Concept Drift Detection and Adaptation with Hierarchical Hypothesis Testing

Shujian Yu, Zubin Abraham, Heng Wang, Mohak Shah and José C. Príncipe

Abstract—In a streaming environment, there is often a need for statistical prediction models to detect and adapt to concept drifts (i.e., changes in the underlying relationship between the response and predictor data streams being modeled) so as to mitigate deteriorating predictive performance over time. Various concept drift detection approaches have been proposed in the past decades. However, they do not perform well across different concept drift types (e.g., gradual or abrupt, recurrent or irregular) and different data stream distributions (e.g., balanced and imbalanced labels). This paper presents a novel framework for statistical prediction models (such as a classifier) that detects and also adapts to the various concept drift types, even in the presence of imbalanced data labels. The framework leverages a hierarchical set of hypothesis tests in an online fashion to detect concept drifts and employs an adaptive training strategy to significantly boost its adaptation capability. The performance of the proposed concept drift detection and adaptation framework is compared to benchmark approaches using both simulated and real-world datasets spanning the breadth of concept drift types. The proposed approaches significantly outperform benchmark solutions in terms of precision, delay of detection as well as the adaptability across different concepts, regardless of data characteristics.

Index Terms—Concept drift, hierarchical hypothesis testing, adaptive training, streaming data classification.

I. INTRODUCTION

Effective techniques for analyzing streaming data are required in the era of big data and also pose new challenges in both machine learning and statistics communities [1], [2]. As a common task in streaming data mining, online classification requires the classifier to predict class labels at each time index [3]. However, when the underlying source generating the data is not stationary, the optimal decision rule for the classifier changes over time, a phenomena known as concept drift [4].

Numerous real-world applications such as fraud detection, user preference prediction, email filtering, etc. [5] capture intrinsically changes in the relationship of the incoming data streams. Current approaches to address these concept drifts fall into two categories [3], [6]. The first, automatically adapts the parameters of the statistical model in an incremental fashion as new data is observed or employs an ensemble of classifiers.

The above mentioned adaptation methods either retrain the classifier after a fixed (or adaptive) window size [7]–[9] or modify elements of a specific model when drift is suspected using a heuristic strategy [10], [11], while the above mentioned ensemble methods apply a weighted voting scheme of multiple learners, trained on different windows over the stream, to give the optimal decision [12]–[15]. There is no explicit detection of drifts for these methods, but initiation of new classifiers.

The second approach to address concept drift is to have a concept drift detector in addition to the statistical model, whose purpose is to signal the need for retraining the statistical model once concept drift is detected in the data. Existing methods in this category monitor the error rate or an error-driven statistics and draw upon the statistical learning theory [5], [16]–[18]. Unlike the previous category of concept drift approaches that only mitigated the effect of concept drift, the approaches in the second category enable identifying when a concept drift has occurred. Identifying when a concept drift has occurred is important in instances such as credit card fraud detection application, where it may be necessary to gather data pertaining to a concept drift triggered by a particular fraudsters [19]. This paper focuses on approaches for detecting concept drifts.

In this paper, we present a two-layered hierarchical hypothesis testing framework (HLFR) for concept drift detection and adaptation. This framework is inspired by the hierarchical architecture that was initially proposed for change-point detection [20], a related yet different topic to concept drift detection (see section I-A for more discussions on their relationships). Our work attempts to bring new perspectives to the field of concept drift detection and adaptation with the recent advances in change-point detection and provides the following original contributions. First, we present HLFR, the first hierarchical architecture for concept drift detection, to take the advantage of our recently developed LFR [17] and also significantly reducing false alarms. Second, we present an adaptive training approach instead of the re-training strategy commonly employed, once a drift is confirmed. The motivation is to leverage the knowledge from the previous concept to enhance the classification performance in the following concept, unlike re-training strategies that discard information from the previous concept. We term this improvement adaptive HLFR (A-HLFR). Finally, we present a comprehensive empirical analysis validating the performance improvement of using a hierarchical architecture and adaptive training for concept drift detection and adaptation. The results suggest that in addition to LFR, almost all the other classical single-layer-based concept
drift detection methods such as DDM [5] and EDDM [21] also benefit from these two strategies.

The rest of the paper is organized as follows. The problem formulation of concept drift and the notations used in this paper are presented in Section I-A, the related research is reviewed in section I-B. In section II, we present the HLFR framework and elaborate on the layer-I and layer-II tests employed. In section III, we present A-HLFR, that not only detects but also adapts the underlying statistical model (classifier) to handle concept drifts. In section IV, we carry out extensive experiments on both benchmarking simulated and real-world datasets to demonstrate the superior performance of HLFR (in detecting) and A-HLFR (in detecting and adapting) to concept drifts, and validate the rationale for employing a hierarchical architecture and adaptive training strategy. Finally, we present the conclusion in section V.

A. Problem Formulation

Given a continuous stream of labeled streaming samples \( \{X_t, y_t\}, t = 1, 2, ..., \) where \( X_t \) is a \( d \)-dimensional vector in a pre-defined vector space \( \mathcal{X} = \mathbb{R}^d \) and \( y_t \in \{0, 1\} \), at every time point \( t \), we split the samples into sets \( S_A \) (containing \( n_A \) recent samples) and set \( S_B \) (containing \( n_B \) examples that appeared prior to those in \( S_A \)). A concept drift is said to occur when the mapping \( f(X_t) \rightarrow y_t \) differs for \( S_A \) and \( S_B \).

A closely related topic to concept drift detection is the well-known change-point detection problem in machine learning and statistics community, whereas the goal of the latter is to detect changes in the generating distributions \( P(X_t) \) of the streaming data. The standard tools for change-point detection are methods from statistical decision theory; some reference books include [22]–[25], the representative novel methods can be referred to [20], [26], [27]. In general, these methods usually compute a statistic from the available data that is sensitive to changes between the two sets of examples. The measured values of the statistic are then compared to the expected value under the null hypothesis that both samples are from the same distribution. The resulting \( p \)-value can be seen as a measure of the strength of the drift. A good statistic must be sensitive to detecting changes in data characteristics of samples observed from different distributions.

Although a drift in generating distribution \( P(X_t) \) may result in a change in the learning problem, the detection of any type of distributional change remains a challenge, especially when \( X_t \) is high-dimensional data [6], [17]. In this paper, to achieve high detection accuracy, we adopt the principle of risk minimization [28] and solve the problem directly by monitoring the “significant” drift in the prediction risk (i.e., classification loss) of the underlying predictor rather than the intermediate problem of change-point detection. This is motivated by the fact that any drift of \( P(f(X_t), y_t) \) would imply a drift in \( P(X_t, y_t) \) with probability 1 [29], where \( f \) is the incrementally learned predictor (or classifier).

B. Related works

An extensive review on learning under concept drift is beyond the scope of this paper, and we refer readers to some recently published surveys [30], [31] for more details about concept drift detection and adaptation, and extensive surveys dealing with imbalanced data can be found in [32], [33]. In this section, we only review the works of most relevance to our method, i.e., concept drift detection approaches.

The approach that renewed attention to this problem was the Drift Detection Method (DDM) [5]. The test statistic DDM monitors the sum of overall classification error \( \left( \hat{P}_{error}^{(t)} \right) \) and its empirical standard deviation \( \left( \hat{S}_{error}^{(t)} \right) \) of the drift detection methods such as DDM [5] and EDDM [21] also benefit from these two strategies.

