# Learning with Asynchronous Labels

YU-YANG QIAN, National Key Laboratory for Novel Software Technology, China and School of Artificial Intelligence, Nanjing University, China
ZHEN-YU ZHANG, National Key Laboratory for Novel Software Technology, Nanjing University, China and RIKEN Center for Advanced Intelligence Project, Tokyo, Japan
PENG ZHAO, National Key Laboratory for Novel Software Technology, China and School of Artificial Intelligence, Nanjing University, China
ZHI-HUA ZHOU*, National Key Laboratory for Novel Software Technology, China and School of Artificial Intelligence, Nanjing University, China

Learning with data streams has attracted much attention in recent decades. Conventional approaches typically assume that the feature and label of a data item can be timely observed at each round. In many real-world tasks, however, it often occurs that either the feature or the label is observed firstly while the other arrives with delay. For instance, in distributed learning systems, a central processor collects training data from different sub-processors to train a learning model, whereas the feature and label of certain data items can arrive asynchronously due to network latency. The problem of learning with *asynchronous* feature or label in streams encompasses many applications but still lacks sound solutions. In this paper, we formulate the problem and propose a new approach to alleviate the negative effect of asynchronicity and mining asynchronous data streams. Our approach carefully exploits the timely arrived information and builds an online ensemble structure to adaptively reuse historical models and instances. We provide the theoretical guarantees of our approach and conduct extensive experiments to validate its effectiveness.

CCS Concepts: • **Computing methodologies → Machine learning**.

Additional Key Words and Phrases: Data streams, Asynchronous data, Weakly supervised learning, Ensemble methods

## 1 INTRODUCTION

Machine learning has achieved great success in a variety of applications [12]. In the supervised learning setup, a predictive model is built on a collection of training data and then deployed in the testing phase. However, in many real-world applications, data arrive continuously over time and thus are collected in the form of a stream. Therefore, a learning model for streaming data should be able to update online, respond timely and be robust to environmental uncertainty, which poses new challenges to the design of stream learning algorithms.

In conventional streaming learning, at the beginning of each timestamp, the predictive model receives the feature of a data item and is required to make a prediction, and subsequently, the model will receive the ground-truth label as the feedback at the end of this timestamp. Notably, both the feature and ground-truth label of this current data item can be observed at this round. However, it is common in practice that features and labels could arrive at *different* timestamps due to system delays or network latency. For instance, in IoT monitoring applications, a central processor aims to predict the abnormal status of IoT devices based on its received network communication data (feature) and a monitor device (label). As IoT devices are geographically spread out, the feature and label of a data item usually arrive at different timestamps due to network latency [27]. As shown in Figure 1, at timestamp $t = 1$, the feature $\mathbf{x}_1$ of an IoT device is received instantly, while its label $y_1$

*Corresponding Author

Authors' address: Yu-Yang Qian[1,2]; Zhen-Yu Zhang[1,3]; Peng Zhao[1,2]; Zhi-Hua Zhou[1,2],
1. National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China,
2. School of Artificial Intelligence, Nanjing University, Nanjing, China,
3. RIKEN Center for Advanced Intelligence Project, Tokyo, Japan.

is received at $t = 2$ due to transmission delay; at timestamp $t = 4$, we receive the instant label $y_4$, and its delayed feature $\mathbf{x}_4$ is received at $t = 5$. Such an issue of asynchronicity also occurs in the task of taxi-compliant predictions, where a central server receives videos and customer complaints through the network. We aim to instantly evaluate the probability of customer complaints based on video records. User complaints (label) are usually not reported until they respond, and the video records (feature) require a certain time to be uploaded to the server through the network, which results in the potential asynchronicity in the data streams.

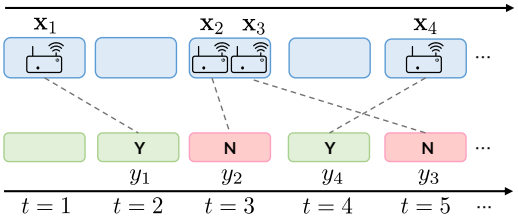



Fig. 1. A typical data stream with asynchronous labels, where we receive a bunch of feature(s) and label(s) at each time. The possible delay at both the label and the feature levels is the key challenge of the asynchronous learning setup.

The asynchronicity issue poses a great challenge to the streaming learning. Most previous streaming learning algorithms [21, 48, 53, 54] usually assume that the feature and label pair is revealed simultaneously, and thus are typically not tailored for data streams with delayed feature or label. There is a line of studies explore the *delayed labels* problem [6, 20, 46], wherein the label of each sample may be disclosed with an unknown delay, whose approaches can be roughly categorized into *wait-and-update* methods and *historical-data-reaccess* methods. Specifically, a wait-and-update method will wait for delayed labels until they arrive and then update the model [3, 32]; historical-data-reaccess algorithms store a handful of representative historical data and handle delayed labels by re-training them [34, 39, 45]. Although these techniques designed for delayed labels can be helpful for tackling the asynchronicity issue, they cannot be directly deployed to address our problems. The wait-and-update methods are too general to capture instantly available information in asynchronous streams, and historical-data-reaccess methods typically need to store plenty of historical data and retrain the model, which is costly in streaming learning. In summary, these two kinds of methods update the model using the fully arrived feature-label pairs, while do not exploit partially arrived features and labels, which are instantly available in our scenario.

In this paper, we initiate the study of *stream learning with asynchronous labels*, where the feature and label of a certain sample within the data stream can arrive at different time stamps. This problem takes place in many real-world tasks but is rarely studied in the literature. We emphasize that the possible delays at both the label and feature levels in streaming learning is one of the key challenges of the asynchronous label learning setup. As opposed to previous algorithms that only handle the simplified setting of delayed labels and update the model only when the feature-label pair both arrived, we aim to exploit the instantly arrived partial feature or label to update the model in an online manner, which is crucial for alleviating the asynchronicity and enjoy both empirical and theoretical improvements in the asynchronous setup. To this end, we propose a novel online ensemble approach named *Learning AsynChronous labels with Hint* (LACH). Our approach first constructs a set of base models to handle the asynchronously arrived features and labels. Based on the essential observation that only part of an instance is delayed while either the feature or the label is instantly available, we extract these instant side information to further benefit the model's online updating procedure. The proposed LACH algorithm enjoys sound regret guarantee and achieves remarkable performance improvements on synthetic examples, benchmark datasets, and real-world applications. Our main contributions are as follows.

(1) We initiate and investigate the challenging problem of stream learning with asynchronous labels, which accommodates many real-world tasks but was not studied in prior works.

(2) We propose a novel online ensemble algorithm with novel designs, in which we maintain a bunch of base models to deal with the asynchronous issue, and then exploit instantly arrived feature or label to further benefit model updating. Theoretical analysis is also presented to justify the rationality of our designed method.
(3) We conduct extensive empirical evaluations on synthetic, benchmark datasets, and real-world applications to demonstrate the superiority of our proposed algorithm.

The rest of this paper is organized as follows. The related works are discussed in Section 2. We formulate the problem in Section 3, followed by our algorithm in Section 4 and theoretical analyses in Section 5. Then we present detailed proofs in Section 6. Experiments on synthetic, benchmark and real-world datasets are presented in Section 7. Finally, we conclude our work in Section 8.

## 2 RELATED WORK

In the following, we discuss the related topics of stream learning with asynchronous labels. To the best of our knowledge, there is no sound solution for this problem yet. Previously streaming learning algorithms [21, 25, 47, 48, 52–54] typically assume that the feature and label pair is disclosed simultaneously, and not tailored for data streams with delayed feedback. For example, Wu et al. [48] propose an online streaming feature selection algorithm based on feature relevance, which maintains a small set of features to maximize the predictive performance with incremental features. Zhao et al. [54] propose an ensemble-based method to adaptively adjust the weights of historical models according to the performance on the current data batch to deal with non-stationary data streams. However, these methods typically assume that labels are revealed instantly after making the prediction, and thus cannot be directly applied to deal with the asynchronicity issue and are unable to mine the structure of the asynchronous data streams.

The most related topic is learning with delayed labels, which considers the simplified setting that the delay only occurs in label level (note that the delay can occur in both label and feature levels in our case) [6, 14, 17, 19, 20, 46, 50, 58]. Below, we briefly discuss the research efforts in this thread, which can be generally categorized into wait-and-update and historical-data-reaccess methods.

**Wait-and-Update Methods.** This line of work typically considers the issue of delayed gradient feedback, i.e., regardless of the feature or label of a data item that comes first, we can wait for the other till it arrives and update the model with delayed feedback. The pioneering work of Weinberger and Ordentlich [46] consider the fixed delay and introduces a 'divide and conquer' method. They construct multiple base models to handle the delayed gradients. Joulani et al. [20] later extend this method to deal with the cases of stochastic and adversarial delays. For a fully adversary case, Quanrud and Khashabi [32] prove that online gradient descent already obtains minimax optimal regret under unknown delays. That is, at every iteration $t$, the learner can directly perform online gradient descent with delayed gradients. Bedi et al. [3] propose an online augmented Lagrangian algorithm to handle the delayed gradients.

Recently, Flaspohler et al. [10] provide a formal reduction of the delayed online learning problem to a non-delayed optimistic online learning problem, and achieve the optimal regret for online convex optimization. Subsequent work [40] investigates the strongly convex functions in online learning with unknown delays scenarios, and establishes a better regret bound. Similar results are further extended to projection-free online learning [41, 42] and bandit online learning [43]. Goyal et al. [13] proposed an SSGD (Synchronized SGD) method which aggregates gradients on all the nodes to update the current model. These approaches consider the general delayed feedback but do not take the instant side information into consideration, thus are not able to tackle the challenges of both label and feature delays and become sub-optimal in the asynchronous data streams.

**Parameters**: Predictor set $\mathcal{W}$, feature space $\mathcal{X}$, loss function $\ell$, time horizon $T$ (optional).
At each time stamp $t = 1, \ldots, T$:
1. Observe feature $\mathbf{x}_t \in \mathcal{X}$ or label $y_t \in \mathcal{Y}$
2. Receive $k$ delayed label(s) $\{y_{t-d_1}, \ldots, y_{t-d_k}\}$ and $k'$ delayed feature(s) $\{\mathbf{x}_{t-d'_1}, \ldots, \mathbf{x}_{t-d'_{k'}}\}$
3. Use asynchronous feedback to update the model $\mathbf{w}_t \in \mathcal{W}$ (and make prediction)
4. The label $y_t \in \mathcal{Y}$ or the feature $\mathbf{x}_t \in \mathcal{X}$ is scheduled to be received after $d_t$ time stamps

Fig. 2. The protocol of streaming data learning with asynchronous labels. The feature or label of a sample can arrive at different time stamps with an unfixed delay.

**Historical-data-Reaccess Methods.** Some works consider dealing with the delayed labels by storing historical data and retraining the model. Souza et al. [36] seek to classify evolving data streams with infinitely delayed labels by a clustering algorithm. Plasse and Adams [30] assume the data are generated from a Gaussian distribution and propose an algorithm based on forgetting factor. Su et al. [39] use two models to deal with the issue: a predictor to make predictions and an extra time-delay model to estimate delays. They use the EM algorithm to optimize two models jointly. Saito et al. [34] design a dual learning algorithm that alternatively optimizes a predictor and bias estimator based on historical data. Wang et al. [45] transform the delays into discrete slots and estimate which slot the label will fall into based on observed data. AdaDelay [37] adds a penalty to the delayed gradients. They decrease the learning rates of delayed gradients to alleviate the delayed gradient problem. Albeit with promising empirical performance, the theoretical properties of those methods are generally unclear. Besides, these methods typically need to store the entire dataset and retrain the model, which is costly in practice streaming data learning scenarios [4, 9, 18].

The above-mentioned wait-and-update and historical-data-reaccess methods mainly consider delays in the label level, while in asynchronous streaming learning problems, we further need to consider delays in the feature level and mine the instant arrived feature or label to alleviate the negative impact of asynchronous feedback and benefit model's online updating. Meanwhile, we aim to explore the instantly arrived partial information of features or labels to update the model in an online fashion. However, wait-and-update methods are too general to capture instant information in asynchronous data streams, and historical-data-reaccess methods typically need to store historical data and retrain the model at each round, which is costly in the data stream learning. Although the techniques designed for tackling the delayed labels are greatly helpful for our purpose, all those methods do not consider the more challenging issue of potential delays in the feature level and thus become sub-optimal in the asynchronous scenarios. The key challenge of the asynchronous label learning setup is the possible delays in both label and feature levels in the streaming learning. Unlike previous methods that only take the delayed labels into consideration, we explore non-asynchronous information to update the models in a timely manner to improve the performance both empirically and theoretically.

