Adversarial Attacks for Drift Detection

Fabian Hinder, Valerie Vaquet, and Barbara Hammer*

Bielefeld University – Inspiration 1, 33619 Bielefeld – Germany

Abstract. Concept drift refers to the change of data distributions over time. While drift poses a challenge for learning models, requiring their continual adaption, it is also relevant in system monitoring to detect malfunctions, system failures, and unexpected behavior. In the latter case, the robust and reliable detection of drifts is imperative. This work studies the shortcomings of commonly used drift detection schemes. We show that they are prone to adversarial attacks, i.e., streams with undetected drift. In particular, we give necessary and sufficient conditions for their existence, provide methods for their construction, and demonstrate this behavior in experiments.

1 Introduction

Data from the real world is often subject to continuous changes known as concept drift [1, 2, 3]. Such can be caused by seasonal changes, changed demands, aging of sensors, etc. Concept drift not only poses a problem for maintaining high performance in learning models [2, 3] but also plays a crucial role in system monitoring [1]. In the latter case, the detection of concept drift is crucial as it enables the detection of anomalous behavior. Examples include machine malfunctions or failures, network security, environmental changes, and critical infrastructures. This can be done by detecting (unexpected) drifts [4, 1, 5]. In these contexts, the ability to robustly detect drift is essential.

In addition to problems such as noise and sampling error, which challenge all statistical methods, drift detection faces a special kind of difficulty when the drift follows certain patterns that evade detection. In this work, we study those specific drifts that we will refer to as "drift adversarials". Similar to adversarial attacks in classification, that exploit model properties to force wrong decisions [6], drift adversarials exploit weaknesses in the detection methods, and thus allow significant concept drift to occur without triggering alarms posing major issues for monitoring systems. Besides the construction of drift adversarials, the presented theory also provides tools to check whether a specific drift detector is provably correct.

This paper is structured as follows: First (Section 2) we recall the definition of concept drift and define the two setups for which we will construct our adversarials. In Section 3, we construct drift adversarials that exploit which data is used for the analysis. Here, we mainly focus on two window approaches. In the last part (Section 4) we perform a numerical evaluation of our considerations and conclude the work (Section 5).

*Funding from the European Research Council (ERC) under the ERC Synergy Grant Water-Futures (Grant agreement No. 951424) and funding in the scope of the BMBF project KI Akademie OWL under grant agreement No 01IS24057A is gratefully acknowledged.

Algorithm 1 Two Window Drift Detector (without memory management)

```
procedure DriftDetection($(x_i)_{i=1}^n$ data stream, $\theta$ detection threshold)
    for $(W_1, W_2) \in \mathcal{W}(n)$ do                    ▷ iterate over all considered window pairs
        $p \leftarrow$ Test($\{x_i \mid i \in W_1\}, \{x_i \mid i \in W_2\}$)   ▷ Test drift between $W_1$ and $W_2$
        if $p < \theta$ then
            Alert drift
        end if
    end for
end procedure
```

2 Concept Drift and Drift Detection

Most machine learning research focuses on the batch setup where one considers a fixed data set as i.i.d. random variables $X_1, \ldots, X_n$ following some distribution $\mathcal{D}$ on the data space $\mathcal{X}$. However, in many scenarios, data is obtained as a stream over time and is thus prone to potential changes of the underlying distribution, a phenomenon known as *concept drift* [1, 3]. In such *finite sample* setup, drift is typically defined in a *sample-wise sense*, that is two samples not having the same distribution, i.e., $\exists i, j : \mathbb{P}_{X_i} \neq \mathbb{P}_{X_j}$ [3]. *Drift detection* refers to the task of deciding whether or not the stream is affected by drift. One issue of this setup is that one cannot estimate the distribution $\mathbb{P}_{X_i}$ the single sample X_i follows [1].

To analyze this task theoretically, we consider an extension building on distribution processes [1] describing the *limiting case.* We model a time $\mathcal{T}$ indexed family of probability measures $\mathcal{D}_t$ on $\mathcal{X}$ together with an observation probability P_T on $\mathcal{T}$ [1]. A *stream* consists of dated data points $(X_1, T_1), (X_2, T_2), \ldots$ such that a data point X_i observed at time t follows the distribution $\mathcal{D}_t$, i.e., $T_i \sim P_T$ and $X_i \mid T_i = t \sim \mathcal{D}_t$. *Concept drift* occurs if the chance of observing two different distributions is larger zero [1], i.e., $\mathbb{P}[\exists i, j : \mathbb{P}_{X_i} \neq \mathbb{P}_{X_j}] > 0$. Notice that this definition is not limited to abrupt drift but all kinds of drift, such as gradual or recurring, and the statistical nature resolves the estimation problem.

