

Flexible and Efficient Drift Detection without Labels

Nelvin Tan
American Express
thongcainelvin.tan@aexp.com

Yu-Ching Shih
American Express
yuching.shih1@aexp.com

Dong Yang
American Express
dong.yang@aexp.com

Amol Salunkhe
American Express
amol.salunkhe@aexp.com

Abstract—Machine learning models are being increasingly used to automate decisions in almost every domain, and ensuring the performance of these models is crucial for ensuring high quality machine learning enabled services. Ensuring concept drift is detected early is thus of the highest importance. A lot of research on concept drift has focused on the supervised case that assumes the true labels of supervised tasks are available immediately after making predictions. Controlling for false positives while monitoring the performance of predictive models used to make inference from extremely large datasets periodically, where the true labels are not instantly available, becomes extremely challenging. We propose a flexible and efficient concept drift detection algorithm that uses classical statistical process control in a label-less setting to accurately detect concept drifts. We show empirically that under computational constraints, our approach has better statistical power than previous known methods. Furthermore, we introduce a new semi-supervised drift detection framework to model the scenario of detecting drift (without labels) given prior detections, and show how our drift detection algorithm can be incorporated effectively into this framework. We demonstrate promising performance via numerical simulations.

Index Terms—Concept drift, unsupervised learning, machine learning, time complexity, classification, big data

I. INTRODUCTION

Machine Learning has been adopted as a popular technique for providing data-driven prediction capabilities, and is at the core of several otherwise complicated or intractable tasks. The ability to generalize and extrapolate from data has made its usage attractive as a general approach to data driven problem solving. However, the generalizability of models relies on an important assumption of data *stationarity*, which states that the data points in the data stream should be identically and independently distributed (i.i.d.), i.e., datapoints are generated from the same distribution [1]. While this is a reasonable assumption, in practice it is often violated. The data that models work with in production might become significantly different than the static dataset that the model was trained on – a phenomenon named *concept drift* [2], [3].

We consider the *abrupt* drift setting, where a new concept occurs in a short period of time (e.g. at the beginning of COVID-19 in March 2020, stock prices suddenly changed). Following the parlance of [4], we focus on non-online drift detection setting with whole-window detection, where the entire window is used in the detection. Most importantly, we consider the label-less (i.e., unsupervised) setting where labels of the new data (i.e., during production) are unavailable.

A. Problem Setup

Formally, a concept drift is the problem where there exists a time t such that

$$P_t(\mathbf{x}, y) \neq P_{t+1}(\mathbf{x}, y), \quad (1)$$

where P_t denotes the joint probability distribution of the features $\mathbf{x} \in \mathbb{R}^m$ and target $y \in \mathbb{R}$. By the chain rule of probability, we have

$$P_t(\mathbf{x}, y) = P_t(\mathbf{x}) \cdot P_t(y|\mathbf{x}), \quad (2)$$

which implies the following types of drift [5]:

- **Virtual drift.** We have $P_t(\mathbf{x}) \neq P_{t+1}(\mathbf{x})$ while $P_t(y|\mathbf{x}) = P_{t+1}(y|\mathbf{x})$.
- **Real drift.** We have $P_t(y|\mathbf{x}) \neq P_{t+1}(y|\mathbf{x})$ while $P_t(\mathbf{x}) = P_{t+1}(\mathbf{x})$.
- **Mixture drift.** Mixture of virtual drift and actual drift, namely, $P_t(y|\mathbf{x}) \neq P_{t+1}(y|\mathbf{x})$ and $P_t(\mathbf{x}) \neq P_{t+1}(\mathbf{x})$.

Without access to the labels y , we can only detect virtual drift and mixture drift. Note that virtual drift may still cause problems for the inference model, since changes in $P_t(y|\mathbf{x})$ may change the error of the inference model, even if $P_t(y|\mathbf{x}) = P_{t+1}(y|\mathbf{x})$ [6]. In this paper, we mainly look at the following problem setup:

Drift detection without labels (problem 1). Given a datastream, we have two windows, the *reference* window/set \mathcal{S}_R and the *detection* window/set \mathcal{S}_D . Furthermore, we have access to the *training* set \mathcal{S}_T that was used to train the inference model and is of the same distribution as \mathcal{S}_R . The datapoints in the reference window are labeled whereas the datapoints in the detection window are unlabeled and may (or may not) have suffered a concept drift. Let us denote the distribution of \mathcal{S}_T and \mathcal{S}_R as P_R and the distribution of \mathcal{S}_D as P_D . The goal is to detect whether $P_R(\mathbf{x}) = P_D(\mathbf{x})$.

Drift detection with prior detections (problem 2). We also introduce a new distinct semi-supervised setting for drift detection: In this setting, we assume that prior detections have already taken place. Specifically, we have access to prior training, reference, and detection data that has been used for prior drift detection. Furthermore, we also have access to the result of the drift detection. Mathematically, we say that given

$$\{\mathcal{S}_T^{(i)}, \mathcal{S}_R^{(i)}, \mathcal{S}_D^{(i)}, z^{(i)}\}_{i=1}^M \quad \text{where } i \in \{1, \dots, M\}, \quad (3)$$

where there are M prior drift detections and we used $\mathcal{S}_T^{(i)}, \mathcal{S}_R^{(i)}, \mathcal{S}_D^{(i)}$ for the i th drift detection to identify the outcome $z^{(i)} \in \{0, 1\}$, where 1 signifies that a drift has taken place and 0 signifies otherwise. With prior knowledge (3), the goal is to detect a drift given a new triplet $\{\mathcal{S}_T, \mathcal{S}_R, \mathcal{S}_D\}$ (as explained in the point above).

Remark. The motivation behind setup 2 is the scenario where a company has done many drift detections in the past using labeled data, which leads to high accuracy (which motivates the assumption of knowing $z^{(i)}$) of detection but is unsustainable. Therefore, at some point of time, they decide to switch to drift detection with unlabeled data. To the best of our knowledge, this framework has been applied to several scenarios [7] but not in concept drift detection.

Big data regime. We consider the big data regime where there is a huge volume of data (e.g., finance). In this regime, the runtime of a drift detection approach becomes very important. We seek an approach that allows us to control the false positive rate, but at the same time is statistically powerful and runtime efficient.

Notation. Throughout the paper, the function $\log(\cdot)$ has base e , and we make use of Bachmann-Landau asymptotic notation (i.e., O). $\mathbf{0}_n$ denotes a vector of zeros of length n , $\mathbf{1}_n$ denotes a vector of ones of length n , and \mathbf{I}_n denotes the $n \times n$ identity matrix.

B. Related Work

a) *Detection with labels:* Popular methods for detecting concept drift rely on updating models as new data becomes available and when changes in data are detected. These methods assume the availability of labeled data [2], [8]–[11] – we refer the interested reader to this survey [5].

b) *Detection without labels:* While the use of labeled data for retraining and updating models is largely unavoidable, its use for the purpose of drift detection might be impractical as labeling is time consuming, expensive and in some cases, not a possibility at all [12], [13]. Moreover the need for constant validation of the learned model leads to wasted labels, which are discarded when the model is found to be stable [14]. This has motivated the development of *unlabeled* or *label-less* drift detection techniques which monitor changes to the feature distribution, as an indicator of drift. To stick to the theme of non-online and whole-window drift detection without labels, we will review related work in this area below – we refer readers who are interested in other variants of the unsupervised drift detection to this survey [4].