## 3 PROBLEM FORMULATION

We formulate the problem of streaming learning with asynchronous labels as an iterative game between a learner and an adversary, similar to the protocol of classic online learning [5]. At the beginning of each timestamp, the adversary sends a data item to the learner, in which its feature $\mathbf{x}_t$ is sampled from a fixed set $\mathcal{X} \in \mathbb{R}^N$ and its ground-truth label $y_t$ from label space $\mathcal{Y} = \{1, ..., C\}$. Due to the transmission delays, the learner might receive the feature and label at different time stamps. Specifically, the learner might only observe a part of the current data item, i.e., its feature

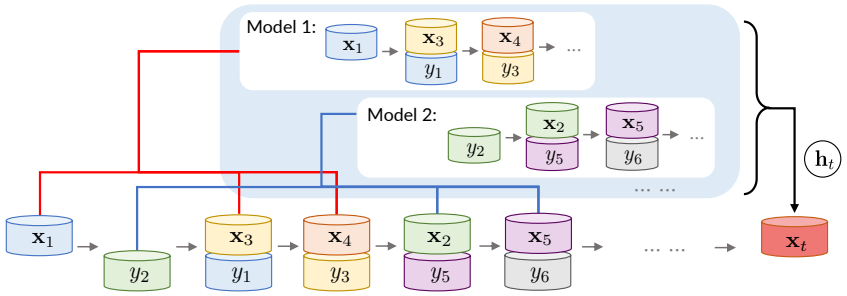



Fig. 3. Handling asynchronous labels by multiple base models. We divide the data stream into several sub-sequences according to asynchronously arrived features and labels. Each sub-sequence is addressed by a base model in which the feedback is non-asynchronous.

or label, and receive (multiple) features or labels of previous instances at the same time. If the learner only receives $\mathbf{x}_t$ and its $y_t$ is not available, the learner requires to output a classifier $\mathbf{w}_t$ from a fixed set $\mathcal{W}$ and make prediction $f(\mathbf{w}_t; \mathbf{x}_t)$ instantly, where $f : \mathcal{X} \times \mathcal{W} \mapsto \mathbb{R}$ is a hypothesis function. In addition, the learner is required to update the model in an online fashion for streaming data. We denote the time interval between data generation and its arrival by $d_t \in \mathbb{N}$, that is, the feature or label of a data item is received with an unfixed delay $d_t \in \mathbb{N}$. When $d_t \equiv 0$, we recover the standard (non-asynchronous) streaming learning setting. Throughout this work, we assume that for every data item, at least one part, i.e., $\mathbf{x}_t$ or $y_t$, can be observed at each time. Otherwise, the learning model can hardly make effective updates unless imposing strong prior knowledge on the data generation processes. We summarize the overall protocol in Figure 2.

## 4 PROPOSED METHOD

In this section, we establish the algorithm for streaming learning with asynchronous labels. In our approach, we handle delays at both the feature and label levels in the data stream while simultaneously utilizing instantly arriving features or labels. In particular, we construct a set of base models to handle the asynchronously arrived features and labels, and meanwhile, introduce a hint sequence to the online model updating process by exploring the instant side information and timely updating each base model. Finally, we introduce two novel mechanisms to improve the algorithm's storage and time efficiency by dynamically adjusting the base model pool.

### 4.1 Handling Asynchronous Labels

As a first step, we aim to handle the delays that occur at both the feature and label levels. Intuitively, if we can exploit the structure of the delayed sequence in order to bridge the gap between the timestamp of receiving the asynchronous data item and the current timestamp, we can explore historical data to alleviate the negative effect of asynchronicity. To this end, we first reconstruct the data stream according to the asynchronously arrived features and the labels. Enlightened by the insightful work of Joulani et al. [20], we construct several base models and propose a scheduling algorithm to handle the asynchronicity in both the feature and label levels.

Specifically, at each time, we select an available base model from the model pool to update and make the prediction according to the available features or the labels. The algorithm runs as follows. At each time, a base model is available if it has received the feedback corresponding to its previous prediction, e.g., the delayed feature or label arrives. Before receiving the feedback, we consider a base model as unavailable because it is waiting for the feedback. When we need to make

a prediction, we use an available base model because it is ready to make predictions. If no such base model exists, we create a new one to be used.

Our scheduling algorithm equals to splitting the original stream of length $T$ into several sub-sequences $\mathcal{T}_1, \mathcal{T}_2, \ldots$ with corresponding base models, respectively. Each sub-sequence $\mathcal{T}_i = \{i, i + d_i^1, i + d_i^2, \ldots\}$ is a sub-problem handled by the $i$-th base model, where the feature and label arrive in time. We give an example for explanation. As shown in Figure 3, model 1 and model 2 are employed for the sub-sequence $t = 1, 3, 4$ and $t = 2, 5, 6$ respectively, where each base model is handling its own sub-problem without any asynchronicity.

We notice that there does exist some instant side information available at each time, such as the instantly arrived features or labels. It is crucial to exploit these instant side information for model updating in streaming data learning. Unlike previous studies waiting for the arrival of both delayed feature and label, we explore these non-asynchronous information to update the base models in a timely manner to improve the model's performance. To this end, we introduce a *hint sequence* that can be generated by these non-asynchronous information for model updating [33]. For each base model, we then perform the following two-step updates,

$$\begin{aligned} \widehat{\mathbf{w}}_t &= \arg\min_{\mathbf{w}\in\mathcal{W}} \ \eta\langle \nabla \ell_{t-d'_t}(\mathbf{w}_{t-d'_t}), \mathbf{w}\rangle + \|\mathbf{w} - \widehat{\mathbf{w}}_{t-d'_t}\|_2^2, \\ \mathbf{w}_t &= \arg\min_{\mathbf{w}\in\mathcal{W}} \ \eta\langle \mathbf{h}_t, \mathbf{w}\rangle + \|\mathbf{w} - \widehat{\mathbf{w}}_t\|_2^2, \end{aligned} \tag{1}$$

where $\{\mathbf{h}_t\}_{t=1}^T$ is the hint (vector) sequence and $\nabla \ell_{t-d'_t}(\mathbf{w}_{t-d'_t}) = \nabla \ell(f(\mathbf{w}_{t-d'_t}; \mathbf{x}_{t-d'_t}), y_{t-d'_t})$ is the gradient of the matching feature-label pair $(\mathbf{x}_{t-d'_t}, y_{t-d'_t})$ we received at time $t$ (i.e., the feature or the label of time $t - d'_t$ is received with a delay of $d'_t$), $\eta > 0$ is step size.

The hint sequence $\{\mathbf{h}_t\}_{t=1}^T$ serves as a guessing of the next move, which is currently unknown. Provably, a more precise guessing of the next move will result in a better performance [8, 33]. While our primary focus here is on prediction problems, we remark that an accurate estimation of the next move can also be considered a rehearsal for the future [62], which is beneficial in decision-making problems. In the proposed method, we exploit instant features or labels to generate the hint sequence $\{\mathbf{h}_t\}_{t=1}^T$ and use this hint sequence to help updating the model. We emphasize that the property of instantly model updating by capturing side information plays a vital role in our algorithm to tackle the key challenge of the problem, as we explore instant arrived feature or label to alleviate the negative impact of asynchronous feedback and benefit model's online updating.

We summarize the proposed algorithm in Algorithm 1. Note that when the budget for the number of generated base models is limited, we propose two mechanisms to dynamically add and drop the base models. More details about the algorithm's efficiency consideration are given in Section 4.3.

### 4.2 Mining Instant Information

In the proposed approach, we introduce a hint sequence to explore instant features or labels to update the model and improve prediction performance. In the following, we discuss the implementation details for timely hints generation. Specifically, when the labels are delayed, we explore instant feature to generate hints by online model ensemble; when the features are delayed, we explore instant labels by sketches on previous data features.

**Exploring Instant Feature by Ensemble**. When the label is delayed, we propose a hints generation method by online ensemble [57, 60] to explore the current feature. By the observation that the base models with better recent performance will be more accurate on the current unlabeled data in many practice scenarios, we combine the base models to estimate the gradient of the current unlabeled instance. Specifically, we measure the quality of each base model by the cumulative

**Algorithm 1** Learning AsynChronous labels with Hint

**Input:** Step size $\eta$

Initialize model queues $Q_{\text{all}}$ and $Q_{\text{ready}}$, where $Q_{\text{all}}$ and $Q_{\text{ready}}$ are empty at the beginning
**for** $t = 1, \ldots, T$ **do**
  **if** receive $k$ delayed label(s) $y_{t-d_1}, \ldots, y_{t-d_k}$ and $k'$ delayed feature(s) $\mathbf{x}_{t-d'_1}, \ldots, \mathbf{x}_{t-d'_{k'}}$ **then**
    Check if exists a pair $(\mathbf{x}_{t-d'_t}, y_{t-d'_t})$, add model $\widehat{\mathbf{w}}_{t-d'_t}$ and gradient $\nabla \ell_{t-d'_t}(\mathbf{w}_{t-d'_t})$ in $Q_{\text{ready}}$
  **end if**
  **if** $Q_{\text{ready}}$ is empty **then**
    Initialize a new model $\widehat{\mathbf{w}}_t$
  **else**
    Use $Q_{\text{ready}}$'s earliest model $\widehat{\mathbf{w}}_{t-d'_t}$ and update by $\nabla \ell_{t-d'_t}(\mathbf{w}_{t-d'_t})$ with (1) to get $\widehat{\mathbf{w}}_t$
    Remove $\widehat{\mathbf{w}}_{t-d'_t}$ from $Q_{\text{ready}}$, remove $\mathbf{w}_{t-d'_t}$ from $Q_{\text{all}}$
  **end if**
  Send delayed pair $(\mathbf{x}_{t-d'_t}, y_{t-d'_t})$ and instant $\mathbf{x}_t$ or $y_t$, $N_t = |Q_{\text{all}}|$ to Algorithm 2 to get hint $\mathbf{h}_t$
  Update model $\mathbf{w}_t$ by $\mathbf{h}_t$ with (1), add $\mathbf{w}_t$ into $Q_{\text{all}}$
**end for**

discounted loss. For the $i$-th base model $\mathbf{w}_i$, the cumulative discounted loss is defined as

$$L_t(\mathbf{w}_i) = (1 - \gamma_{\text{ft}}) L_{t-1}(\mathbf{w}_i) + \gamma_{\text{ft}} \ell_{t-d'_t}(\mathbf{w}_i), \tag{2}$$

where $L_t(\mathbf{w}_i)$ is the cumulative loss for base model $\mathbf{w}_i$ at time $t$, and $\ell_{t-d'_t}(\mathbf{w}_i)$ is the delayed loss feedback we received at time $t$ for the $\mathbf{w}_i$. The discounted factor $\gamma_{\text{ft}}$ plays a role of determining the importance of recent performance, i.e., if the distribution changes slowly, we will use a smaller $\gamma_{\text{ft}}$ which means the performance on previous data is more reliable. This is also widely used in tracking time-varying control system [15] and non-stationary online learning and prediction [44, 55].

Next, we combine the base models to obtain $\mathbf{h}_t$ according to their cumulative discounted loss in an exponential weighted manner. Suppose we have a total of $N_t$ base models at time $t$, let $\beta \in \Delta_{N_t}$ be a weight distribution defined on base models. We define the weight of model $\mathbf{w}_i$ as

$$\beta_i = \frac{\exp\left(-L_t(\mathbf{w}_i)\right)}{\sum_{j=1}^{N_t} \exp\left(-L_t(\mathbf{w}_j)\right)}. \tag{3}$$

We then estimate the labels and obtain $\mathbf{h}_t$,

$$\widetilde{y}_t = \sum_{i=1}^{N_t} \beta_i f(\mathbf{w}_i; \mathbf{x}_t); \quad \mathbf{h}_t = \nabla \ell(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), \widetilde{y}_t), \tag{4}$$

where $\widetilde{y}_t$ is the weighted combination of base models using weights $\beta$.

**Exploring Instant Label by Sketches**. When the feature information is delayed, we generate the hints by exploring instant labels, in which we sketch the historical data stream by maintaining a mean vector of each class. Inspired by the previous work of Li et al. [26], for each class, we maintain a *discounted label sketch*, which is defined as follows,

$$\mathbf{m}_t^c = (1 - \gamma_{\text{lb}}) \mathbf{m}_{t-1}^c + \gamma_{\text{lb}} \mathbf{x}_{t-d'_t}^c, \tag{5}$$

where $\mathbf{m}_t^c$ is the discounted label sketch for the $c$-th class at time $t$, and $c \in \{1, ..., C\}$. $\mathbf{x}_{t-d'_t}^c$ is the feature we receive at time $t$ which belongs to the $c$-th class. We define $\mathbf{m}_1^c = \mathbf{x}_1^c$ as the first instance belonging to the $c$-th class. Discounted factor $\gamma_{\text{lb}}$ determines importance of newly received features.

**Algorithm 2** Hints Generation by Exploring Instant Data

**Input:** Discounted factor $\gamma_{\text{ft}}$ and $\gamma_{\text{lb}}$, number of base models $N_t$

**if** receive delayed feature-label pair $(\mathbf{x}_{t-d'_t}, y_{t-d'_t})$ from Algorithm 1 **then**
  **for** $i = 1, \ldots, N_t$ **do**
    Update the cumulative discounted loss for the $i$-th base model by (2)
  **end for**
  Update the label sketch by (5)
**end if**
**if** observed current feature $\mathbf{x}_t$ **then**
  Get weights for each base model by (3)
  Obtain the hint $\mathbf{h}_t$ using unlabeled data by (6)
**else if** observed current label $y_t$ **then**
  Obtain the hint $\mathbf{h}_t$ using label sketch by (4)
**end if**
Send the hint $\mathbf{h}_t$ to Algorithm 1

Based on label sketches, we approximate feature by exploring instant label and obtain $\mathbf{h}_t$ by

$$\mathbf{h}_t = \nabla \ell(f(\widehat{\mathbf{w}}_t; \mathbf{m}_t^{y_t}), y_t), \tag{6}$$

where $\mathbf{m}_t^{y_t}$ is the historical label sketch of the class $y_t$, which approximates the missing feature.