Following that, a few new approaches have been proposed, aiming at improving DDM from different perspectives. Drift Detection Method for Online Class Imbalance (DDM-OCI) [16] address the limitation of DDM when data is imbalanced. However, DDM-OCI triggers a number of false positives due to an inherent weakness in the model. DDM-OCI assumes that the concept drift in imbalanced streaming data classification is indicated by the change of underlying true positive rate (i.e., minority-class recall). This hypothesis unfortunately does not account for scenarios when concept drift occurs without affecting minority-class recall. For instance, it is very possible that the underlying concept drifts from imbalanced data to balanced data, while the true positive rate \( (P_{tp}) \), positive predictive value \( (P_{pp}) \) and F-measure remain unchanged. This type of drift is unlikely to be detected by DDM-OCI. The test statistic used by DDM-OCI \( \hat{R}_{tpr}^{(t)} \) is also not approximately distributed as \( \mathcal{N}(P_{tpr}^{(t)} + \sqrt{P_{error}^{(t)}}(1 - P_{error}^{(t)})/t) \) under the stable concept [2]. Hence, the tpr rationale of constructing confidence levels specified in [5] is not suitable with the null distribution of \( \hat{R}_{tpr}^{(t)} \). This is the reason DDM-OCI triggers false positives quickly and frequently. PerfSim [18] also dealt with imbalanced data. Different from DDM-OCI, PerfSim initially took into consideration all 4 entries in a confusion matrix. It employed and tracked a single test statistic, the cosine similarity of four rates, to determine the occurrence of concept drift over consecutive batch data. However, the threshold used to distinguish concept

\[ \hat{R}_{tpr}^{(t)} = \hat{R}_{tpr}^{(t-1)} + \eta \hat{R}_{tpr}^{(t-1)} (1 - \eta) \text{ where } \eta \text{ denotes a time decaying factor [16].} \]
drift was user-specified and did not have statistical meaning. On the other hand, Exponentially weighted moving average (EWMA) charts [35] was adapted for concept drift detection in ECDD [3] to control the computational complexity and false positive rate. The test statistic ECDD monitors is the Bernoulli parameter \( p \) estimated from the overall classification error, and a significant increase or decrease of \( p \) indicates a concept drift.

Linear Four Rates (LFR) [17] was recently proposed to address the limitation of single test statistic by monitoring the four rates associated with the confusion matrix of the data stream. Although, LFR outperforms aforementioned methods, it still triggered a number of false alarms. Thus, a novel architecture (or framework), which can take advantage of LFR but also control the number of false alarms, becomes an essential task for concept drift detection.

II. HIERARCHICAL LINEAR FOUR RATES (HLFR)

This paper presents a two-layered hierarchical hypothesis testing framework (HLFR) for concept drift detection. Once a potential drift is detected by the Layer-I test of HLFR, the Layer-II test is performed to confirm (or deny) the validity of the suspected drift. The results of the Layer-II feed back to the Layer-I of the framework, reconfiguring and restarting Layer-I as needed. The HLFR framework implements a sequential hypothesis testing [36], [37], and the two layers cooperate closely by exchanging information about the detected drift to improve online detection ability\(^3\), as shown in Fig.1.

HLFR treats the underlying classifier as a black-box and does not make use of any of its intrinsic properties. This modular property of the framework allows it to be deployed alongside any classifier (\( k \)-nearest neighbors, multilayer perceptron, etc.) unlike concept drift detectors that are designed to only work with linear discriminant classifiers [38], support vector machines (SVM) [39], etc. This paper selects soft margin SVM as the baseline classifier due to its accuracy and robustness [40]. We also test the drift detection performance using other classifiers in experiments.

Given the robustness of detecting concept drift by monitoring the four rates of the confusion matrix streams (more specifically, true positive rate (\( P_{tpr} \)), true negative rate (\( P_{tnr} \)), positive predictive value (\( P_{ppv} \)) and negative predictive value (\( P_{nnv} \)), even in the presence of imbalanced class labels [17], HLFR uses LFR in its Layer-I test. The second layer of HLFR uses the test statistic (or quantity) strictly related to that used at the Layer-I to conduct a permutation test. If a drift is confirmed, the HLFR framework signals a detection, otherwise the Layer-I detection output is considered to be a false positive and the test (eventually retrained) restarts to assess forthcoming data.

Experiments on synthetic dataset and real world applications demonstrate that HLFR outperforms state-of-the-art methods by significantly reducing false positives and guaranteeing low false negatives and detection delays. HLFR, is summarized in Algorithm 1.

**Algorithm 1 Hierarchical Linear Four Rates (HLFR)**

**Require:** Data \( \{X_t, y_t\}_{t=1}^{∞} \) where \( X_t \in \mathbb{R}^d, y_t \in \{0, 1\} \).

**Ensure:** Concept drift time points \( \{T_{cd}\} \).

1: for \( t = 1 \) to \( ∞ \) do
2: Perform Layer-I hypothesis testing.
3: if (Layer-I detects potential drift point \( T_{pot} \)) then
4: Perform Layer-II hypothesis testing on \( T_{pot} \)
5: if (Layer-II confirms the potentiality of \( T_{pot} \)) then
6: \( \{T_{cd}\} ← T_{pot} \)
7: else
8: Discard \( T_{pot} \); Reconfigure and restart Layer-I
9: end if
10: end if
11: end for

A. Layer-I Hypothesis Testing

Layer-I uses a drift detection algorithm that works in an online fashion. This work uses our recently proposed Linear Four Rates (LFR) [17] as the Layer-I test, as it has been experimentally proven that LFR outperforms benchmarks in terms of precision, recall and delay of detection in majority of the cases.

The underlying concept for LFR strategy is straightforward: under a stable concept (i.e., \( P(X_t, y_t) \) remains unchanged), \( P_{pfr}, P_{tnr}, P_{ppv}, P_{nnv} \) remains the same. Thus, a significant change of any \( P_\star (\star \in \{pfr, trnr, ppv, nnp\}) \) implies a change in underlying joint distribution \( P(X_t, y_t) \), or concept. More specifically, at each time instant \( t \), LFR conducts statistical tests with the following null and alternative hypothesis:

\[
H_0 : \forall \star, P(\hat{P}_\star^{(t-1)}) = P(\hat{P}_\star^{(t)}) \\
H_A : \exists \star, P(\hat{P}_\star^{(t-1)}) \neq P(\hat{P}_\star^{(t)})
\]

The concept is stable under \( H_0 \) and is considered to have potential drift if \( H_A \) holds. Intuitively, LFR should be more sensitive to any drifts regardless of their types, as it keeps track of four rates simultaneously. On the contrary, almost all the previous works focus on single specific statistic: DDM and ECDD use the overall error rate, EDDM focuses on the

---

\(^3\) Note that the Layer-I and Layer-II test can also be executed separately and independently, i.e., merely operating the Layer-I test, the HLFR framework reduces to our previously proposed LFR [17]; merely operating the Layer-II test may also achieve satisfactory results at the cost of expensive computation.
The LFR is summarized in Algorithm 2. During implementation, LFR modifies $P_s^{(t)}$ with $R_s^{(t)}$ as employed in [16], [31] (also see footnote 2). $R_s^{(t)}$ is essentially a weighted linear combination of the classifier’s previous performance and current performance. Given that $R_s^{(t)}$ follows a weighted i.i.d. Bernoulli distribution (see proof in [17]), we are able to obtain the bound table “BoundTable” by Monte-Carlo simulations. Having computed the bounds, the framework considers that a concept drift is likely to occur and sets the warning signal ($\text{warn.time} \leftarrow t$), when any $R_s^{(t)}$ crosses the corresponding warning bounds (warn.bd) for the first time. If any $R_s^{(t)}$ reaches the corresponding detection bound (detect.bd), the concept drift is affirmed at ($\text{detect.time} \leftarrow t$). Interested readers can refer to [17] for more details.