The most relevant work is that of [15], which surveyed and analyzed the suitability of various distance measures for quantifying concept drift. Their key findings are:

- Wasserstein distance and maximum mean discrepancy (MMD) scale well to high feature dimensions, but are impractical for large volumes of data (i.e. large sample sizes) due to high runtime complexity that depends on the size of the data.
- Hellinger distance, Kullback–Leibler divergence, Kolmogorov–Smirnov distance, and total variation distance

can be numerically approximated for univariate data of any size. However, they do not scale up well to higher feature dimensions. Kolmogorov–Smirnov distance cannot be extended beyond bivariate case, and the rest (Hellinger distance, Kullback–Leibler divergence, and total variation distance) are limited by the exponential complexity of frequency/density calculation. Hellinger distance and Kullback–Leibler divergence have a closed-form solution for the multivariate Gaussian, but not for general multivariate distributions.

Overall, they recommended the Hellinger distance for drift detection in univariate or low-dimensional data. With the goal of improving efficiency for MMD, [16, Section 6] calculated MMD over pair of datapoints (instead of taking MMD between the two sets), then obtained their test statistic by taking the average of the MMD – this has asymptotic Gaussian null distribution via the central limit theorem. While this is very efficient, the test statistic has very high variance. To lower the variance, while still retaining an asymptotically Gaussian null distribution, [17] introduce the idea of batching, where the algorithm calculates the MMD over batches of datapoints, and the obtain the test statistic by averaging the MMD. [18] followed this up by showing that specifically for the MMD with permutation testing and sub-exponential data distributions, there exists a sample compression technique that is able to improve efficiency while also maintaining power. Our batching approach works for high-dimensional data and any distance measure (not just for MMD), makes no distribution assumption, and is runtime efficient. Furthermore, our method is fully unsupervised, model independent, task independent (allows for classification or regression).

Other related works on unsupervised drift detection include: [19] studied unlabeled drift detection for classification-type inference models, [20] studied drift detection with target class imbalance, [14] developed a drift detection that works with the SVM classifier and further improved it in [21] to work with an ensemble of classifiers, [22] focused on detecting drifts caused regional density changes, [23] studied unlabeled sequence anomaly in multi-dimensional sequences, and [24], [25] studied the use of Shapley values, which are obtained by processing the features, to detect and also explain drift detection.

C. Main Contributions

Our main contributions are as follows:

- We introduce a flexible and efficient framework (Algorithm 1) that uses the idea of batching for drift detection without labels – the flexible choice being the option to choose the statistical distance between batches of data. We rigorously analyze the runtime complexity and show that it is more efficient compared to its non-batching counter parts, i.e., doing permutation testing on the distances obtained from the entire window/set. Numerical simulations are conducted to show that batching is statistically more powerful than not batching, given a computational/runtime constraint.

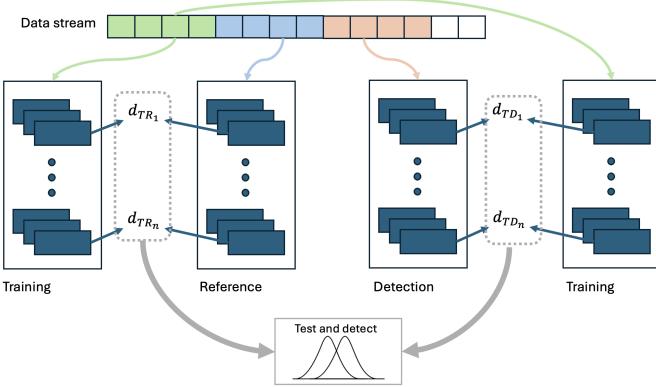


Fig. 1. Visualization of Algorithm 1.

- We introduce a distinct semi-supervised framework for the case where the analyst has access to information about prior detections (see Section I-A). To this end, we provide an algorithm (Algorithm 3) for this problem, that builds upon our batching approach, and analyze its performance via simulations.

II. ALGORITHM FOR PROBLEM 1

Our framework and algorithm is presented in Algorithm 1, and a visual representation is presented in Figure 1. Some remarks are due:

- **Non parametric and parallelizable.** Our algorithm does not make any distribution assumption on the data provided. Our algorithm can be parallelized by computing the distances between batches in parallel.
- **Distance metric.** Our algorithm is flexible in the sense that we can choose the distance metric to use – some examples are the Wasserstein-1 distance (also known as earth mover’s distance (EMD)), maximum mean discrepancy (MMD), and Kullback–Leibler (KL) divergence.
- **Paired test.** We use the paired t-test instead of the 2-sample t-test because the two samples $\{d_{TR_1}, \dots, d_{TR_n}\}$ and $\{d_{TD_1}, \dots, d_{TD_n}\}$ are dependent of each other, through the involvement of the training set \mathcal{S}_T in their calculation.
- **Use of training set.** One might point out that we could just compute batched distances between the (i) union of the training and reference sets, and (ii) the detection set, and then follow up with a one-sample t-test on the batched distances. This approach however is not as general as ours since certain distances, when computed, are always positive even for samples drawn from the same distribution (e.g., the computation of EMD [26]).

a) *Assumptions:* Our algorithm assumptions follow from those of the one-sample t-test.

- **Independence.** The differences in distances $d_{TR_1} - d_{TD_1}, \dots, d_{TR_n} - d_{TD_n}$ have to be independent of each other. This requires the datapoints in the batched triplets $(\mathcal{S}_{T_1}, \mathcal{S}_{R_1}, \mathcal{S}_{D_1}), \dots, (\mathcal{S}_{T_n}, \mathcal{S}_{R_n}, \mathcal{S}_{D_n})$ to be independent of each other.