To this end, we exploit the instant features or labels in the data stream by hints generation mechanisms. These two hints generation mechanisms play essential roles in making our approach successful, in which we explore the historical information and instant feature or label to approximate the current gradient and alleviate the asynchronicity through the hint sequence. We summarize our hints generation mechanisms in Algorithm 2.

### 4.3 Model Pool Management

The number of the base models in our algorithm is dependent on the asynchronicity of the feature or label's arriving timestamp. In practice, however, the number of generated base models is usually limited due to the storage budget. In this section, we propose two mechanisms that reduce the number of base models to handle asynchronous labels more efficiently under the storage constraints.

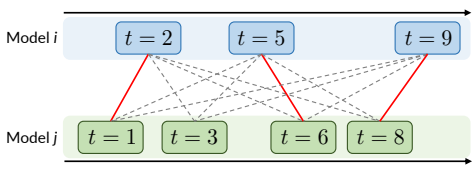



Fig. 4. Illustration of asynchronicity matching based on bipartite graph matching to deal with efficiency problem. We use bipartite graph matching (red line connected) to determine the closest pair of base models.

**Oldest Dropping.** A simple mechanism to control the number of base models is to clip extra generated models, that is, maintain a pool and drop the oldest base model if reaching budget limit.

**Asynchronicity Matching.** To further employ the structure of the asynchronous labels to deal with the efficiency problem, we propose the asynchronicity matching method. Specifically, we maintain a timestamp sequence for each base model, which indicates the timestamps each base model is handling. When the number of base models exceeds the memory budget, we use the bipartite graph matching algorithm to find the similar model pairs and then only maintain one of them and drop the other. The bipartite graph matching algorithm works as follows:

(1) *Initialization of the Bipartite Graph.* First, we create a bipartite graph where each vertex represents a base model, and the edges between the vertices are weighted by the difference in timestamps when they were added to the model pool. We consider that the more similar two models are, the smaller the weight of the edge between them. We aim to find similar model pairs by pairing all vertices and minimizing the sum of the weights of each model pair.
(2) *Improving the Matching Iteratively.* We use the Hungarian algorithm [22] to optimize the matching. The core of the Hungarian algorithm is a series of steps that iteratively improve the matching. In each iteration, we select a subset of edges that form a match, and then adjust the weights of unmatched vertices. This is done by changing the weights to reduce the overall cost of the matching while maintaining the feasibility of the solution. Specifically, we modify the weights so that the sum of the weights of the edges in the match is minimized.
(3) *Converging to Optimal Matching.* These iterations continue until no further improvements can be made, indicating that the matching has reached its optimal state. The optimality criterion is based on maximum similarity, which in our case is equivalent to minimizing the timestamp difference between models.
(4) *Merging Models based on Asynchronicity Matching.* After obtaining the optimal match, we identify pairs of models with the strongest similarity. We then randomly retain only one model from each of these pairs, effectively merging similar models and reducing redundant ones. This greatly improves the computational and storage efficiency of our algorithm.

In summary, our method seeks a matching that maximizes the total similarity, which denotes the strongest similarity between timestamp pairs, and merges the similar models to improve the efficiency. For example, As shown in Figure 4, the time sequence that $\text{Model}_i$ and $\text{Model}_j$ are handling is $t = 2, 5, 9$ and $t = 1, 3, 6, 8$ respectively. The weight of an edge between two different timestamps indicates the similarity between two timestamps, i.e., the higher weight will be if two timestamps are closer. We then use Hungarian algorithm [22] for bipartite graph matching to determine the overall similarity between two models, and merge the most similar models together to reduce the number of base models.

Moreover, there is a recent proposal named "Learnware" aiming at leveraging trained machine learning models submitted by developers globally to a learnware market, which facilitates future users in creating machine learning applications without starting from scratch while preserving the privacy of both developers' and users' data [59]. The key of this is a carefully designed Learnware *specification* that allows for the identification and reassemble of helpful models without compromising data privacy [59]. Consequently, the model specification could be potentially used to enhance our model pool management strategies, which can be explored in the future.

## 5 THEORETICAL RESULT

In this section, we theoretically analyze the proposed LACH algorithm. We first show the performance guarantee of our proposal. Then, we further illustrate the theoretical advantages of mining instant information to alleviate asynchronicity, especially when the underlying distribution shifts slowly in the data stream. At last, we provide theoretical analysis for the proposed model pool management approach. The table of notations is presented in Table 1.

We consider the setting of online convex optimization [16, 29]. We suppose that each feature $\mathbf{x}_t$ is generated from a set $\mathcal{X}$ and make the following assumptions.

**Assumption 1.** The feasible model domain $\mathcal{W}$ is convex and contains the origin $\mathbf{0}$, and the diameter of the domain $\mathcal{W}$ is at most $D$, i.e., $\|\mathbf{w} - \mathbf{w}'\|_2 \leq D$ for any $\mathbf{w}, \mathbf{w}' \in \mathcal{W}$.

**Assumption 2.** The norm of the gradients of loss functions over the domain $\mathcal{W}$ is bounded by $G$, i.e., $\|\nabla \ell_t(\mathbf{w})\|_2 \leq G$, for all $\mathbf{w} \in \mathcal{W}$ and $t \in [T]$.

Table 1. Table of notations.

| Notation | Description | Notation | Description |
|---|---|---|---|
| $\mathbf{w}_t$ | prediction model at time $t$ | $(\mathbf{x}_t, y_t)$ | feature label pair |
| $f$ | hypothesis function $t$ | $N_t$ | number of base models |
| $\mathcal{W}$ | feasible model domain | $\mathbf{h}_t$ | hint at time $t$ |
| $D$ | diameter of the domain $\mathcal{W}$ | $G$ | upper bound of gradients |
| $\gamma_{\text{ft}}$ | discounted factor of discounted loss | $\gamma_{\text{lb}}$ | discounted factor of label sketch |
| $V_T$ | rateofthedistributionchange | $\epsilon_t$ | inaccuracyofthebasemodel |

**Assumption 3.** The loss function $\ell_t(\mathbf{w})$ is convex, non-negative and $L$-smooth, i.e., $\|\nabla\ell_t(\mathbf{w}) - \nabla\ell_t(\mathbf{w}')\|_2 \leq L\|\mathbf{w} - \mathbf{w}'\|_2$ for any $\mathbf{w}, \mathbf{w}' \in \mathcal{W}$ and $t \in [T]$.

We note that these properties hold for most of the commonly used loss functions, including the quadratic loss and most other loss functions.

In streaming data learning, we are required to make a prediction at each round. We introduce the widely used *expected regret* in online learning literature [5, 16, 35] as a performance measure for streams learning with asynchronous labels problem, that is,

$$\mathbb{E}[R_T] = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - \min_{\mathbf{w}\in\mathcal{W}} \sum_{t=1}^{T} \ell_t(\mathbf{w})\right],$$

where $\ell_t(\mathbf{w}) \triangleq \ell(f(\mathbf{w}; \mathbf{x}_t), y_t)$ is the loss function at round $t$. The random quantities are predictors $\mathbf{w}_1, \mathbf{w}_2, \ldots$ that generated by the learner, which depend on the delayed length $d_t$ at each time.

We first analyze the expected regret of the proposed LACH algorithm. At the first step, we require to bound the expected number of generated base models. Let $N_T$ be the number of base models created by Algorithm 1 at time $T$. Assuming the delays are independently and identically sampled from a fixed distribution by the environment, then we have

**Lemma 1.** *Assume $d_1, \ldots, d_T$ a sequence of i.i.d. random variables with finite expected value, then*

$$\mathbb{E}[N_T] \leq \mathbb{E}[d_1] + \sqrt{4\mathbb{E}[d_1] \cdot \log T} + 2\log T + 2.$$

The above result is based on the fact that although $d_t$ can be as large as $T$, both its expectation and variance are upper bounded by $\mathbb{E}[d_1]$.

Now we are ready to propose the main theorem.

**Theorem 1.** *Assume $d_1, d_2, \ldots, d_T$ is a sequence of i.i.d. random variables with finite expected value, under Assumption 1 and 2, the expected regret of Algorithm 1 after $T$ time steps satisfies*

$$\mathbb{E}[R_T] \leq O\left(\sqrt{(\mathbb{E}[d_1] + \log T)\sum_{t=1}^{T} \|\nabla\ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2}\right),$$

*where $\nabla\ell_t(\mathbf{w}_t) \triangleq \nabla\ell(f(\mathbf{w}_t; \mathbf{x}_t), y_t)$ is gradient of the loss we suffer for making the prediction at $t$.*

**Proof Sketch.** The main technique here is to employ the structure of the subsequences scheduled by Algorithm 1, where each subsequence is synchronized. Therefore, we can bridge the gap between the timestamp of receiving the matching feature-label pair $t + d_t$ and the current timestamp $t$. The current instantaneous loss of each base model can be decomposed as follows.

$$\begin{aligned}\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) &\leq \langle\nabla\ell_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^*\rangle \\ &= \underbrace{\left\langle\nabla\ell_t(\mathbf{w}_t) - \mathbf{h}_t, \mathbf{w}_t - \widehat{\mathbf{w}}_{t+d_t}\right\rangle}_{\texttt{term (a)}} + \underbrace{\left\langle\mathbf{h}_t, \mathbf{w}_t - \widehat{\mathbf{w}}_{t+d_t}\right\rangle}_{\texttt{term (b)}} + \underbrace{\left\langle\nabla\ell_t(\mathbf{w}_t), \widehat{\mathbf{w}}_{t+d_t} - \mathbf{w}^*\right\rangle}_{\texttt{term (c)}}.\end{aligned}$$

These three terms will be bounded individually. For term (a), intuitively, it should be small if the hint $\mathbf{h}_t$ is close to true gradient $\nabla \ell_t(\mathbf{w}_t)$. We bound term (a) by the stability lemma (see Lemma 4) which implies $\|\mathbf{w}_t - \widehat{\mathbf{w}}_{t+d_t}\| \leq \eta \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2$. For term (b) and term (c), we appeal to the Bregman proximal inequality (see Lemma 3) which explores the update step in (1). Then we combine the regret bound of each base model together to obtain the overall regret guarantee. □

**Remark 1.** Theorem 1 exhibits that the regret can be bounded by the expected delay length and the gap between the true gradient $\nabla \ell_t(\mathbf{w}_t)$ and hint $\mathbf{h}_t$. Our proposed algorithm aims to minimize the second term. Namely, we explore the instant feature or label to approximate the true gradient $\nabla \ell_t(\mathbf{w}_t)$ by the hint sequence $\{\mathbf{h}_t\}_{t=1}^T$. The exploitation of instant data is more beneficial when distribution changes slowly, in which case the cumulative discounted losses and sketches better indicate the model reusability. We discuss this in Corollary 2. Notice that in the worst-case scenario, our method recovers back to the regret guarantee in previous work [20].

Next, we propose the following corollary to demonstrate that by the proposed instant information mining approach, our method achieves provable lower regret especially when the underlying distribution of the stream changes slowly, which is theoretically illustrated as follows.

**Corollary 2.** *Under the assumptions in Theorem 1 and Assumption 3, the expected regret of Algorithm 1 after $T$ time steps satisfies*

$$\mathbb{E}[R_T] \leq O\left(\sqrt{(\mathbb{E}[d_1] + \log T) \cdot (V_T + \sum_{t=1}^{T} \epsilon_t)}\right),$$

*where $V_T = \sum_{t=2}^{T} \max_{j \in [N_T]} \|\mathbf{w}_t^* - \mathbf{w}_{t-j}^*\|_2^2$ measures the rate of the distribution change, which is the sum of the maximum difference of the current best model $\mathbf{w}_t^*$ and previous best model $\mathbf{w}_{t-j|j\in[N_T]}^*$ over all timestamps. If $V_T$ is large, it means that the underlying best model at current timestamp is very different from one within the previous $N_T$ timestamp, which indicates that the distribution is changing rapidly. Here $\epsilon_t = \max_{j \in [N_T]} \|\mathbf{w}_{t-j} - \mathbf{w}_{t-j}^*\|_2^2$ is the inaccuracy of the base model $\mathbf{w}_{t-j}$.*

**Remark 2.** In Corollary 2, term $V_T$ measures the rate of the distribution change, i.e., small $V_t$ means low distribution change while large $V_t$ means rapid distribution change, which is at most $O(T)$. It could be much smaller in a stationary environment. As a benefit, mining instant information in our algorithm safeguards previous worst-case regret bound $O(\sqrt{\mathbb{E}[d_1]T})$ [20] and meanwhile achieves great improvement in easier environments when the underlying distribution shifts slowly. We give more empirical evidence as support in Section 7. Besides, compared with previous online learning with optimistic methods [8, 33, 56] which are too general to mine the structure of the asynchronous data streams, we focus on designing efficient hint sequences to practically tackle the asynchronicity issue and enjoy a tighter regret in benign environments.

