a binary gating function for each neuron. This allows us to use gating to switch off neurons in $\mathbf{v}$ that have a chance probability of being affected by noisy instances. As we train our RRBM-DD over mini-batches of data, this approach will effectively switch off the RRBM-DD training procedure for every new instance that is denoted as adversarial by our underlying Gaussian noise model. The robust energy function for RRBM-DD is expressed as follows:

$$\begin{aligned} E_R(\mathbf{v}, \tilde{\mathbf{v}}, \mathbf{h}, \mathbf{z}, \mathbf{g}) =& \frac{1}{2} \sum_{i=1}^{V} g_i (v_i - \widetilde{v}_i)^2 - \sum_{i=1}^{V} v_i a_i - \sum_{j=1}^{H} h_j b_j - \sum_{k=1}^{Z} z_k c_k \\ & - \sum_{i=1}^{V} \sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{j=1}^{H} \sum_{k=1}^{Z} h_j z_k u_{jk} + \frac{1}{2} \sum_{i=1}^{V} \frac{(\widetilde{v}_i - \widetilde{b}_i)^2}{\tilde{\sigma}_i^2}, \end{aligned} \tag{42}$$

where the first (added) term stands for the gating interactions between v_i and $\tilde{v}_i$, the original energy function models the clean data, the last (added) term stands for the noise model, and $\tilde{b}_i$ and $\tilde{\sigma}_i^2$ stands for mean and variance of the poisoning attack. If RRBM-DD assumes that the i-th neuron in $\mathbf{v}$ is corrupted by the adversarial instance ($g_i = 0$) then $\tilde{v}_i \sim \mathcal{N}(\tilde{v}_i | \tilde{b}_i; \tilde{\sigma}_i^2)$.

4.4 Metric for evaluating robustness to poisoning attacks in streaming scenarios

Concept drift detectors are commonly evaluated by the prequential accuracy (Hidalgo et al., 2019) or prequential AUC/G-mean (Korycki et al., 2019) of the underlying classifier.

The reasoning behind it is that the better the detection of drift offered by the evaluated algorithm, the faster and more accurate classifier adaptation becomes. However, this setup is not sufficient for evaluating drift detectors in the adversarial concept drift setting. As we want to evaluate the robustness of a drift detector to poisoning attacks, we should attack the data stream with various levels of intensity to determine the breaking point of each algorithm. However, simple evaluation of a performance metric over varying levels of adversarial concept drift intensity does not give us information on how much our algorithm deteriorates over increasingly poisoned data.

We assume that we evaluate the robustness of drift detectors in a controlled experimental environment, where we can inject a desired level of adversarial concept drift into any benchmark data stream. Let us denote by $\mathbf{L} = [l_1, \cdots, l_a]$ the set of adversarial poisoning levels, ordered in the ascending order $l_1 < l_a$ from the smallest to the highest level of adversarial drift injection. Following the taxonomy introduced in Sect. 3, we can understand those levels as:

- for instance-based poisoning attacks—the percentage of instances in a mini-batch considered as adversarial in the data stream;
- for concept-based poisoning attacks—the number of adversarial concepts injected into the data stream.

Inspired by works in noisy data classification (Sáez et al., 2016), we introduce a Relative Loss of Robustness (RLR) that compares the performance of a given algorithm on a clean data stream against its performance on a data stream with l-th level of adversarial concept drift injected:

$$RLR_l = \frac{M_0 - M_l}{M_0}. \tag{43}$$

This is a popular approach used to evaluate the impact of noise on classifiers and can be directly adapted to the setting of adversarial drift detection. While it offers a convenient measure over a single level of poisoning, it becomes increasingly difficult to use when we want to evaluate the robustness of an algorithm over multiple poisoning levels. To alleviate this drawback, we propose an aggregated version of RLR measure over all considered adversarial poisoning levels:

$$RLR = \frac{\sum_{l=1}^{\#\mathbf{L}} \omega_l RLR_l}{\#\mathbf{L}}, \tag{44}$$

where ω_l is the importance weight associated to the performance on l-the level of adversarial information introduced to data and $\sum_{l=1}^{\#\mathbf{L}} \omega_l = 1$. The weighting part allows the end-user to adjust the measurements accordingly to the analyzed problem. If the user is interested only in the overall robustness of a given method, then all weights can be set to $1/\#L$. However, if it becomes crucial to select a drift detector that can perform the best even under the high intensity of attacks, the user may assign higher weights to higher levels of adversarial drift injection.

5 Experimental study

This experimental study was designed to evaluate the usefulness of the proposed RRBM–DD and answer the following research questions:

RQ1 Does RRBM-DD algorithm offers improved robustness to adversarial concept drift realized instance-based poisoning attacks?

RQ2 Does RRBM-DD algorithm offers improved robustness to adversarial concept drift realized concept-based poisoning attacks?

RQ3 Is RRBM-DD capable of preserving its robust characteristics when dealing with sparsely labeled data streams?

RQ4 What are the impacts of individual components of RRBM-DD on its robustness to adversarial concept drift?

5.1 Data stream benchmarks

For the purpose of evaluating the proposed RRBM-DD, we selected 12 benachmark data streams, six of which were generated artificially using MOA environment (Bifet et al., 2010a) and other six being real-world data streams coming from various domains, such as security, physics, and proteomics. Such a diverse mix allowed us to evaluate the effectiveness of RRBM-DD over a plethora of scenarios. Using artificial data streams allows us to control the specific nature of drift and where it occurs, while real-world streams offer challenging problems that are characterized by a mix of different learning difficulties. Properties of used data stream benchmarks are given in Table 1. All features in benchmark streams are scaled to [0,1].

5.2 Experimental setup

Here, we will present the details of the experimental study design.

Injection of adversarial concept drift As there are no real-world benchmarks for adversarial concept drift, we will inject poisoning attacks into our 12 benchmark datasets with the following procedures:

- *Instance-based poisoning attacks* are injected by corrupting a ratio $\mathbf{L}_{inst} \in \{0.05, 0.10, 0.15, 0.20, 0.25\}$ of randomly selected instances in the stream by flipping their class labels to a randomly chosen another class .
- *Concept-based poisoning attacks* are injected by generating a number $\mathbf{L}_{conc} \in \{10, 30, 50, 70, 100\}$ of small artificial concepts of $n = 250$ instances using corresponding MOA data generators.

Sparsely labeled data streams For the experiment investigating the robustness under limited access to ground truth, we investigate the 12 benchmark data streams with number of labeled instances $\in \{5\%, 10\%, 15\%, 20\%, 25\%, 30\%\}$.

Reference concept drift detectors As reference methods to the proposed RRBM-DD, we have selected five state of the art concept drift detectors: Early Drift Detection Method (EDDM) (Baena-García et al., 2006), Exponentially Weighted Moving Average for Concept Drift Detection (ECDD) (Ross et al., 2012), Fast Hoeffding Drift Detection

Table 1 Properties of artificial and real data stream benchmarks

Abbr.	Dataset	Instances	Features	Classes	Drift
Artificial data streams					
HYP_I	Hyperplane	1 000 000	10	2	Incremental
LED_S	LED	1 000 000	24	10	Sudden
RBF_G	RBF	1 000 000	40	20	Gradual
RBF_S	RBF	1 000 000	20	10	Sudden
SEA_G	SEA	3 000 000	3	4	Gradual
TRE_S	RandomTree	2 000 000	10	6	Sudden
Real data streams					
ecbdl14	Protein structure prediction	9 600 000	631	2	Mixed
higgs	High-energy physics classification	4 954 752	28	2	Unknown
IntelLab	Intel lab sensors	2 313 153	6	58	Mixed
iot	IoT botnet attacks	7 062 606	115	11	Mixed
kddcup	KDD intrusion detection	3 107 709	41	24	Mixed
susy	Supersymmetric particle detection	2 305 347	18	2	Unknown

Method (FHDDM) (Pesaranghader and Viktor, 2016), Reactive Drift Detection Method (RDDM) (de Barros et al., 2017),and Wilcoxon rank sum test drift detector (WSTD) (de Barros et al., 2018). Parameters of all six drift detectors are given in Table 2.