Algorithm 1 Batched distance (BD) algorithm

```

input: training set  $\mathcal{S}_T$ , reference set  $\mathcal{S}_R$ , detection set  $\mathcal{S}_D$ , significant level  $\alpha$ 
Partition training set into  $n$  batches of size  $k$ , denoted as  $\{\mathcal{S}_{T_1}, \dots, \mathcal{S}_{T_n}\}$ .
Partition reference set into  $n$  batches of size  $k$ , denoted as  $\{\mathcal{S}_{R_1}, \dots, \mathcal{S}_{R_n}\}$ .
Partition detection set into  $n$  batches of size  $k$ , denoted as  $\{\mathcal{S}_{D_1}, \dots, \mathcal{S}_{D_n}\}$ .
Pair the training set batches with the reference window batches and the detection window batches in the following manner:  $(\mathcal{S}_{T_1}, \mathcal{S}_{R_1}), \dots, (\mathcal{S}_{T_n}, \mathcal{S}_{R_n})$  and  $(\mathcal{S}_{T_1}, \mathcal{S}_{D_1}), \dots, (\mathcal{S}_{T_n}, \mathcal{S}_{D_n})$ .
for each pair do
    Compute a distance metric between the samples, denoted as  $d_{TR_1} = \text{dist}(\mathcal{S}_{T_1}, \mathcal{S}_{R_1}), \dots, d_{TR_n} = \text{dist}(\mathcal{S}_{T_n}, \mathcal{S}_{R_n})$  and  $d_{TD_1} = \text{dist}(\mathcal{S}_{T_1}, \mathcal{S}_{D_1}), \dots, d_{TD_n} = \text{dist}(\mathcal{S}_{T_n}, \mathcal{S}_{D_n})$ 
end for
Conduct a paired two-tailed hypothesis t-test: Denote  $\mu$  as the true mean of  $d_{TR_1} - d_{TD_1}, \dots, d_{TR_n} - d_{TD_n}$ . Then we have:  $H_0 : \mu = 0$ ,  $H_1 : \mu \neq 0$ .
Drift is detected if the p-value of the t-test is below  $\alpha$ .
```

- **Normality.** The difference of distance metrics $\{d_{TR_i} - d_{TD_i}\}_{i=1}^n$ have to follow the Gaussian distribution. Due to the robustness of the t-test, this does not matter too much when the number of batches n is large since the mean of differences of distances approaches the Gaussian distribution via the central limit theorem.
- **Equally sized windows.** Requiring n batches of size k from \mathcal{S}_T , \mathcal{S}_R , and \mathcal{S}_D imposes a constraint that all sets need to be of equal sizes. This is not particularly an issue since we can always generalize the algorithm to account for unequal sizes, this can be done by keeping the number of batches as n and dropping the requirement for the batch size k . This results in paired batches of unequal sizes – if the distance measures can still be calculated, then our algorithm would still work.

b) *Runtime complexity:* Given two samples of size N and feature dimension m , let us define the runtime complexity of computing the distance metric as $d(N, m)$. By observing the n iterations of distance computation in Algorithm 1, we have the following runtime result.

Lemma 2.1: The batched distance algorithm in Algorithm 1 has a runtime complexity of $O(n \cdot d(k, m))$, whereas a permutation tests with B permutations (and without batching) gives a runtime complexity of $O(B \cdot d(nk, m))$.

Essentially, as long as $d(N, m)$ is linear in m and polynomial in N , then batching will always have a lower runtime complexity compared to permutation testing without batching. We illustrate through some distance metric choices how batching reduces the runtime complexity of drift detection.

Choice of distances and motivation. We have selected 3 distance metrics: (i) Earth mover’s distance, (ii) Maximum

mean discrepancy, and (iii) Kullback-Leibler divergence. We have included the earth mover's distance and maximum mean discrepancy because they are known to work better for higher dimensions [15]. Note that our list of distance measures is non-exhaustive – see [15] for a survey of distance measures that can be used for drift detection.

Earth mover's distance. More generally, the integral probability metric between two distributions P and Q is defined as

$$d_{\mathcal{F}}(P, Q) = \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{X \sim P}[f(X)] - \mathbb{E}_{Y \sim Q}[f(Y)] \right|, \quad (4)$$

where \mathcal{F} is a class of functions to be chosen and $f : \mathbb{R}^m \rightarrow \mathbb{R}$. Choosing \mathcal{F} to be the class of 1-Lipschitz functions, i.e.,

$$\mathcal{F} = \{f : f \text{ continuous, } |f(\mathbf{x}) - f(\mathbf{y})| \leq \|\mathbf{x} - \mathbf{y}\|\},$$

where $\|\cdot\|$ is a norm, gives us the EMD. The EMD can be estimated by viewing it as an optimal transport problem and the solving it with linear programming [26].

Lemma 2.2: Using the batched distance algorithm in Algorithm 1, the runtime complexity of using the earth mover's distance (EMD) drops from $O(B(n^3k^3 + mn^2k^2))$ to $O(n(k^3 + mk^2))$, where B is the number of iterations in permutation testing.

Proof: It is known that for 2 samples of size N , EMD can be solved in $d(N, m) = O(N^3 + mN^2)$ time [26].

- Without batching, computing the p-value for EMD via permutation testing would require permutation which requires B additional iterations for each distance computed. As a result, computing the p-value for EMD via permutation testing would take $O(B(N^3 + mN^2)) = O(B(n^3k^3 + mn^2k^2))$, where we set $N = nk$.
- For the BD algorithm, setting $d(k, m) = O(k^3 \log k)$ in Lemma 2.1 gives us the runtime complexity.

□

Maximum mean discrepancy. Choosing \mathcal{F} in (4) to be the unit ball in a reproducing kernel Hilbert space gives us the MMD. We choose our kernel function to be the radial basis function kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp \left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{2\sigma^2} \right).$$

Given a sample size of n_* , the MMD can be estimated by the U-statistic [16]:

$$\begin{aligned} & \frac{1}{n_*(n_* - 1)} \sum_{i=1}^n \sum_{j \neq i} k(\mathbf{x}_i, \mathbf{x}_j) \\ & + \frac{1}{n_*(n_* - 1)} \sum_{i=1}^n \sum_{j \neq i} k(\mathbf{y}_i, \mathbf{y}_j) \\ & - \frac{2}{n_*^2} \sum_{i=1}^n \sum_{j=1}^n k(\mathbf{x}_i, \mathbf{y}_j). \end{aligned} \quad (5)$$

Lemma 2.3: Using the batched distance algorithm in Algorithm 1, the runtime complexity of using the maximum mean discrepancy (MMD) drops from $O(Bmn^2k^2)$ to $O(nmk^2)$, where B is the number of iterations in permutation testing.

Proof: It is known that for 2 samples of size N , MMD can be computed in $d(N, m) = O(mN^2)$ time – this can be derived by observing (5).

- Without batching, computing the p-value for MMD via permutation testing would take $O(Bmn^2k^2) = O(Bmn^2k^2)$, where we set $N = nk$.
- For the BD algorithm, setting $d(k, m) = O(mk^2)$ in Lemma 2.1 gives us the runtime complexity.

□

Kullback-Leibler divergence. The KL divergence between two distributions P and Q is defined as

$$D_{KL}(P \| Q) = \mathbb{E}_{X \sim P} \left[-\log \frac{P(X)}{Q(X)} \right]. \quad (6)$$

KL divergence can be computed by using k -nearest-neighbour density estimation as an intermediate step [27].

Lemma 2.4: Using the batched distance algorithm in Algorithm 1, the runtime complexity of using the maximum mean discrepancy (MMD) drops from $O(Bmn^2k^2)$ to $O(nmk^2)$, where B is the number of iterations in permutation testing.

Proof: It is known that for 2 samples of size N , KL divergence can be computed in $d(N, m) = O(mN^2)$ time [27].