As discussed in Section 4.3, we add efficient model pool management mechanisms into Algorithm 1 to deal with the asynchronous labels more effectively when the storage budget is limited. These mechanisms have good empirical performance and enjoy the following theoretical guarantee. As shown below, Corollary 3 shows that the proposed efficient model pool management method enjoys a sound performance guarantee.

**Corollary 3.** *Let $K$ be the budget of base models, under the assumptions in Theorem 1, the regret of Algorithm 1 after $T$ time steps satisfies*

$$R_T \leq O\left(\sqrt{K\sum_{t=1}^{T}\|\nabla \ell_t(\mathbf{w}_t)-\mathbf{h}_t\|_2^2}+\sum_{t=1}^{T}\mathbb{1}\{d_t>K\}\right).$$

Corollary 3 illustrates that our method still enjoys theoretical guarantee even when the budget of the base model pool is limited and we have to drop some base models, which validates the effectiveness of our proposed efficient model pool management mechanisms.

## 6 PROOF

In this section, we provide detailed proof of our main theoretical results.

### 6.1 Preliminary

We first introduce several technical lemmas used in the proofs. The following projection lemma is commonly used to analyze gradient descent algorithms.

**Lemma 2** (Projection Lemma)**.** *Let $\mathcal{W}$ be a closed convex set. Then, any update of the form $\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}[\mathbf{w}_t - \nabla]$ satisfies the following inequality*

$$\langle \mathbf{w}_{t+1}-\mathbf{w}^*, \nabla\rangle \leq \frac{1}{2}\|\mathbf{w}_t-\mathbf{w}^*\|_2^2-\frac{1}{2}\|\mathbf{w}_{t+1}-\mathbf{w}^*\|_2^2-\frac{1}{2}\|\mathbf{w}_{t+1}-\mathbf{w}_t\|_2^2$$

*for any $\mathbf{w}^* \in \mathcal{W}$.*

We then introduce the Bregman proximal inequality (Chen and Teboulle [7], Lemma 3.2), which is essential to the analysis of first-order optimization methods based on Bregman divergence.

**Lemma 3** (Bregman Proximal Inequality)**.** *Let $\mathcal{W}$ be a convex set and $f : \mathcal{W} \mapsto \mathbb{R}$ be a convex function on $\mathcal{W}$. Given a convex regularizer $\psi : \mathcal{W} \mapsto \mathbb{R}$, we denote its induced Bregman divergence as $\mathcal{D}_\psi(\mathbf{x},\mathbf{y}) = \psi(\mathbf{x}) - \psi(\mathbf{y}) - \langle\nabla\psi(\mathbf{y}), \mathbf{x}-\mathbf{y}\rangle$. Then, any update of the form*

$$\mathbf{w}_k = \arg\min_{\mathbf{w}\in\mathcal{W}}\left\{f(\mathbf{w})+\mathcal{D}_\psi\left(\mathbf{w},\mathbf{w}_{k-1}\right)\right\}$$

*satisfies the following inequality*

$$f\left(\mathbf{w}_k\right)-f(\mathbf{u}) \leq \mathcal{D}_\psi\left(\mathbf{u},\mathbf{w}_{k-1}\right)-\mathcal{D}_\psi\left(\mathbf{u},\mathbf{w}_k\right)-\mathcal{D}_\psi\left(\mathbf{w}_k,\mathbf{w}_{k-1}\right)$$

*for any $\mathbf{u} \in \mathcal{W}$.*

Furthermore, we analyze the online mirror descent frameworks using the following stability lemma (Chiang et al. [8], Proposition 7).

**Lemma 4** (Stability Lemma)**.** *Let $\mathcal{W}$ be a closed convex set. Consider the following two updates: $\mathbf{w}_* = \arg\min_{\mathbf{w}\in\mathcal{W}}\langle\mathbf{a},\mathbf{w}\rangle+\mathcal{D}_\psi(\mathbf{w},\mathbf{c})$ and $\mathbf{w}'_* = \arg\min_{\mathbf{w}\in\mathcal{W}}\langle\mathbf{a}',\mathbf{w}\rangle+\mathcal{D}_\psi(\mathbf{w},\mathbf{c})$. When the regularizer $\psi : \mathcal{W} \mapsto \mathbb{R}$ is 1-strongly convex with respect to the norm $\|\cdot\|$, we have*

$$\left\|\mathbf{w}_*-\mathbf{w}'_*\right\| \leq \|(\nabla\psi(\mathbf{c})-\mathbf{a})-(\nabla\psi(\mathbf{c})-\mathbf{a}')\|_* = \|\mathbf{a}-\mathbf{a}'\|_*.$$

By choosing the regularizer as $\psi(\mathbf{w}) = \|\mathbf{w}\|_2^2$, it is easy to verify that the induced Bregman divergence is $\mathcal{D}_\psi(\mathbf{w},\mathbf{w}') = \|\mathbf{w}-\mathbf{w}'\|_2^2$, and therefore we can employ Lemma 3 and Lemma 4 to analysis our online learning algorithm.

Besides the above technical lemmas, we introduce the following decomposable monotone loss for theoretical analysis.

**Definition 1** (Decomposable Monotone Loss)**.** We say a loss function $\ell : \mathbb{R}\times\mathbb{R} \mapsto \mathbb{R}$ is decomposable monotone if its gradient can be represented as

$$\nabla_{\mathbf{w}}\ell(f(\mathbf{w};\mathbf{x}), y) = g(y)\cdot\nabla_{\mathbf{w}}f(\mathbf{w};\mathbf{x}),$$

where $g(\cdot)$ is a monotone function respect to $y$.

Notice that the decomposable monotone property holds for most of the common loss functions, including the square loss, logistic loss, hinge loss, cross-entropy loss, and most other loss functions. We provide some examples with discussions here.

**Example 1.** For square loss $\ell(f(\mathbf{w};\mathbf{x}), y) = (f(\mathbf{w};\mathbf{x}) - y)^2$, we have

$$\nabla_{\mathbf{w}}\ell(f(\mathbf{w};\mathbf{x}), y) = 2(f(\mathbf{w};\mathbf{x}) - y)\cdot\nabla_{\mathbf{w}}f(\mathbf{w};\mathbf{x}) = g(y)\cdot\nabla_{\mathbf{w}}f(\mathbf{w};\mathbf{x}),$$

where $g(y) = 2(f(\mathbf{w};\mathbf{x}) - y)$ is a monotone function with respect to $y$ for $y \in \mathbb{R}$.

**Example 2.** For hinge loss $\ell(f(\mathbf{w};\mathbf{x}), y) = \max(1 - yf(\mathbf{w};\mathbf{x}), 0)$, we have

$$\begin{aligned}\nabla_{\mathbf{w}}\ell(f(\mathbf{w};\mathbf{x}), y) &= \begin{cases} -y\cdot\nabla_{\mathbf{w}}f(\mathbf{x},\mathbf{w}), & yf(\mathbf{w};\mathbf{x}) < 1 \\ 0, & \text{else}\end{cases} \\ &= \big(y\cdot\mathbb{1}\{y\cdot f(\mathbf{w};\mathbf{x}) < 1\}\big)\cdot\nabla_{\mathbf{w}}f(\mathbf{w};\mathbf{x}) = g(y)\cdot\nabla_{\mathbf{w}}f(\mathbf{w};\mathbf{x}),\end{aligned}$$

where $g(y) = y\cdot\mathbb{1}\{y\cdot f(\mathbf{w};\mathbf{x}) < 1\}$ is a monotone function with respect to $y$ for $y \in [-1, 1]$ and $f(\mathbf{w};\mathbf{x}) \in [-1, 1]$.

**Example 3.** For cross-entropy loss $\ell(f(\mathbf{w};\mathbf{x}), y) = y\log(f(\mathbf{w};\mathbf{x})) + (1-y)\log(1 - f(\mathbf{w};\mathbf{x}))$,

$$\nabla_{\mathbf{w}}\ell(f(\mathbf{w};\mathbf{x}), y) = \frac{(y - f(\mathbf{w};\mathbf{x}))}{(1 - f(\mathbf{w};\mathbf{x}))f(\mathbf{w};\mathbf{x})}\nabla_{\mathbf{w}}f(\mathbf{w};\mathbf{x}) = g(y)\cdot\nabla_{\mathbf{w}}f(\mathbf{w};\mathbf{x}),$$

where $g(y) = \frac{(y - f(\mathbf{w};\mathbf{x}))}{(1-f(\mathbf{w};\mathbf{x}))f(\mathbf{w};\mathbf{x})}$ is a monotone function with respect to $y$ for $y \in [0, 1]$.

The above examples demonstrate that the decomposable monotone property holds for square losses, hinge losses, cross-entropy losses. Other loss functions can be analyzed in a similar manner.

### 6.2 Proof of Lemma 1

We prove Lemma 1 based on the main techniques proposed by Joulani et al. [20]. We denote by $M_t = \sum_{s=1}^{t-1}\mathbb{1}\{s + d_s \geq t\}$ the total number of missing features or labels when the forecaster is making a prediction at time $t$, then we have

$$N_T = \max_{1\leq t\leq T}(M_t) + 1.$$

With a probability of at least $1 - 1/T$, the following inequality holds,

$$\max_{1\leq t\leq n} M_t \leq \mathbb{E}[d_1] + 2\log T + \sqrt{4\mathbb{E}[d_1]\log T}.$$

Taking into account that $\max_{1\leq t\leq T} M_t \leq T$, we have the following inequality

$$\mathbb{E}\left[\max_{1\leq t\leq T}(M_t)\right] \leq \mathbb{E}[d_1] + 2\log T + \sqrt{4d\log T} + 1 = B_T + 1.$$

We define $B_T \coloneqq \mathbb{E}[d_1] + 2\log T + \sqrt{4\mathbb{E}[d_1]\log T}$. Therefore,

$$\mathbb{E}[N_T] = \mathbb{E}\left[\max_{1\leq t\leq T}(M_t)\right] + 1 \leq \mathbb{E}[d_1] + 2\log T + \sqrt{4\mathbb{E}[d_1]\log T} + 2 = B_T + 2.$$

Thus, we complete the proof of Lemma 1. □

### 6.3 Proof of Theorem 1

As a starting point, we first analyze the regret guarantee for each model, and then combine them to analyze the whole algorithm. Following the scheduling algorithm described in Algorithm 1, the $i$-th base model is handling the following non-asynchronous subsequence

$$\mathcal{T}_i = \{i, i + d_i^1, i + d_i^2, i + d_i^3, \ldots\}.$$

Without loss of generality, we assume that at time $t + d_t$, we receive the delayed label $y_t$. We first decompose the instantaneous regret at time $t$ for the $i$-th base model as follows

$$\begin{aligned}\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) &\leq \langle \nabla \ell_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \\ &= \underbrace{\left\langle \nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t, \mathbf{w}_t - \widehat{\mathbf{w}}_{t+d_t} \right\rangle}_{\texttt{term (a)}} + \underbrace{\left\langle \mathbf{h}_t, \mathbf{w}_t - \widehat{\mathbf{w}}_{t+d_t} \right\rangle}_{\texttt{term (b)}} + \underbrace{\left\langle \nabla \ell_t(\mathbf{w}_t), \widehat{\mathbf{w}}_{t+d_t} - \mathbf{w}^* \right\rangle}_{\texttt{term (c)}}.\end{aligned}$$

We individually bound these three terms. Specifically, we use Lemma 4 to bound term (a) and appeal to Lemma 3 to bound term (b) and (c). In the following, we present the precise arguments.