In this paper, we are interested in constructing scenarios containing drift that is not detected. We will first investigate this task theoretically by examining the limiting case leveraging the definition by distribution processes. This also allows us to prove guarantees. Afterward, we study the finite case and derive an algorithm for the construction of drift adversarials.

3 Adversarial Attacks for Drift Detection

Most drift detectors process data on sliding windows using some statistical tool, most commonly a metric [1] (see Algorithm 1). This allows for two natural attack scenarios: *Metric Adversarials* construct distributions indistinguishable by the metric, while *Window Adversarials* exploit the data selection stage.

3.1 Metric Adversarials

The most commonly used drift detectors are based on learning models referring to the optimal model or model accuracy to detect drift [3]. However, as already pointed out in [7] this approach is flawed and can be exploited in many cases.

Table 1: Overview of improper adversarial functions for common windowing schemes used in drift detection (assuming Lebesgue measure $P_T = \lambda$). Cases with Boundary Effects (BE) are marked. Proofs in ArXiv version [8].

$\mathcal{T}$	$\mathcal{W}$	Adv_0 (Theorem 1)	BE
$\mathbb{R}$	$([t-l,t],[t,t+l])\ t \in \mathcal{T}$ two sliding windows	$f(t) = f(t+l)$ l-periodic functions	✗
$\mathbb{R}_{\geq 0}$	$([0,a],[t,t+l])\ t \geq a$ fixed reference window	$f(t) = f(t+l)$ for $t \geq a$ and $a^{-1}\int_0^a f(t)\mathrm{d}t = l^{-1}\int_a^{a+l} f(t)\mathrm{d}t$ l-periodic after a with same mean	
$\mathbb{R}_{\geq 0}$	$([0,t],[t,t+l])\ t \geq a$ growing reference window	$a^{-1}\int_0^a f(s)\mathrm{d}s = f(a) = f(t) \forall t \geq a$ arbitrary before a and then constant	✗

Indeed, the authors provide a constructive proof that can easily be modified to construct a metric adversarial. Other approaches for which metric adversarials can be constructed include methods like the windowing Kolmogorov-Smirnov test that operates feature-wise and thus ignores drifts in correlations as shown in [1] or methods that use deep embeddings for which classical adversarials can be constructed. However, in many cases, it is not possible to construct a metric adversarial, e.g., when the used metric is indeed a metric. We will therefore mainly focus on window adversarials in this paper.

3.2 Window Adversarials for Two-Window-Based Detectors

As most drift detectors work by comparing data from two windows [1] we will focus on this setup. In the following, we consider the limiting and finite case.

The limiting case We refer to the case where we take the sampling rate to infinity so that errors due to sampling vanish and the drift detector becomes a map of the kernel $\mathcal{D}_t$, i.e., the limiting case of Algorithm 1 takes on the form

$$A(\mathcal{D}_t) = \mathbf{1}\left[\sup_{(W_1,W_2)\in\mathcal{W}} d(\mathcal{D}_{W_1}, \mathcal{D}_{W_2}) > 0\right] \tag{1}$$

where $\mathcal{W}$ is the set of all window pairs directly compared by the detector and d is the used metric. Here, A detects drift if $A(\mathcal{D}_t) = 1$. As A cannot have false positives, the window adversarials are given by false negatives which can be constructed as follows:

Theorem 1. *Define the* improper adversarial functions *for A as in Eq.* (1) *as*

$$\mathrm{Adv}_0(A) = \left\{ f : \mathcal{T} \to [0,1] \;\middle|\; P_T(W_2)\int_{W_1} f \;\mathrm{d}P_T = P_T(W_1)\int_{W_2} f \;\mathrm{d}P_T \forall (W_1,W_2) \in \mathcal{W} \right\} \tag{2}$$

then A detects no drift, i.e., $A(\mathcal{D}_t) = 0$, if and only if $t \mapsto \mathcal{D}_t(S) \in \mathrm{Adv}_0(A)$ *for all measurable $S \subset \mathcal{X}$.*

Define the adversarial functions $\mathrm{Adv}(A) \subset \mathrm{Adv}_0(A)$ *as those that are not constant. Then,* $\mathrm{Adv}(A)$ *describes all distribution processes with drift that is not detected by A. In particular, for $f \in \mathrm{Adv}(A)$ and distributions $P \neq Q$ on $\mathcal{X}$, $\mathcal{D}_t = f(t)P + (1-f(t))Q$ is a window adversarial, i.e., $\mathcal{D}_t$ has drift and $A(\mathcal{D}_t) = 0$. Conversely, for every window adversarial $A(\mathcal{D}_t) = 0$ we have*

