Ensemble of Distributed Learners for Online Classification of Dynamic Data Streams

Luca Canzian, Yu Zhang, and Mihaela van der Schaar

***Abstract*—We present a distributed online learning scheme to classify data captured from distributed and dynamic data sources. Our scheme consists of multiple distributed local learners, which analyze different streams of data that are correlated to a common event that needs to be classified. Each learner uses a local classifier to make a local prediction. The local predictions are then collected by each learner and combined using a weighted majority rule to output the final prediction. We propose a novel online ensemble learning algorithm to update the aggregation rule in order to adapt to the underlying data dynamics. We rigorously determine an upper bound for the worst-case mis-classification probability of our algorithm, which tends asymptotically to 0 if the mis-classification probability of the best (unknown) static aggregation rule is 0. Then we extend our algorithm to address challenges specific to the distributed implementation and prove new bounds that apply to these settings. Finally, we test our scheme by performing an evaluation study on several data sets.**

***Index Terms*—Online learning, distributed learning, machine learning, ensemble of classifiers, dynamic streams, concept drift, big data, classification.**

I. Introduction

Recent years have witnessed the proliferation of data-driven applications that exploit the large amount of data captured from distributed, heterogeneous, and dynamic (i.e., whose characteristics are varying over time) data sources. Examples of such applications include surveillance [1], driver assistance systems [2], social multimedia [3], and network monitoring [4]. However, the effective utilization of such high-volume data also involves significant challenges that are the main concern of this work. First, the captured data need to be analyzed *online* (e.g., to make predictions and timely decisions based on these predictions); thus, the learning algorithms need to deal with the time-varying characteristics of the underlying data, i.e., adequately deal with *concept-drift* [5]. Second, the privacy requirements and the communication and sharing costs make it difficult to collect and store all the observed data. Third, the devices that collect the data may be managed by different entities (e.g., multiple hospitals, multiple camera systems, multiple routers, etc.) and may follow policies (e.g., type of information to exchange, rate at which data are collected, etc.) that are not centrally controllable.

Luca Canzian is with Qascom, via Orazio Marinali 87, Bassano del Grappa, Vicenza, Italy. E-mail: luca.canzian@qascom.it

Yu Zhang is with Microsoft Corporation, Sunnyvale CA 94087, USA. E-mail: yuzhan@microsoft.com

Mihaela van der Schaar is with the Department of Electrical Engineering, UCLA, Los Angeles CA 90095, USA. E-mail: mihaela@ee.ucla.edu

This work was performed while the 3 authors were at UCLA and was partially supported by the AFOSR DDDAS grant and the NSF ECCS 1407712 grant.

To address these challenges, we propose an *online ensemble learning* technique, which we refer to as Perceptron Weighted Majority (PWM). Specifically, we consider a set of distributed learners that observe data from different sources, which are correlated to a common event that must be classified by the learners. We focus on binary classification problems.[1] Each learner works in the following manner. For each single instance that enters the system, the learner makes the final classification decision by collecting the local predictions of all the learners and combining them using a weighted majority rule as in [7]–[16]. After making the final prediction, the learner is told the real value, i.e., the *label*, associated to the event to classify. Exploiting such information, the learner updates the aggregation weights adopting a perceptron learning rule [17].

We remark that this paper considers a distributed structure in which each learner waits to receive the local predictions from all the other learners before making a final prediction. However, this does not imply that the physical network topology must be a complete network (i.e., that each learner must be directly connected to all the other learners); in fact, as long as the network is connected, the local predictions can be spread over the network using a multi-hop communication protocol. This setting differs from the typical distributed implementation of online inference algorithms, where at each step each agent exchanges information only by means of local interactions with its neighbors. The algorithm proposed in this paper is open to generalizations to the latter scenario. A detailed study of the implied convergence issue is left for future work.

The main features of the proposed scheme are:

DIS: Distributed data streams. The majority of the existing ensemble schemes proposed in literature assume that the learners make a prediction after observing the same data [8]–[16], [18], [19]. Our approach does not rely on such an assumption, allowing for the possibility that the distributed learners observe *different* correlated data streams. In particular, the statistical dependency among the label and the observation of a learner can be different from the statistical dependency among the label and the observation of another learner, i.e., each data source can have a specific generating process [20].

DYN: Dynamic data streams. Many existing ensemble schemes [7]–[11] assume that the data are generated from a stationary distribution, i.e., that the *concept is stable* [5]. Our scheme is developed and evaluated, both analytically and experimentally, considering the possibility that the data

[1] We remark that a multi-class classifier can be decomposed as a cascade of binary classifiers [6].

streams are dynamic, i.e., they may experience *concept drift* [5].

ONL: Online learning. To deal with dynamic data streams our scheme must learn the aggregation rule "on-the-fly". In this way the learners maintain an up-to-date aggregation rule and are able to track the concept drifts.

COM: Low complexity. Some online ensemble learning schemes, such as [11]–[13], [18], need to collect and store chunks of data, which are later processed to update the aggregation model of the system. This requires a large memory and high computational capabilities, thereby resulting in high implementation cost. Different from these approaches, in our scheme each data is processed only once "on-arrival", thus data are not stored in the system. Only the up-to-date aggregation model is kept in the memory. The local prediction of each learner, which is the only information that must be exchanged, consists of a binary value. Moreover, our scheme is *scalable* to a large number of sources and learners and the learners can be chained in any hierarchical structure.

IND: Independence from local classifiers. Different from [15], [16], [19], our scheme is general and can be applied to different types of local classifiers, such as support vector machine, decision tree, neural networks, offline/online classifiers, etc. This feature is important, because the different learners can be managed by different entities, willing to cooperate in exchanging information but not to modify their own local classifiers. Also, our algorithm does not need any a priori knowledge about the performance of the local classifiers, it automatically adapts the configuration of the distributed system to the current performance of the local classifiers.

DEL: Delayed labels, missing labels, and asynchronous learners. In distributed environments there are many factors that may impact the performance of the learning system. First, because obtaining the information about the label may be both costly and time consuming, one cannot expect that all the learners always observe the label in a timely manner. Some learners can receive the label with delay, or not receive it at all. Second, the learners can be asynchronous, i.e., they can observe data at different time instants. In this paper we first propose a basic algorithm, considering an idealized scenario in which the above issues are not present, and then we extend our scheme to deal with the above issues.

The rest of this paper is organized as follows. Section II reviews the existing literature in ensemble learning techniques. Section III presents our formalism, framework, and algorithm for distributed online learning. Section IV proves a bound for the mis-classification probability of our scheme which depends on the mis-classification probabilities of the best (unknown) static aggregation rule, and of the best (unknown) local classifier. Section V discusses several extensions to our learning algorithm to deal with practical issues associated to the distributed implementation of the ensemble of learners, and proves new bounds that apply to these settings. Section VI presents the empirical evaluation of our algorithm on several data sets. Section VII concludes the paper.

II. Related Works

In this section we review the existing literature on ensemble learning techniques and discuss the differences between the cited works and our paper.

Ensemble learning techniques [21]–[23] combine a collection of base classifiers into a unique classifier. Adaboost [7], for example, trains a sequence of classifiers on increasingly more difficult examples and combines them using a weighted majority rule. Our paper is clearly different with respect to traditional offline approaches such as Adaboost, which rely on the presence of a training set for offline training the ensemble and assume a stable concept.

An online version of Adaboost is proposed in [11]. When a new chunk of data enters the system, the current classifiers are reweighed, a weighted training set is generated, a new classifier (and its weight) is created on this data set, and the oldest classifier is discarded. Similar proposals are made in [12], [13], [18], [24]. Our work differs from these online boosting-like techniques because (i) it processes each instance "on arrival" only once, without the need for storage and reprocessing chunks of data, and (ii) it does not require that the local classifiers are centrally retrained (e.g., in a distributed scenario it may be expensive to retrain the local classifiers or unfeasible if the learners are operated by different entities).

An alternative approach to storing chunks of labeled data consists in updating the ensemble as soon as data flows in the system. [15] and [16] adopt a dynamic weighted majority algorithm, refining, adding, and removing learners based on the global algorithm's performance. [19] proposes a scheme based on two online ensembles with different levels of diversity. The low diversity ensemble is used for system predictions, the high diversity ensemble is used to learn the new concept after a drift is detected. Our work differs from [15], [16], [19] because it does not require that the local classifiers are centrally retrained.

The literature closest to our work is represented by the multiplicative weight update schemes [8]–[10], [14] that maintain a collection of given learners, predict using a weighted majority rule, and update online the weights associated to the learners in a multiplicative manner. Weighted majority [8] decreases the weights of the learners in the pool that disagree with the label whenever the ensemble makes a mistakes. Winnow2 [9] uses a slightly different update rule, but the final effect is the same as weighted majority. In [10] the weights of the learners that agree with the label when the ensemble makes a mistakes are increased, and the weights of the learners that disagree with the label are decreased also when the ensemble predicts correctly. To prevent the weights of the learners which performed poorly in the past from becoming too small with respect to the other learners, [14] proposes a modified version of these schemes adding a phase, after the multiplicative weight update, in which each learner shares a portion of its weight with the other learners. In our algorithm, differently from [8]–[10], [14], the weights are updated in an additive manner and learners can also have negative weights (e.g., a learner that is always wrong would receive a negative weight and could contribute to the system as a learner that is always right).

	DIS	DYN	ONL	COM	IND	DEL
[7]				X	X	
[11]–[13], [18], [24]		X	X		X	
[8]–[10], [14]		X	X	X	X	
[15], [16], [19]		X	X	X		
our work	X	X	X	X	X	X

TABLE I: Comparison among different ensemble learning works.

Finally, we differentiate from all the cited works in another key point: we consider a distributed scenario, allowing for the possibility that the learners observe different data streams. This is the reason why in Section V we extend our learning algorithm to address challenges specific to the distributed implementation.

Table I summarizes the differences between our approach and the cited works in terms of the features described in Section I.

III. Distributed Learning Framework And The Proposed Algorithm

We consider a set of K distributed *learners*, denoted by $\mathcal{K} = \{1, \dots, K\}$. Each learner observes a separate sequence of instances. The time is slotted and the learners are synchronized. Throughout the paper, we use the indices i and j to denote particular learners, the indices n and m to denote particular time instants, the index N to denote the possible infinite time horizon (i.e., for how many slots the system operates), and bold letters to denote vectors.

At the beginning of each time slot n, each learner i observes an instance generated by a *source* $S_i^{(n)}$. Let $\mathbf{x}_i^{(n)} \in \mathcal{X}_i$ denote the multi-dimensional *instance* observed by learner i at time instant n, and $y^{(n)} \in \{-1, 1\}$ denote the corresponding *label*, a common event that the learners have to classify at time instant n. We call the pair $(\mathbf{x}_i^{(n)}, y^{(n)})$ a *labeled instance*. We formally define a source $S_i^{(n)} \triangleq \left\{p_i^{(n)}(\mathbf{x}_i^{(n)}, y^{(n)})\right\}$ for learner i at time instant n as the probability density function $p_i^{(n)}$ over the labeled instance $(\mathbf{x}_i^{(n)}, y^{(n)})$. We write $\mathbf{S}^{(n)} = \left(S_1^{(n)}, \dots, S_K^{(n)}\right)$ for the *vector of sources* at time instant n.

The task of a generic learner i at time instant n is to predict the label $y^{(n)}$. The prediction utilizes the idea of ensemble data mining: each learner adopts an individual classifier to generate a *local prediction*, the local predictions are exchanged, and learner i aggregates its local prediction and the received ones to generate the final prediction $\hat{y}_i^{(n)} \in \{-1, 1\}$. Let $s_i^{(n)} \in \{-1, 1\}$ denote the local prediction of learner i at time instant n. As in [8]–[10], [14], in this paper we assume that the local classifiers are given (i.e., $s_i^{(n)}$ is given $\forall\, i \in \mathcal{K}$) and we focus on the adaptivity of the rule that aggregates the local predictions.

