LEVEL$_{\text{IW}}$: Learning Extreme Verification Latency with Importance Weighting

Muhammad Umer
Rowan University
umerm5@students.rowan.edu

Christopher Frederickson
Rowan University
fredericc0@students.rowan.edu

Robi Polikar
Rowan University
polikar@rowan.edu

Abstract*—Nonstationary streaming data are characterized by changes in the underlying distribution between subsequent time steps. Learning in such environments becomes even more challenging when labeled data are available only at the initial time step, and the algorithm is provided unlabeled data thereafter, a scenario referred to as extreme verification latency. Our previously introduced COMPOSE framework works very well in such settings. COMPOSE is a semi-supervised approach that iteratively labels strategically chosen instances of the next time step using the instances it labeled in the previous time step. COMPOSE originally assumed a significant distribution overlap at consecutive time steps, allowing instances lying in the center of the feature space to be used as the most representative labeled instances from current time step to help label the new data at the next time step. Such an assumption is also inherent in importance weighting based domain adaptation, but only for a single time step with mismatched train and test data distributions. We explore importance weighting not for a single time step matching training / test distributions, but rather matching distributions between two consecutive time steps, and estimate the posterior distribution of the unlabeled data using importance weighted least squares probabilistic classifier. The estimated labels are then iteratively used as the training data for the next time step. We call this algorithm as LEVEL$_{\text{IW}}$, Learning Extreme VErification Latency with Importance Weighting. Our primary goal in doing so is to determine if and when importance weighting provides an advantage over COMPOSE's core support extraction, and whether it provides an alternate solution with reduced parameter sensitivity. Several datasets are used to compare the two approaches, which produced some unique insights.

I. Introduction

One of the basic assumptions made by most traditional machine learning algorithms is that training and test data are drawn from a fixed but unknown distribution (also known as the independent and identically distributed assumption). Such an assumption is often not realistic in many real-world applications, particularly those that are associated with streaming data. Learning from such streaming nonstationary data is sometimes referred to as the concept drift problem [1]–[3].

Given a dataset $\mathcal{D} = \{x_t, y_t\}$, where x_t represents the data obtained at time t, with corresponding labels y_t, two types of drift are commonly encountered: i) *virtual drift*, closely related to *covariate shift* or *the domain adaptation*, characterized by changes in the marginal distribution between two time steps, i.e., $p_t(x) \neq p_{t+1}(x)$ with the posterior distribution remaining unchanged, i.e., $p_t(y|x) = p_{t+1}(y|x)$; and ii) *real drift* or just *concept drift*, characterized by changes in the posterior probability distribution, i.e., $p_t(y|x) \neq p_{t+1}(y|x)$ while marginal data distribution may remain unchanged, i.e., $p_t(x) = p_{t+1}(x)$.

Domain adaptation approaches are designed to handle a specific case of virtual drift, i.e., mismatched source (training) and target (test) data distributions over a single time step [4], [5], where the classification model is trained not on the original training data, but rather on a modified, specifically *weighted*, version of this data through an approach called instance or importance weighting. The modification makes the source domain data behave more like the target domain data distribution. The weighting factor is known as the importance ratio, which is essentially the ratio of target distribution $p_t(x)$ to source distribution $p_s(x)$. The primary task in domain adaptation is then to estimate this ratio directly and use it to weigh the training data. Several approaches have been proposed to estimate the importance ratio, including kernel mean matching (KMM) [6], using a probabilistic classifier such as logistic regression to estimate importance ratio directly [7], and using a Kullback-Leibler based importance estimation procedure (KLIEP) [8] etc. However, many of these approaches typically assume an abundantly available labeled data in the source domain, which is often an unrealistic assumption.

Ensemble based techniques, such as Learn^{++}.NSE [9]–[11], Learn^{++}.NIE [12], DWM [13], and SEA [14] represent a different family of approaches to tackle both virtual and real drift problems, and are capable of tracking data distributions not just over a single time step, but over a streaming setting. However, ensemble approaches also require a large amount of labeled data, and the potential scarcity or the high cost of obtaining labeled data is a major obstacle. In an effort to reduce the amount of required labeled data, semi supervised learning (SSL) approaches have also been used in this scenario where a hypothesis is formed using modest amount of labeled data and more abundant unlabeled data. The primary application domain of SSL techniques has been in stationary environments, but recently the focus has been shifting towards non stationary distributions. SSL approaches, of course, also require labeled data at each time step [15], albeit in smaller quantities.

Network intrusion, web usage and user interest analysis, natural language processing, speech and speaker identification, spam detection, anomaly detection, analysis of financial, climate, medical, energy demand, and pricing data, as well as the analysis of signals from autonomous robots and devices are just a few examples of the applications of the concept drift problem. The concept drift problem poses great difficulty because streaming data are usually unlabeled and unstructured.

All of the above-mentioned approaches for domain adaptation or concept drift further assume that there is (preferably ample) labeled training data in each time step. There are several application domains where obtaining labeled data is expensive or impractical, perhaps beyond an initial investment. Such problems, where only unlabeled data are available in all future time steps of a nonstationary data stream, are referred to as extreme verification latency. Compacted Object Sample Extraction (COMPOSE) [16], [17] is a semi-supervised framework designed to work in such settings with only a limited drift assumption, however, this algorithm is computationally expensive (due to its density estimation module), and extremely sensitive to the parameter selection of its semi-supervised learning module.