Parameter tuning In order to offer a fair and thorough comparison, we perform parameter tuning for every drift detector and for every data stream benchmark. As we deal with a streaming scenario, we use self hyper-parameter tuning (Veloso et al., 2018) that is based on online Nelder & Mead optimization.

Ablation study In order to be able to answer **RQ4** we perform an ablation study to check the impact of individual introduced components on the robustness of the drift detector. We compare RRBM–DD with its simplified versions with only robust online gradient RBM–DD_{RG}, only robust energy function RBM–DD_{RE}, and basic RBM–DD with no explicit robustness mechanisms.

Base classifier In order to ensure fairness in comparison among examined drift detectors they all use adaptive hoeffding decision tree (Bifet and Gavaldà, 2009) as a base classifier.

Evaluation metrics As we deal with drifting data streams, we evaluated examined algorithms using prequential accuracy (Hidalgo et al., 2019) and the proposed RLR metric (see Eq. 44). RLR assumes equal weights assigned to each level of adversarial drift injection.

Windows We used a window size $W = 1000$ for window-based drift detectors and calculating the prequential metrics .

Statistical analysis We used Friedman ranking test with Bonferroni-Dunn post-hoc for determining statistical significance over multiple comparison with significance level $\alpha = 0.05$.

5.3 Experiment 1: evaluating robustness to instance-based poisoning attacks

The first experiment was designed to evaluate the robustness of RRBM–DD and reference drift detectors to instance-based poisoning attacks. For this purpose, we injected five

Table 2 Examined drift detectors and their parameters

Abbr.	Name	Parameters
EDDM	Early drift detection	Warning threshold $\alpha_w \in \{0.90, 0.92, 0.95, 0.98\}$
		Drift threshold $\alpha_d \in \{0.80, 0.85, 0.90.0.95\}$
		Min. no. of errors $e \in \{10, 30, 50, 70\}$
ECDD	EWMA for drift detection	Differentiation weights $\lambda \in \{0.1, 0.2, 0.3, 0.4\}$
		Min. no. of errors $n = \{10, 30, 50, 70\}$
FHDDM	Fast hoeffding drift detection	Sliding window size $\omega \in \{25, 50, 75, 100\}$
		Allowed error $\delta \in \{0.000001, 0.00001, 0.0001, 0.001\}$
RDDM	Reactive drift detection	Warning threshold $\alpha_w \in \{0.90, 0.92, 0.95, 0.98\}$
		Drift threshold $\alpha_d \in \{0.80, 0.85, 0.90.0.95\}$
		Min. no. of errors $e \in \{10, 30, 50, 70\}$
		Min. no. of instances min $\in \{3000, 5000, 7000, 9000\}$
		Max. no. of instances max $\in \{10000, 20000, 30000, 40000\}$
		Warning limit $wL \in \{800, 1000, 1200, 1400\}$
WSTD	Wilcoxon rank sum test	Sliding window size $\omega \in \{25, 50, 75, 100\}$
	Drift detection	Warning significance $\alpha_w \in \{0.01, 0.03, 0.05, 0.07\}$
		Drift significance $\alpha_d \in \{0.001, 0.003, 0.005, 0.007\}$
		Max. no of old instances min $\in \{1000, 2000, 3000, 4000\}$
RRBM–DD	Robust RBM drift detection	Mini–batch size $\mathbf{M} \in \{25, 50, 75, 100\}$
		Visible neurons $\mathbf{V} =$ no. of features
		Hidden neurons $\mathbf{H} \in \{0.25\mathbf{V}, 0.5\mathbf{V}, 0.75\mathbf{V}, \mathbf{V}\}$
		Class neurons $\mathbf{Z} =$ no. of classes
		Learning rate $\eta \in \{0.01, 0.03, 0.05, 0.07\}$
		Gibbs sampling steps $k \in \{1, 2, 3, 4\}$
		Robust gradient confidence $\delta \in \{0.90, 0.92, 0.95, 0.98\}$

different ratios of poisoning attacks into 12 benchmark datasets. Figure 4 depicts the effects of varying levels of adversarial concept drift on the prequential accuracy of the underlying classifier, while Table 3 presents the RLR metric results. Additionally, Figs, 5 and 6 show the visualizations of Friedman ranking test with Bonferroni–Dunn post-hoc on both used metrics.

We can see how instance-based poisoning attacks impact both drift detectors and underlying classifier. For low attack ratios (0.05 and 0.10) we already can observe drop in performance, but all of examined methods can still deliver acceptable performance. With the increase of poisoning attack ratios all reference drift detectors start to fail. They cannot cope with such adversarial streams and are not able to select properly the moment for updating the Adaptive Hoeffding Tree classifier. In many cases their performance starts to approach random decisions, which is a strong indicator of the negative effects that instance-based poisoning attacks have on detectors. This is especially visible in case of EDDM and ECDD that are the weakest performing ones. RRBM–DD outperforms every single competitor, especially for RLR metric where it is the best algorithm over all 12 benchmarks. By analyzing Fig. 4 we can see that RRBM–DD offers significantly higher robustness to increasing levels of adversarial concept drift. For not a single dataset we can observe any sharp decline in performance, even when 25% of instances in the stream are

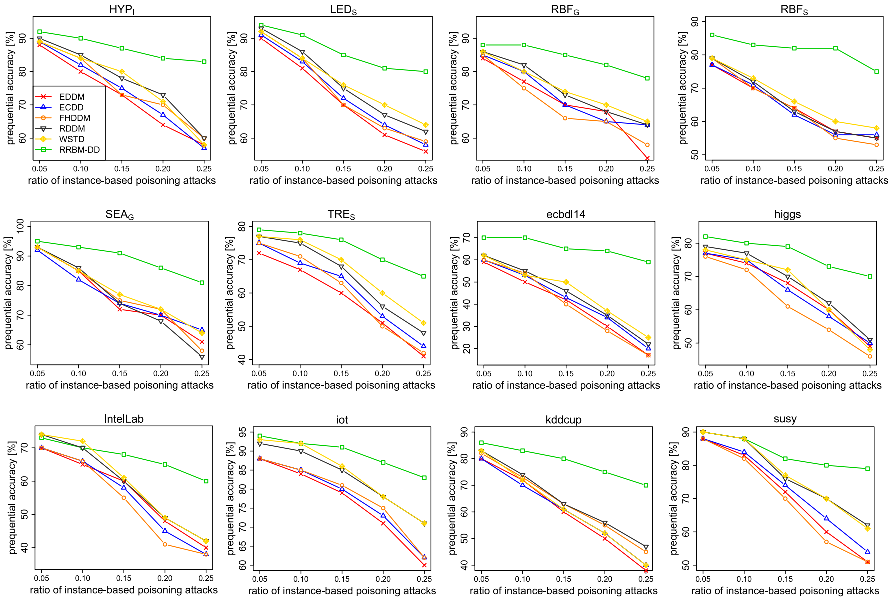

Fig. 4 Relationship between prequential accuracy and ratio of injected adversarial concept drift via instance-based poisoning attacks

corrupted by adversarial attacker. Such situation can be explained by the way standard drift detectors compute the presence of a valid drift. They use statistics, such as mean values or errors, derived directly from data. This makes them highly susceptible to any adversarial attack, as even small corruption of the data being used to compute drift statistics will result in either incorrectly increased sensitivity to any variations in stream, or inhibits sensitivity to valid concept drift. RRBM–DD avoids this pitfall by being a fully trainable drift detector that uses two robustness inducing mechanisms that filter data before our drift detector is updated. This allows RRBM–DD to compute more accurate reconstruction error from mini-batches and use it to correctly react only to valid concept drift.

RQ1 answer Yes, RRBM–DD offers excellent robustness to instance-based poisoning attacks and is not significantly affected by various levels of such adversarial concept drift.