Note that, HLFR includes two modifications to LFR. First, we update the time decaying factor $\eta_s$ with$^4$:

$$\eta_s^{(t)} = \left\{ \begin{array}{ll} \left( \eta_s^{(t-1)} - 1 \right) e^{-\left( R_s^{(t)} - R_s^{(t-1)} \right)} + 1 & R_s^{(t)} \geq R_s^{(t-1)} \\ \left( 1 - \eta_s^{(t-1)} \right) e^{R_s^{(t)} - R_s^{(t-1)}} + (2\eta_s^{(t-1)} - 1) & R_s^{(t)} < R_s^{(t-1)} \end{array} \right.$$  

This adaptation is motivated from the adaptive signal processing domain [42]. The key idea is that a larger time decaying factor shall be used when $R_s$ is increasing, which indicates the classifier tends to perform better with most recent data. Moreover, we applied polynomial curve fittings to BoundTable so that a closed-form function can improve calculations of warning/detection bounds in terms of both memory usage and resolution. Unless otherwise specified, the LFR mentioned in this paper refers to the modified one.

Algorithm 2 Linear Four Rates (Layer-I)

**Require:** Data stream $\{ X_t, y_t \}_{t=1}^{\infty}$ where $X_t \in \mathbb{R}^d$ and $y_t \in \{0, 1\}$; Binary classifier $f(\cdot)$; Time decaying factors $\eta_s$; warn significance level $\delta$; detect significance level $\epsilon$.

**Ensure:** Potential concept drift time stamps $\{ T_{pot} \}$.

1: $R_s^{(0)} \leftarrow 0.5$, $P_s^{(0)} \leftarrow 0.5$ where $\ast \in \{ \text{tpr, tnr, ppv, npv} \}$

and confusion matrix $C^{(0)} \leftarrow \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$;

2: for $t = 1$ to $\infty$ do

3: $\hat{y}_t \leftarrow f(X_t)$

4: $C^{(t)}[\hat{y}_t][y_t] \leftarrow C^{(t-1)}[\hat{y}_t][y_t] + 1$

5: while ($\ast \in \{ \text{tpr, tnr, ppv, npv} \}$) do

6: if ($\ast$ is influenced by ($y_t, \hat{y}_t$)) then

7: $\hat{R}_s^{(t)} \leftarrow \eta_s R_s^{(t-1)} + (1 - \eta_s)1_{(y_t=\hat{y}_t)}$

8: else

9: $\hat{R}_s^{(t)} \leftarrow R_s^{(t-1)}$

10: end if

11: if ($\ast \in \{ \text{tpr, tnr} \}$) then

12: $N_{\ast} \leftarrow C^{(t)}[0,1_{\ast=tpr}] + C^{(t)}[1,1_{\ast=tpr}]$

13: $\hat{P}_s^{(t)} \leftarrow \frac{C^{(t)}[1_{\ast=tpr},1_{\ast=tpr}]}{N_{\ast}}$

14: else

15: $N_{\ast} \leftarrow C^{(t)}[1_{\ast=ppv},0] + C^{(t)}[1_{\ast=ppv},1]$

16: $\hat{P}_s^{(t)} \leftarrow \frac{C^{(t)}[1_{\ast=ppv},1_{\ast=ppv}]}{N_{\ast}}$

17: end if

18: warn.bd, $\ast$ = BoundTable($\hat{P}_s^{(t)}, \eta_s, \delta, N_{\ast}$)

19: detect.bd, $\ast$ = BoundTable($\hat{P}_s^{(t)}, \eta_s, \epsilon, N_{\ast}$)

20: end while

21: if (any $R_s^{(t)}$ exceeds warn.bd., & warn.time is NULL) then

22: warn.time $\leftarrow t$

23: else if (no $R_s^{(t)}$ exceeds warn.bd., & warn.time is not NULL) then

24: warn.time $\leftarrow$ NULL

25: end if

26: if (any $R_s^{(t)}$ exceeds detect.bd, ) then

27: detect.time $\leftarrow t$

28: relearn $f(\cdot)$ by $\{ X_t, y_t \}_{t=\text{detect.time}}$ or wait for sufficient instances;

29: reset $R_s^{(t)}$, $\hat{P}_s^{(t)}$, $C^{(t)}$ as step 1;

30: $\{ T_{pot} \} \leftarrow t$.  

31: end if

32: end for

B. Layer-II Hypothesis Testing

The second-level testing aims at validating detections raised by the Layer-I, and is activated only when the Layer-I detection occurs. In particular, we rely on the value $T_{pot}$ provided by the Layer-I testing to partition the streaming observations into two subsequences (aiming at representing observations before and after the suspected drift instant $T_{pot}$) and a different statistical hypothesis test for comparing the inherent properties of these two subsequences to assess a possible variations in the joint distribution $P(f(X_t), y_t)$.

In this section, we present the permutation test procedure (see Algorithm 3) used in Layer-II. Permutation test is theoretically well founded and does not require apriori information about the monitored process or nature of the drift [43]. We want to emphasise that the selection of the test statistic used at Layer-II should be strictly related to that used at Layer-I. To this end, we choose zero-one loss over the ordered train-test split, as

$^4$Note that, this time-varying representation of $\eta_s^{(t)}$ is hyperparameter free and self-bounded between $(2\eta_s^{(t-1)} - 1, 1)$ with a “sigmoid” like curve.

$^5$We estimate risk with zero-one loss on testing samples herein.
Algorithm 3 Permutation Test (Layer-II)

Require: Potential drift time $T_{pot}$; Permutation window size $\tilde{W}$; Algorithm A; Permutation number $\tilde{P}$; Significant rate $\eta$.

Ensure: Test decision (True positive or False positive?).

1: $S_{ord} \leftarrow$ streaming segment before $T_{pot}$ of length $W$.
2: $S'_{ord} \leftarrow$ streaming segment after $T_{pot}$ of length $W$.
3: Train classifier $f_{ord}$ at $S_{ord}$ using A.
4: Test classifier $f_{ord}$ at $S'_{ord}$ to get the zero-one loss $\hat{E}_{ord}$.
5: for $t = 1$ to $P$ do
6: $(S_i, S'_i)$ $\leftarrow$ random split of $S_{ord} \cup S'_{ord}$.
7: Train classifier $f_i$ at $S_i$ using A.
8: Test classifier $f_i$ at $S'_i$ to get the zero-one loss $\hat{E}_i$.
9: end for
10: if $\frac{\sum_{i=1}^{P} 1[\hat{E}_{ord} \leq \hat{E}_i]}{1+P} \leq \eta$ then
11: decision$\rightarrow T_{pot}$ is True positive.
12: else
13: decision$\rightarrow T_{pot}$ is False positive.
14: end if
15: return decision

III. ADAPTIVE HIERARCHICAL LINEAR FOUR RATES (A-HLFR)

Although HLFR can be used for streaming data classification with concept drifts (just like its DDM [5], EDDM [21] and STEPD [34] counterparts), naively retraining a new classifier after each concept drift detection severely deteriorates its classification performance. This stems from the fact that once a drift is confirmed, it discards all the (relevant) information from previous experience and uses only limited samples from current concept to retrain a classifier. This reaction scheme, though reasonable, always gives rise to a “significant” classification accuracy drop near the drift points, since it is very difficult (or impossible) to represent the new concept well with an extremely small number of samples using re-training scheme. In fact, a promising framework for streaming data classification under concept drifts should satisfy the following properties [31], [44]: 1) accommodate a variety of concept drift environments (e.g., gradual or abrupt, recurrent or irregular); 2) retain and extract knowledge from past experience when such knowledge is relevant; and 3) learn from possibly imbalanced data with good performance on both minority and majority class data.