Clearly, domain adaptation and concept drift problems are related, though algorithms for each make different assumptions. Primarily, domain adaptation problems are single time step problems where training and test data marginal distribution differ, but have the same support and same posterior distributions. Concept drift problems are, however, streaming data problems, where the test data in any given time step becomes the training data during the next time step once they are labeled. Concept drift problems typically assume at least a gradual (or limited) drift assumption, but do not require stationary posteriors or same support. In this contribution, we explore whether and when well-established, computationally efficient domain adaptation approaches can be used for concept drift problems associated with extreme verification latency, i.e., when labeled data is available during the initialization step, followed by only unlabeled nonstationary data. We show that the answer to this question is affirmative, when indeed the original domain adaptation assumptions are satisfied, i.e., the class conditional distributions at consecutive time steps share support, and posterior distributions do not change.

II. Related Work

A. *Extreme verification latency*

Tracking distributions in a continuously changing environment becomes more challenging if labeled data are not available at every time step. Marss describes a scenario where true class labels are only available sometime after the classifier has made a prediction on current environment [18], a scenario referred to as verification latency. The duration of this lag may vary with time, and if this lag is infinite, i.e., no labeled data are ever received after initialization, such a scenario is referred to as an extreme verification latency. If this scenario is also characterized by a nonstationary environment, we call this *initially labeled non stationary environment* (ILNSE) or simply initially labeled streaming environment (ILSE) [16]. In our prior work, we described the COMPOSE framework [16], [17] that works in such environments. COMPOSE is an iterative, semi-supervised framework that runs every time a new batch of unlabeled data is received. At time step t, the new and currently unlabeled data is combined with the labeled data from the previous time step $t-1$, and are used to train a semi-supervised learning (SSL) algorithm to label those unlabeled instances from the current time step t. Any SSL algorithm can be used with COMPOSE, though we prefer cluster and label due to its lower computational complexity and generally good performance. The core support extraction (CSE) step then extracts those newly labeled data drawn from the center region of current distribution (i.e., core supports). Core supports are then used as the labeled data during the next time step's SSL algorithm. The CSE step samples observations from dense areas of the labeled data, as such areas are most likely to best represent incoming unlabeled data.

Other related algorithms that can learn in extreme verification latency scenario include Arbitrary sub-populations tracker (APT) [19], stream classification algorithm guided by clustering (SCARGC) [20], and micro-clusters for classification (Mclassification) [21]. The central premise of each is briefly discussed below.

APT is based on the principle that each class in the data can be represented as a mixture of arbitrarily distributed sub-populations. The APT algorithm makes following assumptions i) The drift is gradual or limited; ii) the drift can be represented as a piecewise linear function; iii) the covariance of each sub-population (a mode in the class conditional distribution) remains constant; iv) each sub-population to be tracked must be present at the initialization; and v) the drift remains constant. The learning strategy of APT is two-fold; first, the optimal one-to-one assignment between labeled instances in time step t and unlabeled instances in time step $t+1$ is determined using expectation maximization (EM). In expectation step, EM predicts which instances are most likely to correspond to a given sub-population, and during the maximization step, it determines which drift parameters maximize the expectation. Then, the classifier is updated to reflect the population parameters of the newly received data and drift parameter relating the previous time step to the current one. When the assumptions are satisfied, APT works very well. However, APT has two primary weaknesses: 1) some of its assumptions often do not hold true, causing a decrease in performance, and 2) it is computationally very expensive [16], even when compared to already expensive COMPOSE.

SCARGC is a clustering based algorithm that repeatedly clusters unlabeled input data, and then classifies the clusters using the labeled clusters from the previous time step [20]. The mapping between the clusters is performed by centroid similarity between current and previous iterations using Euclidean distance. Given the current centroids from the most recent unlabeled clusters and past centroids from the previously labeled clusters, one-nearest neighbor algorithm or support vector machine is used to label the centroid from current unlabeled clusters. SCARGC is computationally efficient but its performance is highly dependent on the clustering phase. It also requires some prior knowledge about the number of classes, and the number of modes for each class in the data, and this requirement may limit the use of this algorithm when such information is not available.

Mclassification uses the idea of micro clusters (MC) [22] to adapt to the changes in the data over time. Micro clusters is a method of storing information about a set of data points without requiring all examples to be retained. A set of labeled MC are first built from the initial labeled data. At each time step, each example from the streaming input data receives its label from the nearest MC. If the addition of an example in its corresponding nearest MC causes its MC radius not to exceed the maximum MC radius defined by the user, this example is added to the nearest MC and its sufficient statistics, i.e., its centroid and micro cluster radius are updated; otherwise a new MC is created. MClassification does not require the number of clusters to be known prior to execution as with APT or SCARGC, however it is also computationally expensive.

B. Importance Weighted Least-Squares Probabilistic Classifier

Importance weighted least-squares probabilistic classifier (**IWLSPC**) combines a probabilistic classification method, called least-squares probabilistic classifier, with the covariate shift adaptation technique. As described in more detail in [23], probabilistic classification is used to estimate the true class-posterior probability $p(y|\mathbf{x})$, modeled through

$$p(y|\mathbf{x}, \boldsymbol{\theta}_y) = \sum_n \theta_{y,n} K(\mathbf{x}, \mathbf{x}_{te,n}) \tag{1}$$

where n is an index on number of instances, $\mathbf{x}_{te,n}$ is the n^{th} test instance, $\boldsymbol{\theta}_y = (\theta_{y,1}, \ldots, \theta_{y,n})$ is the parameter vector, and $K(\mathbf{x}, \mathbf{x}_{te})$ is a Kernel function, typically the Gaussian kernel