Similarly to most ensemble techniques, such as [7]–[16], we consider a *weighted majority* aggregation rule[2], in which learner i maintains a *weight vector* $\mathbf{w}_i^{(n)} \triangleq (w_{i0}^{(n)}, w_{i1}^{(n)}, \dots, w_{ik}^{(n)}) \in \Re^{K+1}$, combines it linearly with the *local prediction vector* $\mathbf{s}^{(n)} \triangleq (1, s_1^{(n)}, \dots, s_k^{(n)})$, and predicts -1 if the result is negative, 1 otherwise, i.e.,

$$\hat{y}_i^{(n)} = \mathrm{sgn}\left(\mathbf{w}_i^{(n)} \cdot \mathbf{s}^{(n)}\right) = \begin{cases} 1 & \text{if } \mathbf{w}_i^{(n)} \cdot \mathbf{s}^{(n)} \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

where $\mathrm{sgn}(\cdot)$ is the sign function (we define $\mathrm{sgn}(0) \triangleq 1$) and $\mathbf{w}_i^{(n)} \cdot \mathbf{s}^{(n)} \triangleq w_{i0}^{(n)} + \sum_{j=1}^{K} w_{ij}^{(n)} s_j^{(n)}$ is the inner product among the vectors $\mathbf{w}_i^{(n)}$ and $\mathbf{s}^{(n)}$. The equation $w_{i0}^{(n)} + \sum_{j=1}^{K} w_{ij}^{(n)} s_j^{(n)} = 0$ defines an hyperplane in $\Re^K$ (the space of the local predictions) which separates the positive predictions (i.e., $\hat{y}_i^{(n)} = 1$) from the negative ones (i.e., $\hat{y}_i^{(n)} = -1$). Notice that in most of the weighted majority schemes proposed in literature, [7]–[16], $w_{i0}^{(n)} = 0$ which constrains the hyperplane to pass through the origin. However, in our paper the weight $w_{i0}^{(n)}$ can be thought of as the weight associated to a "virtual learner" that always sends the local prediction 1, and we introduce it to exploit an additional degree of freedom.

We consider the following rule to update the weight vector $\mathbf{w}_i^{(n)}$ at the end of time instant n:

$$\mathbf{w}_i^{(n+1)} = \begin{cases} \mathbf{w}_i^{(n)} & \text{if } \hat{y}_i^{(n)} = y^{(n)} \\ \mathbf{w}_i^{(n)} + y^{(n)}\mathbf{s}^{(n)} & \text{otherwise} \end{cases} \quad (2)$$

That is, after having observed the true label, learner i compares it with its prediction. If the prediction is correct, the model is not modified. If the prediction is incorrect, the weights of the learners that reported a wrong prediction are decreased by one unit, whereas the weights of the learners that reported a correct prediction are increased by one unit.[3]

Algorithm Perceptron Weighted Majority (PWM)

1: **Initialization**: $w_{ij} = 0, \forall\, i, j \in \mathcal{K}$
2: **For** each learner i and time instant n
3: Observe $\mathbf{x}_i^{(n)}$
4: Obtain $\mathbf{s}^{(n)} = (1, s_1^{(n)}, \dots, s_k^{(n)})$
5: Predict $\hat{y}_i^{(n)} \leftarrow \mathrm{sgn}(\mathbf{w}_i \cdot \mathbf{s}^{(n)})$
6: Observe $y^{(n)}$
7: **If** $y^{(n)} \neq \hat{y}_i^{(n)}$ **do** $\mathbf{w}_i \leftarrow \mathbf{w}_i + y^{(n)}\mathbf{s}^{(n)}$

Since (2) is analogous to the learning rule of a Perceptron algorithm [17], we call the resulting online learning scheme Perceptron Weighted Majority (PWM). We initialize to 0 the weights $w_{ij}^{(1)}$, $i, j \in \mathcal{K}$. Because at the end of each time instant n the value of $w_{ij}^{(n)}$ can remain constant, decrease by one unit, or increase by one unit, $w_{ij}^{(n)}$ is always an integer number.

To summarize, the sequence of events that take place at time instant n for each learner i adopting the PWM algorithm can be described as follows.

[2]We point out that in the distributed detection and data fusion literature [25]–[27] it is known that, for a set of fixed local decision rules, the optimal fusion rule based on the data received from the sensors is a weighted sum of the local decisions.

[3]The proposed update rule increases / decreases the weights by one unit because it assumes that all the mis-classification errors have the same importance. However, the update rule can straightforwardly be modified to take into account differences among mis-classification errors (i.e., among false alarms and mis-detections): in this case the weight increase / decrease must be larger when the most important mis-classification error occurs.

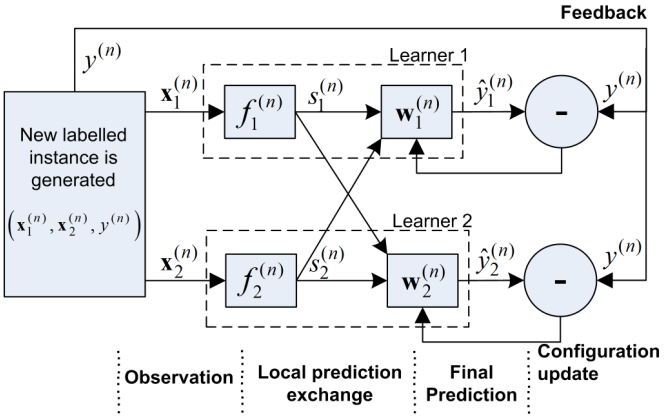

Fig. 1: Illustrative system of two learners adopting the PWM scheme

K	Number of learners
i, j	Indexes to denote particular learners
N	Time horizon
n, m	Indexes to denote particular time instants
$\mathbf{x}_i^{(n)}$	Instance at time instant n
$y^{(n)}$	Label at time instant n
$s_i^{(n)}$	Local prediction of learner i at time instant n
$\mathbf{s}^{(n)}$	Local prediction vector at time instant n
$\hat{y}_i^{(n)}$	Final prediction of learner i at time instant n
$w_{ij}^{(n)}$	Learner i's weight used to combine learner j's local prediction at time instant n
$\mathbf{w}_i^{(n)}$	Learner i's weight vector used to combine the local predictions at time instant n
$\mathbf{D}_N$	A sequence of N labeled instances
$P_i(\mathbf{D}_N)$	Mis-classification probability of the local classifier used by learner i
$P^*(\mathbf{D}_N)$	Mis-classification probability of the most accurate local classifier
$v^*(\mathbf{D}_N)$	Number of local classifiers whose mis-classification probabilities are $P^*(\mathbf{D}_N)$
$\mathbf{w}^O$	Optimal static weight vector
$P^O(\mathbf{D}_N)$	Mis-classification probability of learner i if it uses $\mathbf{w}^O$
$P_i^{PWM}(\mathbf{D}_N)$	Mis-classification probability of learner i if it adopts the PWM scheme

TABLE II: Summary of the considered notations.

1. **Observation:** learner i observes the instance $\mathbf{x}_i^{(n)}$;
2. **Local prediction exchange:** learner i sends its local prediction $s_i^{(n)} = f_i^{(n)}(\mathbf{x}_i^{(n)})$ to the other learners, and receives the local predictions $\mathbf{s}_j^{(n)} = f_j^{(n)}(\mathbf{x}_j^{(n)})$, $\forall\, j \neq i$, from the other learners;
3. **Final prediction:** learner i computes and outputs its final prediction $\hat{y}_i^{(n)} = \mathrm{sgn}\left(\mathbf{w}_i^{(n)} \cdot \mathbf{s}^{(n)}\right)$;
4. **Feedback:** learner i observes the true label $y^{(n)}$;
5. **Configuration update:** learner i updates the weight vector $\mathbf{w}_i^{(n)}$ adopting (2).

Fig. 1 illustrates this sequence of events for a system of two learners, whereas Table II summarizes the most important notations used in this paper.

IV. Performance of PWM

In this section we analytically quantify the performance of PWM in terms of its *empirical mis-classification probability* (shortly, mis-prediction probability), which is defined as the number of prediction mistakes per instance.

We prove two upper bounds for the mis-classification probability of our scheme. The first bound depends on the mis-classification probability of the best (unknown) static aggregation rule, and is particularly useful when the local classifiers are *weak* (i.e., their performance are comparable to random guessing) but their combination can result in an accurate ensemble.[4] The second bound depends on the mis-classification probability of the best (unknown) local classifiers, and is particularly useful when there are accurate local classifiers in the system. We then combine these two bounds into a unique bound. We show that the resulting bound and the mis-classification probability of PWM tend asymptotically to 0 if the mis-classification probability of the best static aggregation rule or the mis-classification probability of the best local classifier tends to 0. Then we formally define the notions of *concept* and *concept drift* and we show that the mis-classification probability of PWM tends to 0 if, for each concept, there exists a (unknown) static aggregation rule whose mis-classification probability (for the considered concept) tends to 0.

Importantly, we remark that PWM is designed in absence of a priori knowledge about the sources and the performance of the local classifiers. We do not need to know a priori whether there are accurate local classifiers or accurate aggregation rules. It is the scheme itself that adapts the configuration of the distributed system to the current performance of the local classifiers.

A. Definitions

Given the sequence of N labeled instances $\mathbf{D}_N \triangleq \left(\mathbf{x}_1^{(n)}, \ldots, \mathbf{x}_K^{(n)}, y^{(n)}\right)_{n=1,\ldots,N}$, we denote by $P_i(\mathbf{D}_N)$ the mis-classification probability of the local classifier used by learner i, by $P^*(\mathbf{D}_N)$ the mis-classification probability of the most accurate local classifier, and by $v^*(\mathbf{D}_N)$ the number of local classifiers whose mis-classification probabilities are $P^*(\mathbf{D}_N)$,[5]

$$
\begin{aligned}
P_i(\mathbf{D}_N) &\triangleq \frac{1}{N}\sum_{n=1}^{N} I\{s_i^{(n)} \neq y^{(n)}\} \\
P^*(\mathbf{D}_N) &\triangleq \min_{i\in\mathcal{K}} P_i(\mathbf{D}_N) \\
v^*(\mathbf{D}_N) &\triangleq |\{i : P_i(\mathbf{D}_N) = P^*(\mathbf{D}_N)\}|
\end{aligned} \tag{3}
$$

where $|\cdot|$ denotes the cardinality of the considered set. Notice that the considered mis-classification probabilities are defined adopting an empirical formulation.

Also, we denote by $P^O(\mathbf{D}_N)$ the mis-classification probability of learner i if it combines the local predictions of all the learners using the *optimal static weight vector* $\mathbf{w}^O$ that minimizes its number of mistakes (notice that $P^O(\mathbf{D}_N)$ and $\mathbf{w}^O$ are the same for all the learners, hence we do not use the

[4]It is known that the combination of weak classifiers can result in a high accurate ensemble [28], in particular when the classifiers are diverse and their errors are independent.

[5]This paper does not distinguish among different classification errors, i.e., among false alarms and mis-detections.

subscript i),

$$\mathbf{w}^O \triangleq \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N} I\{\operatorname{sgn}\left(\mathbf{w}\cdot\mathbf{s}^{(n)}\right)\neq y^{(n)}\}$$

$$P^O(\mathbf{D}_N) \triangleq \frac{1}{N}\sum_{n=1}^{N} I\{\operatorname{sgn}\left(\mathbf{w}^O\cdot\mathbf{s}^{(n)}\right)\neq y^{(n)}\} \quad (4)$$

Remark 1. $P^O(\mathbf{D}_N) \leq P^*(\mathbf{D}_N)$. Indeed learner i can at least achieve the mis-classification probability of the most accurate local classifier using a static weight vector, this is achieved by assigning a positive weight to the local prediction belonging to the most accurate local classifier and zero weights to all the other local predictions.

Remark 2. The computation and adoption of $\mathbf{w}^O$ would require to know in advance, at the beginning of time instant 1, the sequences of local predictions $\mathbf{s}^{(n)}$ and labels $y^{(n)}$, for every time instant $n = 1, \ldots, N$.