We firstly bound term (a). Following Lemma 4 which implies $\left\|\mathbf{w}_t - \widehat{\mathbf{w}}_{t+d_t}\right\|_2 \leq \eta \left\|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\right\|_2$, we have the following inequality,

$$\texttt{term (a)} = \left\langle \nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t, \mathbf{w}_t - \widehat{\mathbf{w}}_{t+d_t} \right\rangle \leq \left\|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\right\|_2 \left\|\mathbf{w}_t - \widehat{\mathbf{w}}_{t+d_t}\right\|_2 \leq \eta \left\|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\right\|_2^2.$$

We then bound term (b) and (c). Based on Lemma 3 and the first update step in (1), we have

$$\texttt{term (b)} = \left\langle \mathbf{h}_t, \mathbf{w}_t - \widehat{\mathbf{w}}_{t+d_t} \right\rangle \leq \frac{1}{\eta}\left[ \left\|\widehat{\mathbf{w}}_{t+d_t} - \widehat{\mathbf{w}}_t\right\|_2^2 - \left\|\widehat{\mathbf{w}}_{t+d_t} - \mathbf{w}_t\right\|_2^2 - \left\|\mathbf{w}_t - \widehat{\mathbf{w}}_t\right\|_2^2 \right].$$

For the second update step in (1), we have

$$\texttt{term (c)} = \left\langle \nabla \ell_t(\mathbf{w}_t), \widehat{\mathbf{w}}_{t+d_t} - \mathbf{w}^* \right\rangle \leq \frac{1}{\eta}\left[ \left\|\mathbf{w}^* - \widehat{\mathbf{w}}_t\right\|_2^2 - \left\|\mathbf{w}^* - \widehat{\mathbf{w}}_{t+d_t}\right\|_2^2 - \left\|\widehat{\mathbf{w}}_{t+d_t} - \widehat{\mathbf{w}}_t\right\|_2^2 \right].$$

By combining the three inequalities, we have

$$\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) \leq \eta \left\|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\right\|_2^2 + \frac{1}{\eta}\left[ \left\|\mathbf{w}^* - \widehat{\mathbf{w}}_t\right\|_2^2 - \left\|\mathbf{w}^* - \widehat{\mathbf{w}}_{t+d_t}\right\|_2^2 - \left\|\widehat{\mathbf{w}}_{t+d_t} - \mathbf{w}_t\right\|_2^2 - \left\|\mathbf{w}_t - \widehat{\mathbf{w}}_t\right\|_2^2 \right].$$

As each base model is handling a non-delayed sequence $\mathcal{T}_i$, using a telescoping summation,

$$\sum_{t \in \mathcal{T}_i} \left( \left\|\mathbf{w}^* - \widehat{\mathbf{w}}_t\right\|_2^2 - \left\|\mathbf{w}^* - \widehat{\mathbf{w}}_{t+d_t}\right\|_2^2 \right) = \left\|\mathbf{w}^* - \widehat{\mathbf{w}}_i\right\|_2^2 - \left\|\mathbf{w}^* - \widehat{\mathbf{w}}_T\right\|_2^2 \leq D^2,$$

where $D$ is the diameter of the domain $\mathcal{W}$. As a result, for the $i$-th base model, we have that its regret $R_T^i$ is bounded as

$$\begin{aligned} R_T^i &\leq \eta \sum_{t \in \mathcal{T}_i} \left\|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\right\|_2^2 + \frac{1}{\eta}\left( D^2 - \sum_{t \in \mathcal{T}_i} \left( \left\|\widehat{\mathbf{w}}_{t+d_t} - \mathbf{w}_t\right\|_2^2 + \left\|\mathbf{w}_t - \widehat{\mathbf{w}}_t\right\|_2^2 \right) \right) \\ &\leq \eta \sum_{t \in \mathcal{T}_i} \left\|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\right\|_2^2 + \frac{1}{\eta} D^2. \end{aligned} \tag{7}$$

Since there are a total of $N_T$ base models, the overall regret is then the summation of all models,

$$R_T = \sum_{i=1}^{N_T} R_T^i \leq \eta \sum_{t=1}^{T} \left\|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\right\|_2^2 + N_T \frac{D^2}{\eta}.$$

The expected regret is upper bounded by

$$\mathbb{E}[R_T] \leq \mathbb{E}\left[\eta \sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2 + \frac{D^2 N_T}{\eta}\right] = \eta \sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2 + \mathbb{E}\left[N_T\right] \frac{D^2}{\eta}$$
$$\leq \eta \sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2 + (B_T + 2)\frac{D^2}{\eta},$$

where $B_T = d + 2\log T + \sqrt{4d \log T}$, the last inequality comes from Lemma 1. By setting an appropriate learning rate with $\eta = \sqrt{(B_T + 2)D^2 / \sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2}$, we have

$$\mathbb{E}[R_T] \leq 2\sqrt{(B_T + 2)D^2 \sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2} = O\left(\sqrt{\left(\mathbb{E}[d_1] + \log T\right) \sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2}\right).$$

Therefore, we complete the proof of Theorem 1. □

### 6.4 Proof of Corollary 2

Corollary 2 illustrates that our approach achieves lower regret when the underlying distribution of the stream changes slowly, which is proved as follows. Under Assumption 3, when the loss function is $L$-smooth with respect to $\mathbf{w}$ and using online ensemble to estimate the missing labels, we have

$$\begin{aligned}
&\|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2 \\
&= \left\|\nabla \ell\big(f(\mathbf{w}_t; \mathbf{x}_t), y_t\big) - \nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), \widetilde{y}_t\big)\right\|_2^2 \\
&\leq \|\nabla \ell\big(f(\mathbf{w}_t; \mathbf{x}_t), y_t\big) - \nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), y_t\big) + \nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), y_t\big) - \nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), \widetilde{y}_t\big)\|_2^2 \\
&\leq 2\left\|\nabla \ell\big(f(\mathbf{w}_t; \mathbf{x}_t), y_t\big) - \nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), y_t\big)\right\|_2^2 + 2\|\nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), y_t\big) - \nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), \widetilde{y}_t\big)\|_2^2 \\
&= 2\,\|\nabla \ell_t(\mathbf{w}_t) - \nabla \ell_t(\widehat{\mathbf{w}}_t)\|_2^2 + 2\|\nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), y_t\big) - \nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), \widetilde{y}_t\big)\|_2^2 \\
&\leq 2L^2 \|\mathbf{w}_t - \widehat{\mathbf{w}}_t\|_2^2 + 2\left\|\nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), y_t\big) - \nabla \ell\big(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), \widetilde{y}_t\big)\right\|_2^2. \qquad \text{(by } L\text{-Smoothness)}
\end{aligned}$$

Then we further upper bound the second term using the property of decomposable monotone loss defined in Definition 1. Assume that the gradient of hypothesis function $f$ is upper bounded by $F$, i.e., $\|\nabla f(\mathbf{w}; \mathbf{x})\|_2^2 \leq F$ for any $\mathbf{w} \in \mathcal{W}$ and $\mathbf{x} \in \mathcal{X}$, the monotone function $g$ is Lipschitz continuous with a constant $L_g$, and the hypothesis function $f$ is Lipschitz continuous with a constant $L_f$, therefore the composition of $g$ and $f$ is Lipschitz continuous with a constant $L_g \cdot L_f$. Then, we have

$$\begin{aligned}
&\|\nabla \ell(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), y_t) - \nabla \ell(f(\widehat{\mathbf{w}}_t; \mathbf{x}_t), \widetilde{y}_t)\|_2^2 \\
&= \|g(y_t)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t) - g(\widetilde{y}_t)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t)\|_2^2 \qquad \text{(by decomposable monotone loss)} \\
&= \left\|g(y_t)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t) - g\left(\sum_{j=1}^{N_T} \beta_j \widehat{y}_j\right)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t)\right\|_2^2 \\
&\leq \max_{j \in [N_T]} \|g(y_t)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t) - g(\widehat{y}_j)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t)\|_2^2 \qquad \text{(by convexity with respect to } \widehat{y}_j) \\
&\leq \max_{j \in [N_T]} \|g(y_t)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t) - g(y_j)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t) + g(y_j)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t) - g(\widehat{y}_j)\nabla f(\widehat{\mathbf{w}}_t; \mathbf{x}_t)\|_2^2 \\
&\leq 2F \cdot L_f \cdot L_g \max_{j \in [N_T]} \|\mathbf{w}_t^* - \mathbf{w}_{t-j}^*\|_2^2 + 2F \cdot L_f \cdot L_g \max_{j \in [N_T]} \|\mathbf{w}_{t-j} - \mathbf{w}_{t-j}^*\|_2^2
\end{aligned}$$

Therefore, substituting the above equation back to (7), we set $\eta \leq 1/(\sqrt{2}F \cdot L_f \cdot L_g)$ to cancel the term $\sum_{t=1}^{T} 2L^2 \|\mathbf{w}_t - \widehat{\mathbf{w}}_t\|_2^2$, sum over $T$, we have:

$$\sum_{t=1}^{T} \|\nabla \ell(\mathbf{w}_t, \mathbf{x}_t, y_t) - \mathbf{h}_t\|_2^2 \leq O\left(V_T + \sum_{t=1}^{T} \epsilon_t\right), \tag{8}$$

in the above inequality, the first term $V_T = \sum_{t=2}^{T} \max_{j\in[N_T]} \|\mathbf{w}_t^* - \mathbf{w}_{t-j}^*\|_2^2$ measures the rate of the distribution change, which is the sum of the maximum difference of the current best model $\mathbf{w}_t^*$ and previous best model $\mathbf{w}_{t-j|j\in[N_T]}^*$ over all timestamps. If $V_T$ is large, it means that the underlying best model at current timestamp is very different from one within the previous $N_T$ timestamp, which indicates that the distribution is changing rapidly. In addition to this, the second term $\epsilon_t = \max_{j\in[N_T]} \|\mathbf{w}_{t-j} - \mathbf{w}_{t-j}^*\|_2^2$ is the inaccuracy of the base model $\mathbf{w}_{t-j}$. The above inequality comes from $(a+b)^2 \leq 2a^2 + 2b^2$. Consequently, substituting (8) back to Theorem 1 yields

$$\mathbb{E}[R_T] \leq O\left(\sqrt{\left(\mathbb{E}[d_1] + \log T\right)\left(V_T + \sum_{t=1}^{T} \epsilon_t\right)}\right).$$

This ends the proof of Corollary 2. □

### 6.5 Proof of Corollary 3

Corollary 3 illustrates that our algorithm still enjoys sound theoretical guarantee when the storage budget is limited. Specifically, we add the base model management mechanisms as described in Section 4.3 into Algorithm 1. At time $t$, when an instance $\mathbf{x}_t$ or $y_t$ has waited for $K$ time intervals in a base model, we delete this oldest base model and reinitialize a new base model with the same $\mathbf{w}$, and put this new base model in the $\mathcal{Q}_{\text{ready}}$. This results in that an extra loss will be suffered every time when the number of missing labels or features $M_t = \sum_{s=1}^{t-1} \mathbb{1}\{s + d_s \geq t\}$ is more than model buffer $K$, and the regret guarantee will depend on the storage buffer $K$.

To prove Corollary 3, our main idea is to upper bound $M_t$, the number of missing labels or features in the asynchronous streams at time $t$. By rewriting $M_t$

$$\begin{aligned} M_t &= \sum_{s=1}^{t-1} \mathbb{1}\{s + d_s \geq t\} = \sum_{i=1}^{t-1} \mathbb{1}\{d_i \geq i\} = \sum_{i=1}^{k} \mathbb{1}\{d_i \geq i\} + \sum_{i=k+1}^{t-1} \mathbb{1}\{d_i \geq i\} \\ &\leq K + \sum_{i=k+1}^{t-1} \mathbb{1}\{d_i > K\} \leq K + \sum_{i=1}^{t} \mathbb{1}\{d_i > K\}. \end{aligned}$$

Therefore, for all $t \in [T]$, we have the following upper bound of $M_T$ with respect to $K$

$$M_t - K \leq \sum_{i=1}^{t} \mathbb{1}\{d_i > K\}.$$

As mentioned above, we will suffer a constant loss $\ell_t(\mathbf{w}_t) \leq GD$ everytime when the feature or the label is missing, where $G$ is the gradient upper bound of as described in Assumption 2, $D$ is the diameter of $\mathcal{W}$ as described in Assumption 1. We can get the overall regret:

$$R_T \leq \eta \sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2 + K\frac{D^2}{\eta} + \sum_{i=1}^{T} \mathbb{1}\{d_i > K\}GD.$$

By setting an appropriate learning rate with the value of $\eta = \sqrt{KD/\sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2}$, we have

$$R_T \leq 2\sqrt{KD^2 \sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2} + \sum_{t=1}^{T} \mathbb{1}\{d_t > K\} GD$$
$$= O\left(\sqrt{K \sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2} + \sum_{t=1}^{T} \mathbb{1}\{d_t > K\}\right).$$

Therefore, we complete the proof of Corollary 3. Note that this regret bound also matches Theorem 1 by setting $K = B_T + 2$. □

## 7 EXPERIMENT

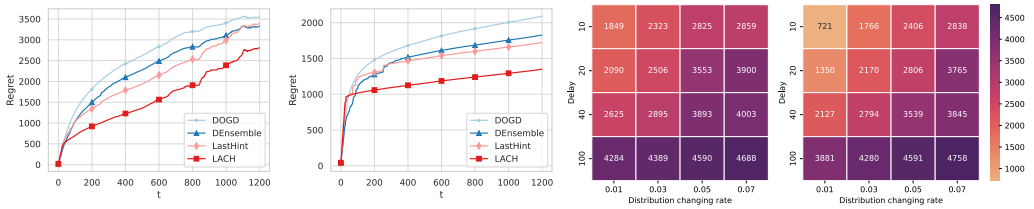
In this section, we examine the performance of the proposed LACH algorithm on synthetic examples, benchmark datasets and real-world applications. Specifically, we evaluate our learning asynchronous labels with hint algorithm in the following three aspects:

(1) **Validation on synthetic data.** We provide intuitive illustrations of the advantages in exploring instant features and labels to deal with the challenge in asynchronous learning, compared with general delayed online learning methods in various non-stationary environments;
(2) **Comparison on benchmark data.** We compare the proposed approach with various contenders on benchmark datasets to demonstrate the superiority of our method;
(3) **Experiment on real-world applications.** We examine the proposed LACH algorithm on two real-world streaming learning tasks with asynchronous labels: IoT monitoring which aims to predict whether an IoT device is functioning properly based on network data, and a loan credit prediction task where the goal is to predict whether the loaner will be overdue.