$$K(\mathbf{x}, \mathbf{x}_{te,n}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}_{te,n}||^2}{2\sigma^2}\right) \tag{2}$$

with kernel width σ serving as the first free parameter for IWLSPC. The parameter vector $\boldsymbol{\theta}_y$ is determined by minimizing the squared error $J_y(\boldsymbol{\theta}_y)$ through quadratic programming

$$\begin{aligned} J_y(\boldsymbol{\theta}_y) &= \frac{1}{2}\int (p(y|\mathbf{x}; \boldsymbol{\theta}_y) - p(y|\mathbf{x}))^2 p_{te}(\mathbf{x}) d\mathbf{x} \\ &= \frac{1}{2}\boldsymbol{\theta}_y^T \mathbf{Q} \boldsymbol{\theta}_y - \mathbf{q}_y^T \boldsymbol{\theta}_y + \frac{\lambda}{2}\boldsymbol{\theta}_y^T \boldsymbol{\theta}_y \end{aligned} \tag{3}$$

where the last term is a regularization term to minimize overfitting through the algorithms's second free parameter λ, and where $\mathbf{Q}$ – an $n_{te} \times n_{te}$ matrix – and $\mathbf{q}_y = (q_{y,1}, \ldots, q_{y,n_{te}})$ are approximated using the adaptive importance sampling technique, through the *importance weight* defined as

$$w(\mathbf{x}) = \frac{p_{te}(\mathbf{x})}{p_{tr}(\mathbf{x})} \tag{4}$$

The quantities $\mathbf{Q}$ and $\mathbf{q}_y$ are then obtained as follows

$$Q_{n,n'} = \int K(\mathbf{x}, \mathbf{x}_{te,n}) K(\mathbf{x}, \mathbf{x}_{te,n'}) p_{tr}(\mathbf{x}) w(\mathbf{x}) d\mathbf{x} \tag{5}$$

$$q_{y,n} = p(y) \int K(\mathbf{x}, \mathbf{x}_{te,n}) p_{tr}(\mathbf{x}|y) w(\mathbf{x}) d\mathbf{x} \tag{6}$$

where $p_{tr}(\mathbf{x}|y)$ denotes the training input density for class y. These quantities are then approximated by replacing the integral over $\mathbf{x}$ with averaging over the training data $\mathbf{x}_{te,n}$ [23] to solve for $\boldsymbol{\theta}_y$, which in turn is used to determine the class-posterior probability $p(y|x; \boldsymbol{\theta}_y)$ through Equation 1. Given a test instance $\mathbf{x}_{te}$, the class label y_{te} is finally estimated as

$$\hat{y}_{te} = \underset{y}{\operatorname{argmax}}\, p(y|\mathbf{x}_{te}; \boldsymbol{\theta}_y). \tag{7}$$

The critical parameter in model selection for IWLSPC is kernel width σ, which is obtained through importance weighted cross validation (IWCV) [24] in IWLSPC's original description in [23] and is updated each time step separately. In this effort, we used parameter sweep, keeping the parameter constant through out experiment, not only because such a cross validation is unrealistic for each time step in a streaming environment, but we also wanted to determine the sensitivity of this parameter on the algorithm classification performance. The importance weights in Equation (4) are estimated through unconstrained least squares importance fitting (uLSIF) [25] as done in [23].

In summary then, we suitably modified IWLSPC – originally proposed for only single time step problems, where it was used to match the divergence in the training and test distributions on a non-streaming data application (i.e., to handle covariate shift or domain adaptation problems), and extended its use to problems in which i) data arrive in a continuous streaming fashion, where concept drift is occurring possibly at every time step, and perhaps more importantly ii) data arrive with extreme verification latency. The pseudocode of the original IWLSPC is given below.

Algorithm 1 IWLSPC

```
Inputs: unconstrained least squares importance fitting uLSIF
        - Importance weighted cross validation IWCV
1: Receive training data $x_{tr}$
2: Receive test data $x_{te}$
3: Run uLSIF to estimate importance weights
4: Run IWCV to estimate Gaussian kernel width $\sigma$
5: Compute Gaussian Kernel Function using $\sigma$ as defined in Equation 2
6: Estimate parameter $\theta_y$ by minimizing squared error $J_y(\theta_y)$ as defined in Equation 3
7: Use $\theta_y$, and the Gaussian Kernel function to compute posterior probability as defined in Equation 1.
```

III. Approach

The **COMP**acted **O**bject **S**ample **E**xtraction (COMPOSE) framework was introduced in [16] to address the extreme verification latency problem in an ILSE setting, i.e., to learn drifting concepts from a streaming non stationary environment that provides only unlabeled data after initialization. The basic assumption behind COMPOSE is the same assumption used by most concept drift algorithms: that the data drifts gradually between two time steps. Under this gradual drift assumption, it is reasonable to expect that the class-conditional distributions of any class will have significant overlap at each consecutive time steps. In fact, this is the reason why core-support extraction allows COMPOSE to track the drifting distribution. On the other hand, *significant overlap* is also the motivation behind using the importance weighting approach as such overlap is likely to result in satisfying the two important assumptions of importance weighting approaches: i) shared support of class-conditional distributions at two consecutive time steps; and ii) posterior distribution for each class remains the same (or at least, changes very little). If that is the case, the SSL and core support extraction steps of COMPOSE can be replaced with an importance weighting based approach. On the other hand, importance weighting, as used in domain adaptation, is intended for a single time step scenario with mismatched training and test datasets, whereas COMPOSE is intended to be used in streaming datasets with nonstationary distributions. Iteratively applying importance weighting, where each consecutive time step serve as the traditional source and target datasets, allows us to cast the importance weighting in a streaming data environment - of course, with the caveat that we are working in an extreme verification latency environment. Hence, we name our approach Learning Extreme VErification Latency with Importance Weighting: LEVEL$_{\text{IW}}$

The pseudocode and implementation details of this approach are described below and summarized in Algorithm 2.