Moreover, we denote by $P_i^{PWM}(\mathbf{D}_N)$ the mis-classification probability of learner i if it adopts the PWM scheme,

$$P_i^{PWM}(\mathbf{D}_N) \triangleq \frac{1}{N}\sum_{n=1}^{N} I\{\operatorname{sgn}\left(\mathbf{w}^{(n)}\cdot\mathbf{s}^{(n)}\right)\neq y^{(n)}\} \quad (5)$$

where $w_{ij}^{(1)} = 1$, $\forall\, i, j$, and $\mathbf{w}_i^{(n)}$ evolves according to (2). We denote by $P^{PWM}(\mathbf{D}_N)$ the average mis-classification probability of the distributed system if all the learners adopt the PWM scheme,

$$P^{PWM}(\mathbf{D}_N) \triangleq \frac{1}{K}\sum_{i=1}^{K} P_i^{PWM}(\mathbf{D}_N) \quad (6)$$

Remark 3. In this section $P^{PWM}(\mathbf{D}_N) = P_i^{PWM}(\mathbf{D}_N)$, $\forall\, i$, because the weight vectors of the learners are equally initialized and we assumed that the learners are synchronized and always observe the labels, hence $\mathbf{w}_i^{(n)}$ and $\mathbf{w}_j^{(n)}$ evolve in the same way and $P_i^{PWM}(\mathbf{D}_N) = P_j^{PWM}(\mathbf{D}_N)$, $\forall\, i, j$. However, in Section V we describe several extensions to our online learning algorithm, in which $\mathbf{w}_i^{(n)}$ and $\mathbf{w}_j^{(n)}$ evolve differently, and consequently $P_i^{PWM}(\mathbf{D}_N) \neq P_j^{PWM}(\mathbf{D}_N)$, $i \neq j$.

B. Bounds for PWM mis-classification probability

In this subsection we derive the following results. Lemma 1 proves a bound for $P^{PWM}(\mathbf{D}_N)$ as a function of $P^O(\mathbf{D}_N)$. Lemma 2 proves a bound for $P^{PWM}(\mathbf{D}_N)$ as a function of $P^*(\mathbf{D}_N)$. Theorem 1 combines these two bounds into a unique bound. Finally, as a special case of Theorem 1, Theorem 2 shows that $P^{PWM}(\mathbf{D}_N)$ converges to 0 if $P^*(\mathbf{D}_N)$ or $P^O(\mathbf{D}_N)$ converge to 0.

Lemma 1. *For every sequence of labeled instances $\mathbf{D}_N$, the mis-classification probability $P^{PWM}(\mathbf{D}_N)$ is bounded by* $\mathbf{B}_1(\mathbf{D}_N) \triangleq 2KP^O(\mathbf{D}_N) + \dfrac{K(K+1)}{N}$.

Proof: See Appendix A. ∎

Remark 4. Lemma 1 shows that it is not always beneficial to have many learners in the system. On one hand, an additional learner can decrease the benchmark prediction probability $P^O(\mathbf{D}_N)$. On the other hand, it increases the number of learners K, and as a consequence the maximum number or errors needed to approach the benchmark weight vector $\mathbf{w}^O$ increases. The final impact on $P^{PWM}(\mathbf{D}_N)$ depends on which of the two effects is the strongest. As an extreme case, if the K learners are such that the product $2KP^O(\mathbf{D}_N)$ is larger than 1, then the bound $\mathbf{B}_1(\mathbf{D}_N)$ becomes meaningless. In the other extreme case, if the K learners are such that the optimal static weight vector $\mathbf{w}^O$ allows to predict always correctly the labeled instances $\mathbf{D}_N$, i.e., $P^O(\mathbf{D}_N) = 0$, then $P^{PWM}(\mathbf{D}_N) \leq \frac{K(K+1)}{N}$. In this case, the bound increases quadratically in the number of learners K, but decreases linearly in the number of instances N.

We define the function $f(x,y) \triangleq 2x + \frac{K+1}{2Ny} + \sqrt{\left(\frac{K+1}{2Ny}\right)^2 + \frac{2(K+1)x}{Ny}}$.

Lemma 2. *For every sequence of labeled instances $\mathbf{D}_N$, the mis-classification probability $P^{PWM}(\mathbf{D}_N)$ is bounded by* $\mathbf{B}_2(\mathbf{D}_N) \triangleq f\left(P^*(\mathbf{D}_N), v^*(\mathbf{D}_N)\right)$.

Proof: See Appendix B. ∎

Remark 5. If the best local classifier always predicts correctly the labeled instances $\mathbf{D}_N$, i.e., $P^*(\mathbf{D}_N) = 0$, then $P^{PWM}(\mathbf{D}_N) \leq \frac{K+1}{N\cdot v^*(\mathbf{D}_N)}$. This bound is $K \cdot v^*(\mathbf{D}_N)$ times better than the bound in Remark 4.

Remark 6. Asymptotically, for $N \to +\infty$, $\mathbf{B}_1(\mathbf{D}_N) \to 2KP^O(\mathbf{D}_N)$ and $\mathbf{B}_2(\mathbf{D}_N) \to 2P^*(\mathbf{D}_N)$. On one hand, if the local classifiers are weak (i.e., $P^*(\mathbf{D}_N) \simeq 0.5$) but their aggregation is very accurate (i.e., $P^O(\mathbf{D}_N) \ll 1$), the first bound is usually stricter than the second. On the other hand, if the performance of the best local classifier is comparable with the performance of the optimal static aggregation rule (i.e., $P^*(\mathbf{D}_N) \simeq P^O(\mathbf{D}_N)$), the second bound is K times stricter than the first one. Notice that also the bound computed in [8], for the multiplicative update rule, depends linearly on the accuracy of the best classifier.

In the following theorem, we combine $\mathbf{B}_1(\mathbf{D}_N)$ and $\mathbf{B}_2(\mathbf{D}_N)$ into a unique bound.

Theorem 1. *For every sequence of labeled instances $\mathbf{D}_N$, the mis-classification probability $P^{PWM}(\mathbf{D}_N)$ is bounded by* $\mathbf{B}(\mathbf{D}_N) \triangleq \min\left\{\mathbf{B}_1(\mathbf{D}_N), \mathbf{B}_2(\mathbf{D}_N), 1\right\}$.

Proof: We simply combine Lemmas 1 and 2, and the fact that the mis-prediction probability cannot be larger than 1. ∎

Importantly, notice that the bound $\mathbf{B}(\mathbf{D}_N)$ is valid for any time horizon N and for any sequence of labeled instances $\mathbf{D}_N$. As a particular case, if the time horizon tends to infinity and there exists either 1) a static aggregation weight vector whose mis-classification probability tends to 0 (i.e., $P^O(\mathbf{D}_N) \to 0$), or 2) a local classifier whose mis-classification probability tends to 0 (i.e., $P^*(\mathbf{D}_N) \to 0$), we obtain that the mis-classification probability of PWM tends to 0 as well. Notice that $P^*(\mathbf{D}_N) \to 0$ is a specific case of $P^O(\mathbf{D}_N) \to 0$, because $P^O(\mathbf{D}_N) \leq P^*(\mathbf{D}_N)$. Hence, in the statement of the following theorem we consider only the case $P^O(\mathbf{D}_N) \to 0$.

Theorem 2. *If* $\lim_{N\to+\infty} P^O(\mathbf{D}_N) = 0$*, then* $\lim_{N\to+\infty} P^{PWM}(\mathbf{D}_N) = 0$.

Proof: $P^{PWM}(\mathbf{D}_N) \leq 2KP^O(\mathbf{D}_N) + \frac{K(K+1)}{N}$ and the right hand side tends to 0 for $N \to +\infty$. ∎

C. Bound in the Presence of Concept Drifts

Given two time instants n and m, $n > m$, we write $S_i^{(n)} = S_i^{(m)}$ if the labeled instances $(\mathbf{x}_i^{(n)}, y_i^{(n)})$ and $(\mathbf{x}_i^{(m)}, y_i^{(m)})$ are independently sampled from the same distribution. We write $\mathbf{S}^{(n)} = \mathbf{S}^{(m)}$ if $S_i^{(n)} = S_i^{(m)}$, $\forall\, i \in \mathcal{K}$. As in [5], we refer to a particular vector of sources as a *concept*. The expression *concept drift* [4], [5], [12]–[16], [18], [19], [29]–[31] refers to a change of concept that occurs in a certain time instant. According to [5], we say that at time instant n there is a concept drift if $\mathbf{S}^{(n+1)} \neq \mathbf{S}^{(n)}$.

Theorem 2 states that $P^{PWM}(\mathbf{D}_N) \to 0$ if $P^O(\mathbf{D}_N) \to 0$. Unfortunately, in presence of concept drifts it is highly improbable that $P^O(\mathbf{D}_N) \to 0$. In fact, the accuracies of the local classifiers can change consistently from one concept to another, and the best weight vector to aggregate the local predictions changes accordingly. In the following we generalize the result of Theorem 2 considering an assumption that is more realistic if there are concept drifts.

We denote by $\mathbf{D}_{N_c}$ a sequence of N_c labeled instances generated by the concept $\mathbf{S}_c^{(n)}$. We say that the concept $\mathbf{S}_c^{(n)}$ is *learnable* if, $\forall\, \mathbf{D}_{N_c}$,

$$\lim_{N_c\to+\infty} \min_{\mathbf{w}_{i,c}^O} \frac{1}{N_c} \sum_{n=1}^{N_c} I\{\text{sgn}\left(\mathbf{w}_{i,c}^O \cdot \mathbf{s}^{(n)}\right) \neq y^{(n)}\} = 0 \quad (7)$$

That is, the concept $\mathbf{S}_c^{(n)}$ is learnable if there exists a static weight vector $\mathbf{w}_{i,c}^O$ whose asymptotic mis-classification probability, over the labelled instances generated by that concept, tends to 0.

Theorem 3. *If* $\mathbf{D}_N$*, for* $N \to +\infty$*, is generated by a finite number of learnable concepts and a finite number of concept drifts occurred, then* $\lim_{N\to+\infty} P^{PWM}(\mathbf{D}_N) \to 0$.

Proof: See Appendix C. ∎

Remark 7. Theorem 2 requires the existence of a *unique weight vector*, $\mathbf{w}^O$, whose mis-classification probability over the labeled instances generated by *all* concepts converges to 0. Theorem 3 requires the existence of *one weight vector for concept*, $\mathbf{w}_{i,c}^O$, whose mis-classification probability over the labeled instances generated by concept $\mathbf{S}_c^{(n)}$ converges to 0.

V. Extended PWM

So far we have considered an idealized setting in which all the learners always observe an instance at the beginning of the time instant (i.e., they are synchronous), and they always observe the corresponding label at the end of the time instant. In a distributed environment one cannot expect that these assumptions are always satisfied: sometimes the learners can be asynchronous, receive the label with delay, or not receive it at all. In this section we address these challenges proposing, for each of them, a modification to the basic PWM scheme introduced in Section III, and we extend Theorems 1 and 3 for each modified version of PWM.[6] At the end of this section we explicitly write the *extended PWM* algorithm that includes all the proposed modification to jointly deal with all the considered challenges.

A. Delayed and Out-Of-Order Labels

In some cases the true label corresponding to a time instant n is observed with delay. For example, in a distributed environment one learner can observe the label immediately, and communicate it to the other learners at a later stage. In this subsection we show that our algorithm can be modified in order to deal with this situation, with a price to pay in terms of increased memory.

We denote by $d_i^{(n)}$ the number of time slots after which learner i observes the n-th label. We assume that $d_i^{(n)}$ is not known a priori, but is bounded by a maximum delay $\overline{d}_i$, $\forall\, n$, which is known. Also, we allow for the possibility that the labels are received out of order (e.g., it is possible that learner i observes the label $y^{(n+1)}$ before the label $y^{(n)}$), but we assume that, when a label is received, the time instant it refers to is known.