In this section, we present a modified HLFR framework to meet the above criteria simultaneously (especially for the second one). We term it adaptive hierarchical linear four rates (A-HLFR). A-HLFR makes a simple yet strategic modification to HLFR: replacing re-training scheme in HLFR framework with an adaptive learning strategy. Specifically, we substitute SVM with adaptive SVM [45], [46] once a concept drift is confirmed. The pseudocode of A-HLFR is the same as Algorithm 1. The only exception comes from the layer-I test, where the re-training scheme with standard SVM (line 28 in Algorithm 2) is substituted with adaptive SVM.

A. Adaptive SVM - Motivations and Formulations

A fundamental difficulty for learning supervised models once a concept drift is confirmed, is that the training samples from new and previous concepts are drawn from different or shifted distributions. A short detection delay (especially for state-of-the-art concept drift detection algorithms) results in extremely limited training samples from the new concept. These limited training samples from the new concept, coupled with the fact that it may be likely that consecutive concepts are closely related or relevant, inspires the idea of adapting the previous models with samples from the new concept to boost the concept drift adaptation capability. Note that, although sample weighting scheme [7] provides a plausible solution, especially for ensembles of classifiers [47], the effectiveness is not salient through vast of simulations when using single classifier.

In the context of the earlier mentioned problem formulation, we are required to conduct a binary classification task with respect to a new concept in a newly observed primary dataset $D = \{(X_i, y_i)\}_{i=1}^N$, where only a limited number of examples are labeled. However, the previously observed fully-labeled auxiliary dataset $D^a = \{(X_i^a, y_i^a)\}_{i=1}^N$ and its corresponding auxiliary classifier $f^a$ should also be considered. The dataset $D$ is drawn from a distribution that is related to, but different from, the distribution of $D^a$ in a way unknown to the learner. To classify samples in $D$, the auxiliary classifier $\hat{f}^a$ may not perform well since it is biased to the new distribution. On the other hand, a new classifier trained only from the limited examples in $D$, although unbiased, may suffer from a high variance.

To achieve a better bias-variance tradeoff, we employ adaptive SVM (A-SVM), initiated in [45], [46], to adapt $\hat{f}^a$ to $D$. Intuitively, the key idea of A-SVM is to learn an adaptive classifier $\hat{f}$ from $\hat{f}^a$ by regularizing the distance between $\hat{f}$ and $\hat{f}^a$, which can be formulated as follows:

\[
\min_w \frac{1}{2} \|w - w^a\|^2 + C \sum_{i=1}^N \xi_i \quad \text{s.t.} \quad \xi_i \geq 0, \quad y_i w^T \phi(X_i) \geq 1 - \xi_i, \forall (X_i, y_i) \in D \tag{1}
\]

where $w^a$ are the classifier parameters estimated from $D^a$. This modified objective function seeks to reduce the distance between $w$ and $w^a$ while minimizing the classification error. A brief discussion on the property of A-SVM is available in Appendix A.

As an example, consider the simulations on USENET1 dataset in section IV-B, the average number of training samples from new concept for DDM [5], EDDM [21], DDM-OCI [16], STEPD [34], LFR [17] and HLFR, after a drift is confirmed, are 26, 36, 26, 19, 17 and 17, respectively.

1A possible exception comes from the scenario when the two consecutive concepts are completely opposite.

2In this work, our discussion is based on kernel SVMs, since linear SVMs are special cases of kernel models when $\phi(X) = X$, where $\phi$ represents a feature mapping to project sample vector $X$ into feature vector $\phi(X)$ in a space of higher or infinite dimension.
IV. Experiments

This section presents six sets of experiments that demonstrate the superiority of HLFR and A-HLFR over baseline approaches, in terms of concept drifts detection and adaptation. Section IV-A demonstrates the benefits of a hierarchical architecture over single-layer-based approaches for concept drift detection. Section IV-B validates the benefits of HLFR on concept drift detection over classical methods such as DDM [5], EDDM [21], DDM-OCI [16], STEPD [34] as well as our recently proposed LFR [17], using quantitative metrics and visual evaluation. Section IV-C, presents experiments of HLFR with various classifiers (soft-margin SVM classifier, k-nearest neighbor (KNN) and quadratic discriminant analysis (QDA)) to demonstrate the superiority of HLFR is independent of the choice of classifiers. The following sections focus on experiments that validate the effectiveness of using an adaptive training method to improve the capacity of concept drift adaptation. Specifically, in section IV-D, we validate, with real world data, the potency of A-HLFR over other competing methods. In section IV-E, we empirically demonstrate that the benefits of adaptive training are not limited to HLFR framework, i.e., it provides a general solution for appropriate concept drift adaptation. Note that, we also conduct a brief analysis on the computational complexity for each approach in section IV-F to have a deeper understanding. All the experiments reported in this work were conducted in MATLAB 2015a under a Windows 10 64bit operating system.

A. Benefits of a Hierarchical Architecture

Before evaluating the HLFR framework, we first evaluate the benefits offered by the proposed hierarchical architecture. Note that the proposed hierarchical architecture may be integrated with an existing concept drift approach, by incorporating the second layer test to reduce false positives. This section compares single-layer-based approaches (such as LFR, DDM and EDDM) to its hierarchical architecture counterpart, in terms of false positive and false negatives. Even though parameter tuning of single-layer-based approaches (such as, decreasing the warning and detection thresholds of DDM) can be used to control the number of detected potential drift points, the reduction of false positives often comes at the cost of reducing true positives. However, in the hierarchical architecture, for a given parameter setting of the single-layer-based approach, it is often possible to reduce false positives with no decrease in true positives.

For the purpose of this evaluation, the benchmark USENET1 [14] dataset is used. This dataset simulates a stream of email messages from different topics, that are sequentially presented to a user who then labels them as interesting or junk according to his/her personal interests. It consists of 5 time periods of 300 examples. At the end of each period, the user’s interest in a topic alters in order to simulate the occurrence of concept drift.

We consider four learning scenarios of the LFR framework, with warning and detection significant levels set to

\[ \{\delta_* = 0.01, \epsilon_* = 0.001\}, \{\delta_* = 0.01, \epsilon_* = 0.0001\}, \{\delta_* = 0.01, \epsilon_* = 0.00005\} \text{ and } \{\delta_* = 0.01, \epsilon_* = 0.00001\}, \]

respectively. For each of the scenarios, we compare the respective HLFR performance in detecting concept drifts (Fig. 2(a) and Fig. 2(c)). Similarly, Fig. 2(b) and Fig. 2(d) plots the results of the benchmark DDM [5] and EDDM [21] approach for different learning scenarios, along with their respective hierarchical architecture counterpart that used an additional permutation test layer. The first and third rows of all the subplots are the results of the hierarchical architecture equivalent of the baseline approaches presented in the second and fourth row respectively.