LEVEL$_{\text{IW}}$ takes advantage of the importance weighted least squares probabilistic classifier (IWLSPC) as a subroutine [23], and hence serves as a wrapper approach. At initial time step $t = 0$, LEVEL$_{\text{IW}}$ receives data $\mathbf{x}$ with their corresponding labels y, and initializes the test data $\mathbf{x}_{te}^{t=0}$ to initial data $\mathbf{x}$ received, and sets their corresponding labels $y_{te}^{t=0}$ equal to the initial labels y. Then, the algorithm iteratively processes the data, such that at each time step t, a new unlabeled test dataset $\mathbf{x}_{te}^{t}$ is first received, the previously unlabeled test data from previous time step $\mathbf{x}_{te}^{t-1}$, which is now labeled by the IWLSPC subroutine, becomes the labeled training data $\mathbf{x}_{tr}^{t}$ for the current time step, and similarly the labels y_{te}^{t-1} obtained by IWLSPC during the previous time step become the labels of the current training data $\mathbf{x}_{tr}^{t}$. The training data at the current time step $\mathbf{x}_{tr}^{t}$, the corresponding label information at the current time step y_{tr}^{t}, the kernel bandwidth value σ and the unlabeled test data at the current time step $\mathbf{x}_{te}^{t}$ are then passed onto the IWLSPC algorithm, which predicts the labels y_{te}^{t} for the test unlabeled data. The entire process is then iteratively repeated.

Recall that the central premise behind COMPOSE is core support extraction, where core supports are those labeled instances located at the densest part (typically the core or central region) of the feature space, which can be used to represent the drifted distribution at the next time step, subject to gradual drift assumption. This assumption results in significant overlap between the data distributions at two consecutive time steps, which is illustrated in Fig. 1(a) where the yellow circle represents the core support region at time step t, denoted as CS_t. X_{t-1} represents the *now-labeled* data from the previous time step, and X_t represents the unlabeled

Algorithm 2 LEVEL$_{\text{IW}}$

```
Inputs: Importance weighted least squares probabilistic classifier - IWLSPC; Kernel bandwidth value $\sigma$
1: At $t = 0$, receive initial data $\mathbf{x} \in X$ and the corresponding labels $y \in Y = 1, \ldots, C$.
   Set $\mathbf{x}_{te}^{t=0}$ = $\mathbf{x}$
   Set $y_{te}^{t=0}$ = $y$
2: for $t = 1, ....,$ do
3:     Receive new unlabeled test data $\mathbf{x}_{te}^{t} \in X$
4:     Set $\mathbf{x}_{tr}^{t}$ = $\mathbf{x}_{te}^{t-1}$
5:     Set $y_{tr}^{t}$ = $y_{te}^{t-1}$
6:     Call IWLSPC with $\mathbf{x}_{tr}^{t}, \mathbf{x}_{te}^{t}, y_{tr}^{t}$, and $\sigma$ to estimate $y_{te}^{t}$
7: end for
```

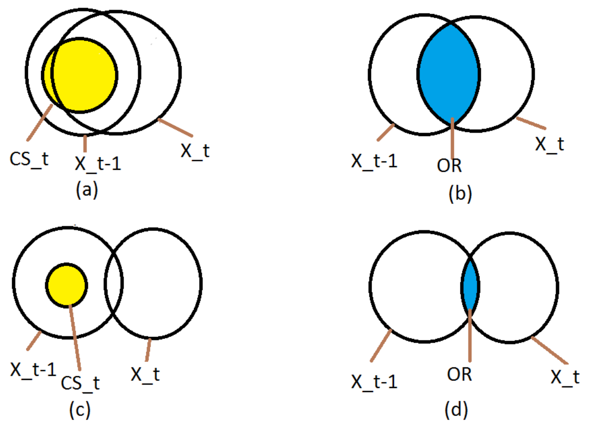

Fig. 1. Core support extraction process in COMPOSE vs. importance weighting in LEVEL$_{\text{IW}}$: (a) yellow region represents the core supports extracted using COMPOSE, which works well when there is significant overlap; (b) blue region represents the overlapping region of two distributions, training instances in this region receive more weight using LEVEL$_{\text{IW}}$; (c) yellow region represents the core supports extracted using COMPOSE, which does not work well when there is only marginal overlap; (d) blue region represents the marginally overlapping regions of two distributions, LEVEL$_{\text{IW}}$ still works by assigning higher weights to instances in this region.

data received in the current time step, t.

When there is such significant overlap, there is also significant shared support, and LEVEL$_{\text{IW}}$ assigns higher weights to those instances that are drawn from the shared support region, as shown in Fig. 1(b). Here, the overlapping region, denoted as OR, is shaded in blue, X_{t-1} and X_t represent the labeled data currently available, and the newly received unlabeled data, respectively. The weighting scheme then seeks to match the distributions in the two consecutive time steps. The primary advantage of using the importance weighting approach over the core support extraction of the original COMPOSE algorithm can be best observed when the drifting scenario is not as gradual, and the overlapping regions of the two distributions at two consecutive time steps is only marginal. In such cases, the core support region at time step t may no longer adequately represent the distribution at time step $t + 1$, as illustrated in Fig. 1(c). However, as long as there is some shared support – however marginal – importance weighting can still be applied to those instances in that region, albeit more aggressively, as indicated in Fig. 1(d).