Algorithm 2 Construction of Drift Adversarials

```
function ConstructDriftAdversarial($P, Q$ sampling distributions, $\mathcal{W}(n)$ windowing scheme to be attacked)
    $\mathbf{W}_n \leftarrow [\mathbf{1}]$
    for $(W_1, W_2) \in \mathcal{W}(n)$ do
        $\mathbf{W}_n \leftarrow \mathbf{W}_n + \left[|W_1|^{-1} \sum_{i=1}^{n} \mathbf{1}[i \in W_1] e_i - |W_2|^{-1} \sum_{i=1}^{n} \mathbf{1}[i \in W_2] e_i\right]$
    end for
    $v \leftarrow$ Solve($\mathbf{W}_n x = 0$)                    ▷ Interpret $\mathbf{W}_n$ as a matrix
    $v \leftarrow \frac{v - \min_i v_i}{\max_i v_i - \min_i v_i}$
    $x \leftarrow []$
    for $i = 1, \ldots, n$ do
        $x \leftarrow x + [$Sample$(v_i P + (1 - v_i) Q)]$
    end for
    return $x$
end function
```

$t \mapsto \mathcal{D}_t(S) \in \text{Adv}_0(A)$. Therefore, if $\text{Adv}(A) = \emptyset$ then A detects every drift assuming d is a metric.

Proof. All proofs can be found in the ArXiv version [8]. □

We want to stress that the adversaials do not depend on the metric d but only the considered windows and that the drift detector detects every possible drift if and only if $\text{Adv}(A) = \emptyset$. This can for example be achieved by combining multiple drift detectors as $\text{Adv}(\,(A, B)\,) \subset \text{Adv}(A) \cap \text{Adv}(B)$.

Most detectors use a sliding window for the current distribution of fixed length. There are three main strategies for the reference window: 1) fixed, 2) growing, and 3) sliding with fixed length [1]. Furthermore, there are two update strategies: Either the update is performed after every single data point, which in the limit is for every time point, or by considering chunks of data points. For the latter, we can hide arbitrary drifts within a chunk allowing for trivial adversarials. For point-wise updates and any of the aforementioned reference windows, we present the adversarial functions in Table 1.

The finite case Analog to the limiting case we can also consider the case of finite samples $X_1, \ldots, X_n$. In this case, the windows refer to which samples are considered, i.e., $W_1, W_2 \subset [n]$. We will denote the set of all considered window pairs for n samples by $\mathcal{W}(n)$ together with a normalized distance measure, e.g., a statistic test, and a decision threshold θ this leads to Algorithm 1. Usually, there is some memory management so that we do not have to store the entire stream which however depends on the windowing scheme $\mathcal{W}(n)$.

We can encode the window selection $\mathcal{W}(n)$ into a single weight matrix $\mathbf{W}_n$ that encodes the pair (W_1, W_2) as the vector $w = |W_1|^{-1} \sum_{i \in W_1} e_i - |W_2|^{-1} \sum_{i \in W_2} e_i$ where e_i is the i-th coordinate vector. This representation is quite useful as it for example allows us to write the biased MMD – a kernel-based probability metric commonly used in drift detection [1] – of the i-th window as $(\mathbf{W}_n^\top K \mathbf{W}_n)_{ii}$ where $K_{ij} = k(X_i, X_j)$ is the kernel matrix. For our purpose, it is useful as the kernel of $\mathbf{W}_n$ can be used to construct drift adversarials. To do so choose $v \in [0,1]^n$ with $\mathbf{W}_n v = 0$ and then sample $X_i \sim v_i P + (1 - v_i) Q$. This idea is represented in Algorithm 2. If v is not constant and $P \neq Q$ then

Table 2: Result of numerical analysis. 90%/10%-quintile of obtained p-values (500 runs). Correct result is $p = 0$, lining marks adversarials according to theory. The number in brackets is the length of the initial reference window.

Dataset / $\mathcal{W}$	fixed (100)	fixed (150)	grow (100)	grow (150)	slideing
Periodic	$\underline{0.63/0.28}$	0.00/0.00	0.00/0.00	0.00/0.00	$\underline{0.39/0.27}$
Rand.Const (100)	$\underline{0.59/0.31}$	$\underline{0.54/0.31}$	$\underline{0.39/0.27}$	$\underline{0.46/0.28}$	0.00/0.00
Rand.Const (150)	0.00/0.00	$\underline{0.50/0.25}$	0.00/0.00	$\underline{0.45/0.24}$	0.00/0.00
Rand.Per. (100)	$\underline{0.49/0.26}$	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00
Rand.Per. (150)	0.00/0.00	$\underline{0.65/0.22}$	0.00/0.00	0.02/0.00	0.00/0.00

the distributions differ for some X_i, i.e., there is drift in the sample-wise sense, the mean distributions of the samples in W_1 and W_2 however coincide for all $(W_1, W_2) \in \mathcal{W}(n)$ which is what Algorithm 1 line 3 is testing for. There are ways to increase the quality by choosing $v \in \{0,1\}^n$ or trying to avoid fast oscillations as such streams are similar to non-drifting streams. Notice, that there is a close connection between the limiting and the finite setup given by sampling adversarial functions (Theorem 1) equidistant to obtain v. Yet, $\{v \mid \mathbf{W}_n v = 0\}$ can be much larger than $\mathrm{Adv}(A)$ due to boundary effects (BE in Table 1).