5.4 Experiment 2: evaluating robustness to concept-based poisoning attacks

Experiment 2 was designed as a follow-up to Experiment 1. Here, we want to investigate a much more challenging scenario of concept-based poisoning attacks on data streams. For this purpose, we injected five different numbers of adversarial concepts into 12 benchmark datasets. Figure 7 depicts the effects of varying levels of adversarial concept drift on the prequential accuracy of the underlying classifier, while Table 4 presents the RLR metric results. Additionally, Figs. 8 and 9 show the visualizations of Friedman ranking test with Bonferroni–Dunn post-hoc on both used metrics.

Experiment 2 further confirms the observations made during analysis of experiment 1. State-of-the-art drift detectors offer no robustness to adversarial attacks and can be very easily fooled or damaged by even a very small number of poisoning attacks. What is very

Table 3 RLR for RRBM–DD and reference drift detectors under instance-based poisoning attacks

Dataset	EDDM	ECDD	FHDDM	RDDM	WSTD	RRBM–DD
HYP$_I$	0.55	0.58	0.67	0.64	0.63	0.85
LED$_S$	0.61	0.62	0.71	0.71	0.74	0.90
RBF$_G$	0.54	0.55	0.58	0.56	0.61	0.77
RBF$_S$	0.49	0.47	0.50	0.54	0.52	0.73
SEA$_G$	0.67	0.71	0.70	0.74	0.77	0.86
TRE$_S$	0.44	0.43	0.48	0.49	0.51	0.72
ecbdl14	0.40	0.45	0.52	0.56	0.53	0.69
higgs	0.71	0.73	0.77	0.81	0.82	0.94
IntelLab	0.41	0.44	0.46	0.48	0.49	0.69
iot	0.73	0.77	0.81	0.84	0.87	0.95
kddcup	0.63	0.60	0.67	0.65	0.69	0.83
susy	0.65	0.71	0.74	0.76	0.73	0.85
avg. rank	5.88	4.12	3.73	3.22	3.05	1.00

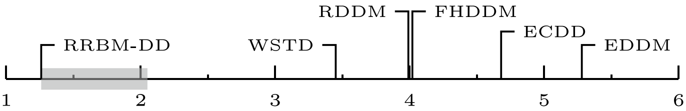

Fig. 5 The Bonferroni–Dunn test for comparison among drift detectors under instance-based poisoning attacks, based on prequential accuracy

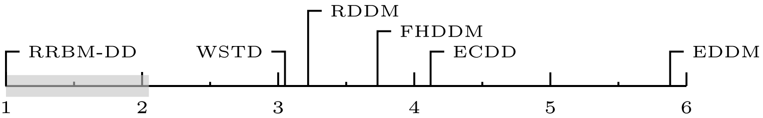

Fig. 6 The Bonferroni–Dunn test for comparison among drift detectors under instance-based poisoning attacks, based on RLR

important to notice is the difference in impact of concept-based poisoning attacks versus their instance-based counterparts. Here we can see catastrophic effects of injecting adversarial concepts into the stream, as every single reference drift detectors converges at a point where it behaves worse than a random guess. This can be interpreted as adversarial data hijacking the updating process of the underlying classifier and forcing it to adapt to the malicious information. RRBM–DD once again offers excellent robustness, while maintaining a very good valid drift detection - as evident from the high prequential accuracy of Adaptive Hoeffding Tree associated with it. Of course concept-based poisoning attacks have stronger effect on RRBM-DD, yet they are not capable of hindering its sensitivity and robustness. First two experiments highlighted the excellent properties of RRBM-DD when dealing with adversarial concept drift, but it would be very beneficial to understand what exactly is the cause of such a performance. We will analyze this using ablation study in experiment 4.

RQ2 answer Yes, RRBM-DD can efficiently handle injection of adversarial concepts, even when they fulfill smoothness and cluster assumptions. This proves that RRBM–DD is

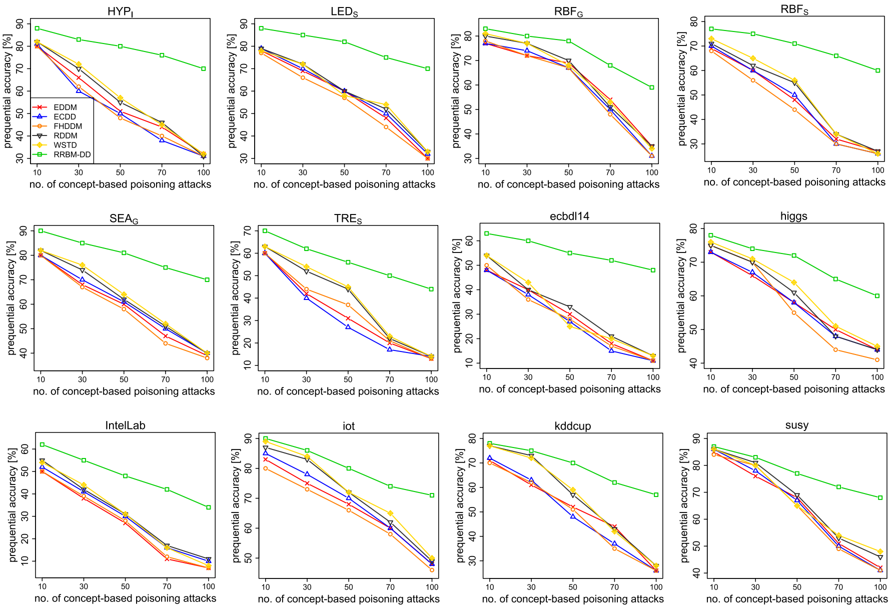

Fig. 7 Relationship between prequential accuracy and number of injected adversarial concept drift via concept-based poisoning attacks

not simply insensitive to noisy data, but can handle truly adversarial scenarios with malicious party using crafted poisoning attacks on data.

5.5 Experiment 3: evaluating robustness under class label sparsity

Third experiment was designed to investigate the behavior of RRDM–DD under sparse access to class labels. As discussed in Sects. 1 and 3 a common pitfall of many algorithms for learning from data streams lies in unrealistic assumption that there is an unlimited access to labeled instances. It is crucial to acknowledge that any algorithm for data stream mining, be it classifier or drift detector, must be flexible enough to work in scenarios where class labels are sparse. We used 12 benchmark data streams to select six different percentages of labeled instances. Label query procedure was repeated 10 times, in order to avoid impacting the results with selecting easy or difficult instances. This resulted in 60 runs for each out of 12 benchmark streams. Figures 10 and 11 present the win-tie-loss plots for comparing the performance of RRBM–DD on sparsely labeled streams under instance-based and concept-based poisoning attacks. Tables 5 and 6 present the RLR metrics for RRDM-DD and reference methods under both types of attacks, while Figs. 12 and 13 show the visualizations of Friedman ranking test with Bonferroni–Dunn post-hoc analysis.

Sparse access to class labels significantly impacts state-of-the-art drift detectors. All of them were designed for fully supervised cases and use both feature values and class labels to compute statistics used in drift detection. By limiting the number of labeled instances they obtain, the estimators of used statistics become less and less reliable. Detectors are forced to make decisions based on a small, potentially non-representative sample and thus are much more prone to errors. This is already a very difficult problem, but

Table 4 RLR for RRBM–DD and reference drift detectors under concept-based poisoning attacks

Dataset	EDDM	ECDD	FHDDM	RDDM	WSTD	RRBM–DD
HYP_I	0.44	0.45	0.50	0.52	0.51	0.78
LED_S	0.48	0.52	0.56	0.55	0.59	0.84
RBF_G	0.35	0.38	0.42	0.44	0.43	0.71
RBF_S	0.27	0.28	0.30	0.33	0.32	0.68
SEA_G	0.51	0.55	0.52	0.55	0.56	0.82
TRE_S	0.22	0.20	0.25	0.27	0.27	0.63
ecbdl14	0.18	0.17	0.23	0.26	0.26	0.60
higgs	0.52	0.56	0.57	0.61	0.63	0.88
IntelLab	0.24	0.26	0.25	0.30	0.32	0.65
iot	0.63	0.65	0.69	0.71	0.74	0.90
kddcup	0.39	0.35	0.44	0.40	0.47	0.76
susy	0.42	0.45	0.46	0.51	0.47	0.80
avg. rank	5.60	4.55	3.10	3.90	2.95	1.00

becomes even more challenging when combined with the presence of adversarial concept drift. RRBM–DD, due to its trainable nature is capable of much better performance even under limited access to class labels. As we track progress of stream using regression-based trends, we can still compute them efficiently from a smaller sample size. Furthermore, the implemented robust gradient can work efficiently with smaller datasets (Holland and Ikeda, 2019), making it suitable for stream processing over mini-batches.