### 7.1 Validation on Synthetic Data

We first illustrate the advantage of mining instant information with hints by a synthetic example in various non-stationary environments. We generate a classification example, each instance $\mathbf{x}_t$ is generated from a standard Gaussian distribution. The potential optimal decision $\mathbf{w}_t^*$ changes overtime with a random Gaussian shift, i.e., $\mathbf{w}_{t+1}^* = \mathbf{w}_t^* + \boldsymbol{\nu}$, where $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$ is a random Gaussian noise vector. The ground-truth label is assigned by $y_t = \text{sgn}(\langle \mathbf{w}_t^*, \mathbf{x}_t \rangle)$ and we use cross-entropy loss. To simulate data streams with asynchronous labels, we reveal label or feature of each instance with a random delay. Delay lengths are i.i.d. sampled from a uniform distribution $U(\mathbb{E}[d_1] - 10, \mathbb{E}[d_1] + 10)$, where $\mathbb{E}[d_1]$ is the expectation of the delay lengths. We set the total iteration $T = 1200$. The learning rate for each method is set as the optimal value according to Theorem 1.

**Implementation Details.** We set different norms of $\mathbf{V}$ to simulate distribution change, specifically, $\|\mathbf{V}\|_2 = 0.01, 0.05$ for slow and fast distribution changing rate, respectively. We compare the regret of our proposed algorithm on synthetic data with three contenders, including two baseline methods DOGD [20] and DEnsemble [32] that do not explore the instant feature or labels; and the LastHint method which takes the last received gradient as a hint. For the synthetic data, at every iteration $t$, we receive a mini-batch data of size 8. We choose the linear model as the classifier and control the diameter of the domain $D = 1$. We set the learning rate for each method is set as the optimal value according to Theorem 1, i.e., $\eta = \sqrt{D^2/(2TG^2)} = \sqrt{1^2/(2 \times 1200 \times 8 \times 2^2)} \approx 0.003$ for DOGD; and $\eta = \sqrt{(B_T + 2)D^2/\sum_{t=1}^{T} \|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|_2^2} \approx \sqrt{20 \times 1^2/(1200 \times 8 \times 0.01 \times 2)} \approx 0.3$ for our proposed LACH. We use 1×Nvidia GeForce RTX 3090 and train for about 10 minutes per task.




(a) Distribution changes fast (b) Distribution changes slow (c) Heat map comparison

Fig. 5. (a) & (b) Results of regret comparison on synthetic data with different distribution changing rates. We compare our LACH with two baseline methods only considering general delayed feedback: DOGD and DEnsemble; LastHint is a simplified version of our proposal that sets the last gradient as a hint. (c) Heat map of accumulated regrets of the left (DOGD) and right (LACH) concerning different expected delay lengths $\mathbb{E}[d_1]$ and different distribution changing rates.

**Comparison Results.** We report the comparison results in Figure 5. The performance of our algorithm and its simplified version (LastHint) is better than the DOGD and DEnsemble, which validates the effectiveness of exploring instant data to estimate its ground-truth gradient. Besides, compared with LastHint method, our method is better than a simple reuse of the last model's gradient, because the cumulative discounted loss can better adapt to the changing environment than just reusing the last gradient. These results further validate our method's effectiveness for exploiting instant information of asynchronous data.

**Discussion on Hints Quality.** Note that our performance will become better when distribution changes slowly, as shown in Figure 5 (a) and (b), which matches the theoretical result in Corollary 2. The reason is that when distribution changes slowly, our designed hint sequence will become more accurate and LACH will enjoy a better performance.

On the other side, when the distribution changes dramatically, our hint sequence will become less accurate and suffers a performance drop, but still outperforms the methods that do not exploit instant information, which further validates Theorem 1 that our algorithm still has a performance guarantee in the worst-case scenario.

We further show a heat map of the accumulated regret of the DOGD and our approach concerning different expected delay lengths and different distribution changing rates in Figure 5 (c). Our LACH significantly improves the performance compared to the DOGD method in cases where the delay length is short and distribution changes slowly. The phenomenon also matches our theoretical results and validates the proposed method's effectiveness. When the distribution changes slowly, the cumulative discounted loss can correctly indicate the accuracy of each model on the current unlabeled data and the label sketches can better estimate the missing label or feature, hence we can estimate current gradients more accurately, while the DOGD method cannot exploit such properties as it does not explore the instant data in asynchronous streams.

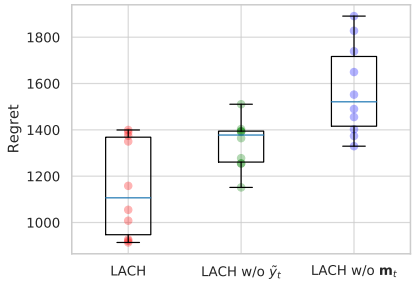



Fig. 6. Ablation study of our proposed method, we compare LACH with its variants: (i) LACH w/o $\widetilde{y}_t$, which does not explore the unlabeled feature by employing online ensemble; and (ii) LACH w/o $\mathbf{m}_t$, which does not use the label sketch to estimate the missing feature.

**Ablation Studies.** To validate the effectiveness of exploring instant features and labels respectively, we quantitatively evaluate our proposed LACH and its variants: (i) LACH w/o $\widetilde{y}_t$, which does not explore the unlabeled feature by online ensemble; and (ii) LACH w/o $\mathbf{m}_t$, which does not use the label sketch to estimate the missing feature. As shown in Figure 6, the feature and label exploration both make the regret lower, which indicates our proposed method exploits the instant information to better estimate the gradient and improve the performance.

### 7.2 Benchmark Datasets

We examine the performance of the proposed LACH algorithm on real-world applications with other contenders to show its superiority.

**Datasets.** We conduct the empirical comparisons on four datasets in two applications: the human activity recognition task (UCI-HAR, UCI-HHAR, WISDM-AR) and the online gender recognition task (Portraits) as follows.

- UCI-HAR (Human Activity Recognition dataset) [1]. This dataset has been collected from 30 subjects performing six different activities (Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing, Laying). It contains over 1,600 data items and has 128 features. It consists of inertial sensor data that was collected using a smartphone carrier. Our goal is to classify what a person does based on a stream of sensor data.
- UCI-HHAR (Heterogeneity Human Activity Recognition dataset) [38]. A dataset devised to benchmark human activity recognition in real-world contexts, which has more diversity than HAR. Specifically, the dataset is gathered with a variety of different device models and use scenarios, in order to reflect sensing heterogeneities to be expected in real deployments. It contains over 37,000 data items and has 128 features for each data item. The goal is to classify the action of the subject.
- WISDM (WIreless Sensor Data Mining Action Recognition dataset) [24]. Wisdm uses a smartphone-based application to collect a total of 33 participants' accelerometer data. Using its embedded accelerometer and gyroscope sensor data, this dataset contains 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. It contains over 1,200 data items and has 315 features. The goal is to infer what a person is doing based on a stream of sensor data.
- Portraits [11]. We take another benchmark dataset named Portraits, which comprises photos of high school seniors taken across the 1900s-2010s. The dataset contains 37,921 frontal-facing American student photos with 3,072 features that allow us to glimpse into historical visual records. The fashions and habits change over time, and thus it is a distribution-changing stream. The goal is to classify the gender of each portrait.

**Contenders.** We compare LACH with seven contenders, including four previous methods designed for online learning with delayed loss, a self-training approach, a distributed machine learning approach, a recent advanced causal-based method, and a simplified version of our approach that directly uses the last gradient as hint.

- DOGD [32] aims to deal with the delayed label issue, which directly applies online gradient descent to update the model when receiving a delayed loss feedback;
- DEnsemble [20] splits the time horizon into sub-sequences and uses a set of non-delayed base online algorithms to handle each sub-sequence;
- FF [30] uses forgetting factors to handle the delayed labels in the data stream;
- Dual [34] handles label delays with a dual learning algorithm that alternatively optimizes a predictor and a bias estimator that estimates the model's performance for the unlabeled data;

Table 2. Comparisons on benchmark datasets. On each dataset, 5 test runs were conducted, and the average as well as standard deviation of accuracy and AUC score are presented. The best one is emphasized in bold.

| Method | Measure | UCI-HAR | | WISDM | | UCI-HHAR | | Portraits | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\mathbb{E}[d] = 200$ | $\mathbb{E}[d] = 300$ | $\mathbb{E}[d] = 200$ | $\mathbb{E}[d] = 300$ | $\mathbb{E}[d] = 200$ | $\mathbb{E}[d] = 300$ | $\mathbb{E}[d] = 200$ | $\mathbb{E}[d] = 300$ |
| OGD | ACC | 65.44 ± 5.99 | 57.48 ± 4.88 | 56.33 ± 7.57 | 50.58 ± 8.47 | 44.47 ± 8.15 | 37.79 ± 7.54 | 62.19 ± 2.77 | 61.53 ± 2.57 |
| | AUC | 85.14 ± 2.91 | 78.15 ± 2.40 | 75.61 ± 3.05 | 70.85 ± 3.25 | 72.44 ± 5.53 | 66.54 ± 5.06 | 62.62 ± 4.36 | 60.82 ± 4.00 |
| DEnsemble | ACC | 64.17 ± 4.43 | 58.24 ± 3.64 | 48.25 ± 6.26 | 41.88 ± 7.03 | 43.66 ± 5.80 | 37.15 ± 6.07 | 58.09 ± 4.53 | 57.47 ± 4.58 |
| | AUC | 80.85 ± 2.43 | 76.31 ± 1.92 | 77.68 ± 4.73 | **73.37 ± 3.93** | 70.66 ± 4.65 | 65.74 ± 5.52 | 59.07 ± 4.76 | 58.13 ± 4.26 |
| Self-Train | ACC | 68.90 ± 5.70 | 61.13 ± 4.57 | 57.33 ± 8.04 | 50.94 ± 7.85 | **45.33 ± 6.82** | 38.52 ± 6.30 | 61.20 ± 2.04 | 61.29 ± 1.84 |
| | AUC | 85.35 ± 3.07 | 79.41 ± 2.62 | 76.21 ± 2.97 | 71.63 ± 3.35 | 72.73 ± 4.81 | 66.24 ± 4.08 | 63.03 ± 4.27 | 61.31 ± 3.83 |
| FF | ACC | 69.64 ± 6.12 | 57.77 ± 6.67 | 57.13 ± 7.82 | 50.56 ± 7.63 | 44.58 ± 6.51 | 37.86 ± 5.77 | 61.11 ± 2.11 | 60.13 ± 2.45 |
| | AUC | 85.45 ± 3.44 | **82.37 ± 3.91** | 76.35 ± 3.14 | 71.75 ± 3.53 | 72.04 ± 4.10 | 65.72 ± 3.58 | 62.94 ± 4.29 | 61.19 ± 3.81 |
| Dual | ACC | 67.23 ± 3.15 | 60.25 ± 3.15 | 56.21 ± 4.36 | 50.15 ± 6.26 | 43.87 ± 5.12 | 36.25 ± 4.15 | 60.35 ± 1.29 | 61.11 ± 1.41 |
| | AUC | 85.12 ± 2.11 | 79.34 ± 1.74 | 75.97 ± 2.16 | 70.01 ± 2.15 | 71.42 ± 3.25 | 66.98 ± 5.51 | 62.10 ± 4.15 | 59.36 ± 3.26 |
| AdaDelay | ACC | 66.59 ± 6.49 | 59.23 ± 5.32 | 55.93 ± 7.56 | 50.25 ± 8.44 | 43.37 ± 8.32 | 35.32 ± 7.64 | 61.87 ± 1.32 | 60.13 ± 2.45 |
| | AUC | 84.05 ± 2.90 | 78.21 ± 2.37 | 75.73 ± 3.07 | 70.95 ± 3.29 | 71.57 ± 5.65 | 65.85 ± 5.07 | 62.34 ± 4.21 | 58.39 ± 3.75 |
| GrangerCausal | ACC | 66.14 ± 3.12 | 59.51 ± 4.98 | 56.15 ± 4.26 | 50.13 ± 7.05 | 43.91 ± 6.67 | 35.44 ± 6.25 | 61.15 ± 1.98 | 60.52 ± 2.78 |
| | AUC | 83.36 ± 1.16 | 78.98 ± 2.25 | 75.87 ± 3.25 | 70.88 ± 2.98 | 72.01 ± 5.28 | 65.98 ± 4.15 | 61.93 ± 3.75 | 59.05 ± 2.18 |
| LastHint | ACC | 66.86 ± 4.84 | 60.94 ± 2.79 | 48.86 ± 8.31 | 41.47 ± 4.85 | 42.24 ± 6.45 | 37.22 ± 5.24 | 62.33 ± 6.02 | 60.64 ± 6.77 |
| | AUC | 79.55 ± 5.56 | 76.05 ± 4.74 | **77.96 ± 2.38** | 72.95 ± 1.86 | 69.60 ± 6.20 | 66.29 ± 4.93 | 64.88 ± 6.42 | 63.52 ± 6.52 |
| **LACH** | ACC | **69.92 ± 3.88** | **61.98 ± 3.03** | **57.53 ± 8.08** | **52.07 ± 8.91** | 44.90 ± 4.46 | **38.57 ± 4.11** | **62.59 ± 6.59** | **61.87 ± 4.35** |
| | AUC | **87.33 ± 4.66** | 79.61 ± 3.19 | 74.34 ± 4.07 | 71.19 ± 3.60 | **73.95 ± 5.21** | **70.23 ± 3.50** | **66.29 ± 6.93** | **64.97 ± 7.14** |

- Self-Train [23] is a weakly supervised method that randomly initializes a model, then repeatedly minimizes empirical risks based on pseudo labels generated by the last classifier;
- AdaDelay [37] is an asynchronous distributed machine learning method that adds a penalty to the learning rate of delayed gradients;
- GrangerCausal [49] is a causal inference-based method that designs a Granger-causal regularization mechanism to identify the most appropriate historical model for prediction;
- Lasthint is a simplified version of our proposal that directly applies the last gradient as hint.