- Without batching, computing the p-value for MMD via permutation testing would take $O(Bmn^2k^2) = O(Bmn^2k^2)$, where we set $N = nk$.
- For the BD algorithm, we have $d(k, m) = O(mk^2)$ in Lemma 2.1 gives us the runtime complexity.

□

III. EXPERIMENTS FOR PROBLEM 1

The code for all the experiments in this section is presented in [28].

A. Comparing batch number and size

We study how varying k/n affects the performance of the batched distance algorithm. This study this via numerical simulations instead of analyzing the statistical power analytically because the analytical approach is not feasible – namely it is difficult to characterize the formula for the variance of estimated distances (and this variance is required for a formula-type analysis of the power).

We fix the total size $nk = 5000$ and vary the ratio

$$\frac{k}{n} = \left\{ \frac{5}{1000}, \frac{25}{200}, \frac{50}{100}, \frac{100}{50}, \frac{200}{25}, \frac{250}{20}, \frac{500}{10} \right\}.$$

For each ratio, we run 100 simulations to calculate the FPR and FNR. We set $\alpha = 0.05$, $m = 100$, and sample the training set and reference set i.i.d. from $\mathcal{N}(\mathbf{0}_m, \mathbf{I}_m)$. We have chosen drifts that are difficult for the algorithm to detect, so that the changes will be more pronounced. We look at a few situations:

- No drift.** The detection window is sampled i.i.d. from $\mathcal{N}(\mathbf{0}_m, \mathbf{I}_m)$. Results are shown in Figure 2(a) – there is no clear relationship between FPR and k/n .
- Mean drift in all feature dimensions.** The detection window is sampled i.i.d. from $\mathcal{N}(0.03 \cdot \mathbf{1}_m, \mathbf{I}_m)$. Results

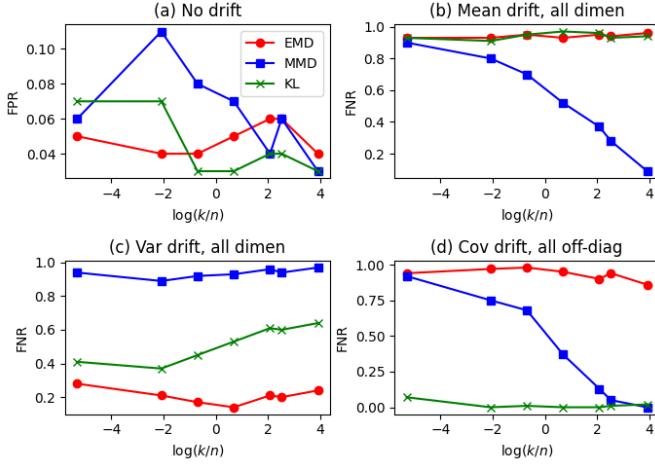


Fig. 2. Plots of how the ratio k/n affects the BD algorithm.

are shown in Figure 2(b) – for MMD, FNR decreases and k/n increases. There is no clear relationship between FNR and k/n for EMD and KL.

- 3) **Variance drift in all feature dimensions.** The detection window is sampled i.i.d. from $\mathcal{N}(\mathbf{1}_m, 1.01 \cdot \mathbf{I}_m)$. Results are shown in Figure 2(c) – for both EMD and KL, as k/n is increased, FNR first drops and then increases. There is no clear relationship between FNR and k/n for MMD.
- 4) **Covariance drift in all feature dimensions.** The detection window is sampled i.i.d. from $\mathcal{N}(\mathbf{1}_m, \Sigma_m)$. The diagonal entries of Σ_m are all ones, and the off-diagonals are all 0.07. Results are shown in Figure 2(d) – for MMD, FNR decreases and k/n increases. There is no clear relationship between FNR and k/n for EMD and KL.

Overall, MMD works better with a higher k/n whereas EMD and KL work better with a lower k/n (around 0.5 to 2). We also point out that increasing k/n makes the BD algorithm run slower, and therefore recommend k/n to be around 1 to 2.

B. Comparing with other approaches

We consider EMD, MMD, and KL divergence, and compare their performances on synthetic datasets with other approaches, namely Kolmogorov–Smirnov test with Bonferroni correction (KS-BC), and permutation testing. Note that EMD, MMD, and KL divergence are estimated using the algorithms that we referenced in Lemma 2.2, Lemma 2.3, and Lemma 2.4. We introduce other approaches that allow us to control the FPR:

- 1) **Kolmogorov–Smirnov test with Bonferroni correction [29, Chapter 13.3].** This approach uses multiple univariate testing, with a test for each dimension. For each dimension i , we test, using the Kolmogorov–Smirnov test, if the dataset from the i th dimension of $\mathcal{S}_T \cup \mathcal{S}_R$ is equivalent to the dataset from

the i th dimension of \mathcal{S}_D . If the p-value at any dimension is less than the significant level α , then the algorithms detects a drift. We have chosen Bonferroni correction due to its simplicity and conservative nature – there are other correction methods [29, Chapter 13] that an analyst could consider.

- 2) **Permutation testing [29, Chapter 13.5].** This approach uses multivariate testing on all dimensions. This approach computes the chosen distance (e.g., EMD, MMD, KL) between $\mathcal{S}_T \cup \mathcal{S}_R$ and \mathcal{S}_D . The p-value is then obtained by carrying out a permutation test with $B = 100$ permutations – see Algorithm 2.

Algorithm 2 Permutation testing

input: training set \mathcal{S}_T , reference set \mathcal{S}_R , detection set \mathcal{S}_D , significant level α
 Compute a distance metric $d = \text{dist}(\mathcal{S}_T \cup \mathcal{S}_R, \mathcal{S}_D)$.
for iteration $b \in \{1, \dots, B\}$ **do**
 Permute $\mathcal{S}_T \cup \mathcal{S}_R \cup \mathcal{S}_D$ at random. Call the first $2nk$ permuted observations $\mathcal{S}_T^* \cup \mathcal{S}_R^*$ and call the remaining nk observations \mathcal{S}_D^* .
 Compute $d_b^* = \text{dist}(\mathcal{S}_T^* \cup \mathcal{S}_R^*, \mathcal{S}_D^*)$.
end for
 The p-value is given by $\frac{1}{B} \sum_{b=1}^B \mathbb{1}\{|d_b^*| \geq |d|\}$.
 Drift is detected if the p-value is below α .

Runtime of KS-BC. For completeness, we provide the runtime complexity of Kolmogorov–Smirnov test with Bonferroni correction (KS + BC), which is $O(nk \log(nk))$. To see why, note that computing the KS test statistic for data of size $O(nk)$ requires sorting all $O(nk)$ observations and therefore takes $O(nk \log(nk))$ time [30]. Since we have m dimensions, the total runtime is $O(mnk \log(nk))$.

Experiment details. We study the following approaches: (i) batch distance algorithm with EMD, MMD, and KL (denoted as EMD-BD, MMD-BD, and KL-BD for short), (ii) permutation testing with EMD, MMD, and KL (denoted as EMD-PT, MMD-PT, and KL-PT for short), and (iii) KS-BC. Recall that we our focus is the big data regime, where we have limited time (i.e., computational resource) but large volumes of data. Therefore, in our experiments, we constrain the total number of iteration for each approach to around 10^8 , and restrict the number of feature dimensions to $m = 100$ and the number of permutations to $B = 100$. The choices of the remaining parameters are shown in Table I.