RQ3 answer Yes, RRBM-DD is capable of efficiently handling sparsely labeled data streams, without sacrificing its accuracy and robustness to adversarial concept drift.

5.6 Experiment 4: ablation study

The fourth and final experiment was designed in a form of ablation study. Here, we want to switch off different components of RRBM–DD to gain understanding what is the source of its desirable performance and under what specific scenarios which component offers the most improvement to our drift detector. Tables 7 and 8 show the LRL metric performance averaged over fully and sparsely labeled data streams for four settings - the proposed RRBM–DD model, its basic version RBM–DD without any robustness

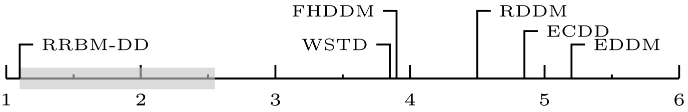

Fig. 8 The Bonferroni–Dunn test for comparison among drift detectors under concept-based poisoning attacks, based on prequential accuracy

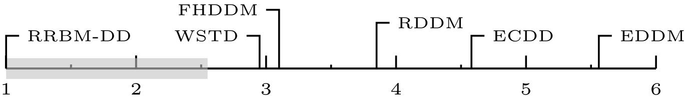

Fig. 9 The Bonferroni–Dunn test for comparison among drift detectors under concept-based poisoning attacks, based on RLR

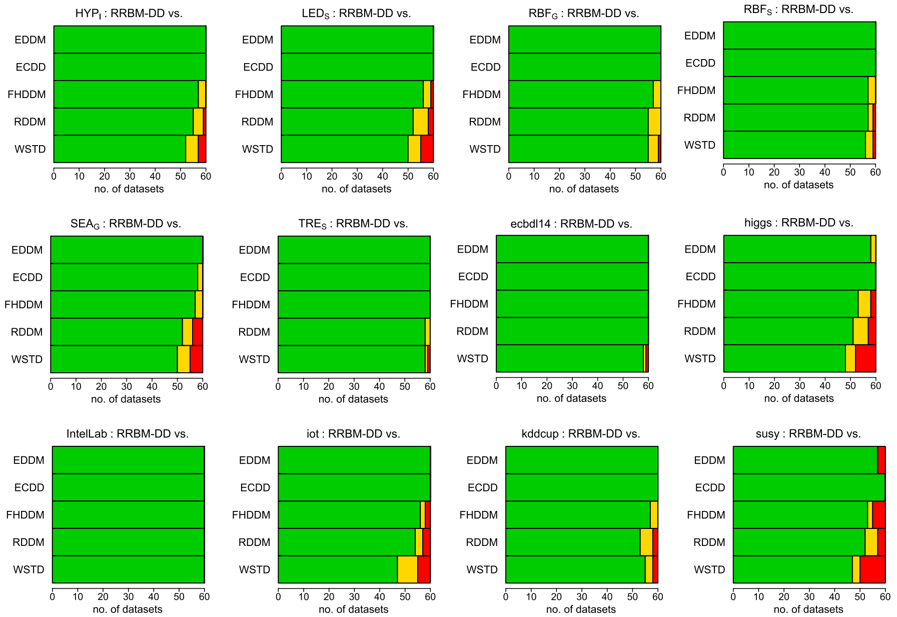

Fig. 10 Comparison of RRBM–DD with reference drift detectors under instance-based poisoning attacks and sparsely labeled assumption (only 10% labeled instances) with respect to the number of wins (green), ties (yellow), and losses (red), according to a pairwise F-test with statistical significance level $\alpha = 0.05$. Prequential accuracy used as a metric. 60 runs per dataset were obtained from six different labeling budgets, each repeated 10 times with random selection of instances to be labeled

enhancements (discussed in Sect. 4.1), RBM–DD$_{RG}$ using only robust gradient calculation and RBM–DD$_{RE}$ using only robust energy function for estimating network state.

The basic RBM–DD performs only slightly better than state-of-the-art drift detectors. This shows that the excellent performance of RRBM–DD in the face of adversarial concept drift cannot be only contributed to its trainable nature or used neural model. One must notice that both gradient and energy function of RRBM–DD offer significant improvements on their own, but in specific conditions. Robust gradient offers higher boost to robustness when dealing with instance-based poisoning attacks. This can be explained by the scaling approach used in it (see Eqs. 39– 41), as it filters out instances that differ significantly from the core concept of each class. This makes it very robust to instance by instance attacks, as they will not affect significantly the training procedure of RRBM–DD. On the other hand the robust energy function (see Eq. 42) offers much better performance when dealing with concept-based poisoning attacks. This can be contributed to noise model that is embedded in said function. It allows to model not instances, but entire noisy distributions, effectively cutting out adversarial concepts (i.e., disjuncts of instances) from strongly impacting the weight update of RRBM-DD.

RQ4 answer. RRBM–DD benefits from interplay between both of its robustness enhancing components. They offer diverse, yet mutually complementary functionalities and their combination makes RRDM-DD robust to various types of adversarial attacks.

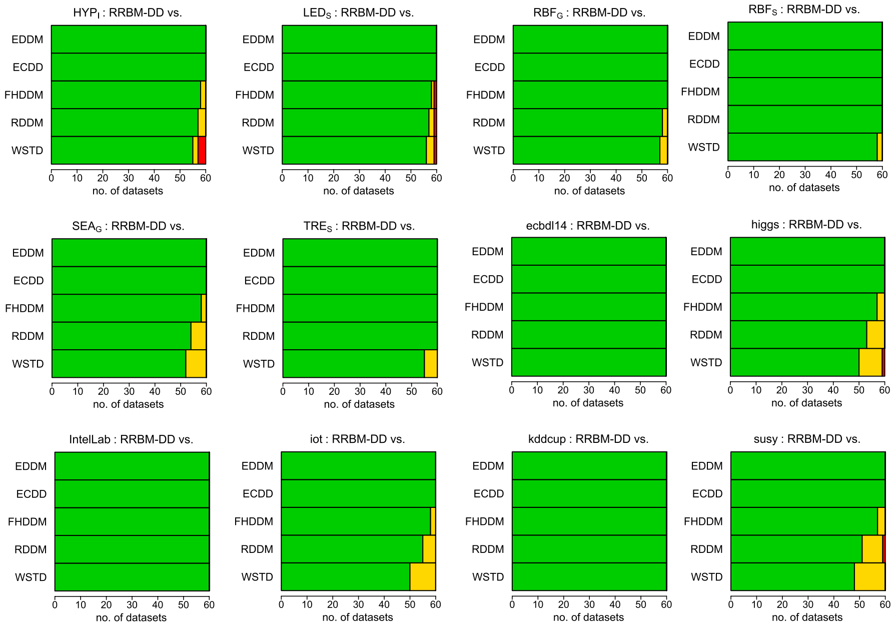

Fig. 11 Comparison of RRBM–DD with reference drift detectors under concept-based poisoning attacks and sparsely labeled assumption with respect to the number of wins (green), ties (yellow), and losses (red), according to a pairwise F-test with statistical significance level $\alpha = 0.05$. Prequential accuracy used as a metric. 60 runs per dataset were obtained from six different labeling budgets, each repeated 10 times with random selection of instances to be labeled

Table 5 RLR for RRBM–DD and reference drift detectors under instance-based poisoning attacks and sparsely labeled data. Results averaged over all labeling budgets, presented with standard deviation