IV. Experiments and Results

A. Experimental Setup

Fifteen synthetic datasets and one real dataset are used in the evaluation and comparison of LEVEL_{IW} and COMPOSE. Some of these datasets (indicated by an asterix in Table I) were originally provided by us in our prior works of [12] and [16], and others are provided by the authors of SCARGC in [20], and then provided at one convenient web site (https://sites.google.com/site/nonstationaryarchive/) for machine learning community. Table I briefly describes the important characteristics of these datasets. We analyze the algorithm behavior from two perspectives: the raw average accuracy of LEVEL_{IW} and COMPOSE, shown in Table II, and a more detailed parameter sensitivity based analysis shown in Tables III, and IV. From a raw classification accuracy perspective, we see that it is difficult to pick a clear winner between LEVEL_{IW} and COMPOSE - in some cases one algorithm does better, and in some cases the other, with some advantage in favor of COMPOSE when there is significant between-class overlap. The advantage of COMPOSE appears to be dramatic when the class-overlap lasts for the entire period of experiment, as was the case for datasets, such as *2CDT* and *2CHT*. This between-class overlap coupled by a drifting environment ultimately leads to a significant change in the posterior probability distribution $p(y|x)$ of classes for these datasets, hence violating one of the main assumptions of importance weighting approaches. Further analysis of results and the dataset properties revealed that the ability of COMPOSE to perform well even under significant between-class overlap is in fact due to a crucial piece of information provided to COMPOSE, through one of the free-parameters of its SSL module. Parameter sensitivity analysis, described in detail below, shows that the advantage shifts towards LEVEL_{IW}, when the user is not able to provide optimal free-parameters.

Recall that COMPOSE uses a clustering-based SSL algorithm to track the drifting distributions. The cluster-and-label (as used in our experiments), or label propagation, or other clustering-based SSL algorithms are able to identify the structure in the data from few labeled instances, and they do so reasonably well even when there is overlap among the clusters. In a drifting scenario, as long as at least part of the feature space from which there is labeled information is not completely overlapping with another class, a clustering-based SSL algorithm (and hence COMPOSE) can recover as soon as the classes separate again (as described in the case-studies below). However, this performance is subject to correct choice of the SSL's algorithm's primary user-provided free parameter, the number of clusters k in the data, to which the algorithms tend to be rather sensitive.

To better understand the underlying behavior of each algorithm, let's consider a few specific datasets as case studies: $i)$ *1CSurr*, a two-class, two dimensional problem where one class circumnavigates the other class, $ii)$ *UG_2C_2D*, where two unimodal bi-dimensional Gaussians circle around each other; and $iii)$ *keystroke*, 10-dimensional, four-class dataset with a complex drift scenario. In the *1CSurr* dataset, both classes are drawn from a Gaussian distribution, however the circumnavigating distribution has a much smaller covariance than the other distribution (that stay stationary). What makes this dataset particularly challenging is that the distribution of the class with smaller variance in each dimension first moves around the other class, completely circumnavigating the perimeter with some – but not complete overlap – followed by diagonally crossing over the second class distribution with *complete* overlap. We note that "overlap", in this context, refers to overlapping class distributions (which is bad for the classifiers), and not to overlap between two consecutive timesteps of a particular class distribution (which is good for the classifiers). Fig. 2 illustrates this dataset in six different time snapshots.

TABLE I
Dataset Description

Datasets	# of classes	# of features	Cardinality	Drift interval	Class Over-lap
1CDT	2	2	16000	400	no
1CHT	2	2	16000	400	no
1CSurr	2	2	55283	600	yes
2CDT	2	2	16000	400	yes
2CHT	2	2	16000	400	yes
4CE1CF	5	2	173250	750	no
4CR	4	2	144400	400	no
4CRE-V2	4	2	183000	1000	yes
5CVT	5	2	40000	1000	yes
FG_2C_2D*	2	2	200000	2000	no
GEARS_2C_2D	2	2	200000	2000	no
MG_2C_2D*	2	2	200000	2000	yes
UG_2C_2D*	2	2	100000	1000	yes
UG_2C_3D*	2	3	200000	2000	yes
UG_2C_5D*	2	5	200000	2000	yes
keystroke	4	10	1600	200	DNK

TABLE II
Average Accuracy

DATASETS	COMPOSE (GMM)	LEVEL_{IW}
1CDT	99.8563	**99.9250**
1CHT	99.3462	**99.5187**
1CSurr	89.7210	**91.3025**
2CDT	**95.9169**	58.3250
2CHT	**89.6350**	52.1500
4CE1CF	93.9038	**97.7403**
4CR	99.9896	**99.9924**
4CRE-V2	**92.3026**	24.1011
5CVT	45.0988	33.1021
FG_2C_2D	95.5025	**95.7115**
GEARS_2C_2D	95.8284	**97.7380**
MG_2C_2D	**93.2010**	85.4445
UG_2C_2D	**95.7130**	74.3370
UG_2C_3D	**95.2056**	64.6890
UG_2C_5D	**92.1265**	80.1720
keystroke	87.2125	**90.5625**