PWM is modified as follow. Learner i maintains in memory all the local prediction vectors that refer to the not yet observed labels. As soon as learner i receives the label $y^{(m)}$, it computes the prediction $\hat{y}_i^{(m)} = \text{sgn}(\mathbf{w}_i^{(n)} \cdot \mathbf{s}^{(m)})$ which it would have made at time instant m with the current weight vector $\mathbf{w}_i^{(n)}$, and updates the weight vector according to

$$\mathbf{w}_i^{(n+1)} = \begin{cases} \mathbf{w}_i^{(n)} & \text{if } \hat{y}_i^{(m)} = y^{(m)} \\ \mathbf{w}_i^{(n)} + y^{(m)}\mathbf{s}^{(m)} & \text{otherwise} \end{cases} \quad (8)$$

This update rule is similar to (2), but now the updates may happen with delays. In particular, since different learners experience different delays, the weight vectors $\mathbf{w}_i^{(n)}$ and $\mathbf{w}_j^{(n)}$, $i \neq j$, follow different dynamics.

Theorem 4. *For every sequence of labeled instances* $\mathbf{D}_N$*,* $P^{PWM}(\mathbf{D}_N)$ *is bounded by* $\mathbf{B}(\mathbf{D}_N) + \frac{\sum_{i=1}^K \overline{d}_i}{NK}$.

Proof: See Appendix D. ∎

Remark 8. The term $\frac{\sum_{i=1}^K \overline{d}_i}{NK}$ can be interpreted as the maximum loss for the delayed labels.

Theorem 5. *If* $\mathbf{D}_N$*, for* $N \to +\infty$*, is generated by a finite number of learnable concepts and a finite number of concept drifts occurred, then* $\lim_{N\to+\infty} P^{PWM}(\mathbf{D}_N) \to 0$.

Proof: The proof applies the same is methodology as the proof of Theorem 4. The number of errors a generic learner i makes in each concepts can be divided into two contributions. The first contribution represents the number of errors i makes over the sequence of labeled instances it observes. Theorem 3 proves that such a term tends to 0. The second contribution represents the number of errors learner i makes over the label instances whose labels are not observed. Such a term is bounded by $\frac{\overline{d}_i}{N}$, that tends to 0. ∎

[6] Notice that Theorem 3 is a more general version of Theorem 2, and hence we do not need to extend also Theorem 2.

B. Missing Labels

In a distributed environment one cannot expect that all the learners always receive the label, in particular in those scenarios in which obtaining the information about the label may be both costly and time consuming. In this subsection we show that our scheme can be easily extended to deal with situations in which the true labels are only occasionally observed.

Let $g_i^{(n)} \triangleq 1$ if learner i observes the label $y^{(n)}$ at the end to time instant n, $g_i^{(n)} \triangleq 0$ otherwise. The following update rule represents the natural extension of (2) to deal with missing labels:

$$\mathbf{w}_i^{(n+1)} = \begin{cases} \mathbf{w}_i^{(n)} & \text{if } g_i^{(n)} = 0 \text{ or } \hat{y}_i^{(n)} = y^{(n)} \\ \mathbf{w}_i^{(n)} + y^{(n)}\mathbf{s}^{(n)} & \text{otherwise} \end{cases} \tag{9}$$

That is, learner i updates the weight vector $\mathbf{w}_i^{(n)}$ only when it observes the true label and it recognizes it made a prediction error. Notice that different learners observe different labels; therefore, the weight vectors $\mathbf{w}_i^{(n)}$ and $\mathbf{w}_j^{(n)}$, $i \neq j$, follow different dynamics.

Now we consider a simple model of missing labels and we derive the equivalent for the Theorems 1 and 3. We assume that $g_i^{(n)}$ is an independent and identically distributed (i.i.d.) process, $\forall\, i$, and denote by μ the probability that $g_i^{(n)} = 1$, $0 < \mu < 1$.[7] That is, at the end of a generic time instant n learner i observes the label with probability μ.

Denote by N_e^{PWM} the number of prediction errors observed by learner i, i.e., the number of times i observes the label and recognizes it made a prediction mistake. We define the function

$$\lambda\left(y, z\right) \triangleq \sqrt{\frac{1}{2z}\ln\frac{1}{y}} \tag{10}$$

Theorem 6. *Given the sequence of instances $\mathbf{D}_N$, for any level of confidence $\epsilon > 0$ such that $\lambda(\epsilon, \tilde{L}_i^{PWM}) \leq \mu$, with probability at least $1-\epsilon$ we have that $P^{PWM}(\mathbf{D}_N)$ is bounded by $\dfrac{\mathbf{B}(\mathbf{D}_N)}{\mu - \lambda\left(\epsilon, N_e^{PWM}\right)}$.*

Proof: See Appendix E. ∎

Remark 9. The denominator $\mu - \lambda(\epsilon, N_e^{PWM})$, which is lower than 1, can be interpreted as the maximum loss for the missing labels. Notice that, for any given level of confidence ϵ, the function $\lambda\left(\epsilon, N_e^{PWM}\right)$ is decreasing in the number of observed errors N_e^{PWM}, and tends to 0 if $N_e^{PWM} \to +\infty$. As a consequence, the bound defined by Theorem 6 tends to $\mathbf{B}(\mathbf{D}_N)$ divided by the probability to observe a label μ.

Theorem 7. *If $\mathbf{D}_N$, for $N \to +\infty$, is generated by a finite number of learnable concepts and a finite number of concept drifts occurred, then* $\lim_{N\to+\infty} P^{PWM}(\mathbf{D}_N) \to 0$.

Proof: If the number of errors $\overline{N}_{e,c}^{PWM}$ PWM makes in concept c is finite, then $\frac{\overline{N}_{e,c}^{PWM}}{N} \to 0$. If $\overline{N}_{e,c}^{PWM}$ is unbounded, then by a Chernoff-Hoeffding bound [32] we have $N_{e,c}^{PWM} = \mu \cdot \overline{N}_{e,c}^{PWM}$ with probability 1. Using the same arguments as in the proof of Theorem 3, we can say that $\frac{N_{e,c}^{PWM}}{N} \to 0$, hence $\frac{\overline{N}_{e,c}^{PWM}}{N} \to 0$. Therefore, $P^{PWM}(\mathbf{D}_N) = \frac{\sum_c \overline{N}_{e,c}^{PWM}}{N} \to 0$. ∎

[7] We can extend the analysis considering an observation probability μ_i that depends on the learner i. The results would be similar to those obtained with a unique μ, but the notations would be much messier.

C. Asynchronous Learners

Another important factor that may impact the performance of an online learning distributed system is the synchronization among the learners. So far we have assumed that each learner observes an instance in every time instant. However, in many practical scenarios different learners may capture instances in different time instants, and they can have different acquisition rates. In this subsection we extend our scheme to deal with this situation.

PWM is modified as follow. A learner does not send a local prediction when it does not observe the instance; however, it can still output a final prediction exploiting the local predictions received from the other learners. A generic learner i maintains two weight vectors: $\mathbf{w}_{i,s}^{(n)}$ and $\mathbf{w}_{i,a}^{(n)}$. At the time instants in which all the learners observe the instances (i.e., when the learners are synchronized), learner i aggregates all the local predictions using $\mathbf{w}_{i,s}^{(n)}$ and then, after having observed the label, updates $\mathbf{w}_{i,s}^{(n)}$ using (2). At the time instants in which some learners do not observe the instances (i.e., when the learners are not synchronized), learner i set to 0 the non received local predictions (i.e., it treats the learners that do not observe the instances as "abstainer"), aggregate the local predictions using $\mathbf{w}_{i,a}^{(n)}$, and then, after having observed the label, updates $\mathbf{w}_{i,a}^{(n)}$ using (2) (notice that the weights of the abstainers are not modified).

Given the sequence of labelled instances $\mathbf{D}_N$, we denote by M the number of times in which the instances are jointly observed by all the learners. We define the synchronization index $\alpha \triangleq \frac{N-M}{N}$. Notice that $0 \leq \alpha \leq 1$, the lower α the more synchronized the learners.

Algorithm Extended PWM

Initialization: $w_{ij,s} = w_{ij,a} = 0,\ \forall\, i, j \in \mathcal{K}$
For each learner i and time instant n
  If $s_j^{(n)}$ is received $\forall\, j \in \mathcal{K}$
    $\hat{y}_i^{(n)} \leftarrow \text{sgn}(\mathbf{w}_{i,s} \cdot \mathbf{s}^{(n)})$
  Else
    For each j such that $s_j^{(n)}$ is not received **do** $s_j^{(n)} \leftarrow 0$
    $\hat{y}_i^{(n)} \leftarrow \text{sgn}(\mathbf{w}_{i,a} \cdot \mathbf{s}^{(n)})$
  For each instant $m \leq n$ such that $y^{(m)}$ is observed
    If $s_j^{(m)} \neq 0\ \forall\, j$
      If $y^{(m)} \neq \text{sgn}(\mathbf{w}_{i,s} \cdot \mathbf{s}^{(m)})$ **do** $\mathbf{w}_{i,s} \leftarrow \mathbf{w}_{i,s} + y^{(m)}\mathbf{s}^{(m)}$
    Else
      If $y^{(m)} \neq \text{sgn}(\mathbf{w}_{i,a} \cdot \mathbf{s}^{(m)})$ **do** $\mathbf{w}_{i,a} \leftarrow \mathbf{w}_{i,a} + y^{(m)}\mathbf{s}^{(m)}$

Theorem 8. *Given the sequence of instances $\mathbf{D}_N$, $P^{PWM}(\mathbf{D}_N)$ is bounded by $\mathbf{B}(\mathbf{D}_N) + \alpha$.*

Proof: See Appendix F. ∎

Remark 10. The synchronization index α can be interpreted as the maximum loss for non synchronized learners. If the learners are always synchronous (i.e., $\alpha = 0$), Theorem 8 is equal to Theorem 1.

Theorem 9. *If $\mathbf{D}_N$, for $N \to +\infty$, is generated by a finite number of learnable concepts and a finite number of concept drifts occurred, then* $\lim_{N\to+\infty} P^{PWM}(\mathbf{D}_N) \leq \alpha$.

Proof: Using the notations and considerations of the proof of Theorem 8 we can state that, for a generic learner, the mis-classification probability in the sequence $\overline{\mathbf{D}}_M$ tends to 0 (because we can apply Theorem 3), whereas the mis-classification probability over the instances in which some learners do not observe the instances is bounded by α. ■

Remark 11. Different from Theorems 3, 5, and 7, in Theorem 9 the mis-classification probability does not tend to 0. In fact, the consequence of non synchronized learners is that a learner does not have, in all the time instances, the local predictions of all the other learners, and this lack of information may result in a mis-classification.

Remark 12. Theorem 9 can be used as a tool to design the acquisition protocol adopted by the learners. Indeed, if we know that the concepts are learnable and we have to satisfy a mis-classification probability constraint P_{mis}, we can choose the acquisition protocol such that the synchronization index α is equal to or lower than P_{mis}.

VI. Experiments

In this section we evaluate empirically the basic PWM algorithm and the extended PWM algorithm we proposed in Sections III and V, respectively. In order to compare PWM with other state-of-the-art ensemble learning techniques that do not deal with a distributed environment, in the first set of experiments (Subsection VI-A) all the learners observe the same data stream, but they are pre-trained on different data sets and hence their local predictions are in general different. In the second set of experiments (Subsection VI-B), different learners observe different data streams. In this case we compare PWM against a learner that predicts using only its local prediction, and analyze the impact on their performance of delayed labels, missing labels, and asynchronous learners.

A. Unique Data Stream

In this subsection we test PWM and other state-of-the-art solutions using real data sets that are generated from a unique data stream. First, we shortly describe the data sets, then we discuss the results.

1) Real Data Sets: We consider four data sets, well known in the data mining community, that refer to real-world problems. In particular, the first three data sets are widely used by the literature dealing with concept drift (which is the closest to our work), because they exhibit evident drifts.

R1: Network Intrusion. The network intrusion data set, used for the KDD Cup 1999 and available in the UCI archive [33], consists of a series of TCP connection records, labeled either as normal connections or as attacks. For a more detailed description of the data set we refer the reader to [4], that shows that the network intrusion data set contains non-stationary data. This data set is widely used in the stream mining literature dealing with concept drift [4], [13], [19], [34].

R2: Electricity Pricing. The electricity pricing data set holds information for the Australian New South Wales electricity market. The binary label (up or down) identifies the change of the price relative to a moving average of the last 24 hours. For a more detailed description of this dataset we refer the reader to [35]. An appealing property of this data set is that it contains drifts of different types, due to changes in consumption habits, the seasonability, and the expansion of the electricity market. This data set is widely used in the stream mining literature dealing with concept drift [16], [19], [35]–[40].