Dataset	EDDM	ECDD	FHDDM	RDDM	WSTD	RRBM–DD
HYP$_I$	0.47 ± 0.11	0.49 ± 0.10	0.61 ± 0.12	0.55 ± 0.16	0.57 ± 0.13	0.80 ± 0.08
LED$_S$	0.52 ± 0.09	0.51 ± 0.09	0.63 ± 0.10	0.61 ± 0.14	0.63 ± 0.11	0.83 ± 0.05
RBF$_G$	0.40 ± 0.16	0.42 ± 0.12	0.47 ± 0.09	0.45 ± 0.15	0.50 ± 0.11	0.69 ± 0.09
RBF$_S$	0.37 ± 0.06	0.38 ± 0.05	0.41 ± 0.07	0.40 ± 0.09	0.44 ± 0.06	0.63 ± 0.07
SEA$_G$	0.56 ± 0.12	0.62 ± 0.09	0.59 ± 0.09	0.57 ± 0.14	0.66 ± 0.10	0.81 ± 0.08
TRE$_S$	0.30 ± 0.05	0.32 ± 0.04	0.35 ± 0.06	0.32 ± 0.08	0.38 ± 0.08	0.63 ± 0.04
ecbdl14	0.33 ± 0.07	0.36 ± 0.06	0.45 ± 0.09	0.42 ± 0.11	0.46 ± 0.10	0.60 ± 0.05
higgs	0.58 ± 0.14	0.62 ± 0.10	0.65 ± 0.12	0.64 ± 0.16	0.70 ± 0.11	0.82 ± 0.07
IntelLab	0.28 ± 0.08	0.30 ± 0.06	0.34 ± 0.06	0.31 ± 0.10	0.36 ± 0.08	0.60 ± 0.04
iot	0.65 ± 0.09	0.68 ± 0.06	0.72 ± 0.07	0.71 ± 0.11	0.77 ± 0.10	0.88 ± 0.07
kddcup	0.55 ± 0.05	0.53 ± 0.09	0.57 ± 0.10	0.55 ± 0.13	0.60 ± 0.10	0.77 ± 0.06
susy	0.60 ± 0.11	0.63 ± 0.08	0.67 ± 0.09	0.65 ± 0.13	0.68 ± 0.13	0.79 ± 0.07
Avg. rank	5.17	4.55	4.13	3.25	2.90	1.00

Table 6 RLR for RRBM–DD and reference drift detectors under concept-based poisoning attacks and sparsely labeled data. Results averaged over all labeling budgets, presented with standard deviation

Dataset	EDDM	ECDD	FHDDM	RDDM	WSTD	RRBM–DD
HYP_I	0.28 ± 0.07	0.29 ± 0.06	0.37 ± 0.10	0.34 ± 0.13	0.37 ± 0.13	0.69 ± 0.06
LED_S	0.25 ± 0.05	0.32 ± 0.05	0.44 ± 0.08	0.41 ± 0.10	0.46 ± 0.08	0.75 ± 0.07
RBF_G	0.23 ± 0.06	0.26 ± 0.07	0.33 ± 0.07	0.29 ± 0.12	0.34 ± 0.14	0.62 ± 0.06
RBF_S	0.18 ± 0.04	0.19 ± 0.06	0.20 ± 0.04	0.18 ± 0.05	0.22 ± 0.04	0.59 ± 0.04
SEA_G	0.40 ± 0.12	0.42 ± 0.13	0.44 ± 0.10	0.41 ± 0.15	0.45 ± 0.12	0.74 ± 0.08
TRE_S	0.09 ± 0.03	0.11 ± 0.04	0.14 ± 0.04	0.13 ± 0.04	0.17 ± 0.04	0.51 ± 0.05
ecbdl14	0.08 ± 0.02	0.07 ± 0.02	0.11 ± 0.02	0.09 ± 0.03	0.13 ± 0.02	0.48 ± 0.05
higgs	0.39 ± 0.10	0.40 ± 0.08	0.44 ± 0.11	0.41 ± 0.13	0.48 ± 0.15	0.79 ± 0.08
IntelLab	0.12 ± 0.04	0.14 ± 0.04	0.16 ± 0.05	0.13 ± 0.05	0.18 ± 0.04	0.58 ± 0.09
iot	0.51 ± 0.13	0.52 ± 0.16	0.56 ± 0.12	0.55 ± 0.16	0.60 ± 0.13	0.81 ± 0.11
kddcup	0.23 ± 0.03	0.20 ± 0.05	0.24 ± 0.05	0.19 ± 0.05	0.28 ± 0.03	0.68 ± 0.05
susy	0.31 ± 0.07	0.34 ± 0.07	0.36 ± 0.09	0.37 ± 0.10	0.39 ± 0.10	0.70 ± 0.07
avg. rank	5.38	4.14	4.66	3.72	2.10	1.00

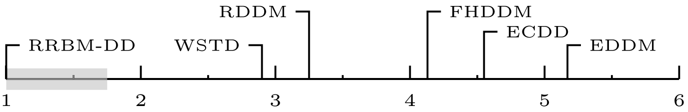

Fig. 12 The Bonferroni–Dunn test for comparison among drift detectors under instance-based poisoning attacks and sparsely labeled data, based on RLR and all examined labeling ratios

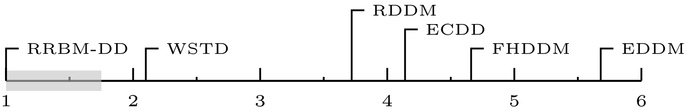

Fig. 13 The Bonferroni–Dunn test for comparison among drift detectors under concept-based poisoning attacks and sparsely labeled data, based on RLR and all examined labeling ratios

6 Conclusions and future works

Summary. In this paper we have presented a novel scenario of learning from data streams under adversarial concept drift. We proposed that two types of change may be present during the stream processing. Valid concept drift represents natural changes in the underlying data characteristics and must be correctly detected in order to allow a smooth adaptation of the stream classification system. Adversarial concept drift represents a poisoning attack from a malicious party that aims at hindering or disabling the stream classification system. We discussed the negative impacts of adversarial drift on classifiers and proposed a taxonomy of two types of attacks: instance-based and concept-based. We examined state-of-the-art concept drift detectors and concluded that none of them is capable of handling adversarial instances. To address this challenging task we proposed a novel approach that

Table 7 Ablation results for RRBM–DD according to RLR under instance-based poisoning attacks. Results averaged over fully and sparsely labeled benchmarks, presented with standard deviation

Dataset	RBM-DD	RBM–DD$_{RG}$	RBM–DD$_{RE}$	RRBM-DD
HYP$_I$	0.58 ± 0.10	0.77 ± 0.09	0.64 ± 0.09	0.82 ± 0.09
LED$_S$	0.66 ± 0.12	0.80 ± 0.07	0.72 ± 0.10	0.84 ± 0.06
RBF$_G$	0.52 ± 0.11	0.66 ± 0.09	0.59 ± 0.10	0.72 ± 0.09
RBF$_S$	0.47 ± 0.08	0.58 ± 0.09	0.53 ± 0.07	0.66 ± 0.08
SEA$_G$	0.68 ± 0.11	0.78 ± 0.10	0.74 ± 0.10	0.82 ± 0.10
TRE$_S$	0.41 ± 0.10	0.59 ± 0.06	0.48 ± 0.08	0.65 ± 0.05
ecbdl14	0.48 ± 0.11	0.58 ± 0.07	0.53 ± 0.09	0.61 ± 0.07
higgs	0.73 ± 0.11	0.82 ± 0.09	0.77 ± 0.10	0.85 ± 0.08
IntelLab	0.38 ± 0.08	0.59 ± 0.08	0.52 ± 0.08	0.66 ± 0.08
iot	0.77 ± 0.10	0.86 ± 0.05	0.83 ± 0.08	0.90 ± 0.05
kddcup	0.61 ± 0.11	0.72 ± 0.08	0.66 ± 0.10	0.79 ± 0.07
susy	0.70 ± 0.12	0.80 ± 0.09	0.73 ± 0.09	0.83 ± 0.09
avg. rank	4.00	2.05	2.95	1.00