**Implementation Details.** In the following, we provide the details for the implementation of the proposed learning asynchronous labels with hint algorithm. For the human activity recognition datasets, we choose the dense network with a 1D-convolution feature extractor as base model. The learning rate and the batch size are set as 0.003 and 128 respectively, and we train 500 iterations for each data batch. For the Portraits dataset, we choose the dense network with a 2D-convolution feature extractor. The learning rate and the batch size are set as 0.0003 and 128 respectively, and we train 500 iterations for each data batch.

For all the experiments, we set the discounted factor $\gamma_{\text{ft}} = 0.8$ in online ensemble (2) and $\gamma_{\text{lb}} = 0.5$ in label sketch (5) without careful tuning. We formulate a data stream with $T = 10$ for all the datasets, where at each iteration $t$ a batch of data (about 100 data items) arrives. We conduct empirical experiments on cases of $\mathbb{E}[d] = 200$ and $\mathbb{E}[d] = 300$.

**Comparison Results.** We report the comparison results on four benchmark datasets in Table 2. Specifically, we perform five test runs on each benchmark dataset and compare the accuracy and AUC scores. Our LACH achieves the highest average accuracy on three out of four datasets, which indicates that our approach outperforms contenders in various streaming learning tasks with asynchronous labels. DOGD and DEnsemble methods do not perform well, providing evidence that it is crucial to explore instant data in asynchronous label streaming learning tasks. A simplified version of our proposed approach also performs better than DEnsemble, further validating the effectiveness of exploring the instant data.

Compared with FF and AdaDelay method which exploit delayed gradient with weighted combination, our method shows superiority in most cases because our hints generated by online ensemble

Table 3. Quantitative analysis on real-world IoT Monitoring dataset. For each method, 5 test runs were conducted, and we present the average as well as the standard deviation of accuracy (ACC) and AUC score. The best one is emphasized in bold.

| Method | DOGD | DEnsemble | FF | Dual | Self-Train | AdaDelay | LastHint | GrangerCausal | **LACH (Ours)** |
|---|---|---|---|---|---|---|---|---|---|
| ACC | 68.51 ± 2.8 | 67.14 ± 2.6 | 69.29 ± 1.4 | 73.53 ± 2.3 | 68.94 ± 2.1 | 71.13 ± 2.6 | 74.37 ± 2.1 | 73.41 ± 3.1 | **76.30 ± 2.3** |
| AUC | 77.54 ± 0.8 | 75.06 ± 0.9 | 77.78 ± 1.2 | 81.28 ± 1.0 | 79.34 ± 1.4 | 80.16 ± 0.7 | 81.50 ± 1.1 | 80.85 ± 1.3 | **83.68 ± 0.9** |

and label sketch are more effective and closer to the true gradient. Additionally, our proposed LACH method outperforms the Dual method as it not only explores the instant unlabeled data, but also explores the instant labels, thereby enhancing its effectiveness in asynchronous label learning tasks. Furthermore, compared with GrangerCausal which employs causal-based method to identify the most appropriate historical model, our method also outperforms it in most cases because we explore the instant data to reuse historical models more accurately. Our LACH algorithm also outperforms the Self-Train approach as it uses a novel scheduling algorithm to deal with the asynchronous labels, which exploits the structure of the delayed sequence to bridge the gap between the time of receiving delayed gradient and the current timestamp.

### 7.3 Real-world Applications

In this section, we further validate the performance of our proposed method on two real-world applications, IoT monitoring and a loan credit prediction task.

**Dataset.** The IoT Monitoring Data [27] contains public IoT traffic traces captured in the IoT environment DS2OS. This dataset addresses the lack of public botnet datasets, especially for the IoT. It suggests real traffic data gathered from 9 commercial IoT devices authentically infected by Mirai and BASHLITE. The dataset contains 7,062,606 entries and 113 variables, including timestamps, network communications of different IoT devices (feature) and the normality (label). Our goal is to predict whether the IoT device is running normally. We use logistic regression as base models. The delays are determined by the network latency.

Besides, we also investigate the performance of our proposed method on a real-world Loan data. The LendingClub Loan Data[1] is collected from LendingClub, which contains complete loan data for all loans issued through 2007 to 2018, a total of 139 months. The dataset contains about 1,300 thousand entries and 78 variables, including the current loan status (Fully Paid, Charged Off) and the latest payment information. Features include credit scores, number of finance inquiries, addresses including zip codes and state, and collections. Each data contains a delayed label feedback because the loan has a term (36 months or 60 months), and we can not receive the overdue information until the term has finished, therefore there exists the asynchronous labels issue in this application. Our goal is to predict whether the loaner will be overdue.

**Implementation Details.** We use a three-layer dense neural network as the base model, and split the dataset into 100 time intervals (i.e., $T = 100$). The delays are determined by the latency of network traces. We compare results of LACH with seven contenders, DOGD, DEnsemble, FF, Self-Train, Dual learning method, AdaDelay and LastHint methods. The learning rate is set to be 0.05, the batchsize is set as 128, and we train 200 iterations for each data batch.

**Comparison Results.** We first report the average and standard deviation of the accuracy score and AUC score on IoT datasets in Table 3, which shows that our approach has a promising performance compared to the other seven comparative methods. We compare the results of our

[1]The dataset can be downloaded from https://www.kaggle.com/wordsforthewise/lending-club

Table 4. Quantitative analysis on real-world loan credit prediction task. For each method, 5 test runs were conducted, and we present the average as well as the standard deviation of accuracy (ACC) and AUC score. The best one is emphasized in bold.

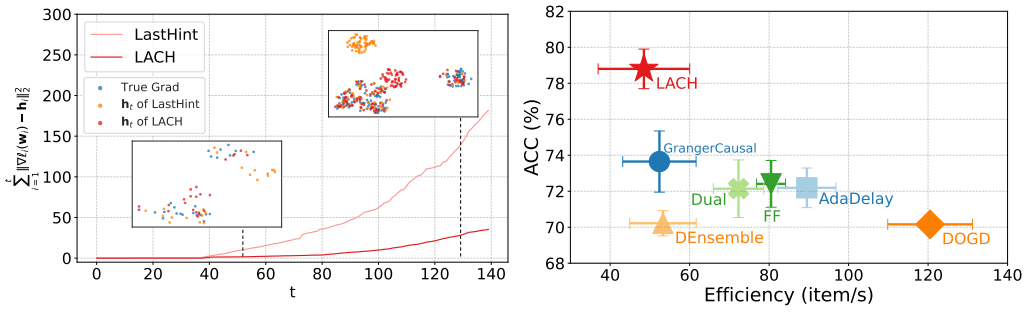
| Method | DOGD | DEnsemble | FF | Dual | Self-Train | AdaDelay | LastHint | GrangerCausal | **LACH (Ours)** |
|---|---|---|---|---|---|---|---|---|---|
| ACC | 70.16 ± 0.3 | 70.22 ± 0.7 | 72.21 ± 1.3 | 72.15 ± 1.6 | 71.47 ± 0.4 | 72.19 ± 1.1 | 71.08 ± 1.7 | 73.45 ± 1.7 | **78.80 ± 1.1** |
| AUC | 78.77 ± 0.4 | 76.19 ± 0.2 | 79.31 ± 1.5 | 79.23 ± 1.0 | 78.84 ± 0.2 | 79.15 ± 1.4 | 78.43 ± 0.5 | 79.25 ± 1.3 | **82.78 ± 0.6** |

learning asynchronous labels with hint with seven contenders, the DOGD, DEnsemble, FF, Self-Train, AdaDelay, Dual and LastHint methods. The DOGD and DEnsemble methods do not perform well on IoT Data, as they are not able to employ the instant information of the data. The FF and AdaDelay improved performances by exploring delayed feedback, but they fail to capture the asynchronous structure of the data stream. Our LACH performs better than the Self-Train method in this case, as the delayed label problem makes the model can not be updated instantly, and the error of the model will accumulate over time, hence the pseudo-label provided by self-training will be inaccurate. Dual method addresses the delayed label problem with a dual learning algorithm, our method outperforms it because we can explore instant information and update the model to deal with the delays at both label and feature levels in the asynchronous labels streaming learning setup. The LastHint method performs poorly in the IoT task. LACH also outperforms the LastHint method as the exploration of instant data by online ensemble and label sketch mechanisms makes our hint sequence closer to true gradients while simply reusing the last model's gradient fails.

Secondly, we report the average and standard deviation of the accuracy score and AUC score on the loan dataset in Table 4, which shows that our approach has a promising performance compared to the other eight comparative methods. The DOGD and Ensemble method do not perform well on Loan Data, as they are not able to employ the instant unlabeled data. Compared with Dual method, our LACH also show a better performance because it not only explores the instant unlabeled data, but also explores the instant labels, thereby enhancing its effectiveness in asynchronous label learning tasks. Furthermore, our method also outperforms the GrangerCausal which employs causal-based method to identify the most appropriate historical model, because we explore the instant data to reuse historical models more accurately. Our LACH performs better than Self-Training method in this case, as the delayed label problem makes the model can not be updated instantly, and the error of the model will accumulate over time, hence the pseudo-label provided by self-training will be inaccurate. The LastHint method performs poorly in the loan forecasting application, as the labels are delayed for a long time period and simply reusing the last model's gradient fails. Our LACH also outperforms the LastHint method as the exploration of unlabeled data by online ensemble strategy makes our hints closer to the true gradients.

**Modular Analysis.** To better validate the effectiveness of our designed hints generated by mining instant information. We investigate the accuracy of the hints, i.e., the term $\|\nabla \ell_t(\mathbf{w}_t) - \mathbf{h}_t\|$ in Theorem 1. We compare the hinting error of our LACH method with LastHint. As shown in Figure 7 (a), the hints generated by our proposed LACH method are more accurate as the quantity $\sum_{i=1}^{t} \|\nabla \ell_i(\mathbf{w}_i) - \mathbf{h}_i\|_2^2$ of our proposal is lower, which indicates that our method can employ instant data to hint the future gradient better.

We also visualize the hint vectors through the data stream to show the effectiveness of our proposed method. As shown in the subfigures in Figure 7 (a), the t-SNE analysis demonstrates that at the beginning, hints of both methods perform poorly. However, as time grows, the hints generated by our learning asynchronous labels with hint method are more accurate than the LastHint method and closer to the true gradient, which means that our proposal successfully hints the future gradients by carefully mining instant information of the data.




(a) Modular Analysis (b) Efficiency Comparison

Fig. 7. (a) Modular analysis in IoT data of our LACH compared with LastHint, which takes the last model's gradient as hint. We validate the quality of model reuse by comparing the hinting error $\sum_{i=1}^{t} \|\nabla \ell_i(\mathbf{w}_i) - \mathbf{h}_i\|_2^2$.In addition to this, the t-SNE analyses of hints $\{\mathbf{h}_t\}$ illustrate that LACH can reuse base models to update the model more accurately. (b) Efficiency comparison by evaluating the accuracy and efficiency (items processed per second) of different algorithms on the loan dataset. We report the mean and standard deviation over five runs. An algorithm closer to the top-right corner indicates superior efficiency and performance.

**Efficiency Comparison.** We also compare the efficiency of different algorithms. Specifically, we evaluate the efficiency (items processed per second) and accuracy of various algorithms on the loan credit prediction task. An algorithm that plots closer to the top-right corner indicates superior efficiency and performance since it achieves a better performance with higher efficiency. As demonstrated in Figure 7 (b), the DOGD method is the most efficient one, but it yields the poorest performance as it does not consider to alleviate the asynchronicity issue. Though the ensemble-based methods, GrangerCausal and FF, exhibit slower speeds, they accomplish superior performance. Our LACH, albeit with a slight compromise on efficiency because we carefully exploit the timely arrived information and build an online ensemble structure to adaptively combine historical models and instances, our method attains the best performance among all the methods.