We run 100 simulations to calculate the false positive rate (FPR) and the false negative rate (FNR) for every case considered. In each simulation, we use $\alpha = 0.05$. Furthermore, the training set and reference set are sampled i.i.d. from $\mathcal{N}(\mathbf{0}_m, \mathbf{I}_m)$. We look at a few scenarios:

- 1) **No drift.** The detection window is sampled i.i.d. from $\mathcal{N}(\mathbf{0}_m, \mathbf{I}_m)$. The FNR will always be 0, therefore we will only compute the FPR.
- 2) **Mean drift in all feature dimensions.** The detection window is sampled i.i.d. from $\mathcal{N}(\zeta \cdot \mathbf{1}_m, \mathbf{I}_m)$. We vary

TABLE I
PARAMETER CHOICES FOR THE DIFFERENT APPROACHES SUCH THAT THE TOTAL NUMBER OF ITERATIONS IS AROUND 10^8 .

Approach	Complexity	Parameters
EMD-BD	$O(n(k^3 + mk^2))$	$n = 50, k = 100$
EMD-PT	$O(B(n^3k^3 + mn^2k^2))$	$nk = 76$
MMD-BD	$O(mnk^2)$	$n = 100, k = 100$
MMD-PT	$O(Bmn^2k^2)$	$nk = 100$
KL-BD	$O(mnk^2)$	$n = 100, k = 100$
KL-PT	$O(Bmn^2k^2)$	$nk = 100$
KS-BC	$O(mnk \log(nk))$	$nk = 87, 900$

ζ from the set $\{0.01, 0.02, 0.03, 0.04\}$. The FPR will always be 0, therefore we will only compute the FNR.

- 3) **Variance drift in all feature dimensions.** The detection window is sampled i.i.d. from $\mathcal{N}(\mathbf{1}_m, \zeta \cdot \mathbf{I}_m)$. We vary ζ from the set $\{1.005, 1.01, 1.05, 1.10\}$. The FPR will always be 0, therefore we will only compute the FNR.
- 4) **Covariance drift in all feature dimensions.** The detection window is sampled i.i.d. from $\mathcal{N}(\mathbf{1}_m, \Sigma_m)$. The diagonal entries of Σ_m are all ones, and the off-diagonals are all ζ . We vary ζ from the set $\{0.05, 0.06, 0.07, 0.08\}$. The FPR will always be 0, therefore we will only compute the FNR.

The experiment results are displayed in Table II. To supplement the numerical results, we also provide visual plots for the results of mean drift in all feature dimensions (scenario 2) in Figure 3(a), the results of variance drift in all feature dimensions (scenario 3) in Figure 3(b), and the results of Covariance drift in all feature dimensions (scenario 4) in Figure 3(c). We make the following observations:

- 1) **No drift.** From Table II, we above that the FPRs are around 0.05, indicating that it is well controlled.
- 2) **Mean drift in all feature dimensions.** From Table II and Figure 3(a), we observe that KS-BC and MMD-BD both performed well, with MMD-BD significantly outperforming MMD-PT. The other approaches are less powerful, performing similarly.
- 3) **Variance drift in all feature dimensions.** From Table II and Figure 3(b), we observe that EMD-BD and KL-BD performed the best, with EMD-BD, MMD-BD, and KL-BD outperforming EMD-PT, MMD-PT, and KL-PT.
- 4) **Covariance drift in all feature dimensions.** From Table II and Figure 3(c), we observe that KL-BD performed the best, with KL-BD and MMD-BD significantly outperforming KL-PT and MMD-PT. In this setting EMD-PT outperformed EMD-BD. Also, as expected, KS-BC is unable to detect drifts between features.

Overall observation. Each distance performed well for a different type of drift, and this implies the following recommendations: If mean drift is suspected, then using MMD-BD or KS-BC might be a viable option. If variance drift is suspected, then using EMD-BD or KL-BD might be a viable option. If covariance drift is suspected, then using KL-BD might be a viable option. Overall, whenever a distance

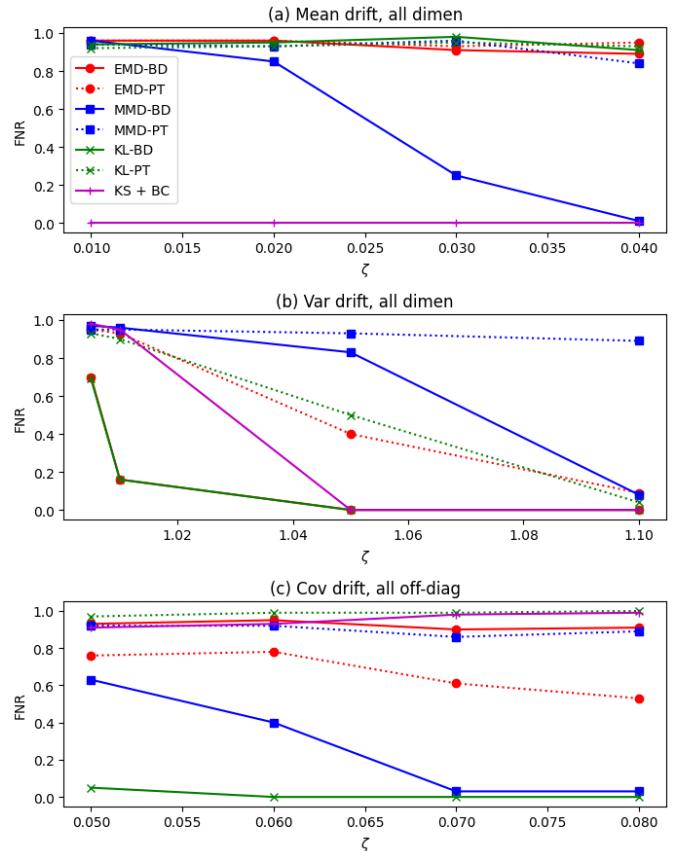


Fig. 3. Plots comparing the different approaches.

is effective at detecting a certain drift (i.e., mean, variance, or covariance), then using that particular distance with the batched distance algorithm outperforms permutation testing, validating the superiority of the batched distance algorithm.

Why batching helps. When we have a runtime budget, batching leads to better statistical power as compared to permutation testing. This is because in order to meet the runtime budget, permutation testing has to reduce the size of its input datasets. The BD algorithm, being more efficient, can use a significantly larger input dataset to meet the same runtime budget. The ‘gain’ in dataset size for the BD algorithm leads to a better statistical power.

Since each distance has its own strengths, one might wonder whether we could do better by combining the outputs of different approaches. This spurs us in the direction of combining the outputs of several approaches to conduct drift detection, which leads us into the next part of the paper.