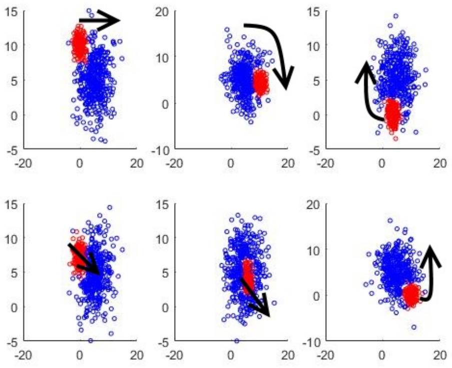

Fig. 2. Six different snapshots of the *1CSurr* dataset with arrows representing the drift direction of red class throughout the experiment

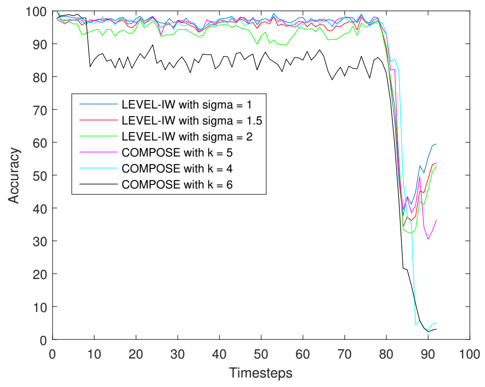

Fig. 3. Accuracy comparison of COMPOSE vs. LEVEL$_{\text{IW}}$ on 1CSurr for various values of their free parameters.

Fig. 3 shows the classification accuracy of both LEVEL$_{\text{IW}}$ and COMPOSE with different values of their respective primary parameters: k, the number of clusters for COMPOSE and σ, the value of the kernel bandwidth for LEVEL$_{\text{IW}}$. During the first 80 time steps, corresponding to the modest class overlap during the circumnavigation of one class around the other, LEVEL$_{\text{IW}}$ performance is consistently high for all three values of σ, whereas COMPOSE performance is significantly lower for $k = 6$, which is only 1 higher than the optimal value of $k = 5$ (note that k is not the number of classes). A dramatic drop in performance is seen for both algorithms and for all cases at time step 80, which corresponds the class crossover; since there is complete class overlap during this interval, the performance drop is to be expected. With no labeled data coming thereafter, no algorithm can recover from this scenario – with or without optimal parameter values – though LEVEL$_{\text{IW}}$ performance is still more *consistent* and *stable* for different values of σ, while COMPOSE performance fluctuates significantly for different values of k.

The *UG_2C_2D* dataset is also a two-class problem where two classes are rotating around each other. There is class overlap at all time steps, but with varying degree: the class overlap increases between time steps 0 and 20, decreases between time steps 20 and 40. Then, between time steps 60

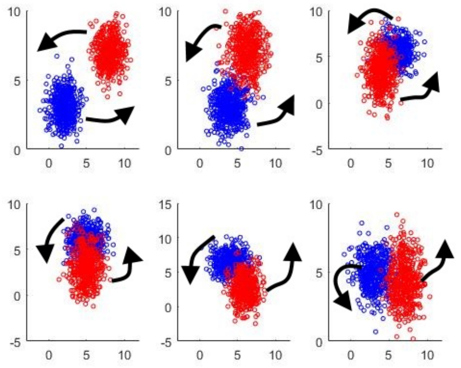

Fig. 4. UG_2C_2D data for six different snapshots with arrows representing the drift direction of classes throughout the experiment

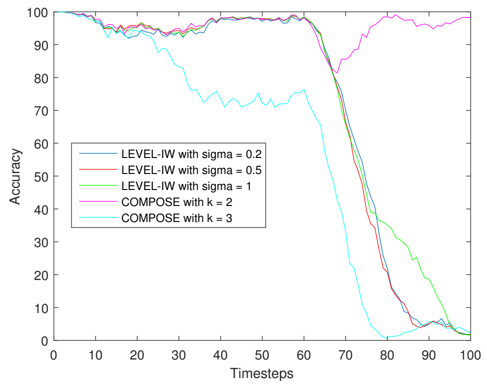

Fig. 5. Accuracy comparison of COMPOSE vs. LEVEL$_{\text{IW}}$ on UG_2C_2D.

and 70 overlap increases again, but this time significantly. After time step 70, the classes separate. Fig. 4 illustrates the behavior of the dataset at six different time snapshots.

This is a more interesting dataset and explains the performance and behavior of both algorithms very clearly, as shown in Fig. 5. Both algorithms do well when there is only modest overlap, and when using the optimal values of their free parameters, $k = 2$ and $\sigma = 0.5$. However, while LEVEL$_{\text{IW}}$ performance is consistent and stable for other σ values that differ significantly from its optimal value, COMPOSE performance drops dramatically for $k = 3$, the smallest change possible from its optimal value. At time step 60, when there is significant class-overlap, both algorithms see a drop in their performance (as expected); however COMPOSE is able to recover when the classes separate, whereas LEVEL$_{\text{IW}}$ cannot. Of course, COMPOSE recovery is subject to correct and exact choice of the optimal value of $k = 2$, as it too cannot recover for $k = 3$.