R3: Forest Cover Type. The forest cover type data set from UCI archive [33] contains cartographic variables of four wilderness areas of the Roosevelt National Forest in northern Colorado. Each instance refers to a 30×30 meter cell of one of these areas and is classified with one of seven possible classes of forest cover type. Our task is to predict if an instance belong to the first class or to the other classes. For a more detailed description of this dataset we refer the reader to [41]. The forest cover type data set contains drifts because data are collected in four different areas. This data set is widely used in the stream mining literature dealing with concept drift [13], [39], [42], [43].

R4: Credit Card Risk Assessment. In the credit card risk assessment data set, used for the PAKDD 2009 Data Mining Competition [44], each instance contains information about a client that accesses to credit for purchasing on a specific retail chain. The client is labeled as good if he was able to return the credit in time, as bad otherwise. For a more detailed description of this dataset we refer the reader to [44]. This data set does not contain drifts because the data were collected during one year with a stable inflation condition. In fact, to the best of our knowledge, the only work dealing with concept drift that uses this data set is [19].

2) Results: In this experiment we compare our scheme with other state-of-the-art ensemble learning algorithms. Table III lists the considered algorithms, the corresponding references, the parameters we adopted (that are equal to the ones used in the corresponding papers, except for the window size that is obtained following a tuning procedure), and their performance in the considered data sets.

We shortly describe the considering algorithms in the following. Average Majority (AM) [4] is a simple static scheme that gives the same weight to the all the local predictions. Adaboost (Ada) [7] trains a sequence of classifiers on increasingly more difficult examples and combines them using a weighted majority rule. First it weights the training data set to increase the importance of the examples in which the current ensemble fails, then it creates a new learner on the weighted training data set, and finally it assigns a weight to the learner based on its performance on the weighted data set. Fan's Online Adaboost (OnAda) and Wang's Online Adaboost (Wang) represent online versions of Adaboost. When a new chunk of data enters the system the current classifiers are

reweighed, a weighted training set is generated, and a new classifier (and its weight) is created on this data set. OnAda maintains the most recent classifiers, whereas Wang maintains classifiers with the highest prediction accuracy on the current chunk of data. Diversity for Dealing with Drifts (DDD) [19] is a scheme based on two online ensembles with different levels of diversity. The low diversity ensemble is used for system predictions, the high diversity ensemble is used to learn the new concept after a drift is detected. Weighted Majority (WM) [8] maintains a collection of given learners, predicts using a weighted majority rule, and updates the weights associated to the learners in a multiplicative manner, by decreasing the weights of the learners in the pool that disagree with the label whenever the ensemble makes a mistakes. In the WM variant proposed in [10] (Blum) the weights of the learners that agree with the label when the ensemble makes a mistakes are increased, and the weights of the learners that disagree with the label are decreased also when the ensemble predicts correctly. The WM variant proposed in [14] (TrackExp) tries to prevent the weights of the learners which performed poorly in the past from becoming too small with respect to the other learners, by adding a phase, after the multiplicative weight update, in which each learner shares a portion of its weight with the other learners

For each data set we consider a set of 8 learners and we use logistic regression classifiers for the learners' local predictions. Each local classifier is pre-trained using an individual training data set and kept fixed for the whole simulation (except for the OnAda, Wang, and DDD schemes, in which the base classifiers are retrained online). The training and testing procedures are as follow. From the whole data set we select 8 training data sets, each of them consisting of Z sequential records. Z is equal to $5,000$ for the data sets **R1** and **R3**, and $2,000$ for **R2** and **R4**. Then we take other sequential records ($20,000$ for **R1** and **R3**, and $8,000$ for **R2** and **R4**) to generate a set in which the local classifiers are tested, and the results are used to train offline Adaboost. Finally, we select other sequential records ($20,000$ for **R1** and **R3**, $21,000$ for **R2**, and $26,000$ for **R4**) to generate the testing set that is used to run the simulations and test all the considered schemes.

Table III reports the final mis-classification probability in percentages (i.e., multiplied by 100) obtained for each data set for the considered schemes. For the first three data sets, which exhibit concepts drifts, the schemes that update their models after each instance (DDD, WM, Blum, TrackExp, and PWM) outperform the static schemes (AM and Ada) and the schemes that update their model after a chunk of instances enters the system (OnAda and Wang). This result shows that the static schemes are not able to adapt to changes in concept, and the schemes that need to wait for a chunk of data adapt slowly because 1) they have to wait for the last instance of the chuck before updating the model, and 2) a chuck of data can contain instances belonging to different concepts, hence the model built on it can be inaccurate to predict the current concept.

Importantly, in the first three data sets PWM outperforms all the other schemes, whereas the second best scheme is WM. The gain of PWM (in terms of reduction of the mis-classification probability) with respect to WM is about 34% for **R1**, 38% for **R2**, and 71% for **R3**. We remark that the main differences among our scheme and WM are 1) the weights update rule (additive vs. multiplicative), and 2) the weight $w_{i0}^{(n)}$ associated to the virtual learner that always sends the local prediction 1. To investigate the real reason of the gain of PWM we tested also a version of PWM in which $w_{i0}^{(n)} = 0$, $\forall\, n$, obtaining the following percentage of mis-classifications in the first three data sets: 0.23, 14.4 and 4.1. Hence, the weight $w_{i0}^{(n)}$ can slightly help to increase the accuracy of the distributed system, but the main reason why PWM outperforms WM in these data sets is the update rule.

Differently from the first three data sets, in **R4**, the data set that does not contain drifts, Ada, OnAda, and Wang outperform the other schemes. In fact, they exploit many stored labeled instances to build their models, and this results in more accurate models when the data are generated from a static distribution.

B. Different Data Streams

In this subsection we evaluate PWM using synthetic data sets in which different learners observe different data streams, and analyze the impact of delayed labels, missing labels, and asynchronous learners. First, we shortly describe the data sets, then we discuss the results.

1) Synthetic Data Sets: We consider three synthetic data sets to carry on different experiments. The first data set represents a separating hyperplane that rotates slowly, we use it to simulate gradual drifts [5], [19], [37]. Similar data sets are widely adopted in the stream mining literature dealing with concept drift [4], [13], [19], [31]. In the third data set, similarly to [45], each learner observes a local event that is embedded in a zero-mean Gaussian noise. Concept drifts occur because the accuracies of the observations evolve following Markov processes. The third data is a simple Gaussian distributed data set in which the concept is stable. We use this data set because we can analytically compute the optimal mis-classification probability $P^O(\mathbf{D}_N)$ and investigate how strict the bound $\mathbf{B}(\mathbf{D}_N)$ is.

S1: Rotating Hyperplane. Each learner i observes a 3-dimensional instance $x_i^{(n)} = (x_{i,1}^{(n)}, x_{i,2}^{(n)}, x_{i,3}^{(n)})$ that is uniformly distributed in $[-1\ 1]^3$, and is independent from $x_i^{(m)}$, $n \neq m$, and from $x_j^{(m)}$, $i \neq j$. The label is a deterministic function of the instances observed by the first $\underline{K} < K$ learners (the other $K - \underline{K}$ learners observe irrelevant instances). Specifically, $y^{(n)} = 1$ if $\sum_{i=1}^{\underline{K}} \sum_{\ell=1}^{3} \theta_{i,\ell}^{(n)} x_{i,\ell}^{(n)} \geq 0$, $y^{(n)} = 0$ otherwise. The parameters $\theta_{i,\ell}^{(n)}$ are unknown and time-varying. As in [31], each $\theta_{i,\ell}^{(1)}$ is independently generated according to a zero-mean unit-variance Gaussian distribution $\mathcal{N}(0,1)$, and $\theta_{i,\ell}^{(n)} = \theta_{i,\ell}^{(n-1)} + \delta_{i,\ell}^{(n)}$ where $\delta_{i,\ell}^{(n)} \sim \mathcal{N}(0, 0.1)$.

S2: Distributed Event Detection. Each learner i monitors the occurrence of a particular local event. Let $e_i^{(n)} \triangleq 1$ if the local event monitored by learner i occurs at time instant n, $e_i^{(n)} \triangleq -1$ otherwise. $e_i^{(n)}$ is an i.i.d. process, the probability that $e_i^{(n)} = 1$ is 0.05, $\forall\, i, n$. The observation of learner i is $\mathbf{x}_i^{(n)} = e_i^{(n)} + \beta_i^{(n)}$, where $\beta_i^{(n)}$ is an i.i.d. zero-mean Gaussian

Abbreviation	Name of the Scheme	Reference	Parameters	Performance			
				R1	R2	R3	R4
AM	Average Majority	[4]	–	3.07	41.8	29.5	34.1
Ada	Adaboost	[7]	–	5.25	41.1	57.5	**19.7**
OnAda	Fan's Online Adaboost	[11]	Window size: $W = 100$	2.25	41.9	39.3	19.8
Wang	Wang's Online Adaboost	[12]	Window size: $W = 100$	1.73	40.5	32.7	19.8
DDD	Diversity for Dealing with Drifts	[19]	Diversity parameters: $\lambda_l = 1$, $\lambda_h = 0.1$	0.72	39.7	24.6	20.0
WM	Weighted Majority algorithm	[8]	Multiplicative parameter: $\beta = 0.5$	0.29	22.9	14.1	67.4
Blum	Blum's variant of WM	[10]	Multiplicative parameters: $\beta = 0.5$, $\gamma = 1.5$	1.64	37.3	22.6	68.1
TrackExp	Herbster's variant of WM	[14]	Mult. and sharing parameters: $\beta = 0.5$, $\alpha = 0.25$	0.52	23.1	14.8	22.0
PWM	**Perceptron Weighted Majority**	our work	–	**0.19**	**14.3**	**4.1**	**31.5**

TABLE III: The considered schemes, their parameters, and their percentages of mis-classifications in the data sets **R1**-**R4**

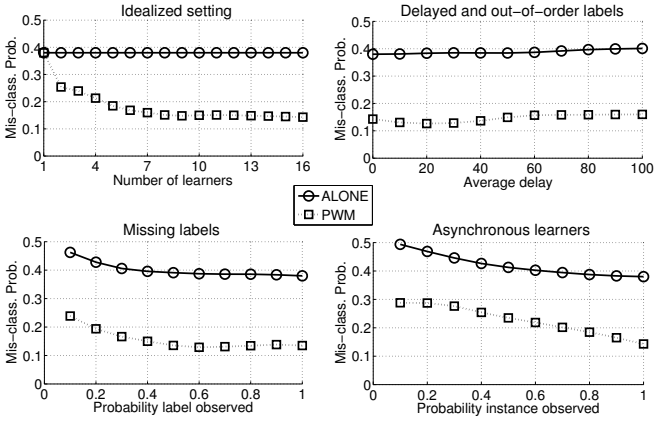

Fig. 2: mis-classification probability of learner 1 if it predicts alone and if it uses PWM, for the data set **S1**

process. To simulate concept drifts, we assume that a source can be in two different states: good or bad. In the good state $\beta_i^{(n)} \sim \mathcal{N}(0, 0.5)$, in the bad state $\beta_i^{(n)} \sim \mathcal{N}(0, 1)$. The state of the source evolves as a Markov process with a probability 0.01 to transit from one state to the other.

S3: Gaussian Distribution. The labels are generated according to a Bernoulli process with parameter 0.5, and the instance $\mathbf{x}^{(n)} = (x_1^{(n)}, \ldots, x_K^{(n)})$ is generated according to a K-dimensional Gaussian distribution $\mathbf{x}^{(n)} \sim \mathcal{N}\left(y^{(n)} \cdot \mu, \Sigma\right)$, where Σ is the identity matrix. That is, if the label is 1 (-1) each component $x_i^{(n)}$ is independently generated according to a Gaussian distribution with mean μ $(-\mu)$ and unitary variance. A generic learner i observes only the component $x_i^{(n)}$ of the whole instance $\mathbf{x}^{(n)}$.