## 8 CONCLUSION

In this paper, we initiate the study of stream learning with asynchronous labels, which accommodates a variety of real-world applications but is not thoroughly considered in the literature. We investigate this challenging problem, where the possible delays of both labels and features in the data stream can make it difficult for learners to online update and respond quickly. To this end, we propose a novel online ensemble approach named Learning AsynChronous label with Hint (LACH) algorithm to handle the asynchronous data, and simultaneously mine the instant arrived feature or label to alleviate the negative impact of asynchronous feedback and benefit model's online updating. Specifically, we first divide the data stream into several non-asynchronous sub-sequences, each handled by a base model, and then use hint sequence to extract information from instant data to further benefit the online model updating procedure. The proposed approaches are equipped with nice theoretical guarantees: by regret analysis, we justify the usefulness of the model ensemble and mining instant feature or label to handle the asynchronous labels, especially when the underlying distribution shifts slowly in the data stream. Empirical studies on synthetic examples verify the advantage of exploring instant data, and extensive experiments on various real-world applications validate the effectiveness of our algorithm.

In future work, we will investigate the possibility of jointly addressing the asynchronicity and other challenges in open-environment machine learning [61]. For example, we may consider to handling the conjunction of the asynchronous issue together with the non-stationary distribution shift [2, 51] or emerging new classes [28, 31].

## ACKNOWLEDGMENTS

This research was supported by National Science and Technology Major Project (2022ZD0114800), National Science Foundation of China (61921006, 62206125), National Postdoctoral Program for Innovative Talent, and China Postdoctoral Science Foundation (2023M731597).

## REFERENCES

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A Public Domain Dataset for Human Activity Recognition using Smartphones. In *Proceedings of the 21st European Symposium on Artificial Neural Networks*. 3.

[2] Yong Bai, Yu-Jie Zhang, Peng Zhao, Masashi Sugiyama, and Zhi-Hua Zhou. 2022. Adapting to Online Label Shift with Provable Guarantees. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*. 29960–29974.

[3] Amrit Singh Bedi, Alec Koppel, and Ketan Rajawat. 2019. Asynchronous online learning in multi-agent systems with proximity constraints. *IEEE Transactions on Signal and Information Processing over Networks* 5, 3 (2019), 479–494.

[4] Hamed R. Bonab and Fazli Can. 2018. GOOWE: Geometrically Optimum and Online-Weighted Ensemble Classifier for Evolving Data Streams. *ACM Transactions on Knowledge Discovery from Data* 12, 2 (2018), 25:1–25:33.

[5] Nicolo Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge university press.

[6] Olivier Chapelle. 2014. Modeling delayed feedback in display advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1097–1105.

[7] Gong Chen and Marc Teboulle. 1993. Convergence Analysis of a Proximal-Like Minimization Algorithm Using Bregman Functions. *SIAM Journal on Optimization* 3, 3 (1993), 538–543.

[8] Chao-Kai Chiang, Tianbao Yang, Chia-Jung Lee, Mehrdad Mahdavi, Chi-Jen Lu, Rong Jin, and Shenghuo Zhu. 2012. Online Optimization with Gradual Variations. In *Proceedings of the 25th Annual Conference on Computational Learning Theory (COLT)*. 6.1–6.20.

[9] João Duarte, João Gama, and Albert Bifet. 2016. Adaptive Model Rules From High-Speed Data Streams. *ACM Transactions on Knowledge Discovery from Data* 10, 3 (2016), 30:1–30:22.

[10] Genevieve Flaspohler, Francesco Orabona, Judah Cohen, Soukayna Mouatadid, Miruna Oprescu, Paulo Orenstein, and Lester Mackey. 2021. Online Learning with Optimism and Delay. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*. 3363–3373.

[11] Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, Crystal Lee, Philipp Krähenbühl, and Alexei A. Efros. 2017. A Century of Portraits: A Visual Historical Record of American High School Yearbooks. *IEEE Transactions on Computational Imaging* 3, 3 (2017), 421–431.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[13] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *ArXiv preprint* arXiv:1706.02677 (2017).

[14] Maciej Grzenda, Heitor Murilo Gomes, and Albert Bifet. 2020. Delayed labelling evaluation for data streams. *Data Mining and Knowledge Discovery* 34, 5 (2020), 1237–1266.

[15] Lei Guo, Lennart Ljung, and Pierre Priouret. 1993. Performance analysis of the forgetting factor RLS algorithm. *International Journal of Adaptive Control and Aignal Processing* 7, 6 (1993), 525–537.

[16] Elad Hazan. 2016. Introduction to Online Convex Optimization. *Foundations and Trends in Optimization* 2, 3-4 (2016), 157–325.

[17] Amélie Héliou, Panayotis Mertikopoulos, and Zhengyuan Zhou. 2020. Gradient-free Online Learning in Continuous Games with Delayed Rewards. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 4172–4181.

[18] Juan Isidro González Hidalgo, Silas G. T. C. Santos, and Roberto S. M. Barros. 2022. Dynamically Adjusting Diversity in Ensembles for the Classification of Data Streams with Concept Drift. *ACM Transactions on Knowledge Discovery from Data* 16, 2 (2022), 31:1–31:20.

[19] Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. 2022. Multi-Agent Online Optimization with Delays: Asynchronicity, Adaptivity, and Optimism. *Journal of Machine Learning Research* 23, 78 (2022), 1–49.

[20] Pooria Joulani, András György, and Csaba Szepesvári. 2013. Online Learning under Delayed Feedback. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 1453–1461.

[21] Bartosz Krawczyk, Leandro L Minku, Joao Gama, Jerzy Stefanowski, and Michal Wozniak. 2017. Ensemble learning for data stream analysis: A survey. *Information Fusion* 37 (2017), 132–156.

[22] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1-2 (1955), 83–97.

[23] Ananya Kumar, Tengyu Ma, and Percy Liang. 2020. Understanding self-training for gradual domain adaptation. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 5468–5479.
[24] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel Moore. 2010. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter* 12, 2 (2010), 74–82.
[25] Qingyang Li, Zhiwen Yu, Huang Xu, and Bin Guo. 2022. Human-machine interactive streaming anomaly detection by online self-adaptive forest. *Frontiers of Computer Science* 17, 2 (2022), 172317.
[26] Yu-Feng Li, James T Kwok, and Zhi-Hua Zhou. 2009. Semi-supervised learning using label mean. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*. 633–640.
[27] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. 2018. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing* 17, 3 (2018), 12–22.
[28] Xin Mu, Kai-Ming Ting, and Zhi-Hua Zhou. 2017. Classification Under Streaming Emerging New Classes: A Solution Using Completely-Random Trees. *IEEE Transaction on Knowledge and Data Engineering* 29, 8 (2017), 1605–1618.
[29] Francesco Orabona. 2019. A modern introduction to online learning. *ArXiv preprint* arXiv:1912.13213 (2019).
[30] Joshua Plasse and Niall M. Adams. 2016. Handling delayed labels in temporally evolving data streams. In *Proceedings of the 4th IEEE International Conference on Big Data*. 2416–2424.
[31] Yu-Yang Qian, Yong Bai, Zhen-Yu Zhang, Peng Zhao, and Zhi-Hua Zhou. 2023. Handling New Class in Online Label Shift. In *Proceedings of the 23rd IEEE International Conference on Data Mining (ICDM)*. 1283–1288.
[32] Kent Quanrud and Daniel Khashabi. 2015. Online Learning with Adversarial Delays. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*. 1270–1278.
[33] Alexander Rakhlin and Karthik Sridharan. 2013. Online Learning with Predictable Sequences. In *Proceedings of the 26th Annual Conference on Computational Learning Theory (COLT)*. 993–1019.
[34] Yuta Saito, Gota Morishita, and Shota Yasui. 2020. Dual Learning Algorithm for Delayed Conversions. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1849–1852.
[35] Shai Shalev-Shwartz et al. 2011. Online learning and online convex optimization. *Foundations and Trends in Machine Learning* 4, 2 (2011), 107–194.
[36] Vinicius MA Souza, Diego F Silva, Gustavo EAPA Batista, and João Gama. 2015. Classification of Evolving Data Streams with Infinitely Delayed Labels. In *Proceedings of the 14th IEEE International Conference on Machine Learning and Applications*. 214–219.
[37] Suvrit Sra, Adams Wei Yu, Mu Li, and Alexander J Smola. 2015. AdaDelay: Delay Adaptive Distributed Stochastic Convex Optimization. *ArXiv preprint* arXiv:1508.05003 (2015).
[38] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 127–140.
[39] Yumin Su, Liang Zhang, Quanyu Dai, Bo Zhang, Jinyao Yan, Dan Wang, Yongjun Bao, Sulong Xu, Yang He, and Weipeng Yan. 2021. An attention-based model for conversion rate prediction with delayed feedback via post-click calibration. In *Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence (IJCAI)*. 3522–3528.
[40] Yuanyu Wan, Wei-Wei Tu, and Lijun Zhang. 2022. Online strongly convex optimization with unknown delays. *Machine Learning* 111, 3 (2022), 871–893.
[41] Yuanyu Wan, Wei-Wei Tu, and Lijun Zhang. 2022. Online frank-wolfe with arbitrary delays. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*. 19703–19715.
[42] Yuanyu Wan, Yibo Wang, Chang Yao, Wei-Wei Tu, and Lijun Zhang. 2022. Projection-free Online Learning with Arbitrary Delays. *ArXiv preprint* arXiv:2204.04964 (2022).
[43] Yuanyu Wan, Chang Yao, Mingli Song, and Lijun Zhang. 2024. Improved Regret for Bandit Convex Optimization with Delayed Feedback. *ArXiv preprint* arXiv:2402.09152 (2024).
[44] Jing Wang, Peng Zhao, and Zhi-Hua Zhou. 2023. Revisiting Weighted Strategy for Non-stationary Parametric Bandits. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 7913–7942.
[45] Yanshi Wang, Jie Zhang, Qing Da, and Anxiang Zeng. 2020. Delayed Feedback Modeling for the Entire Space Conversion Rate Prediction. *ArXiv preprint* arXiv:2011.11826 (2020).
[46] Marcelo J. Weinberger and Erik Ordentlich. 2002. On delayed prediction of individual sequences. *IEEE Transactions on Information Theory* 48, 7 (2002), 1959–1976.
[47] Muning Wen, Runji Lin, Hanjing Wang, Yaodong Yang, Ying Wen, Luo Mai, Jun Wang, Hai-Feng Zhang, and Weinan Zhang. 2023. Large sequence models for sequential decision-making: a survey. *Frontiers of Computer Science* 17, 6 (2023), 176349.

[48] Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. 2012. Online feature selection with streaming features. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 5 (2012), 1178–1192.

[49] Min Yang, Ying Shen, Xiaojun Chen, and Chengming Li. 2020. Multi-source Domain Adaptation for Sentiment Classification with Granger Causal Inference. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1849–1852.

[50] Xiao Zhang, Haonan Jia, Hanjing Su, Wenhan Wang, Jun Xu, and Ji-Rong Wen. 2021. Counterfactual reward modification for streaming recommendation with delayed feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 41–50.

[51] Yu-Jie Zhang, Zhen-Yu Zhang, Peng Zhao, and Masashi Sugiyama. 2023. Adapting to continuous covariate shift via online density ratio estimation. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*. 29074–29113.

[52] Zhao Zhang, Yong Zhang, Da Guo, Shuang Zhao, and Xiaolin Zhu. 2023. Communication-efficient federated continual learning for distributed learning system with Non-IID data. *Science China Information Sciences* 66, 2 (2023).

[53] Zhen-Yu Zhang, Yu-Yang Qian, Yu-Jie Zhang, Yuan Jiang, and Zhi-Hua Zhou. 2022. Adaptive Learning for Weakly Labeled Streams. In *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2556–2564.

[54] Peng Zhao, Le-Wen Cai, and Zhi-Hua Zhou. 2020. Handling concept drift via model reuse. *Machine Learning* 109, 3 (2020), 533–568.

[55] Peng Zhao, Xinqiang Wang, Siyu Xie, Lei Guo, and Zhi-Hua Zhou. 2021. Distribution-Free One-Pass Learning. *IEEE Transaction on Knowledge and Data Engineering* 33 (2021), 951 – 963. Issue 3.

[56] Peng Zhao, Yu-Jie Zhang, Lijun Zhang, and Zhi-Hua Zhou. 2020. Dynamic Regret of Convex and Smooth Functions. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*. 12510–12520.

[57] Peng Zhao, Yu-Jie Zhang, Lijun Zhang, and Zhi-Hua Zhou. 2024. Adaptivity and Non-stationarity: Problem-dependent Dynamic Regret for Online Convex Optimization. *Journal of Machine Learning Research* 25, 98 (2024), 1–52.

[58] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhiming Ma, and Tie-Yan Liu. 2017. Asynchronous Stochastic Gradient Descent with Delay Compensation. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 4120–4129.

[59] Zhi-Hua Zhou and Zhi-Hao Tan. 2023. Learnware: Small models do big. *Science China Information Sciences* 67, 1 (2023), 112102.

[60] Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms.* CRC press.

[61] Zhi-Hua Zhou. 2022. Open-environment machine learning. *National Science Review* 9, 8 (2022), nwac123.

[62] Zhi-Hua Zhou. 2022. Rehearsal: Learning from prediction to decision. *Frontiers of Computer Science* 16, 4 (2022), 164352.