IV. ALGORITHMS FOR PROBLEM 2

Recall that under this setting, we have access to prior detections (see Section I-A). We do not use permutation testing to generate the p-values as they are impractical in the big data regime, especially in this setting (problem 2), since we are dealing with significantly more datasets (i.e., training, reference, and detection sets) compared to the previous setting

TABLE II
EXPERIMENT RESULTS FOR VARIOUS METHODS FOR DRIFT DETECTION WITHOUT LABELS.

Approach	No drift (FPR)	Mean drift with varying ζ (FNR)				Var drift with varying ζ (FNR)				Cov drift with varying ζ (FNR)			
		0.01	0.02	0.03	0.04	1.005	1.01	1.05	1.10	0.05	0.06	0.07	0.08
EMD-BD	0.04	0.96	0.96	0.91	0.89	0.7	0.16	0	0	0.93	0.95	0.9	0.91
EMD-PT	0.04	0.96	0.96	0.93	0.95	0.95	0.93	0.4	0.09	0.76	0.78	0.61	0.53
MMD-BD	0.03	0.96	0.85	0.25	0.01	0.97	0.96	0.83	0.08	0.63	0.4	0.03	0.03
MMD-PT	0.04	0.94	0.93	0.96	0.84	0.95	0.95	0.93	0.89	0.92	0.92	0.86	0.89
KL-BD	0.06	0.94	0.95	0.98	0.91	0.69	0.16	0	0	0.05	0	0	0
KL-PT	0.04	0.92	0.93	0.95	0.93	0.93	0.9	0.5	0.04	0.97	0.99	0.99	1
KS-BC	0.05	0	0	0	0	0.98	0.95	0	0	0.91	0.93	0.98	0.99

Algorithm 3 Classifier algorithm

input: training set \mathcal{S}_T , reference set \mathcal{S}_R , detection set \mathcal{S}_D , prior detections $\{\mathcal{S}_T^{(i)}, \mathcal{S}_R^{(i)}, \mathcal{S}_D^{(i)}, z^{(i)}\}_{i=1}^M$ where $i \in \{1, \dots, M\}$, batch number n and batch size k for BD algorithm, choice of classifier, and classifier threshold ξ .

Initialize the classifier training set \mathcal{C} and output type (p-values or test statistics).

for $i \in \{1, \dots, M\}$ **do**

 Compute outputs $p_1^{(i)}, p_2^{(i)}, p_3^{(i)}$ using BD algorithm with EMD (for $p_1^{(i)}$), MMD (for $p_2^{(i)}$), KL (for $p_3^{(i)}$) on $\{\mathcal{S}_T^{(i)}, \mathcal{S}_R^{(i)}, \mathcal{S}_D^{(i)}\}$.

 Compute output $p_4^{(i)}$ using KS + BC on $\{\mathcal{S}_T^{(i)}, \mathcal{S}_R^{(i)}, \mathcal{S}_D^{(i)}\}$, where we output the minimum p-values amongst all the dimensions.

 Add $\{p_1^{(i)}, p_2^{(i)}, p_3^{(i)}, p_4^{(i)}, z^{(i)}\}$ to \mathcal{C} .

end for

Standardize \mathcal{C} to give $\tilde{\mathcal{C}}$.

Train the chosen classifier on $\tilde{\mathcal{C}}$.

Compute outputs (p-values or test statistics) p_1, p_2, p_3 using BD algorithm with EMD (for p_1), MMD (for p_2), KL (for p_3) on $\{\mathcal{S}_T, \mathcal{S}_R, \mathcal{S}_D\}$.

Compute output (p-value or test statistics) p_4 using KS + BC on $\{\mathcal{S}_T, \mathcal{S}_R, \mathcal{S}_D\}$, where we choose the minimum output amongst all the dimensions.

Standardize $\{p_1, p_2, p_3, p_4\}$ and feed it into the trained classifier to obtain the classifier probability p_c .

Drift is detected if $p_c \leq \xi$.

(problem 1). We first consider two ways to threshold the combined p-values:

- Taking average.** The most obvious and naive approach is to take the average of the p-values of the different approaches. We then threshold the average p-value to make the decision, i.e., declare drift if the average p-value is below the significant level α , and declare no drift otherwise.
- Perceptron learning** A slightly more sophisticated way is to use perceptron learning to learn the linear combination of the p-values before thresholding. Specifically, the perceptron algorithm is used to learn the weights $\mathbf{w} \in \mathbb{R}^4$

from the p-values $\mathbf{p} \in \mathbb{R}^4$ (we have 4 p-values) such that

$$z = \mathbb{1}\{\mathbf{w}^\top \mathbf{p} + \alpha > 0\} = \begin{cases} 1 & \text{if } -\mathbf{w}^\top \mathbf{p} < \alpha \\ 0 & \text{if } -\mathbf{w}^\top \mathbf{p} \geq \alpha. \end{cases}$$

Note that α here is fixed, and would not be learnt by the algorithm.

A more sophisticated way is to consider the general classifier algorithm presented in Algorithm 3, where we can choose whether to use p-values or test statistics as features. We do not use permutation testing to generate the outputs as they are impractical in the big data regime. The main question is to determine if we can do better given that we have knowledge about prior detections and are allowed to combine the outputs of different approaches. We use the following choice of classifiers:

- Logistic regression (LR).** The algorithm predicts the probability of a binary outcome using a linear combination of input features passed through a sigmoid function. The probability of a class can be predicted by calculating the sigmoid of a weighted sum of the input features.
- K-nearest neighbour (kNN).** Classifies or predicts data points based on the majority class or average value of its 'k' nearest neighbors in the feature space. The probability of a class can be predicted by the proportion of neighbors belonging to that class.
- Multi-layer perceptron (MLP).** A feedforward neural network with multiple layers that learns complex patterns by adjusting the weights of connections between neurons. The probability of a class can be predicted by applying the softmax function on the output layer's activations.

The FPR can be controlled by adjusting the probability threshold used in each of the algorithms presented above.

V. EXPERIMENTS FOR PROBLEM 2

The code for all the experiments in this section is presented in [28]. We first generate our training set and then generate our testing set before testing our classifier on the testing set.

Training data generation. We set $n = 50$, $k = 100$, $m = 100$, and the training set and reference set are sampled i.i.d. from $\mathcal{N}(0_m, I_m)$. We look at the following scenarios (which are exactly the same as those used in Section III-B):

- No drift.** The detection set is sampled i.i.d. from $\mathcal{N}(0_m, I_m)$.

- 2) **Mean drift in all feature dimensions.** The detection window is sampled i.i.d. from $\mathcal{N}(\zeta \cdot \mathbf{1}_m, \mathbf{I}_m)$. We vary ζ from the set $\{0.01, 0.02, 0.03, 0.04\}$.
- 3) **Variance drift in all feature dimensions.** The detection window is sampled i.i.d. from $\mathcal{N}(\mathbf{1}_m, \zeta \cdot \mathbf{I}_m)$. We vary ζ from the set $\{1.005, 1.01, 1.05, 1.10\}$.
- 4) **Covariance drift in all feature dimensions.** The detection window is sampled i.i.d. from $\mathcal{N}(\mathbf{1}_m, \Sigma_m)$. The diagonal entries of Σ_m are all ones, and the off-diagonals are all ζ . We vary ζ from the set $\{0.05, 0.06, 0.07, 0.08\}$.