2) Results: In the first set of experiments we adopt the synthetic data set **S1** to evaluate the mis-prediction probability of a generic learner, which we refer to as learner 1, when it predicts by its own (ALONE), and when it adopts PWM. We consider a set of $K = 16$ learners, in which the last 8 learners observe irrelevant instances. For each simulation we generate a data set of $1,000$ instances. We use non pre-trained online logistic regression classifiers for the learners' local predictions. We run $1,000$ simulations and average the results. The final results are reported in the four sub-figures of Fig. 2, and are discussed in the following.

The top-left sub-figure shows how the mis-classification probability of learner 1 varies, in the idealized setting (i.e., without the issues described in Section V), with respect to the the number of learners that PWM aggregates. If there is only one learner, ALONE and PWM are equivalent, but the gap between the performance obtainable by ALONE and the performance achievable by PWM increases as the number of learners that PWM aggregates increases. In particular, if the local predictions of all the learners are aggregated, the mis-classification probability of PWM is less than half the mis-classification probability of ALONE. Notice that the performance of PWM remains constant from 8 to 16 learners, and this is a positive result because the last 8 learners observe irrelevant instances. PWM automatically gives them a low weight such that their (noisy) local predictions do not influence the final prediction. In fact, the simulation for $K = 16$ learners shows that the average absolute weight of the first 8 learners is about twice the average absolute weight of the last 8 learners. In all the following experiments we consider $K = 16$ learners.

Now we assume that learner 1 observes the labels after some time instants, and each delay is uniformly distributed in $[0 \;\; D]$. The top-right sub-figure shows how the mis-classification probability varies with respect to the average delay $\frac{D}{2}$. We can see that the delay does not affect considerably the performance, in fact both mis-classification probabilities slightly increases if the delay increases and the gap between them remain constant.

In the next experiment we analyze the impact of missing labels on the performance of learner 1. The bottom-left sub-figure shows how the mis-classification probability varies with respect to the probability that learner 1 observes a label. Even when the probability of observing a label is 0.1, the mis-classification probability of PWM is about half the mis-classification probability of ALONE. This gain is possible because learner 1, adopting PWM, automatically exploits the fact that the other learners are learning.

Similar considerations are valid when learner 1 observes an instance with a certain probability (see the bottom-right sub-figure), which can be interpreted as the reciprocal of the arrival rate. The impact on the mis-classification probabilities of missing instances is stronger (i.e., the mis-classification probabilities are higher) than the impact of missing labels. In fact, when instances are not observed, not only learner 1 does not update the weight vector, it also waits more time between two consecutive predictions, hence the concept between two consecutive predictions can change consistently. When the probability of observing an instance is 0.1, the gain of PWM, with respect to ALONE, is about 40%.

In the second set of experiments we use a similar set-up as in the first set of experiments, but we adopt the synthetic

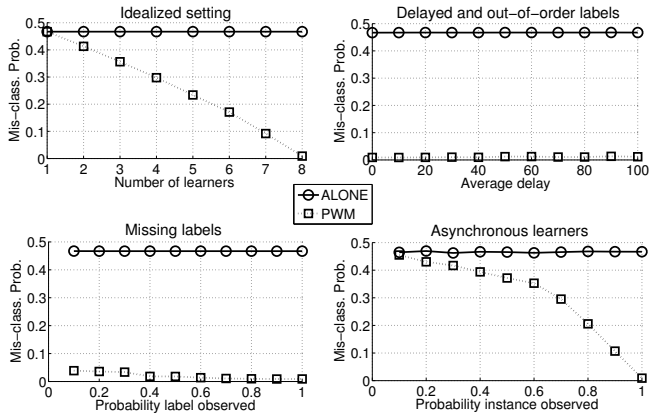

Fig. 3: mis-classification probability of learner 1 if it predicts alone and if it uses PWM, for the data set **S2**

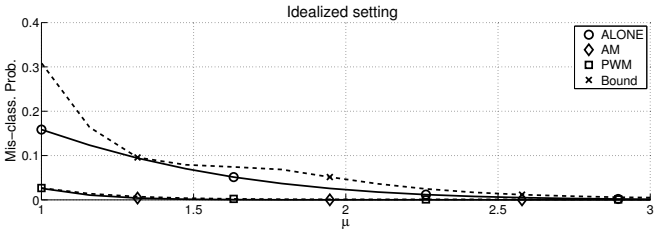

Fig. 4: The bound $\mathbf{B}(\mathbf{D}_N)$ and the mis-classification probability of learner 1 if 1) it predicts by its own, 2) it uses AM, 3) it uses PWM, for the data set **S3**

data set **S2**. We consider a set of $K = 8$ learners and for each simulation we generate $10,000$ instances. Each learner uses a non pre-trained online logistic regression classifiers to learn the best threshold to adopt to classify the local event. We run 100 simulations and average the results. The final results are reported in the four sub-figures of Fig. 3, and are briefly discussed in the following. The top-left sub-figure shows that the mis-classification probability of PWM decreases linearly in the number of learners until the local prediction of all learners are aggregated, in this case the mis-classification probability of PWM is about 0.01, whereas the mis-classification probability of ALONE is about 0.47. As in the first set of experiments, the delay does not affect the performance of the two schemes, and the performance of PWM is much better than the performance of ALONE even when the probability of observing the label is very low. Differently from the first set of experiments, with the data set **S2** the performance of PWM is strongly affected by the synchronicity of the learners, and when the learners observe few instances the mis-classification of PWM becomes close to the mis-classification of ALONE.

In the last experiment we adopt the data set **S3** to investigate how strict the bound $\mathbf{B}(\mathbf{D}_N)$ is. For each simulation we consider $K = 8$ learners and generate a data set of $1,000$ instances. We assume that the local prediction of learner i is -1 if its observation $x_i^{(n)}$ is negative, 1 otherwise. It is possible to show that, given the structure of the problem, this represents the most accurate policy for the local prediction, and the best possible aggregation rule is the average majority (AM). We run $10,000$ simulations and average the results.

Fig. 4 shows the bound $\mathbf{B}(\mathbf{D}_N)$ and the mis-classification probability of learner 1 if 1) it predicts by its own (ALONE), 2) it uses AM, and 3) it uses PWM, varying the parameter μ. If μ is low the instances corresponding to negative and positive labels are similar, hence it is more difficult to predict correctly the labels. Fig. 4 shows that, in this case, the mis-classification probability of PWM is much lower than the bound, and it is very close to the mis-classification probability of AM, that is the best aggregation rule in this scenario. With the increase of μ, the mis-classification probabilities of all the schemes decrease, and the bound become stricter to the real performance of PWM.

Notice that the curve representing the bound has a cusp at about $\mu = 1.75$. In fact, before this value $\mathbf{B}_1(\mathbf{D}_N)$ is stricter than $\mathbf{B}_2(\mathbf{D}_N)$, whereas for $\mu > 1.75$ $\mathbf{B}_2(\mathbf{D}_N)$ is lower than $\mathbf{B}_1(\mathbf{D}_N)$. This agrees with Remark 6: when μ is low the local classifiers are inaccurate (see ALONE), but their ensemble can be very accurate (see AM), and $\mathbf{B}_1(\mathbf{D}_N)$ is stricter than $\mathbf{B}_2(\mathbf{D}_N)$; whereas, when μ is high the local classifiers are very accurate and $\mathbf{B}_2(\mathbf{D}_N)$ becomes stricter than $\mathbf{B}_1(\mathbf{D}_N)$.

VII. Conclusion

We proposed a distributed online ensemble learning algorithm to classify data captured from distributed and dynamic data sources. Our approach requires limited communication, computational, energy, and memory requirements. We rigorously determined a bound for the worst-case mis-classification probability of our algorithm, which depends on the mis-classification probabilities of the best static aggregation rule and of the best local classifier. Importantly, this bound tends asymptotically to 0 if the mis-classification probability of the best static aggregation rule tends to 0. We extended our algorithm and the corresponding bounds such that they can address challenges specific to the distributed implementation, e.g., learners can be asynchronous, receive the label with delay, or not receive it at all. Simulation results show the efficacy of the proposed approach. When applied to real data sets widely used by the literature dealing with dynamic data streams and concept drift, our scheme exhibits performance gains ranging from 34% to 71% with respect to state-of-the-art solutions.

Appendix A
Proof of Lemma 1

Proof: Since $P^{PWM}(\mathbf{D}_N) = P_i^{PWM}(\mathbf{D}_N)$, $\forall\, i$, we can derive the bound with respect to the mis-classification probability $P_i^{PWM}(\mathbf{D}_N)$ of a generic learner i.

The proof departs from [46, Theorem 2], which states that, for a general Perceptron algorithm (i.e., $s_i^{(n)}$ can belong to whatever subset of $\Re$), if $\|\mathbf{s}^{(n)}\| \leq R, \forall\, n$, then for every $\gamma > 0$ and vector $\mathbf{u} \in \Re^{K+1}$, $\|\mathbf{u}\| = 1$, the number of prediction errors $N_e^{PWM}(\mathbf{D}_N)$ of the online Perceptron algorithm on the sequence $\mathbf{D}_N$ is bounded by

$$N_e^{PWM}(\mathbf{D}_N) \leq \left(\frac{R + \sqrt{\gamma D}}{\gamma}\right)^2 \tag{11}$$

where $D = \sum_{n=1}^{m} d_n$, $d_n = \max\left(0, \gamma - y^n(\mathbf{u}\cdot\mathbf{s}^{(n)})\right)$. Starting from this bound, we exploit the structure of our problem (i.e., $s_i^{(n)} \in \{-1, 1\}$) to derive the bound $\mathbf{B}_1(\mathbf{D}_N)$.

Since in our case $\|\mathbf{s}^{(n)}\| = \sqrt{K+1}$, we can consider $R = \sqrt{K+1}$. Notice that the last K elements of $\mathbf{s}^{(n)}$, i.e., the local predictions, represent a particular vertex of an hypercube in $\Re^K$, and the optimal a posteriori weight vector $\mathbf{w}^O$ represents an hyperplane in $\Re^K$ which separates the 2^K vertexes of the hypercube in two subsets $\mathcal{V}_{-1}$ and $\mathcal{V}_1$, representing the vertexes resulting in a negative and positive prediction respectively. Now we consider two scenarios: (1) either $\mathcal{V}_{-1}$ or $\mathcal{V}_1$ are empty; (2) both $\mathcal{V}_{-1}$ and $\mathcal{V}_1$ are not empty.

We consider the first scenario. In this situation the optimal policy $\mathbf{w}^O$ predicts always -1 or 1, independently of the local predictions (this case is not very interesting in practice, but we analyze it for completeness). The geometric interpretation is that the separating hyperplane does not intersect the hypercube. Let γ be the distance between the separating hyperplane and the closest vertex of the hypercube, and $\mathbf{u} = \frac{\mathbf{w}^O}{\|\mathbf{w}^O\|}$. If $\mathbf{w}^O$ predicts correctly the n-th instance, then $y^n(\mathbf{u}\cdot\mathbf{s}^{(n)}) \geq \gamma$, hence $d_n = 0$. If $\mathbf{w}^O$ makes a mistakes in the n-th instance, then $y^n(\mathbf{u}\cdot\mathbf{s}^{(n)}) \leq -\gamma$ and $y^n(\mathbf{u}\cdot\mathbf{s}^{(n)}) \geq -\gamma - 2\sqrt{K}$ (because the closest vertex is $2\sqrt{K}$ distant from the farthest one), therefore $d_n \leq 2\gamma + 2\sqrt{K}$. Hence, we obtain $D \leq 2N^O(\mathbf{D}_N)\left(\gamma + \sqrt{K}\right)$, where $N_e^O(\mathbf{D}_N)$ is the number of mistakes made adopting $\mathbf{w}^O$, and

$$\left(\frac{R+\sqrt{\gamma D}}{\gamma}\right)^2 \leq \left(\frac{\sqrt{K+1} + \sqrt{2N_e^O(\mathbf{D}_N)\gamma\left(\gamma+\sqrt{K}\right)}}{\gamma}\right)^2 \tag{12}$$

The right side of the above inequality is decreasing in γ. Since we can consider other optimal a posteriori weight vectors $\mathbf{w}^O$ and since there is no constraint on how far the separating hyperplane could be with respect to the hypercube, taking the limit for $\gamma \to +\infty$ and dividing everything by N we finally obtain

$$P_i^{PWM}(\mathbf{D}_N) \leq \frac{1}{N}\lim_{\gamma\to+\infty}\left(\frac{R+\sqrt{\gamma D}}{\gamma}\right)^2 \leq 2P^O(\mathbf{D}_N)\ , \tag{13}$$

which is compatible with $P^{PWM}(\mathbf{D}_N) \leq \mathbf{B}_1(\mathbf{D}_N)$.