Table 8 Ablation results for RRBM–DD according to RLR under concept-based poisoning attacks. Results averaged over fully and sparsely labeled benchmarks, presented with standard deviation

Dataset	RBM-DD	RBM–DD$_{RG}$	RBM–DD$_{RE}$	RRBM-DD
HYP$_I$	0.38 ± 0.11	0.44 ± 0.10	0.65 ± 0.06	0.72 ± 0.07
LED$_S$	0.50 ± 0.09	0.57 ± 0.10	0.68 ± 0.12	0.78 ± 0.11
RBF$_G$	0.36 ± 0.15	0.45 ± 0.11	0.56 ± 0.07	0.64 ± 0.08
RBF$_S$	0.25 ± 0.05	0.38 ± 0.05	0.50 ± 0.05	0.61 ± 0.05
SEA$_G$	0.43 ± 0.10	0.57 ± 0.11	0.68 ± 0.10	0.78 ± 0.10
TRE$_S$	0.16 ± 0.04	0.31 ± 0.06	0.42 ± 0.09	0.55 ± 0.08
ecbdl14	0.14 ± 0.04	0.29 ± 0.05	0.40 ± 0.08	0.51 ± 0.07
higgs	0.51 ± 0.13	0.65 ± 0.11	0.76 ± 0.10	0.83 ± 0.11
IntelLab	0.20 ± 0.04	0.41 ± 0.07	0.53 ± 0.08	0.64 ± 0.08
iot	0.64 ± 0.13	0.69 ± 0.12	0.74 ± 0.10	0.83 ± 0.11
kddcup	0.26 ± 0.06	0.53 ± 0.05	0.62 ± 0.05	0.72 ± 0.05
susy	0.37 ± 0.11	0.50 ± 0.09	0.63 ± 0.07	0.75 ± 0.08
avg. rank	4.00	2.90	2.10	1.00

is robust to the presence of adversarial instances: robust restricted Boltzmann machine drift detector. We presented how this fully trainable neural model can be used for efficient tracking of trends in the stream and accurate drift recognition. We enhanced our drift detector with improved gradient calculation method and energy function with an embedded noise model. These modifications made our model robust to adversarial concept drift, while still being sensitive to valid drifts. Finally, we have introduced relative loss of robustness, novel measure for evaluating the performance of drift detectors and other streaming algorithms under adversarial concept drift.

Main conclusions. The efficacy of RRBM–DD was evaluated on the basis of extensive, four part experimental study. First two experiments compared RRBM–DD with five state-of-the-art drift detectors under instance-based and concept-based poisoning attacks. Obtained results clearly backed-up our observations that none of existing drift detectors displays robustness to adversarial instances. RRBM–DD offered not only excellent detection rates of valid concept drift, but also high robustness to all types and intensity levels of adversarial attacks. These characteristics were valid in both fully and sparsely labeled data streams, making RRBM-DD a highly suitable algorithm for a plethora of real-life applications. Finally, we have conducted ablation study to understand the impact of RRBM-DD components on its robustness. Interestingly, both components were useful in specific tasks. Robust gradient algorithm was contributing most when dealing with instance-based poisoning attacks, while robust energy function offered significant gains for RRBM-DD when handling concept-based poisoning attacks.

Future works. RRBM–DD opens several directions for future research. We plan to modify its cost functions in order to tackle the issue of simultaneous adversarial concept drift and dynamic class imbalance, as well as develop deep architectures based on RRBM–DD.

Declarations

Funding This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflicts of interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Adeli, E., Thung, K., An, L., Wu, G., Shi, F., Wang, T., & Shen, D. (2019). Semi-supervised discriminative classification robust to sample-outliers and feature-noises. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 41*(2), 515–522.

Baena-García, M., Campo-Avila, J., Fidalgo-Merino, R., Bifet, A., Gavald, R., & Morales-Bueno, R. (2006). Early drift detection method. In *4th ECML PKDD international workshop on knowledge discovery from data streams*, p 77–86.

Barddal, J. P., Gomes, H. M., Enembreck, F., & Pfahringer, B. (2017). A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software, 127*, 278–294.

Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the seventh SIAM international conference on data mining*, April 26–28, 2007, Minneapolis, Minnesota, USA, SIAM, pp 443–448.

Bifet, A., Gavaldà, R. (2009). Adaptive Learning from Evolving Data Streams. In *Advances in Intelligent data analysis viii, 8th international symposium on intelligent data analysis, IDA 2009*, Lyon, France, August 31–September 2, 2009. Proceedings, Springer, Lecture Notes in Computer Science (vol. 5772, pp. 249–260).

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive Online Analysis. *Journal of Machine Learning Research, 11*, 1601–1604.

Bifet, A., Holmes, G., & Pfahringer, B. (2010). Leveraging bagging for evolving data streams. in: machine learning and knowledge discovery in databases. In *European conference, ECML PKDD 2010, Barcelona, Spain, September 20–24, 2010, Proceedings, Part I, Springer, lecture notes in computer science,* (vol. 6321, pp. 135–150).

Bifet, A., Hammer, B., & Schleif, F. (2019). Recent trends in streaming data analysis, concept drift and analysis of dynamic data sets. In *27th European symposium on artificial neural networks, ESANN 2019*, Bruges, Belgium, April 24–26, 2019.

Biggio, B., & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognit, 84*, 317–331.

Biggio, B., Nelson, B., & Laskov, P. (2012). Poisoning attacks against support vector machines. In *Proceedings of the 29th International conference on machine learning, ICML 2012*, Edinburgh, Scotland, UK, June 26–July 1, 2012, icml.cc / Omnipress.

Blanco, I. I. F., del Campo-Ávila, J., Ramos-Jiménez, G., Bueno, R. M., Díaz, A. A. O., & Mota, Y. C. (2015). Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering, 27*(3), 810–823.

Bojchevski, A., & Günnemann, S. (2019). Adversarial attacks on node embeddings via graph poisoning. In Chaudhuri K, Salakhutdinov R (Eds.) *Proceedings of the 36th international conference on machine learning, ICML 2019, 9–15 June 2019*, Long Beach, California, USA, PMLR, Proceedings of Machine Learning Research (vol. 97, pp. 695–704).

Brzezinski, D., & Stefanowski, J. (2014). Reacting to different types of concept drift: the accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems, 25*(1), 81–94.

Cano, A., & Krawczyk, B. (2020). Kappa updated ensemble for drifting data stream mining. *Machine Learning, 109*(1), 175–218.

Chatterjee, A., Gerdes, M. W., & Martinez, S. G. (2020). Statistical explorations and univariate timeseries analysis on COVID-19 datasets to understand the trend of disease spreading and death. *Sensors, 20*(11), 3089.

Choi, K., Fazekas, G., Cho, K., & Sandler, M. B. (2018). The effects of noisy labels on deep convolutional neural networks for music tagging. *IEEE Transactions on Emerging Topics in Computational Intelligence 2*(2), 139–149.

Cohen, G., Sapiro, G., & Giryes, R. (2020). Detecting adversarial samples using influence functions and nearest neighbors. In *2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020*, Seattle, WA, USA, June 13–19, 2020, IEEE, (pp. 14441–14450(.

de Barros, R. S. M., & de Carvalho Santos, S. G. T. (2018). A large-scale comparison of concept drift detectors. *Information Sciences, 451–452*, 348–370.

de Barros, R. S. M., de Lima Cabral, D. R., Gonçalves, P. M. & de Carvalho Santos, S. G. T. (2017). RDDM: reactive drift detection method. *Expert Systems with Applications, 90,* 344–355.

de Barros, R. S. M., Hidalgo, J. I. G., & de Lima Cabral, D. R. (2018). Wilcoxon rank sum test drift detector. *Neurocomputing, 275*, 1954–1963.

de Lima Cabral, D. R., & de Barros, R. S. M. (2018). Concept drift detection based on fisher's exact test. *Information Sciences 442–443*, 220–234.

dos Reis, D.M., Flach, P.A., Matwin, S., & Batista, G.E.A.P.A. (2016). Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In: Krishnapuram B, Shah M, Smola AJ, Aggarwal CC, Shen D, Rastogi R (Eds.) *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, August 13–17, 2016 (pp 1545–1554).