From Fig. 2, it can be concluded that the hierarchical architecture presents an intrinsic advantage over single-layer-based methods. The hierarchical architecture does not influence the performance of given single-layer-based methods if it already perform well and reduces false positives made by the single-layer-based methods. The second-layer test is a flexible module within hierarchical architecture and it can be logically combined with any other single-layer-based method in practice. It is also worth noting that the second-layer test is not limited to permutation test.

B. Concept Drift Detection with HLFR

In this section, we compare the performance of the HLFR framework against popular concept drift benchmark approaches. Five benchmark algorithms are considered for eval-
In the following drift detection tasks, the base classifier used is a soft margin SVM (regularization parameter \(C = 1\)) with linear kernel (except for USENET1 [14], where RBF kernel with kernel size 1 is selected). The first stream, denoted “SEA” [5], represents abrupt drift with label noise. This dataset has 60000 examples, 3 attributes. Attributes are numeric between 0 and 10, only two are relevant. There are 4 concepts, 15000 examples each, with different thresholds for the concept function, which is if the sum of two relevant features is larger than threshold then example label is 0. Threshold values are 8, 9, 7 and 9.5. The second stream, denoted “Checkerboard” [15], presents a more challenging concept drift with label noise, where the examples are sampled uniformly from the unit square and the labels are set by a checkerboard with 0.2 tile width. At each concept drift, the checkerboard is rotated by an angle of \(\pi/8\) radians. The third stream, the most challenging synthetic dataset, denoted “Rotating hyperplane”, constitutes 60000 examples and 5 uniformly-spaced abrupt concept drift points. Within any of the two adjacent abrupt drift points, there are 10000 examples demonstrating a slow gradual concept drift which is specified by the \((k,t)\) pairs, where \(k\) denotes the total number of dimensions whose weights are changing and \(t\) measures the magnitude of such change in attributes. In our experiments, the \((k,t)\) pairs are set to \((2,0.1), (2,0.5), (2,1.0), (5,0.1), (5,0.5)\) and \((5,1.0)\) successively. The last stream, USENET1 [14], represents drifts synthesized from real data. Table I summarized the data properties and drift types for each stream\(^{10}\). A yes indicates that the stream data have corresponding data property or concept drift type, and vice versa. Clearly, the selected datasets span the breadth of concept drift types.

Each stream was independently generated 100 times, and \(P = 1000\) reshuffling splits were used in HLFR. We summarized the detected concept drift detection points for each method over these 100 independent trails. Fig. 3 compares the detection results of the various approaches. As can be seen, HLFR and LFR significantly outperform the other four approaches in terms of their ability to detect concept drifts early. The two approaches also significantly outperform the other approaches by triggering fewer false detections while also missing fewer concept drift points. HLFR further improves on LFR by reducing even the few false positives triggered by LFR.

Quantitative evaluations for Precision and Recall are presented in Fig. 4: the rows correspond to performance measurements and columns to different datasets. The detection Precision is significantly improved with HLFR while the Recall of HLFR and LFR are similar (except for Rotating hyperplane dataset). This is not surprising, as the purpose of Layer-II test serves to confirm or deny the potentiality of layer-I detection results. Layer-II cannot compensate for the errors of missing a detection made by Layer-I test. The relatively lower Recall is explained by the fact that Layer-II test is conservative, i.e., it may deny true positives although the probability is very low. STEPD seems to provide much better Recall on SEA and Rotating hyperplane datasets. However, the results are meaningless in practice as it triggers significantly more false alarms (as seen in the fifth row of Fig. 3(a) and Fig. 3(c)). Additionally, the detection Precision of STEPD on these two datasets are consistently less than 0.15, which further discredits its high Recall values. Table II summarized the detection delays for all competing algorithms, the displayed values outside brackets indicate ensemble averages, while the numbers in brackets denote standard deviations. The results of quantitative evaluations corroborate the qualitative observations.

### TABLE I: Summary of properties of selected datasets

| Data property | SEA | Checkerboard | Hyperplane | USENST1 |
|---------------|-----|--------------|------------|---------|
| high dimensional | no  | no           | no         | yes     |
| imbalance     | no  | no           | yes        | no      |
| recurrent     | yes | no           | yes        | yes     |
| gradual       | yes | yes          | yes        | yes     |

\(^{10}\)We re-implemented STEPD using chi-square test with Yates’s continuity correction, as it is equivalent to Fisher’s exact test used in the original paper [8].

\(^{11}\)These hyper parameters were not optimized. The parameter \(d\) in STEPD corresponds to a \(p\)-value and is thus set to a standard significance value. The parameters used for DDM and EDDM were taken as recommended by their authors.

### TABLE II: Detection delay for all competing algorithms

| Algorithms   | SEA     | Checkerboard | Hyperplane | USENST1 |
|--------------|---------|--------------|------------|---------|
| HLFR         | 482(502)| 55(37)       | 120(85)    | 17(7)   |
| LFR          | 458(486)| 56(36)       | 127(112)   | 17(7)   |
| DDM          | 1209(450)| 69(56)      | 125(128)   | 26(17)  |
| EDDM         | 939(550)| 93(50)       | 166(121)   | 36(8)   |
| DDM-OCI      | 844(461)| 58(40)       | 198(112)   | 26(17)  |
| STEPD        | 463(423)| 57(58)       | 140(118)   | 19(7)   |

\(^{12}\)We define high dimensional if the number of attributes of streaming samples, i.e., the dimensionality of \(X_0\), is larger than 10, as majority of benchmark datasets in concept drift detection community is less than 5 [3], [5]. Besides, imbalance means that the ratio of the number of samples in minority class to the number of samples in majority class is less than 20%.
(b) Precision over Checkerboard dataset

(d) USENET1 dataset

(f) Recall over Checkerboard dataset

Fig. 3: Comparison between the histograms of detected drift points over (a) SEA dataset; (b) Checkerboard dataset; (c) Rotating hyperplane dataset and (d) USENET1 dataset. The red columns denote the ground truth of drift points, the blue columns represent the histogram of detected drift points generated from 100 Monte-carlo simulations.

Fig. 4: Summary of Precision and Recall over SEA, Checkerboard, Rotating hyperplane and USENET1 datasets. The X-axis in each figure represents the pre-defined detection delay range, whereas the Y-axis denotes the corresponding Precision and Recall values. For a specific delay range, a higher Precision or Recall value suggests better performance.
C. Performance of HLFR is Independent of Classifier

In this section, we show that the superiority of HLFR over other approaches is independent of the classifier. To this end, instead of using a soft-margin SVM, two different other classifiers, i.e., k-nearest neighbor (KNN) classifier and quadratic discriminant analysis (QDA), are applied separately on all the competing algorithms. Fig. 5 shows the concept drift detection results over USENET1 dataset and Checkerboard datasets using these two classifiers, which, obviously, coincide with the simulation results in Section IV-B. HLFR consistently produces the best performance when compared to the baseline approaches. Note that, although almost all the methods perform poorly on the Checkerboard dataset using the QDA classifier, only HLFR can provide reasonable detection results on the third, fifth, sixth and seventh drifts.

![Comparison between the concept drift detection results over USENET1 and Checkerboard datasets using KNN or QDA classifier.](image)

Fig. 5: Comparison between the concept drift detection results over USENET1 and Checkerboard datasets using KNN or QDA classifier. The red lines denote the ground truth of drift points. The blue columns represent the histogram of detected drift points generated from 100 Monte-carlo simulations.

D. Classification With Concept Drift Using HLFR

To improve concept drift adaptation capacity of HLFR, we modify the classifier initialization strategy in section III, i.e., substituting the re-training strategy with an adaptive training method once a concept drift is confirmed. We term this improvement “Adaptive Hierarchical Linear Four Rates” (A-HLFR).