Now we consider the second scenario. In this case the separating hyperplane intersects the hypercube. Among all the optimal a posteriori weight vectors $\mathbf{w}^O$, we want to consider the one which separates the vertexes in $\mathcal{V}_{-1}$ from the vertexes in $\mathcal{V}_1$ with the largest margin possible in order to find the strictest bound defined by (11). However, since the bound must be valid for every linearly separable sets of vertexes $\mathcal{V}_{-1}$ and $\mathcal{V}_1$, we have to consider the worst case possible (in term of separating margin) with respect to the sets $\mathcal{V}_{-1}$ and $\mathcal{V}_1$. It is easy to see that the worst case corresponds to the situation in which one vertex must be separated by all the vertexes it is connected with through an edge, and the best separating hyperplane must be equidistant from all these vertexes. Since the distances between the vertexes and the separating hyperplane are invariant with respect to translation and rotation of both the hypercube and the hyperplane, we consider the hypercube with vertexes $\mathbf{v} = (v_1, \ldots, v_K)$, $v_i \in 0, 2$, and we want to find the hyperplane defined by the parameters $(a_0, \ldots, a_K)$ which separates with the largest margin the vertex $\mathbf{v}^{(0)} = (0, \ldots, 0)$ from the vertexes $\mathbf{v}^{(i)} = (0, \ldots, 0, 2, 0, \ldots, 0)$ having 2 in position i, $i = 1, \ldots, K$. Imposing that the signed distance between $\mathbf{v}^{(0)}$ and the separating hyperplane is the opposite of the signed distance between $\mathbf{v}^{(i)}$ and the separating hyperplane, we obtain

$$\frac{a_0 + v_1^{(0)}a_1 + \ldots + v_K^{(0)}a_K}{\sqrt{\sum_{i+1}^K a_i^2}} = -\frac{a_0 + v_1^{(i)}a_1 + \ldots + v_K^{(i)}a_K}{\sqrt{\sum_{i+1}^K a_i^2}}$$
$$\to a_0 = -a_i \tag{14}$$

Repeating the same procedure for every vertex $\mathbf{v}^{(i)}$, $i = 1, \ldots, K$, we obtain that the best separating hyperplane must satisfy $a_0 = -a_i$, $\forall\, i$, hence the distance between it and $\mathbf{v}^{(0)}$ is $\frac{|a_0|}{\sqrt{K a_0^2}} = \frac{1}{\sqrt{K}}$. This means that we are always able to find an optimal a posterior weight vector $\mathbf{w}^O$ which separates the local prediction vectors which a margin of at least $\gamma = \frac{1}{\sqrt{K}}$. Notice also that the maximum distance between the hyperplane defined by $\mathbf{w}^O$ and a vertex in the hypercube is $2\sqrt{K} - \frac{1}{\sqrt{K}}$. Define $\gamma = \frac{1}{\sqrt{K}}$ and $\mathbf{u} = \frac{\mathbf{w}^O}{\|\mathbf{w}^O\|}$. If $\mathbf{w}^O$ predicts correctly the n-th instance, then $y^n(\mathbf{u}\cdot\mathbf{s}^{(n)}) \geq \gamma$, hence $d_n = 0$. If $\mathbf{w}^O$ makes a mistakes in the n-th instance, then $y^n(\mathbf{u}\cdot\mathbf{s}^{(n)}) \leq -\gamma$ and $y^n(\mathbf{u}\cdot\mathbf{s}^{(n)}) \geq -2\sqrt{K} + \frac{1}{\sqrt{K}}$, therefore $d_n \leq 2\sqrt{K}$. Finally, we obtain $D \leq 2\sqrt{K}N_e^O(\mathbf{D}_N)$ and

$$P_i^{PWM}(\mathbf{D}_N) \leq \frac{1}{N}\left(\frac{R+\sqrt{\gamma D}}{\gamma}\right)^2 \leq 2KP^O(\mathbf{D}_N) + \frac{K(K+1)}{N}$$

■

APPENDIX B
PROOF OF LEMMA 2

Proof: Since $P^{PWM}(\mathbf{D}_N) = P_i^{PWM}(\mathbf{D}_N)$, $\forall\, i$, we can derive the bound with respect to the mis-classification probability $P_i^{PWM}(\mathbf{D}_N)$ of a generic learner i.

PWM updates its weight vector only on those instances in which it makes a mistake. We denote with the superscript n the parameters of the system during the n-th mistake. We have

$$\|\mathbf{w}_i^{n+1}\|^2 = \|\mathbf{w}_i^n + y^n\mathbf{s}^n\|^2 = \|\mathbf{w}_i^n\|^2 + \|\mathbf{s}^n\|^2 + 2y^n\mathbf{w}_i^n\cdot\mathbf{s}^n$$
$$\leq \|\mathbf{w}_i^n\|^2 + \|\mathbf{s}^n\|^2 = \|\mathbf{w}_i^n\|^2 + K + 1 \tag{15}$$

where the first inequality is valid because the system makes an error, hence $y^n\mathbf{w}_i^n\cdot\mathbf{s}^n \leq 0$. By applying a straightforward inductive argument we obtain $\|\mathbf{w}_i^{n+1}\|^2 \leq n(K+1)$.

To simplify the notations, we denote by z the number of errors made by the most accurate classifier and by v^* the number of most accurate classifiers, i.e., $z = NP^*(\mathbf{D}_N)$ and $v^* = v^*(\mathbf{D}_N)$ After the system makes n errors, the weight associated to the most accurate classifiers is at least $n - 2z$ (it increases by one unit at least $n - z$ times, and decreases by one unit at most z times). Hence, $\|\mathbf{w}_i^{n+1}\|^2 \geq v^*(n-2z)^2$.

Combining the two above inequalities we obtain $v^*n^2 - (4v^*z + K + 1)n + 4v^*z^2 \leq 0$, which implies

$$n \leq 2z + \frac{K+1}{2v^*} + \sqrt{\left(\frac{K+1}{2v^*}\right)^2 + \frac{2(K+1)x}{v^*}} \quad (16)$$

Dividing by N we obtain $P^{PWM}(\mathbf{D}_N) \leq \mathbf{B}_2(\mathbf{D}_N)$. ■

APPENDIX C
PROOF OF THEOREM 3

Proof: We denote by $\mathbf{w}_{i,c}^0$ the weight vector of learner i at the beginning of a generic concept $\mathbf{S}_c^{(n)}$, and use the superscript n to denote the parameters of the system during the n-th mistake inside the concept $\mathbf{S}_c^{(n)}$. We have

$$\|\mathbf{w}_{i,c}^{n+1}\|^2 = \|\mathbf{w}_{i,c}^n + y^n\mathbf{m}^n\|^2 = \|\mathbf{w}_{i,c}^n\|^2 + \|\mathbf{m}^n\|^2 + 2y^n\mathbf{w}_{i,c}^n \cdot \mathbf{m}^n \leq \|\mathbf{w}_{i,c}^n\|^2 + \|\mathbf{m}^n\|^2 = \|\mathbf{w}_{i,c}^n\|^2 + K + 1$$

where the first inequality is valid because the system makes an error, hence $y^n\mathbf{w}_{i,c}^n \cdot \mathbf{m}^n \leq 0$. By applying a straightforward inductive argument we obtain $\|\mathbf{w}_{i,c}^{n+1}\|^2 \leq \|\mathbf{w}_{i,c}^0\|^2 + n(K+1)$.

Since the concept $\mathbf{S}_c^{(n)}$ is learnable, there exists a unit vector $\mathbf{u} \in \Re^{K+1}$, $\|\mathbf{u}\| = 1$, and $\gamma > 0$ such that $y^n\mathbf{u} \cdot \mathbf{m}^n \geq \gamma$, for every labeled instances of the current concept. Hence, we have $\mathbf{w}_{i,c}^{n+1} \cdot \mathbf{u} = \mathbf{w}_{i,c}^n \cdot \mathbf{u} + y^n\mathbf{m}^n \cdot \mathbf{u} \geq \mathbf{w}_{i,c}^n \cdot \mathbf{u} + \gamma$, from which we obtain $\mathbf{w}_{i,c}^{n+1} \cdot \mathbf{u} \geq \mathbf{w}_{i,c}^0 \cdot \mathbf{u} + n\gamma$.

Combining the two inequalities we have derived we get

$$\sqrt{\|\mathbf{w}_{i,c}^0\|^2 + n(K+1)} \geq \|\mathbf{w}_{i,c}^{n+1}\| \geq \mathbf{w}_{i,c}^{n+1} \cdot \mathbf{u} \geq \geq \mathbf{w}_{i,c}^0 \cdot \mathbf{u} + n\gamma \geq -\|\mathbf{w}_{i,c}^0\| + n\gamma \quad (17)$$

For $n \geq \frac{\|\mathbf{w}_{i,c}^0\|}{\gamma}$ (if this is not valid then n is bounded by $\frac{\|\mathbf{w}_{i,c}^0\|}{\gamma}$ which stricter than the following bound) we obtain

$$\|\mathbf{w}_{i,c}^0\|^2 + n(K+1) \geq \|\mathbf{w}_{i,c}^0\|^2 - 2\|\mathbf{w}_{i,c}^0\|n\gamma + \gamma^2n^2$$
$$\rightarrow \quad n \leq \frac{2\|\mathbf{w}_{i,c}^0\|\gamma + K + 1}{\gamma^2} \quad (18)$$

As shown in the proof of Lemma 1, we can take $\gamma = \frac{1}{\sqrt{K}}$. Hence, we obtain

$$n \leq 2\sqrt{K}\|\mathbf{w}_{i,c}^0\| + K(K+1) \quad (19)$$

In the first concept $\mathbf{w}_{i,1}^0$ is initialized to 0, thus the number of errors at the end of the first concept is upper-bounded by a bounded function of K, and in turns also the norm of the weight vector $\mathbf{w}_{i,2}$ at the beginning of the second concept is bounded by a function of K. Exploiting (19) and using an inductive argument we can conclude that the number of errors at the end of each concept is upper bounded by a bounded function K. Since there are a finite number of concepts, the total number of errors is upper-bounded by a bounded function of K. Finally, dividing it by N, we obtain $P_i^{PWM}(\mathbf{D}_N) \rightarrow 0$, that implies $P^{PWM}(\mathbf{D}_N) \rightarrow 0$. ■

APPENDIX D
PROOF OF THEOREM 4

Proof: Denote by $\overline{y}^{(n)}$ the n-th label observed by learner i, and by $\overline{\mathbf{x}}^{(n)}$ the corresponding instance. Let $\overline{\mathbf{D}}_M = \left((\overline{\mathbf{x}}^{(1)}, \overline{y}^{(1)}), \ldots, (\overline{\mathbf{x}}_\ell^{(1)}, \overline{y}^{(1)}), \ldots, (\overline{\mathbf{x}}_\ell^{(M)}, \overline{y}^{(M)})\right)$ the sequence of the M labeled instances observed by learner i until time instant N. Notice that $N - \overline{d}_i \leq M \leq N$. We can applied Theorem 1 to $\overline{\mathbf{D}}_M$ (the bound of Theorem 1 is valid also for the mis-classification probability of a generic learner i), obtaining $P_i^{PWM}(\overline{\mathbf{D}}_M) \leq \mathbf{B}(\overline{\mathbf{D}}_M)$. The sequence $\overline{\mathbf{D}}_M$ is a permutation of a subset of $\mathbf{D}_N$, hence the number of errors made by the optimal aggregation rule and by the best classifier in $\overline{\mathbf{D}}_M$ cannot by higher than those made in $\mathbf{D}_N$, i.e., $P^O(\overline{\mathbf{D}}_M)M \leq P^O(\mathbf{D}_N)N$ and $P^*(\overline{\mathbf{D}}_M)M \leq P^*(\mathbf{D}_N)N$. The number of errors learner i makes over $\mathbf{D}_N$ adopting the PWM scheme are equal to the number of errors learner i makes over $\overline{\mathbf{D}}_M$ plus the number of errors it makes over the label instances whose labels are not observed. Since the last term is bounded by $\overline{d}_i$, we obtain

$$P_i^{PWM}(\mathbf{D}_N) \leq \frac{1}{N}P_i^{PWM}(\overline{\mathbf{D}}_M)M + \frac{\overline{d}_i}{N} \leq \mathbf{B}(\mathbf{D}_N) + \frac{\overline{d}_i}{K}$$