Ditzler, G., Roveri, M., Alippi, C., & Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine, 10*(4), 12–25.

Dong, Y., Fu, Q., Yang, X., Pang, T., Su, H., Xiao, Z., & Zhu, J. (2020). Benchmarking adversarial robustness on image classification. In *2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020*, Seattle, WA, USA, June 13–19, 2020, IEEE (pp. 318–328).

Elsayed, G. F., Shankar, S., Cheung, B., Papernot, N., Kurakin, A., Goodfellow, I. J., & Sohl-Dickstein, J. (2018). Adversarial examples that fool both computer vision and time-limited humans. *Advances in neural information processing systems 31: annual conference on neural information processing systems 2018, NeurIPS 2018, 3–8 December 2018* (pp. 3914–3924). Canada: Montréal.

Frénay, B., & Verleysen, M. (2014). Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems, 25*(5), 845–869.

Gama, J., & Castillo, G. (2006). Learning with local drift detection. In *Advanced Data mining and applications, second international conference, ADMA 2006*, Xi'an, China, August 14-16, 2006, *Proceedings, Springer, lecture notes in computer science* (vol. 4093, pp. 42–55).

Gama, J., Medas, P., Castillo, G., & Rodrigues, P.P. (2004). Learning with drift detection. In: Bazzan ALC, Labidi S (Eds.) *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian symposium on artificial intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*, Springer, lecture notes in computer science (vol. 3171, pp. 286–295).

Gao, C., Chen, Y., Liu, S., Tan, Z., Yan, S. (2020). Adversarialnas: Adversarial neural architecture search for gans. In *2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020*, Seattle, WA, USA, June 13–19, 2020, IEEE (pp. 5679–5688).

Goldenberg, I., & Webb, G. I. (2019). Survey of distance measures for quantifying concept drift and shift in numeric data. *Knowledge and Information Systems, 60*(2), 591–615.

Goldenberg, I., & Webb, G. I. (2020). PCA-based drift and shift quantification framework for multidimensional data. *Knowledge and Information Systems, 62*(7), 2835–2854.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfahringer, B., Holmes, G., & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning, 106*(9–10), 1469–1495.

Guzy, F., Wozniak, M. (2020). Employing dropout regularization to classify recurring drifted data streams. In *2020 international joint conference on neural networks, IJCNN 2020*, IEEE (pp. 1–8).

Hidalgo, J. I. G., Maciel, B. I. F., & Barros, R. S. M. (2019). Experimenting with prequential variations for data stream learning evaluation. *Computational Intelligence, 35*(4), 670–692.

Holland, M.J., & Ikeda, K. (2019). Better generalization with less data using robust gradient descent. In Chaudhuri K, Salakhutdinov R (Eds.) *Proceedings of the 36th international conference on machine learning, ICML 2019*, 9–15 June 2019, Long Beach, California, USA, *PMLR, proceedings of machine learning research* (vol .97, pp. 2761–2770).

Huang, D.T.J., Koh, Y.S., Dobbie, G., Pears, R. (2014). Detecting volatility shift in data streams. In: Kumar R, Toivonen H, Pei J, Huang JZ, Wu X (Eds.) 2014 IEEE international conference on data mining, ICDM 2014, Shenzhen, China, December 14-17, 2014 (pp. 863–868). IEEE Computer Society .

Jaworski, M., Rutkowski, L., Staszewski, P., & Najgebauer, P. (2021). Monitoring of changes in data stream distribution using convolutional restricted boltzmann machines. In: Rutkowski L, Scherer R, Korytkowski M, Pedrycz W, Tadeusiewicz R, Zurada JM (Eds.) *Artificial intelligence and soft computing - 20th international conference, ICAISC 2021*, Virtual Event, June 21–23, 2021, *Proceedings, Part I, Springer, Lecture Notes in Computer Science* (vol. 12854, pp. 338–346).

Jothimurugesan, E., Tahmasbi, A., Gibbons, P. B., & Tirthapura, S. (2018). Variance-reduced stochastic gradient descent on streaming data. In *Advances in neural information processing systems 31: annual conference on neural information processing systems 2018, NeurIPS 2018, 3–8 December 2018* (pp. 9928–9937). Montréal., Canada .

Kaneko, T., & Harada, T. (2020). Noise robust generative adversarial networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, Seattle, WA, USA, June 13–19, 2020, IEEE, pp 8401–8411.

Koltchinskii, V. I. (1997). M-estimation, convexity and quantiles. *The Annals of Statistics, 25*(2), 435–477.

Korycki, L., & Krawczyk, B. (2017). Combining active learning and self-labeling for data stream mining. In *Proceedings of the 10th international conference on computer recognition systems CORES 2017*, Polanica Zdroj, Poland, 22-24 May 2017, Advances in Intelligent Systems and Computing (vol. 578, pp. 481–490).

Korycki, Ł., Krawczyk, B. (2018). Clustering-driven and dynamically diversified ensemble for drifting data streams. In *IEEE International Conference on Big Data, Big Data 2018*, Seattle, WA, USA, December 10–13, 2018, IEEE (pp. 1037–1044).

Korycki, Ł., & Krawczyk, B. (2019). Unsupervised Drift Detector Ensembles for Data Stream Mining. In *2019 IEEE international conference on data science and advanced analytics, DSAA 2019*, Washington, DC, USA, October 5–8, 2019, IEEE (pp. 317–325).

Korycki, L., Cano, A., & Krawczyk, B. (2019). Active learning with abstaining classifiers for imbalanced drifting data streams. In *2019 IEEE international conference on big data (big data)*, Los Angeles, CA, USA, December 9-12, 2019, IEEE (pp. 2334–2343).

Krawczyk, B., & Cano, A. (2018). Online ensemble learning with abstaining classifiers for drifting and noisy data streams. *Applied Soft Computing, 68*, 677–692.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Wozniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Inf Fusion, 37*, 132–156.

Li, D., & Li, Q. (2020). Adversarial deep ensemble: Evasion attacks and defenses for malware detection. *IEEE Transactions on Information Forensics and Security 15*, 3886–3900.

Liu, A., Lu, J., & Zhang, G. (2021). Concept drift detection via equal intensity k-means space partitioning. *IEEE Transactions on Cybernetics 51*(6), 3198–3211.

Liu, S., Feng, L., Wu, J., Hou, G., & Han, G. (2017). Concept drift detection for data stream learning based on angle optimized global embedding and principal component analysis in sensor networks. *Computers & Electrical Engineering 58*, 327–336.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: a review. *IEEE Transactions on Knowledge and Data Engineering, 31*(12), 2346–2363.

Lughofer, E. (2017). On-line active learning: A new paradigm to improve practical useability of data stream modeling methods. *Information Sciences, 415*, 356–376.

Maciel, B.I.F., de Carvalho Santos, S.G.T., & de Barros, R.S.M. (2015). A lightweight concept drift detection ensemble. In *27th IEEE international conference on tools with artificial intelligence, ICTAI 2015*, Vietri sul Mare, Italy, November 9-11, 2015, IEEE Computer Society (pp. 1061–1068).

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *6th international conference on learning representations, ICLR 2018*, Vancouver, BC, Canada, April 30 - May 3, 2018, *Conference Track Proceedings*, OpenReview.net.

Mahdi, O. A., Pardede, E., Ali, N., & Cao, J. (2020). Diversity measure as a new drift detection method in data streaming. *Knowledge-Based Systems, 191*, 105227.

Mahjoub, C., Bellanger, J., Kachouri, A., & Bouquin-Jeannès, R. L. (2020). On the performance of temporal granger causality measurements on time series: a comparative study. *Signal Image Video Process, 14*(5), 955–963.

