Timely Event Detection by Networked Learners

Luca Canzian and Mihaela van der Schaar

***Abstract*—We consider a set of distributed learners that are interconnected via an exogenously-determined network. The learners observe different data streams that are related to common events of interest, which need to be detected in a timely manner. Each learner is equipped with a set of local classifiers, which generate local predictions about the common event based on the locally observed data streams. In this work, we address the following key questions: (1) Can the learners improve their detection accuracy by exchanging and aggregating information? (2) Can the learners improve the timeliness of their detections by forming clusters, i.e., by collecting information only from surrounding learners? (3) Given a specific tradeoff between detection accuracy and detection delay, is it desirable to aggregate a large amount of information, or is it better to focus on the most recent and relevant information? To address these questions, we propose a cooperative online learning scheme in which each learner maintains a set of weight vectors (one for each possible cluster), selects a cluster and the corresponding weight vector, generates a local prediction, disseminates it through the network, and combines all the received local predictions from the learners belonging to the selected cluster by using a weighted majority rule. The optimal cluster and weight vector that a learner should adopt depend on the specific network topology, on the location of the learner in the network, and on the characteristics of the data streams. To learn such optimal values, we propose a general online learning rule that exploits only the feedbacks that the learners receive. We determine an upper bound for the worst-case mis-detection probability and for the worst-case prediction delay of our scheme in the realizable case. Numerical simulations show that the proposed scheme is able to successfully adapt to the unknown characteristics of the data streams and can achieve substantial performance gains with respect to a scheme in which the learners act individually or a scheme in which the learners always aggregate all available local predictions. We numerically evaluate the impact that different network topologies have on the final performance. Finally, we discuss several surprising existing trade-offs.**

***Index Terms*—Event detection, clustering, networked learners, online learning, distributed learning, ensemble of classifiers, weighted majority, classification.**

I. Introduction

The world is increasingly information-driven. Vast amounts of data are being produced by diverse sources and in diverse formats including sensor readings, physiological measurements, documents, emails, transactions, tweets, and multimedia files. Many businesses and government institutions are also embracing automation and relying on a variety of sensors and infrastructure to collect, store, and analyze data on a continuous basis. It is becoming critical to endow assessment systems with the ability to process streaming information from sensors in real-time in order to better manage physical systems, derive informed decisions, tweak production processes, and optimize logistics choices [1].

Consider for example a city that is responsible for providing public transport and road emergency services. The city can use information from mobile phone applications such as Waze, augmented with its own road sensors (loop sensors, cameras, etc.) and transport sensors (GPS on buses and trains, etc.), to predict some minutes in advance the amount of traffic in major intersections. These predictions can be used to optimize the transport grid in real-time, provide emergency services (e.g., evacuations and dynamic closures), modify public transport (e.g., dynamic connections between bus/train routes based on current demand), and even control the traffic light systems.

This traffic example shows the need for developing novel analytic algorithms and platforms to support the unique requirements of stream mining applications [2], [3], including 1) handling data with different formats, sources, content, rates and features; 2) analyzing data *in motion*, in order to support *real-time* and *low-latency* requirements; and 3) processing the data in a distributed manner.

The main aim of this paper is to develop a systematic framework for real-time distributed stream-mining solutions for *event detection*. The term "event" is used in this paper as an abstract concept that is characterized by an unknown *duration*, an unknown *label*, and observable *features*. We consider a general network structure with multiple distributed *learners* that observe (parts of) the features of the events, and their final goal is to detect the events by discovering the changes in the values of the labels in a timely manner, before the events end. A learner is a node that relies on a *local classifier* – a processing element that partitions data sets into multiple classes of interest – to pre-process the observed features locally and to make a *local prediction*. We allow the learners to exchange the local predictions over a topology and to aggregate them before deciding on a final label classification. Since information requires time to spread in the network, some learners can receive outdated local predictions, and this can have a negative impact on the final label classification. Therefore, we allow each learner to limit the aggregation only to the most recent local predictions it receives, which is equivalent to selecting a *cluster* of surrounding learners and aggregating only the local predictions of the learners belonging to that cluster. More critically, we allow the learners to dynamically *adapt* the cluster selection and the aggregation model to minimize a parametrizable *cost function*, which will be formally defined as a combination between the *detection accuracy* and the *detection delay*. This component is a key feature of our framework and is particularly relevant in dynamic scenarios where the nature of the information being analyzed is constantly evolving.

Specifically, in our scheme each learner maintains a set of weight vectors (one for each possible cluster), selects a cluster and the corresponding weight vector, generates a local prediction, disseminates it through the network, and combines all the received local predictions from the learners belonging to the selected cluster by using a weighted majority rule. After an event ends, the learners receive a *feedback* about the event. For example, in the traffic application in which the goal is to predict

The authors are with the Department of Electrical Engineering, UCLA, Los Angeles CA 90095, USA. Email: lcanzian@ucla.edu – mihaela@ee.ucla.edu.

This work was partially supported by the AFOSR DDDAS grant.

some minutes in advance whether the amount of traffic in major intersections is above a certain threshold, the correctness of the predictions can be automatically verified after some minutes. The learners can exploit the received feedbacks to improve their cluster and weight vector choices. We propose a specific online learning rule to learn and adapt in parallel the weight vectors of all the possible clusters and to learn the optimal cluster to adopt. We call our distributed learning scheme Cluster Weighted Majority (CWM).

CWM is a general learning algorithm that can be applied to stream mining applications in which the correlation between the observed features and the event to detect is not perfectly known beforehand. In a *realizable scenario*, i.e., when there exist (unknown) weight vectors that would allow the learners to detect all the events *if* they had available all the local predictions, we determine performance guarantees for the misdetection probability and for the prediction delay of CWM, both in absence and in presence of communication errors.