In this section, we perform two case studies using representative real-world concept drift datasets from email filtering and weather prediction domain respectively, aiming to validate the rationale of HLFR on concept drift detection as well as the potency of A-HLFR on concept drift adaptation. Performance is compared to DDM, EDDM, STEPD and LFR. Results of DDM-OCI are omitted as it fails to detect any “reasonable” concept drift points in our selected real world datasets.

The spam filtering dataset [13], which represents email messages from the Spam Assassin Collection13, consisting of 9324 instances and 500 attributes is used. The spam ratio is approximately 20%. It has been demonstrated that spam filtering dataset contains natural concept drifts [13], [49]. Besides, the weather dataset [15], [44], as a subset of the National Oceanic and Atmospheric Administration (NOAA) data14, consisting of daily observations recorded in Offutt Air Force Base in Bellevue, Nebraska, is also used for our study as it spans over 50 years, providing not only cyclical seasonal changes, but also possibly long-term climate change. Daily measurements include a variety of features such as temperature, pressure, visibility, and wind speed. The task is to predict whether it is going to rain from these features. Minority class cardinality varied between 10% and 30% throughout these 50 years.

On Parameter Tuning and Experimental Setting. A common phenomenon for classification of real world streaming data with concept drifts and temporal dependency is that “the more random alarms fire the classifier, the better the accuracy” [50]. Thus, to provide a fair comparison, the parameters of all competing algorithms are tuned to detect similar number of concept drifts (except for LFR [17], in which slightly more potential drift points are allowed). Table III and Table IV summarize the key parameters regarding significant levels (or thresholds) of different algorithms in spam data and weather data respectively. Note that, for spam data, an extensive search for appropriate partition of training set and testing set was performed based on the two criteria: 1) the training set is sufficient to achieve “significant” classification performance on both majority and minority classes; and 2) there is no strong autocorrelation in the training set classification residual sequence. With these two considerations, the length of training set is set to 600. As for the weather data, the training size is set to 120 instances (days), approximately one season as suggested in [44].

| Algorithms | Parameter settings on significant levels (or thresholds) |
|------------|--------------------------------------------------------|
| A-HLFR     | $\delta_s = 0.01, \epsilon_s = 0.0001, \eta = 0.01$   |
| HLFR       | $\delta_s = 0.01, \epsilon_s = 0.0001, \eta = 0.01$   |
| LFR        | $\delta_s = 0.01, \epsilon_s = 0.000001$            |
| DDM        | $\alpha = 3, \beta = 2.5$                            |
| EDDM       | $\alpha = 0.55, \beta = 0.90$                        |
| STEPD      | $w = 0.005, d = 0.0003$                               |

Case study on spam dataset. We first evaluate the algorithms performance on the spam dataset. Fig. 6 plots the concept drift detection results. Before evaluating detection results, we recommend interested readers to refer to [15], in which the k-means and expectation maximization (EM) clustering

13http://spamassassin.apache.org/
14ftp://ftp.ncdc.noaa.gov/pub/data/gsod
Fig. 6: Concept drift detection results on the spam dataset.

algorithms have been applied to the conceptual vectors of spam filtering dataset. According to [13], there are three dominating clusters (i.e., concepts) distributed in different time periods and concept drifts occurred approximately in the neighborhood of point 200 in Region I, point 8000 in Region III, the ending location (point 1800) of Region I as well as the start and ending locations (point 2300 and 6200) of Region II. Besides, there are many abrupt drifts in the Region II. A possible reason for these abrupt and frequent drifts may be batches of outliers or noisy messages. Obviously, the detection results of A-HLFR and HLFR best match these descriptions, except that they both miss a potential drift point around point 1800. LFR detects this point, but it also feeds back some false positives. Other methods, like DDM or EDDM, not only miss obvious drift points, but also report unreasonable drift locations in Region I or Region III.

To further bridge the connection between our detection results and clustering results in [13], a recently developed measurement - Kappa Plus Statistic (KPS) [51], [52] - have been proposed. KPS, defined as $\kappa^+ = \frac{p_0 - p_e}{1 - p_e}$, aims to evaluate data stream classifier performance taken into account temporal dependence as well as the effectiveness (or rationality) of classifier adaptation, where $p_0$ is the classifier’s prequential accuracy [53] and $p_e$ is the accuracy of No-Change classifier.$^{15}$ We segment the training set to approximately 30 periods. The KPS prequential representation over these periods is shown in Fig. 7(a). As can be seen, the HLFR adaptation is most effective in period 1-5 but suffer from a performance drop on period 6-10. These observations coincide with our detection results, as HLFR accurately detected the first drift point with no false positives in Region I, but missed a target in Region II.

Regarding the problem of streaming data classification, several different quantitative measurements are used for a thorough evaluation. First, while overall accuracy (OAC) is an important and commonly used metric, it is inadequate (especially for imbalanced data). Therefore, we also include the F-measure$^{16}$ and G-mean$^{17}$ values. All metrics are obtained at each time step, creating a time-series representation (just like learning curves in the general field of adaptive learning systems [56]). Fig. 8(a)-(c) plot the time series representations of OAC, F-measure and G-mean in the learning scenario. As can be seen, A-HLFR and HLFR typically provide a significant improvement in F-measure and G-mean while maintaining good OAC when compared to their DDM, EDDM and LFR counterparts, with A-HLFR performs slightly better than HLFR. STEPD seems to demonstrate the best overall classification performance for spam data, but A-HLFR and HLFR provide more accurate (or rational) concept drift detections which coincide with cluster assignments results in [13].

Case study on weather dataset. We then evaluate algorithm performance on weather dataset. As the ground-truth drift point location is not available, we only demonstrate the concept drift adaptation comparison results. Fig. 7(b) plots the KPS prequential representations. As can be seen, A-HLFR performs (or updates) best in majority of time segments. Fig. 8(d)-(f) plot the corresponding OAC, F-measure and G-mean time series representations for all competing algorithms. Although the no adaptation$^{18}$ enjoys an overwhelming advantage in OAC compared to DDM, EDDM, LFR, STEPD, it

$^{15}$The No-Change classifier is defined as a classifier that predicts the same label as previously observed, i.e., $\hat{y}_t = y_{t-1}$ for any observation $X_t$ [51], [52].

$^{16}$F-measure, defined as $F\text{-}\text{measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$, explicitly tries to balance precision and recall performance.

$^{17}$G-mean is defined as $G\text{-}\text{mean} = \sqrt{\text{Acc}^+ \times \text{Acc}^-}$, where Acc$^+$ denotes true positive rate and Acc$^-$ denotes true negative rate. It indicates if a classifier is performing well across all classes or just a majority class.

$^{18}$The no adaptation means using the initial trained classifier for prequential classification without any classifier update.
is however invalid as the corresponding F-measure and G-mean tend to be zero as time evolves. This suggests that if no adaptation is adopted, the initial classifier gradually identifying remaining data into a major class, i.e., no rain everyday, which is definitely not realistic. A-HLFR and HLFR can achieve close OAC values to non-adaptive classifier, but significant improvements on F-measure and G-mean. Again, A-HLFR performs slightly better than HLFR.