Mahloujifar, S., Diochnos, D.I., & Mahmoody, M. (2019). The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In: *The thirty-third AAAI conference on artificial intelligence, AAAI 2019, The thirty-first innovative applications of artificial intelligence conference, IAAI 2019*, *The Ninth AAAI symposium on educational advances in artificial intelligence, EAAI 2019,* Honolulu, Hawaii, USA, January 27–February 1, 2019, (pp 4536–4543) AAAI Press .

Masegosa, A. R., Martínez, A. M., Ramos-López, D., Langseth, H., Nielsen, T. D., & Salmerón, A. (2020). Analyzing concept drift: A case study in the financial sector. *Intelligent Data Analysis, 24*(3), 665–688.

Masud, M. M., Woolam, C., Gao, J., Khan, L., Han, J., Hamlen, K. W., & Oza, N. C. (2011). Facing the reality of data stream classification: coping with scarcity of labeled data. *Knowledge and Information Systems 33*(1), 213–244.

Miller, D. J., Xiang, Z., & Kesidis, G. (2020). Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE, 108*(3), 402–433.

Nishida, K., & Yamauchi, K. (2007). Detecting concept drift using statistical testing. In: Corruble V, Takeda M, Suzuki E (Eds.) *Discovery science, 10th international conference, DS 2007*, Sendai, Japan, October 1–4, 2007, Proceedings, Springer, *Lecture notes in computer science* (vol. 4755, pp. 264–269).

Oliveira, G.H.F.M., Minku, L.L., & Oliveira, A.L.I. (2019). GMM-VRD: A Gaussian Mixture model for dealing with virtual and real concept drifts. In *International joint conference on neural networks, IJCNN 2019* Budapest, Hungary, July 14-19, 2019 (pp. 1–8) IEEE .

Oza, N.C., & Russell, S.J. (2001). Online Bagging and Boosting. In: Richardson TS, Jaakkola TS (Eds.) *Proceedings of the eighth international workshop on artificial intelligence and statistics, AISTATS 2001*, Key West, Florida, USA, January 4–7, 2001, Society for Artificial Intelligence and Statistics.

Pears, R., Sakthithasan, S., & Koh, Y. S. (2014). Detecting concept change in dynamic data streams: A sequential approach based on reservoir sampling. *Machine Learning, 97*(3), 259–293.

Pesaranghader, A., & Viktor, H.L. (2016). Fast hoeffding drift detection method for evolving data streams. In *Machine learning and knowledge discovery in databases - European conference, ECML PKDD 2016*, Riva del Garda, Italy, September 19–23, 2016, *Proceedings, Part II, Springer, Lecture Notes in Computer Science* (vol. 9852, pp. 96–111).

Pesaranghader, A., Viktor, H., & Paquet, E. (2018). Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams. *Machine Learning, 107*(11), 1711–1743.

Pinage, F. A., dos Santos, E. M., & Gama, J. (2020). A drift detection method based on dynamic classifier selection. *Data Mining and Knowledge Discovery, 34*(1), 50–74.

Ramírez-Gallego, S., Krawczyk, B., García, S., Wozniak, M., Benítez, J. M., & Herrera, F. (2017). Nearest Neighbor Classification for High-Speed Big Data Streams Using Spark. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47*(10), 2727–2739.

Roseberry, M., Krawczyk, B., & Cano, A. (2019). Multi-label punitive kNN with self-adjusting memory for drifting data streams. *ACM Trans Knowl Discov Data 13*(6):60:1–60:31.

Ross, G. J., Adams, N. M., Tasoulis, D. K., & Hand, D. J. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters 33*(2), 191–198.

Sáez, J. A., Luengo, J., & Herrera, F. (2016). Evaluating the classifier behavior with noisy data considering performance and robustness: The equalized loss of accuracy measure. *Neurocomputing, 176*, 26–35.

Salakhutdinov, R., & Hinton, G.E. (2009). Deep boltzmann machines. In Dyk DAV, Welling M (Eds) *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009*, Clearwater Beach, Florida, USA, April 16–18, 2009, JMLR.org, *JMLR Proceedings* (vol. 5, pp. 448–455(.

Sebastião, R., & Fernandes, J.M. (2017). Supporting the page-hinkley test with empirical mode decomposition for change detection. In *Foundations of intelligent systems - 23rd international symposium,* *ISMIS 2017*, Warsaw, Poland, June 26–29, 2017, *Proceedings, Springer, lecture notes in computer science* (vol. 10352, pp. 492–498).

Sethi, T. S., & Kantardzic, M. M. (2018). Handling adversarial concept drift in streaming data. *Expert Systems with Applications, 97*, 18–40.

Shaker, A., & Hüllermeier, E. (2015). Recovery analysis for adaptive learning from non-stationary data streams: Experimental design and case study. *Neurocomputing, 150*, 250–264.

Sobolewski, P., & Wozniak, M. (2017). SCR: simulated concept recurrence - a non-supervised tool for dealing with shifting concept. *Expert Systems: The Journal of Knowledge Engineering* 34(5).

Su, J., Vargas, D. V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation, 23*(5), 828–841.

Sun, X. (2008). Assessing nonlinear granger causality from multivariate time series. In *Machine learning and knowledge discovery in databases, European conference, ECML/PKDD 2008*, Antwerp, Belgium, September 15–19, 2008, *Proceedings, Part II, Springer, lecture notes in computer science* (vol. 5212, pp. 440–455).

Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N., & Robinson, S. (2017). Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *The workshops of the the thirty-first AAAI conference on artificial intelligence, Saturday,* February 4–9, 2017, San Francisco, California, USA, AAAI Press, AAAI Workshops (vol. WS-17).

Umer, M., Frederickson, C., & Polikar, R. (2019). Vulnerability of covariate shift adaptation against malicious poisoning attacks. In *International joint conference on neural networks, IJCNN 2019 Budapest*, Hungary, July 14–19, 2019, IEEE (pp 1–8).

Veloso, B., Gama, J., & Malheiro, B. (2018). Self hyper-parameter tuning for data streams. In Soldatova LN, Vanschoren J, Papadopoulos GA, Ceci M (Eds.) *Discovery science - 21st international conference, DS 2018, limassol, cyprus, October 29-31, 2018*, *Proceedings, Springer, lecture notes in computer science* (vol. 11198, pp. 241–255).

Wallace, E., Feng, S., Kandpal, N., Gardner, M., & Singh, S. (2019). Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing, EMNLP-IJCNLP 2019*, Hong Kong, China, November 3–7, 2019, Association for Computational Linguistics (pp. 2153–2162).

Wang, Z., Hu, G., & Hu, Q. (2020). Training noise-robust deep neural networks via meta-learning. In *2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020*, Seattle, WA, USA, June 13–19, 2020, IEEE (pp 4523–4532).

Wozniak, M., Graña, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion, 16,* 3–17.

Xiao, C., Li, B., Zhu, J., He, W., Liu, M., & Song, D. (2018). Generating adversarial examples with adversarial networks. In Lang ,J, (Ed). *Proceedings of the twenty-seventh international joint conference on artificial intelligence, IJCAI 2018*, July 13–19, 2018, Stockholm, Sweden, ijcai.org (pp. 3905–3911).

Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., & Roli, F. (2015). Is feature selection secure against training data poisoning? In *Proceedings of the 32nd international conference on machine learning, ICML 2015, Lille, France, 6–11 July 2015, JMLR.org, JMLR workshop and conference proceedings* (vol. 37, pp 1689–1698).

Yu, H., & Webb, G. I. (2019). Adaptive online extreme learning machine by regulating forgetting factor by concept drift map. *Neurocomputing, 343*, 141–153.

Zhang, F., Chan, P. P. K., Biggio, B., Yeung, D. S., & Roli, F. (2016). Adversarial feature selection against evasion attacks. *IEEE Transactions on Cybernetics, 46*(3), 766–777.

Zliobaite, I., Budka, M., & Stahl, F. T. (2015). Towards cost-sensitive adaptation: When is it worth updating your predictive model? *Neurocomputing, 150*, 240–249.