Instead of comparing the mean distribution of two windows, some drift detectors – dubbed block-based in [1] – check for any kind of drift within a single window. Using similar techniques it can be shown that such detectors are not prone to window adversarial attacks. In the next section, we will test our theoretical observations empirically.

4 Empirical Evaluation

To evaluate our methodology we consider two empirical setups: a numerical analysis on synthetic data, and a showcase on data from critical infrastructure.[1]

Synthetic Data We perform a numerical analysis based on the simple two-squares dataset [1] (drift intensity 5). We create the adversarial streams using Algorithm 2 where line 6 is performed by hand to assure $v_i \in \{0,1\}$ with as little changes as possible (see Table 1). Each stream has a length of 1,000 samples, window sizes of the sliding window is 100, and (initial) reference window is 100/150. We use the permutation MMD test [1] with 2,500 permutations and consider the smallest p-value found in the stream. We performed 500 independent runs for each setup. The results are reported in Table 2. As can be seen, there is a (nearly) perfect alignment of our theoretical predictions and the empirical results with only one exceptional case.

Application to Water Distribution Networks Thus far we have considered drift adversarials as a kind of attack where we try to hide the drift from the monitoring system. However, in case we expect certain drifts that we do not want to detect, we can try to construct a drift detector so that the adversarials are exactly the expected drifts. For this study, we explicitly consider water distribution networks from which we obtain pressure measurements [4]. We are interested in leakage detection which can be done via drift detection. However,

[1] The code can be found at `https://github.com/FabianHinder/Drift-Adversarials`

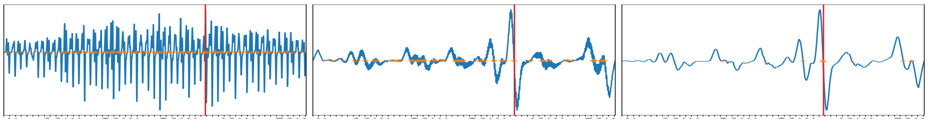

Fig. 1: Shape curve for different window sizes (1 day, $6\frac{1}{2}$ days, 1 week). Red line marks leakage, orange crosses candidate points (transparency is MMD).

as the demands on drinking water are not constant over time, we expect daily (day-night-cycle) and weekly (week-weekend-cycle) patterns, which need to be removed before drift and leakages are directly related [4]. Following [4], we use the Shape Drift Detector [9] which postprocesses the MMD of two consecutive sliding windows to find candidate drift points. The result of different window lengths is presented in Fig. 1. As can be seen, windows of one-day length detects weekends, while a one-week length window mainly detects the leakage as desired. Also notice, that this is not an instability of the algorithm as can be seen by considering the window length of $6\frac{1}{2}$ days (middle figure).

5 Conclusion

In this paper, we introduced the concept of drift adversarials. We showed that many commonly used drift detectors are subject to at least some drift adversarial attacks. We considered the problem from a general theoretical and concrete point of view and evaluated our findings empirically. Furthermore, we investigated the potential of our theory to construct problem-tailored drift detectors which seems to be a promising approach but requires further research.

Our considerations show that drift adversarials pose a major problem but might be numerically unstable. A further analysis is yet subject to future work.

References

[1] F. Hinder, V. Vaquet, and B. Hammer. One or two things we know about concept drift—a survey on monitoring in evolving environments. part a: detecting concept drift. *Frontiers in Artificial Intelligence*, 2024.

[2] A. Bifet and J. Gama. Iot data stream analytics. *Ann. des Télécomm.*, 75(9-10), 2020.

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4), March 2014.

[4] V. Vaquet, F. Hinder, and B. Hammer. Investigating the suitability of concept drift detection for detecting leakages in water distribution networks. In *ICPRAM*, 2024.

[5] G. I. Webb, L. K. Lee, F. Petitjean, and B. Goethals. Understanding concept drift. *CoRR*, abs/1704.00362, 2017.

[6] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI TIT*, 2021.

[7] F. Hinder, V. Vaquet, J. Brinkrolf, and B. Hammer. On the hardness and necessity of supervised concept drift detection. In *ICPRAM*, pages 164–175, 2023.

[8] F. Hinder, V. Vaquet, and B. Hammer. Adversarial attacks for drift detection. *arXiv preprint arXiv:2411.16591*, 2024.

[9] F. Hinder, J. Brinkrolf, V. Vaquet, and B. Hammer. A shape-based method for concept drift detection and signal denoising. In *SSCI*, pages 01–08. IEEE, 2021.