From these two real applications, it is easy to summarize some key observations:
1) There are severe concept drifts in the given data, as the performance of no adaptation deteriorates significantly as time evolves.
2) A-HLFR and HLFR can always demonstrate the best overall performance in terms of OAC, F-measure, G-mean as well as rationality of drift detection, with A-HLFR performs slightly better than HLFR.
3) There is still plenty of room for performance improvement on incremental learning under concept drifts in nonstationary environment, as the OAC, F-measure and G-mean values are far from optimal. In fact, even with the state-of-the-art methods which only focus on automatically adapting classifier behavior (or parameters) to stay up-to-date with the streaming methods which only focus on automatically adapting classifier
4) The ensemble of classifiers seems to be a direction in our future works, but majority of the currently available works using ensemble learning are developed for batch training data [43], which are not applicable in a fully online fashion as the data is given one by one in a sequential manner [31].

E. Benefits of adaptive learning

Finally, we demonstrate, via the application of concept drift adaptation on USENET1 and Checkerboard datasets, that the superiority of adaptive SVM for concept drift adaptation is not limited to the HLFR framework. To this end, we consider the algorithm performance of integrating adaptive SVM into DDM, EDDM, DDM-OCl, STEPD as well as LFR framework. We term this combinations A-DDM, A-EDDM, A-DDM-OCl, A-STEPD and LFR framework.

In Fig. 9, we plotted the Precision and Recall curves of HLFR, LFR, DDM, EDDM, DDM-OCl, STEPD, A-HLFR, A-LFR, A-DDM, A-EDDM, A-DDM-OCl and A-STEPD on USENET1 and Checkerboard, respectively. For better visualization, we separate all the competing algorithms into two groups, group I includes HLFR, A-HLFR, LFR, A-LFR, STEPD and A-STEPD as they always perform better than their counterparts, while group II contains DDM, A-DDM, EDDM, A-EDDM, DDM-OCl and A-DDM-OCl. In each subfigure, the dashed line represents the baseline algorithm without adaptive training (e.g., HLFR), while the solid line denotes its adaptive version (e.g., A-HLFR). Meanwhile, for each baseline algorithm, its adaptive version is marked with the same color for comparison purpose. Obviously, the adaptive training will not affect the performance of concept drift detection. This is because the drift is determined by keeping track of “significant” changes of classification performance, rather than the specific performance measurement itself.

In Fig. 10 and Fig. 11, we plotted the time series representations of OAC, F-measure and G-mean on these two datasets over 100 Monte-carlo simulations. The shading enveloping each curve in the figures represents 95% percent confidence interval. In each sub-figure, the red dashed (or blue solid) line represents mean values for drift detection algorithm with (or without) adaptive training scheme, while the red (or blue) shading envelop represent the corresponding confidence intervals. For almost all the competing algorithms their corresponding adaptive versions achieve much better classification results than the non-adaptive counterparts. This performance boost begins from the first concept drift adaptation and grows gradually with increasing number of adaptations. As seen, A-HLFR and A-LFR achieves more compelling learning performance compared with A-DDM, A-EDDM, A-DDM-OCl and A-STEPD. This also coincides with the quantitative analysis results of concept drift detection shown in Fig. 9. These results empirically validate the potential and superiority of using adaptive classifier techniques for concept drift adaptation, instead of the re-training strategy adopted in previous work. It is also worth noting that the adaptive classifier is not limited to soft-margin SVM. In fact, adaptive logistic regression [57], adaptive single-layer perceptron [58] and adaptive decision tree [59] frameworks all have been developed in recent years with the advance of statistical machine learning. We leave investigations of concept drift adaptation using other adaptive classifiers as future work.

F. On the computational complexity analysis

Having demonstrated the benefits of the hierarchical architecture (precision, recall and detection delay), this section discusses the computational complexity of aforementioned concept drift algorithms, particularly the additional computation cost incurred by incorporating the hierarchical architecture in HLFR, DDM, EDDM, DDM-OCl, STEPD and LFR have a constant time complexity ($O(1)$) at each time point, as all of them follow a single-layer-based hypothesis testing framework that monitors one or four error-related statistics [17]. The time complexity for generating bound tables by LFR to determine the corresponding warning and detection bounds with respect to different rate values $P_a$ is $O(M)$, where $M$ is the number of Monte-Carlo simulations used. However, since the bound tables can be computed offline, the time complexity for looking up the bound table values once $P_a$ is given (see

\[ Admittedly, there is performance gap for DDM or STEPD, the difference is, however, data-dependent. For example, DDM seems to be better than A-DDM in Checkerboard dataset, but this advantage does not hold in USENET1.\]

\[ The comparable performance of A-DDM on Checkerboard dataset results from more times of adaptations, which is however unreasonable as the adaptation alarms are false alarms.\]
Fig. 8: The time series representations of different metrics for all competing algorithms. From (a)-(c): the OAC, F-measure and G-mean representations for spam data. From (d)-(f): the OAC, F-measure and G-mean representations for weather data.

Fig. 9: Summary of Precision and Recall over Checkerboard and USENET1 datasets for all competing algorithms and their adaptive versions. The X-axis in each figure represents the pre-defined detection delay range, whereas the Y-axis denotes the corresponding Precision and Recall values. For a specific delay range, a higher Precision or Recall value suggests better performance.

line 18 and 19 of Algorithm 2) remains $O(1)$ [60]. HLFR is more computational expensive than the single-layer-based hypothesis testing approaches, because of the introduction of Layer-II test, which requires training $P$ classifiers (1000 in this work) for validating the occurrence of a potential concept drift time point\(^\text{21}\). Assuming $O(K)$ is the computational complexity of training a new classifier, the time complexity for HLFR at this potential time point is $O(KP) \gg O(1)$. It should be noted that the $P$ permutations in Layer-II test may be run in parallel, as the classifier trained are independent across the various permutations.

A summary of the average computational cost (in seconds) of all the competing algorithms on USENST1 and Checkerboard datasets, is provided in Table V and Table VI, respectively. As expected, DDM, EDDM, DDM-OCI, STEPD and LFR have low and comparable average detection cost. On the contrary, HLFR consumes much more time. However, the hierarchical
architecture of HLFR introduces a new perspective to the field of concept drift detection, especially its ability to significantly reduce false alarms. To the best of our knowledge, the developed permutation test is currently the only available method that can explicitly remove false positive drift detections. We leave the investigation of more computational efficient layer-II tests as future work.

22 The Diversity for Dealing Drifts (DDD) [61] once considers the impact of false positive detections, but it only aims at improving online classification performance which is robust to false alarms.

**Fig. 10:** The time series representations of different metrics (OAC, F-measure, G-mean) on USENET1 dataset for (a) A-HLFR, HLFR; (b) A-LFR, LFR; (c) A-DDM, DDM; (d) A-EDDM, EDDM; (e) A-STEPD, STEPD; and (f) A-DDM-OCI, DDM-OCI. The red dashed line denotes mean values for adaptive learning methods, the red shading envelop represents 95% confidence interval. The blue solid line denotes mean values for non-adaptive learning methods, the blue shading envelop represents 95% confidence interval.

**TABLE V:** Detection cost comparison on USENET1 dataset.

| Algorithms | Computational complexity | Average detection cost (s.) |
|------------|--------------------------|-----------------------------|
| HLFR       | $O(KP)$                  | 9.9947                      |
| LFR        | $O(1)$                   | 0.1526                      |
| DDM        | $O(1)$                   | 0.0276                      |
| EDDM       | $O(1)$                   | 0.0166                      |
| DDM-OCI    | $O(1)$                   | 0.0220                      |
| STEPD      | $O(1)$                   | 0.0367                      |
Fig. 11: The time series representations of different metrics (OAC, F-measure, G-mean) on Checkerboard dataset for (a) A-HLFR, HLFR; (b) A-LFR, LFR; (c) A-DDM, DDM; (d) A-EDDM, EDDM; (e) A-STEPD, STEPD; and (f) A-DDM-OCI, DDM-OCI. The red dashed line denotes mean values for adaptive learning methods, the red shading envelop represents 95% confidence interval. The blue solid line denotes mean values for non-adaptive learning methods, the blue shading envelop represents 95% confidence interval.