Training data generation. We sample 50 sets of $\{\mathcal{S}_T^{(i)}, \mathcal{S}_R^{(i)}, \mathcal{S}_D^{(i)}\}$ from scenario 1 with $z^{(i)} = 0$, 40 sets of $\{\mathcal{S}_T^{(i)}, \mathcal{S}_R^{(i)}, \mathcal{S}_D^{(i)}\}$ from scenario 2 with $z^{(i)} = 1$ with 10 sets for each ζ , 40 sets of $\{\mathcal{S}_T^{(i)}, \mathcal{S}_R^{(i)}, \mathcal{S}_D^{(i)}\}$ from scenario 3 with $z^{(i)} = 1$ with 10 sets for each ζ , 40 sets of $\{\mathcal{S}_T^{(i)}, \mathcal{S}_R^{(i)}, \mathcal{S}_D^{(i)}\}$ from scenario 4 with $z^{(i)} = 1$ with 10 sets for each ζ .

Testing data generation. We run 100 simulations for each scenario to calculate the FPR and the FNR for every case considered – the cases are exactly the same as those presented in the training data generation (see above). In each simulation, we use $n = 50$, $k = 100$, and $\alpha = 0.05$.

Classifier details. We use the following parameters for each of our classifiers.

- **Logistic regression (LR).** We use threshold $\xi = 0.8$ in Algorithm 3.
- **K-nearest neighbour (kNN).** We use threshold $\xi = 0.85$ in Algorithm 3, and set the number of nearest neighbour to 10.
- **Multi-layer perceptron (MLP).** We use threshold $\xi = 0.8$ in Algorithm 3, 2 hidden layers of sizes 4 and 2, and the ReLU activation function.

The thresholds ξ are chosen to control the FPR to be around 0.05. We use the following shorthands in our experiment results:

- AVG for the average p-value approach, and PL for perceptron learning.
- LR-p, kNN-p, MLP-p for logistic regression, k-nearest neighbour, and multi-layer perceptron, respectively, in Algorithm 3 using p-values as features.
- LR-s, kNN-s, MLP-s for logistic regression, k-nearest neighbour, and multi-layer perceptron, respectively, in Algorithm 3 using test statistics as features.

FPR and FNR. The experimental results for FPR and FNR are shown in Table III. To supplement the numerical results, we provide visual plots for the classifier approach in Figure 4. We make the following observations.

- 1) **No drift.** As expected, the FPR is kept low around 0.05, since we have adjusted the classifier's predicted probability threshold so that the FPR is comparable to those of previous experiments in problem 1.
- 2) **Mean drift in all feature dimensions.** As expected, the more sophisticated approaches LR-s and MLP-s, along with KS-BC, performed the best. This shows that KS-BC is very suitable for mean drift detection and we cannot do significantly better with prior detection information.

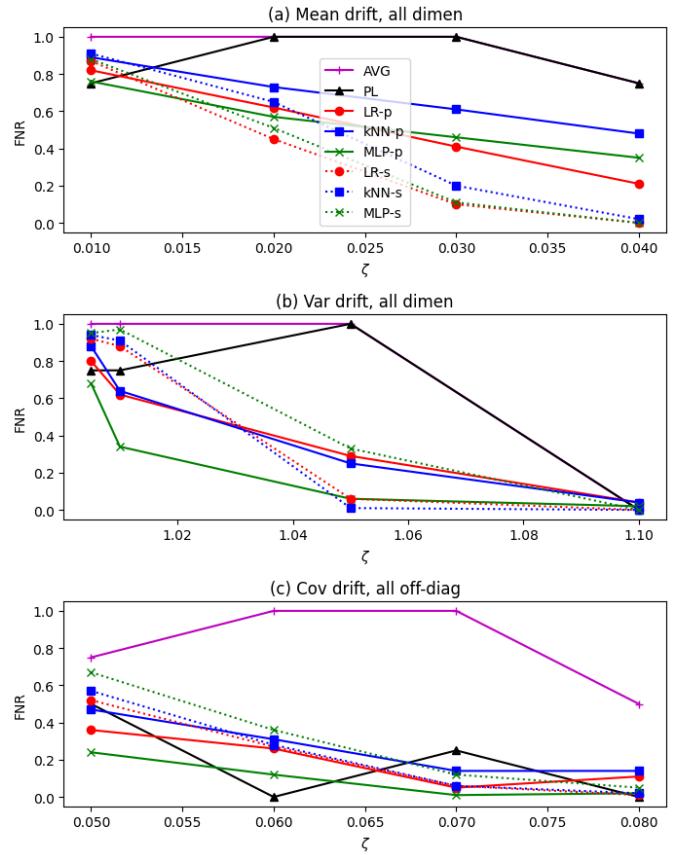


Fig. 4. Plots comparing the different approaches that utilize prior detection information.

- 3) **Variance drift in all feature dimensions.** Overall, the most sophisticated approach, MLP-p, performed the best.
- 4) **Covariance drift, all off-diagonals.** Overall, the most sophisticated approach, MLP-p, performed the best.

Accuracy. The accuracy results for the various detection approaches are as follows are presented in Table IV. We observe that MLP-p is the best when p-values are used as features, and LR-s is the best when test statistics are used as features.

As expected, approaches that use prior detection information generally perform better than those that do not use prior detection information. Furthermore, the most sophisticated classifier (MLP) performed the best in terms of accuracy.

Remark. We admit that the classifier-based fusion of p-values and test statistics may be prone to overfitting. Hence, it would be interesting to design better approaches for this semi-supervised setting – we leave this as a future direction.

VI. CONCLUSION

We considered drift detection without labels in the big data regime where there is a huge volume of data, but there is a constraint on the computational resources (i.e., runtime budget). In this regime, we introduce the batched distance algorithm and show experimentally that it outperforms permutation testing

TABLE III
EXPERIMENT RESULTS FOR VARIOUS METHODS FOR DRIFT DETECTION WITH PRIOR DETECTIONS.