Finally, let's also take a look at the *keystroke* dataset, a real-world dataset that contains the information from the keystrokes dynamics obtained from the users who type a fixed password, *.tie5Roan1*, followed by the *Enter* key 400 times in 8 sessions performed on different days. The task of the classifier is to classify each one of four different users over time according to their typing profile. Fig. 6 shows the classification accuracy

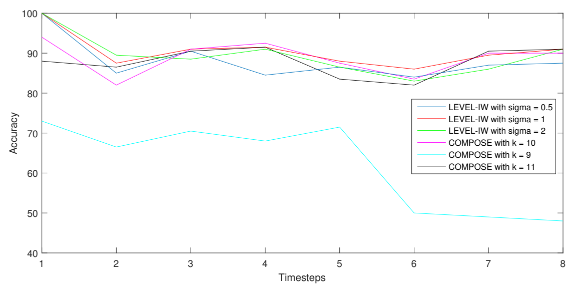

Fig. 6. Accuracy comparison of COMPOSE vs. LEVEL$_{IW}$ on keystroke.

obtained by both LEVEL$_{IW}$ and COMPOSE on this dataset using various values of their respective free parameters. Since this is a 10-dimensional real-world dataset, we do not know (indicated as DNK in Table 1) whether this dataset have class-overlap. However, the robustness of LEVEL$_{IW}$ with respect to the selection of its free parameter, and relative high sensitivity of COMPOSE to its free parameter is also visible on this dataset.

Given these observations, we now take a deeper and more methodical look at the parameter sensitivity of both algorithms.

B. Parameter Sensitivity Analysis

One particular benefit observed above with LEVEL$_{IW}$ appears to be its relative resilience (less sensitivity) to the choice of its primary free parameter, the Gaussian kernel width σ as used by IWLSPC, compared to the number of clusters k as used by the SSL algorithm in COMPOSE. To determine whether this benefit is consistent across the datasets, we conducted a series of parameter sweep experiments, running each algorithm with different values of its free parameter.

Table III shows the results obtained by COMPOSE using cluster-and-label, where for each dataset, we provided the COMPOSE performance with the optimal k value, as well as k incorrectly chosen by just "1." This ± 1 represents the smallest possible change in k around its optimal value. For example, if the optimal value is $k = 4$, the three values of k used for comparison are $k = 3, k = 4$, and $k = 5$. When optimal k is two, the selection of $k = 1$ is, of course, meaningless, as $k = 1$ would result in all instances being classified into the same class. Hence, such cases are indicated as N/A in Table III. We observe that in most datasets changing the value of k from the optimal value just by 1, significantly and catastrophically reduces the average accuracy for that dataset. The datasets where this is observed most dramatically are the ones that show class overlap. Perhaps not so surprisingly, if the data has no class overlap, "increasing" the value of k from the optimal value does not hurt the performance as cluster and label can easily find more clusters in the data.

The free parameter for LEVEL$_{IW}$ is the value of the kernel width σ as used in Gaussian kernel. The parameter sweep range was chosen to cover a range commonly known to work well in other algorithms that use Gaussian kernels, and included the values of 0.01, 0.1, 0.2, 0.5, 1, 1.2, 1.5, 2, and 5. Perhaps not too unsurprisingly, the extreme values never yielded the best results. In Table IV, we show the performance of LEVEL$_{IW}$ for each of the datasets with three different values of σ, representing the smallest and largest values of σ on which the algorithm performed the best, and an additional value in the middle of two. We observed that LEVEL$_{IW}$ is surprisingly robust to such wide fluctuations of σ values of typically five fold, and sometimes as wide as an order of magnitude difference. This outcome shows the consistent and stable performance of LEVEL$_{IW}$, its most prominent advantage over COMPOSE.

TABLE III
ACCURACY WITH THREE DIFFERENT VALUES OF K FOR CLUSTER AND LABEL (COMPOSE)

DATASETS	Reduced k (Accuracy)	Optimal k (Accuracy)	Increased k (Accuracy)
1CDT	N/A	k=2 (99.85)	k=3 (99.76)
1CHT	N/A	k=2 (99.34)	k=3 (98.72)
1CSurr	k=4 (88.38)	k=5 (89.72)	k=6 (77.81)
2CDT	N/A	k=2 (95.91)	k=3 (52.91)
2CHT	N/A	k=2 (89.63)	k=3 (77.33)
4CE1CF	k=4 (78.96)	k=5 (93.90)	k=6 (94.66)
4CR	k=3 (74.88)	k=4 (99.98)	k=5 (99.98)
4CRE-V2	k=3 (25.13)	k=4 (92.30)	k=5 (22.78)
FG_2C_2D	k=3 (68.91)	k=4 (95.50)	k=5 (95.44)
GEARS_2C_2D	N/A	k=2 (95.82)	k=3 (87.99)
MG_2C_2D	k=3 (65.32)	k=4 (93.20)	k=5 (92.07)
UG_2C_2D	N/A	k=2 (95.71)	k=3 (56.28)
UG_2C_3D	N/A	k=2 (95.20)	k=3 (91.46)
UG_2C_5D	N/A	k=2 (92.12)	k=3 (88.03)
keystroke	k=9 (68.62)	k=10 (87.21)	k=11 (81.56)

TABLE IV
ACCURACY WITH THREE DIFFERENT VALUES OF SIGMA (LEVEL$_{IW}$)