TABLE VI: Detection cost comparison on Checkerboard dataset.

| Algorithms  | Computational complexity | Average detection cost (s.) |
|-------------|--------------------------|-----------------------------|
| HLFR        | $O(KP)$                  | 6.4749                      |
| LFR         | $O(1)$                   | 0.1451                      |
| DDM         | $O(1)$                   | 0.0449                      |
| EDDM        | $O(1)$                   | 0.0206                      |
| DDM-OCI     | $O(1)$                   | 0.0153                      |
| STEPD       | $O(1)$                   | 0.1095                      |

V. CONCLUSIONS

This paper presents a hierarchical hypothesis testing framework (HLFR) for concept drift detection. Unlike previous works, HLFR not only detects all possible variants of concept drifts regardless of data properties, it is also independent of the underlying classifier and outperforms existing approaches in terms of earliest detection of concept drift, with the least false alarms and highest precision. Using Adaptive SVM [45], [46] as its base classifier, HLFR can be extended to a concept drift-agnostic framework, i.e., A-HLFR. The performance of HLFR and A-HLFR in detecting and adapting to concept drifts are compared to benchmark approaches using both simulated and real-world datasets that span the gamut of concept drift types.
Results demonstrate that our proposed approaches significantly outperform benchmark approaches in terms of precision, delay of detection as well as the adaptability across different concepts, regardless of data characteristics.

Future works are two folds: firstly, we will continue on improving Layer-I or Layer-II testing methods, especially considering the fact that the employed permutation test is computationally expensive and time consuming. Besides, we are also interested in developing novel frameworks from information theoretic perspective [62], [63]. One possible solution is to track and detect the changes of mutual information between past and current observations. Some initial works have been done.

APPENDIX A
DISCUSSION ON A-SVM

We present a brief analysis to A-SVM (Adapted from [45]). The regularizer term \(|w - w^a|^2\) favors a “minimal” discrepancy from \(w\) to \(w^a\), and consequently, a decision function \(\hat{f}\) that is close to the auxiliary classifier \(\hat{f}^a\). Therefore, the objective function (1) seeks a new decision boundary that is close to the boundary of the auxiliary classifier (in the feature space) and meanwhile separates the labeled examples in \(D\). The regularizer term \(\|w^a\|\) favors a “minimal” discrepancy from \(w^a\) to \(w\), and consequently, a decision function \(\hat{f}\) that is close to the auxiliary classifier \(\hat{f}^a\). Therefore, the objective function (1) seeks a new decision boundary that is close to the boundary of the auxiliary classifier (in the feature space) and meanwhile separates the labeled examples in \(D\) as well. The factor \(C\) in A-SVM balances the contribution between the auxiliary classifier (through the regularizer) and the training examples. The larger \(C\) is, the smaller the influence of the auxiliary classifier is. Thus, one should use a small \(C\) for a “good” auxiliary classifier and vice versa.\(^{23}\) Equation (1) can be rewritten as the following (primal) Lagrangian:

\[
\ell_p = \frac{1}{2}\|w - w^a\|^2 + C\sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \alpha_i y_i w^T \phi(x_i) - (1 - \xi_i) - \sum_{i=1}^{N} \mu_i \xi_i
\] (2)

where \(\alpha_i \geq 0, \mu_i \geq 0\) are Lagrange multipliers. If we minimizing \(\ell_p\) by setting its derivative with respect to \(w\) and \(\xi_i\) to zero, it is straightforward to derive that the decision function \(\hat{f}\) can be written as [46]:

\[
\hat{f}(x) = \hat{f}^a(x) + \sum_{i=1}^{N} \hat{\alpha}_i y_i \kappa(x, x_i)
\] (3)

where \(\hat{\alpha}_i\) is the solution of \(\alpha_i\), \(\kappa\) is a Mercer’s kernel which satisfies \(\kappa(x, x_i) = \langle \phi(x), \phi(x_i) \rangle\). The adapted classifier \(\hat{f}\) can be seen as augmented from the auxiliary classifier \(\hat{f}^a\) with support vectors from the labeled samples in primary dataset \(D\).

\(^{23}\)In this work, we fix \(C = 1\) as employed in [46], [64].

REFERENCES

[1] K. Slavakis, S.-J. Kim, G. Mateos, and G. B. Giannakis, “Stochastic approximation vis-à-vis online learning for big data analytics [lecture notes],” IEEE Signal Processing Magazine, vol. 31, no. 6, pp. 124–129, 2014.

[2] H. Hu, Y. Wen, T.-S. Chua, and X. Li, “Toward scalable systems for big data analytics: A technology tutorial,” IEEE Access, vol. 2, pp. 652–687, 2014.

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

[4] G. Widmer and M. Kubat, “Learning in the presence of concept drift and hidden contexts,” Machine learning, vol. 23, no. 1, pp. 69–101, 1996.

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

[6] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer, “Concept drift detection through resampling,” in ICML, 2014, pp. 1009–1017.

[7] R. Klinkenberg, “Learning drifting concepts: Example selection vs. example weighting,” Intelligent Data Analysis, vol. 8, no. 3, pp. 281–300, 2004.

[8] A. Bifet and R. Gavalda, “Learning from time-changing data with adaptive windowing,” in Proceedings of the 2007 SIAM International Conference on Data Mining. SIAM, 2007, pp. 443–448.

[9] L. Du, Q. Song, and X. Jia, “Detecting concept drift: An information entropy based method using an adaptive sliding window,” Intelligent Data Analysis, vol. 18, no. 3, pp. 337–364, 2014.

[10] G. Hulten, L. Spencer, and P. Domingos, “Mining time-changing data streams,” in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001, pp. 97–106.

[11] M. Black and R. J. Hickey, “Maintaining the performance of a learned classifier under concept drift,” Intelligent Data Analysis, vol. 3, no. 6, pp. 453–474, 1999.

[12] W. N. Street and Y. Kim, “A streaming ensemble algorithm (sea) for large-scale classification,” in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001, pp. 377–382.

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

[14] I. Katakis, G. Tsoumakas, and I. P. Vlahavas, “An ensemble of classifiers for coping with recurring contexts in data streams,” in ECAI, 2008, pp. 763–764.

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

[16] S. Wang, L. L. Minku, D. Ghezzi, D. Caltabiano, P. Tino, and X. Yao, “Concept drift detection for online class imbalance learning,” in Neural Networks (IJCNN), The 2013 International Joint Conference on. IEEE, 2013, pp. 1–10.

[17] H. Wang and Z. Abraham, “Concept drift detection for streaming data,” in 2015 International Joint Conference on Neural Networks (IJCNN). IEEE, 2015, pp. 1–9.

[18] D. K. Antwi, H. L. Viktor, and N. Japkowicz, “The perfis algorithm for concept drift detection in imbalanced data,” in Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on. IEEE, 2012, pp. 619–628.

[19] N. G. Pavlidis, D. K. Tasoulis, N. M. Adams, and D. J. Hand, “Adaptive consumer credit classification,” Journal of the Operational Research Society, vol. 63, no. 12, pp. 1645–1654, 2012.