By applying (6) we we conclude the proof. ■

APPENDIX E
PROOF OF THEOREM 6

Proof: Inside this proof, to simplify the notations, we denote by $n = P^{PWM}(\mathbf{D}_N)N$ the number of errors made by PWM and by $t = N_e^{PWM}$ the number of observed errors. The number of observed errors t is a binomial with parameters n and μ. Exploiting a Chernoff-Hoeffding inequality [32] we can write $P\left[t \leq n(\mu - \gamma)\right] \leq e^{-2\gamma^2 n} \leq e^{-2\gamma^2 t}$, which implies $P\left[n \geq \frac{t}{\mu - \gamma}\right] \leq e^{-2\gamma^2 t}$

Since PWM updates its weight vector only on those instances on which an error is observed, the bound $\mathbf{B}(\mathbf{D}_N)$ for non perfectly observable labels must be interpreted as bounds for the number of observed errors, i.e., $\frac{t}{N} \leq \mathbf{B}(\mathbf{D}_N)$. The two inequalities above, with the change of variable $e^{-2\gamma^2 t} = \epsilon$, imply $P^{PWM}(\mathbf{D}_N) \leq \frac{\mathbf{B}(\mathbf{D}_N)}{\mu - \lambda\left(\epsilon, N_e^{PWM}\right)}$ with probability at least $1 - \epsilon$. ■

APPENDIX F
PROOF OF THEOREM 8

Proof: Denote by $\overline{\mathbf{D}}_M$ the subset of labeled instances of $\mathbf{D}_N$ in which all the learners are synchronized. The number of errors learner i makes over $\mathbf{D}_N$ adopting the PWM scheme is equal to the number of errors learner i makes over $\overline{\mathbf{D}}_M$, plus the number of errors it makes over the $N - M$ label instances in which some learners do not observe the instances. The weight vector $\mathbf{w}_{i,s}^{(n)}$ is used only to predict the instances in $\overline{\mathbf{D}}_M$, and is updated only in these instances. Therefore, we can applied Theorem 1 to $\overline{\mathbf{D}}_M$ (the bound of Theorem 1 is valid also for the mis-classification probability of a generic learner i), obtaining $P_i^{PWM}(\overline{\mathbf{D}}_M) \leq \mathbf{B}(\overline{\mathbf{D}}_M)$. The sequence $\overline{\mathbf{D}}_M$ is a subset of $\mathbf{D}_N$, hence the number of errors made by

the optimal aggregation rule and by the best classifier in $\overline{\mathbf{D}}_M$ cannot by higher than those made in $\mathbf{D}_N$, i.e., $P^O(\overline{\mathbf{D}}_M)M \leq P^O(\mathbf{D}_N)N$ and $P^*(\overline{\mathbf{D}}_M)M \leq P^*(\mathbf{D}_N)N$. Therefore the number of errors made by i in $\overline{\mathbf{D}}_M$ is $P_i^{PWM}(\overline{\mathbf{D}}_M)M \leq \mathbf{B}(\overline{\mathbf{D}}_M)M \leq \mathbf{B}(\mathbf{D}_N)N$, which implies that the contribution of i to $P^{PWM}(\mathbf{D}_N)$ is at most $\mathbf{B}(\mathbf{D}_N) + \alpha$. The proof is concluded by summing the contributions of all learners and dividing the result by K. ■

REFERENCES

[1] J. Albusac, J. J. Castro-Schez, L. M. Lopez-Lopez, D. Vallejo, and L. Jimenez-Linares, "A supervised learning approach to automate the acquisition of knowledge in surveillance systems," *Signal Process.*, vol. 89, no. 12, pp. 2400–2414, Dec. 2009.

[2] P. Heidenreich and A. M. Zoubir, "Fast maximum likelihood DOA estimation in the two-target case with applications to automotive radar," *Signal Process.*, vol. 93, no. 12, pp. 3400–3409, 2013.

[3] X. Zhang, X. Zhao, Z. Li, J. Xia, R. Jain, and W. Chao, "Social image tagging using graph-based reinforcement on multi-type interrelated objects," *Signal Process.*, vol. 93, no. 8, pp. 2178–2189, Aug. 2013.

[4] J. Gao, W. Fan, and J. Han, "On appropriate assumptions to mine data streams: Analysis and practice," in *Proc. IEEE ICDM*, 2007, pp. 143–152.

[5] I. Zliobaite, "Learning under concept drift: an overview," Vilnius University, Faculty of Mathematics and Informatics, Tech. Rep., 2010. [Online]. Available: http://arxiv.org/abs/1010.4784

[6] R. Ducasse, D. S. Turaga, and M. van der Schaar, "Adaptive topologic optimization for large-scale stream mining," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 3, pp. 620–636, 2010.

[7] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, Aug. 1997.

[8] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, Feb. 1994.

[9] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," in *Machine Learning*, 1988, pp. 285–318.

[10] A. Blum, "Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain," *Mach. Learn.*, vol. 26, no. 1, pp. 5–23, Jan. 1997.

[11] W. Fan, S. J. Stolfo, and J. Zhang, "The application of AdaBoost for distributed, scalable and on-line learning," in *Proc. ACM SIGKDD*, 1999, pp. 362–366.

[12] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. ACM SIGKDD*, 2003, pp. 226–235.

[13] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Integrating novel class detection with classification for concept-drifting data streams," in *Proc. ECML PKDD*, 2009, pp. 79–94.

[14] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Mach. Learn.*, vol. 32, no. 2, pp. 151–178, Aug. 1998.

[15] J. Z. Kolter and M. A. Maloof, "Using additive expert ensembles to cope with concept drift," in *Proc. ICML*, 2005, pp. 449–456.

[16] ——, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.

[17] F. Rosenblatt, "The perceptron: a perceiving and recognizing automaton," Cornell Aeronautical Laboratory, Tech. Rep., 1957.

[18] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. ACM SIGKDD*, 2001, pp. 377–382.

[19] L. L. Minku and Y. Xin, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, 2012.

[20] K. Geras and C. Sutton, "Multiple-source cross-validation," in *Proc. ICML*, vol. 28, no. 3, May 2013, pp. 1292–1300.

[21] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.

[22] M. Sewell, "Ensemble learning," *UCL Research Note*, 2008.

[23] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.

[24] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.

[25] P. K. Varshney, *Distributed Detection and Data Fusion*, 1st ed. Springer-Verlag New York, Inc., 1996.

[26] D. Ciuonzo, A. D. Maio, and P. S. Rossi, "A systematic framework for composite hypothesis testing of independent bernoulli trials," *IEEE Signal Process. Lett.*, vol. 22, no. 9, pp. 1249–1253, Sept 2015.

[27] N. Michelusi and U. Mitra, "Cross-layer design of distributed sensing-estimation with quality feedback – part i: Optimal schemes," *IEEE Trans. Signal Process.*, vol. 63, no. 5, pp. 1228–1243, March 2015.

[28] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[29] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proc. ACM PODS*, 2002, pp. 1–16.

[30] S. S. Ho and H. Wechsler, "A martingale framework for detecting changes in data streams by testing exchangeability," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2113–2127, Dec. 2010.

[31] K. Crammer, E. E. Dar, Y. Mansour, and J. W. Vaughan, "Regret minimization with concept drift," in *Proc. COLT*, 2010, pp. 168–180.

[32] S. Boucheron, G. Lugosi, and O. Bousquet, "Concentration inequalities," in *Advanced Lectures in Machine Learning*, 2004, pp. 208–240.

[33] K. Bache and M. Lichman, "UCI machine learning repository," http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences, 2013.

[34] X. Wu, P.-P. Li, and X. Hu, "Learning from concept drifting data streams with unlabeled data," *Neurocomputing*, vol. 92, pp. 145–155, 2012.

[35] M. Harries, "SPLICE–2 comparative evaluation: Electricity pricing," University of New South Wales, School of Computer Science and Engineering, Tech. Rep., 1999.

[36] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. SBIA*, 2004, pp. 286–295.

[37] M. Baena-Garcia, J. del Campo-Avila, R. Fidalgo, and A. Bifet, "Early drift detection method," in *Proc. ECML PKDD*, 2006, pp. 77–86.

[38] L. I. Kuncheva and C. O. Plumpton, "Adaptive learning rate for online linear discriminant classifiers," in *Proc. SSPR*, 2008, pp. 510–519.

[39] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank, "Fast perceptron decision tree learning from evolving data streams," in *Proc. PAKDD*, 2010, pp. 299–310.

[40] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proc. SDM*, 2007.

[41] J. A. Blackard, "Comparison of neural networks and discriminant analysis in predicting forest cover types," Ph.D. dissertation, Colorado State University, 1998.

[42] J. Gama, R. Rocha, and P. M. Gama, "Accurate decision trees for mining high-speed data streams," in *Proc. ACM SIGKDD*, 2003, pp. 523–528.

[43] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proc. ACM SIGKDD*, 2001, pp. 359–364.

[44] "PAKDD data mining competition 2009, credit risk assessment on a private label credit card application," http://sede.neurotech.com.br/PAKDD2009.

[45] P. Willett, P. F. Swaszek, and R. S. Blum, "The good, bad and ugly: distributed detection of a known signal in dependent gaussian noise," *IEEE Trans. Signal Process.*, vol. 48, no. 12, pp. 3266–3279, 2000.

[46] S. Shalev-Shwartz and Y. Singer, "A new perspective on an old perceptron algorithm," in *Proc. COLT*, 2005, pp. 264–278.

Luca Canzian received the B.Sc., M.Sc., and Ph.D. degrees in Electrical Engineering from the University of Padova, Italy, in 2005, 2007, and 2013, respectively. From 2007 to 2009 he worked in Venice, Italy, as an R&D Engineer at Tecnomare, a company providing design and engineering services for the oil industry. From September 2011 to March 2012 he was on leave at the University of California, Los Angeles (UCLA). From January 2013 to April 2014 he was a PostDoc at the Electrical Engineering Department at UCLA. From April 2014 to April 2015 he was a PostDoc at the Computer Science Department at University of Birmingham, UK. Since April 2015 he has been working in Bassano del Grappa, Italy, as an R&D Engineer at Qascom, a company providing design and engineering services for the satellite communication and navigation industry.

Yu Zhang received the Ph.D degree from University of California, Los Angeles, in 2013. He is now a Applied and Data Scientist in Microsoft. His research interests include social network analytics, web data mining, and game theory.

Mihaela van der Schaar is Chancellor's Professor of Electrical Engineering at University of California, Los Angeles. She is an IEEE Fellow since 2010, a Distinguished Lecturer of the Communications Society for 2011-2012, the Editor in Chief of IEEE Transactions on Multimedia (2011 - 2013). She received an NSF CAREER Award (2004), the Best Paper Award from IEEE Transactions on Circuits and Systems for Video Technology (2005), the Okawa Foundation Award (2006), the IBM Faculty Award (2005, 2007, 2008), the Most Cited Paper Award from EURASIP: Image Communications Journal (2006), the Gamenets Conference Best Paper Award (2011) and the 2011 IEEE Circuits and Systems Society Darlington Award Best Paper Award. She holds 33 granted US patents. She is also the founding director of the UCLA Center for Engineering Economics, Learning, and Networks (see http://netecon.ee.ucla.edu). For more information about her research visit: http://medianetlab.ee.ucla.edu/