DATASETS	lowest sigma (Accuracy)	Middle sigma (Accuracy)	Highest sigma (Accuracy)
1CDT	0.2 (99.91)	1 (99.91)	2 (99.92)
1CHT	0.2 (99.40)	1 (99.42)	2 (99.51)
1CSurr	1 (91.30)	1.5 (90.00)	2 (87.79)
2CDT	0.2 (58.32)	0.5 (50.32)	1 (50.48)
2CHT	0.2 (50.10)	0.5 (50.89)	1 (52.15)
4CE1CF	0.2 (97.74)	0.5 (97.12)	1.5 (92.40)
4CR	0.2 (99.99)	1 (99.99)	2 (99.99)
4CRE-V2	0.2 (20.96)	0.5 (20.84)	1 (24.10)
FG_2C_2D	0.2 (95.71)	0.5 (86.41)	1 (94.28)
GEARS_2C_2D	0.2 (97.73)	1 (95.28)	2 (95.36)
MG_2C_2D	0.2 (78.03)	0.5 (78.21)	1.2 (85.44)
UG_2C_2D	0.2 (70.61)	0.5 (71.81)	1 (74.33)
UG_2C_3D	0.1 (61.21)	1 (64.30)	2 (64.68)
UG_2C_5D	0.5 (77.67)	0.5 (80.07)	1.5 (80.17)
keystroke	0.5 (88.12)	1 (90.56)	2 (89.43)

V. CONCLUSION & FUTURE WORK

We described an importance weighting based approach for addressing concept drift in the presence of extreme verification latency, where data with a changing underlying distribution arrive in a continuous streaming fashion with only unlabeled data provided beyond the initialization step. Traditionally, importance weighting is used for domain adaptation problems characterized in a single time step, where the change in distribution is simply the mismatch between training and test dataset, with no consideration of streaming data or extreme verification latency.

The primary contribution of this effort is therefore a modification to the importance weighted least squares probabilistic classifier so that it can work within a) a streaming data environment and b) when there is extreme verification latency. The proposed approach is called Learning Extreme VErification Latency with Importance Weigthing (LEVEL_{IW}) and is compared in accuracy and parameter sensitivity to our prior work in this setting, called COMPOSE. Over fifteen datasets, the two algorithms exchanged claims to better accuracy, with COMPOSE generally performing better when there is significant class-overlap, but only if – and that is an important if – its free-parameter is chosen correctly.

While LEVEL_{IW} showed modest to moderate advantage in accuracy over COMPOSE when the class overlap itself was modest, its primary benefit was revealed when we observed its robustness and higher tolerance (less sensitivity) to fluctuations around the optimal value of its free parameter. Of course, we should note that in all cases where LEVEL_{IW} performed relatively poorly in comparison to COMPOSE, the dataset had experienced drifting posterior probabilities, a violation of the primary assumption of importance sampling approaches.

Further work is needed to further verify whether these general observations generalize to other datasets and scenarios. Specifically, additional work is needed in generating other challenging synthetic dataset scenarios and additional real datasets, and evaluating LEVEL_{IW} on datasets that contain abrupt drift, recurring concepts, feature or class noise, and high dimensional features. It is, after all, a challenging dataset or scenario, or a collection thereof that provide the motivation for the development of specialized algorithms within a specific disciple.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant nos. 1310496 and 1429467.

References

[1] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, 2004.

[2] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[3] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: a survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[4] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. The MIT Press, 2009.

[5] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[6] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," in *Advances in neural information processing systems*, 2006, pp. 601–608.

[7] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning under covariate shift," *Journal of Machine Learning Research*, vol. 10, no. Sep, pp. 2137–2155, 2009.

[8] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Advances in neural information processing systems*, 2008, pp. 1433–1440.

[9] M. Muhlbaier and R. Polikar, "Multiple classifiers based incremental learning algorithm for learning in nonstationary environments," *International Conference on Machine Learning and Cybernetics*, pp. 3618–3623, 2007.

[10] M. Karnick, M. Ahiskali, M. Muhlbaier, and R. Polikar, "Learning concept drift in nonstationary environments using an ensemble of classifiers based approach," *International Joint Conference on Neural Networks*, pp. 3455–3462, 2008.

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

[12] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 2283–2301, 2013.

[13] J. Kolter and M. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, July 2007.

[14] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," *ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 377–382, 2001.

[15] G. Ditzler and R. Polikar, "Semi-supervised learning in nonstationary environments," *International Joint Conference on Neural Networks*, pp. 2741–2748, 2011.

[16] K. B. Dyer, R. Capo, and R. Polikar, "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 12–26, 2014.

[17] R. Capo, A. Sanchez, and R. Polikar, "Core support extraction for learning from initially labeled nonstationary environments using compose," *International Joint Conference on Neural Networks*, pp. 602–608, 2014.

[18] G. R. Marrs, R. J. Hickey, and M. M. Black, "The impact of latency on online classification learning with concept drift," in *International Conference on Knowledge Science, Engineering and Management*. Springer, 2010, pp. 459–469.

[19] G. Krempl, "The algorithm apt to classify in concurrence of latency and drift," *Intelligent Data Analysis*, pp. 223–233, 2011.

[20] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classication guided by clustering on nonstationary envi- ronments and extreme verication latency," *SIAM International Conference on Data Mining*, pp. 873–881, 2015.

[21] V. M. A. Souza, D. F. Silva, G. E. A. P. A. Batista, and J. Gama, "Classification of evolving data streams with infinitely delayed labels," *IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 214–219, 2015.

[22] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," *Very Large Data Bases*, vol. 29, pp. 81–92, 2003.

[23] H. Hachiya, M. Sugiyama, and N. Ueda, "Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition," *Neurocomputing*, vol. 80, pp. 93–101, 2012.

[24] M. Sugiyama, M. Krauledat, and K.-R. MÃžller, "Covariate shift adaptation by importance weighted cross validation," *Journal of Machine Learning Research*, vol. 8, no. May, pp. 985–1005, 2007.

[25] T. Kanamori, S. Hido, and M. Sugiyama, "A least-squares approach to direct importance estimation," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1391–1445, 2009.