Approach	No drift (FPR)	Mean drift with varying ζ (FNR)				Var drift with varying ζ (FNR)				Cov drift with varying ζ (FNR)			
		0.01	0.02	0.03	0.04	1.005	1.01	1.05	1.10	0.05	0.06	0.07	0.08
AVG	0	1	1	1	0.75	1	1	1	0	0.75	1	1	0.5
PL	0.02	0.75	1	1	0.75	0.75	0.75	1	0	0.5	0	0.25	0
LR-p	0.09	0.82	0.62	0.41	0.21	0.8	0.62	0.29	0.04	0.36	0.26	0.05	0.11
kNN-p	0.06	0.89	0.73	0.61	0.48	0.88	0.64	0.25	0.04	0.47	0.31	0.14	0.14
MLP-p	0.09	0.76	0.57	0.46	0.35	0.68	0.34	0.06	0.02	0.24	0.12	0.01	0.02
LR-s	0.05	0.87	0.45	0.1	0	0.92	0.88	0.06	0	0.52	0.27	0.06	0.01
kNN-s	0.07	0.91	0.65	0.2	0.02	0.94	0.91	0.01	0	0.57	0.28	0.06	0.02
MLP-s	0.06	0.88	0.51	0.11	0	0.95	0.97	0.33	0	0.67	0.36	0.12	0.05
EMD-BD	0.07	0.93	0.93	0.93	0.97	0.69	0.16	0	0	0.94	0.94	0.95	0.94
MMD-BD	0.07	0.92	0.88	0.52	0.17	0.93	0.93	0.85	0.4	0.79	0.55	0.37	0.23
KL-BD	0.04	0.96	0.98	0.97	0.93	0.82	0.46	0	0	0.3	0.06	0	0
KS-BC	0.06	0.91	0.48	0.13	0	0.94	0.98	0.88	0.53	0.95	0.95	0.95	0.95

TABLE IV
ACCURACY RESULTS FOR THE VARIOUS DETECTION APPROACHES.

Approach	Accuracy
AVG	23.1%
PL	47.9%
LR-p	64.0%
kNN-p	56.6%
MLP-p	71.4%
LR-s	67.8%
kNN-s	64.3%
MLP-s	61.5%
EMD-BD	35.0%
MMD-BD	41.5%
KL-BD	57.5%
KS-BC	33.0%

in terms of statistical power, while also keeping the false positive rate low. Furthermore, we introduced a new semi-supervised drift detection framework to model the scenario of detecting drift (without labels) given prior detections. Under this setting we showed that our drift detection algorithm can be incorporated effectively into this framework.

REFERENCES

- [1] I. Žliobaitė, “Change with delayed labeling: When is it detectable?” *2010 IEEE International Conference on Data Mining Workshops*, pp. 843–850, 2010.
- [2] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and H. Bouchachia, “A survey on concept drift adaptation,” *ACM Computing Surveys (CSUR)*, vol. 46, 04 2014.
- [3] H. Wang and Z. Abraham, “Concept drift detection for streaming data,” *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, 2015.
- [4] R. N. Gemaque, A. F. J. Costa, R. Giusti, and E. M. Dos Santos, “An overview of unsupervised drift detection methods,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, p. e1381, 2020.
- [5] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, “Learning under concept drift: A review,” *IEEE transactions on knowledge and data engineering*, vol. 31, pp. 2346–2363, 2018.
- [6] T. R. Hoens, R. Polikar, and N. V. Chawla, “Learning from streaming data with concept drift and imbalance: an overview,” *Progress in Artificial Intelligence*, vol. 1, pp. 89–101, 2012.
- [7] D. Lopez-Paz and M. Oquab, “Revisiting classifier two-sample tests,” *International Conference on Learning Representations*, 2017.
- [8] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with drift detection,” *Intelligent Data Analysis*, vol. 8, pp. 286–295, 2004.
- [9] A. Bifet and R. Gavaldà, “Learning from time-changing data with adaptive windowing,” *Proceedings of the 7th SIAM International Conference on Data Mining*, vol. 7, 2007.
- [10] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, “Early drift detection method,” in *Fourth international workshop on knowledge discovery from data streams*, vol. 6, 2006, pp. 77–86.
- [11] P. Gonçalves Jr, S. Santos, R. Barros, and D. Vieira, “A comparative study on concept drift detectors,” *Expert Systems with Applications*, vol. 41, pp. 8144–8156, 12 2014.
- [12] E. Lugofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer, “Recognizing input space and target concept drifts in data streams with scarcely labelled and unlabelled instances,” *Information Sciences*, vol. 355, 03 2016.
- [13] G. Krempel, I. Zliobaite, D. Brzezinski, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowski, “Open challenges for data stream mining research,” *ACM SIGKDD Explorations Newsletter*, 07 2014.
- [14] T. Sethi and M. Kantardzic, “Don’t pay for validation: Detecting drifts from unlabeled data using margin density,” *Procedia Computer Science*, vol. 53, pp. 103–112, 12 2015.
- [15] I. Goldenberg and G. I. Webb, “Survey of distance measures for quantifying concept drift and shift in numeric data,” *Knowledge and Information Systems*, vol. 60, no. 2, pp. 591–615, 2019.
- [16] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” *The Journal of Machine Learning Research*, vol. 13, pp. 723–773, 2012.
- [17] W. Zaremba, A. Gretton, and M. Blaschko, “B-test: A non-parametric, low variance kernel two-sample test,” *Advances in neural information processing systems*, vol. 26, 2013.
- [18] C. Domingo-Enrich, R. Dwivedi, and L. Mackey, “Compress then test: Powerful kernel testing in near-linear time,” *International Conference on Artificial Intelligence and Statistics*, vol. 206, pp. 1174–1218, 2023.
- [19] S. A. Bashir, A. Petrovski, and D. Doolan, “A framework for unsupervised change detection in activity recognition,” *International journal of pervasive computing and communications*, vol. 13, no. 2, pp. 157–175, 2017.
- [20] A. Maletzke, D. Reis, E. Cherman, and G. Batista, “On the need of class ratio insensitive drift tests for data streams,” *International workshop on learning with imbalanced domains: theory and applications*, pp. 110–124, 2018.
- [21] T. S. Sethi and M. Kantardzic, “On the reliable detection of concept drift from streaming unlabeled data,” *Expert Systems with Applications*, vol. 82, pp. 77–99, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417417302439>
- [22] A. Liu, J. Lu, F. Liu, and G. Zhang, “Accumulating regional density dissimilarity for concept drift detection in data streams,” *Pattern Recognition*, vol. 76, pp. 256–272, 2018.
- [23] B. Li, Y.-j. Wang, D.-s. Yang, Y.-m. Li, and X.-k. Ma, “FAAD: an unsupervised fast and accurate anomaly detection method for a multi-dimensional sequence over data stream,” *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 3, pp. 388–404, 2019.

- [24] S. Zheng, S. Zon, M. Pechenizkiy, C. Campos, W. Ipenburg, and H. Harder, "Labelless concept drift detection and explanation," *NeurIPS 2019 Workshop on Robust AI in Financial Services: Data, Fairness, Explainability, Trustworthiness, and Privacy*, 2019.
- [25] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [26] G. Peyré and M. Cuturi, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.
- [27] F. Pérez-Cruz, "Kullback-leibler divergence estimation of continuous distributions," *IEEE international symposium on information theory*, pp. 1666–1670, 2008.
- [28] N. Tan, "Code for flexible and efficient drift detection without labels," https://github.com/nelvintan/drift_detect, 2025.
- [29] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, "An introduction to statistical learning," *Springer*, vol. 112, no. 1, 2013.
- [30] T. Gonzalez, S. Sahni, and W. R. Franta, "An efficient algorithm for the kolmogorov-smirnov and lilliefors tests," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 1, pp. 60–64, 1977.