Finally, we run numerous simulations and we evaluate the performance of the CWM for a variety of system parameters and network topologies. The results show that 1) CWM is able to successfully adapt to the unknown characteristics of the data streams and can achieve substantial performance gains with respect to a scheme in which the learners act individually and with respect to a scheme in which the learners always aggregate all available local predictions; and 2) interesting tradeoffs happen, which depend on the network topology and on the system parameters.

The rest of this paper is organized as follows. Section II reviews the relevant literature. Section III presents our formalism and framework for distributed online learning. Section IV derives the proposed learning rules. Section V proves performance guarantees in the realizable case. Section VI presents the empirical evaluation of our scheme. Section VII concludes the paper.

II. Related Works

Similarly to our approach, in the works dealing with *future event prediction* [4]–[7] the labels are correlated in time and the feedback is received with delay. [4] and [5] exploit only the label time series to make predictions. In addition [4] adopts a Bayesian model in which the distribution of the process generating the labels is known and there exists a belief (constantly updated) about the values of the parameters of the process. Differently from these papers, in our work we assume no a priori knowledge about the statistical properties of the data and we exploit also additional information – the local observations and predictions of distributed learners – to determine the final label class. Additional information is also exploited in [6] and [7]. Specifically, [6] exploits the textual tweets of users to predict box-office revenues of movies, whereas [7] exploits Google Trends indexes to predict Ford's monthly sales. Differently from our approach, none of the cited works in future event prediction consider a distributed scenario and propose a model that adapts online based on the characteristics of the observed data.

Ensemble techniques [8]–[11] build and combine a collection of base classifiers into a unique classifier. Examples of these techniques include bagging [8], boosting [9], and weighted majority [10]. Similarly to our work, most of the ensemble learning techniques are developed assuming no a priori knowledge about the statistical properties of the data. However, differently from our work, such techniques are designed for a centralized scenario; in fact, the base classifiers are not distributed entities, they all observe the same data stream. The focus of ensemble learning is on the statistical advantages of learning with an ensemble and not on the nature of learning under communication constraints. One might cast these techniques within the framework of distributed learning, but they would suffer from many drawbacks. For example, [12], [13] would require an entity that collects and stores all the data recently observed by the learners and that tells the learners how to adapt their local classifiers, which is clearly impractical in real-time stream processing applications characterized by high data rates.

The *estimation* literature [14]–[19] consists of learning agents that are linked together through a network topology. The agents must estimate some parameters based on their local observations and on the continuous sharing and diffusion of information across the network. Similarly to our work, the estimation literature is about learning in a distributed environment under communication constraints. In fact, [15] shows that a classification problem can be cast within the estimation framework. However, there is a major difference between our work and the estimation techniques. In the estimation literature the parameter estimation represents the final objective; hence, the goodness of a scheme is measured by the time required to learn the parameters with a certain accuracy. In our work the focus is on the timeliness of the event detection, and the learning of the parameters (i.e., the weights and the cluster levels) is designed with this objective in mind.

The *decentralized detection* literature [20]–[22] considers a scenario in which a network of nodes provides relevant information about the state of nature to a fusion center, which then solve a standard problem of statistical inference in order to detect the events. The focus of the detection literature is on providing the rules according to which the nodes should decide what to transmit, whereas in our work we provide the rule according to which the nodes learn how to aggregate the information received.

In our previous work [23] we consider a scenario in which distributed learners observe distributed data streams, make local predictions, exchange the predictions, and aggregate them using a weighted majority rule. However, differently from the current work, [23] focuses on designing a protocol to spread the local predictions efficiently in the network and makes the assumption that the classifiers receive all the local predictions before deciding on a final label class. As a consequence, in [23] the network topology and the location of a learner have no impact on the final performance of the learner. Instead, in the current work the learners receive information with delay. As a consequence, differently from [23], in the current work 1) we include the detection delay as a performance metric to investigate, 2) we allow each learners to aggregate only the information belonging to a cluster of learners, 3) we propose a scheme in which each learner has to jointly learn the weights and the cluster, and 4) the network topology and the learner position inside the network play a fundamental role in the performance of each learner.

Importantly, all the above cited works focus only on the accuracy of the prediction, whereas in this paper we also focus on the timeliness of the event detection.

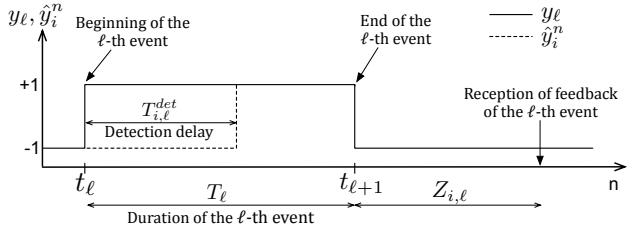

Fig. 1: Illustration of the considered scenario and notations.

III. The Considered Distributed Learning Setting

We consider a set $\mathcal{K} = \{1, \dots, K\}$ of K distributed *learners* and we divide the time into *slots*. At the beginning of the n-th time slot, one *label* $y^n \in \{-1, +1\}$ and K multi-dimensional *instances* $\mathbf{x}_i^n \in \mathcal{X}_i$, $i = 1, \dots, K$, are generated by an unknown (possibly stochastic) process.[1] Each learner i observes the instance $\mathbf{x}_i^n$ and its task is to estimate the value of the label.

Each learner i is equipped with a *local classifier* $f_i^n : \mathcal{X}_i \to \{-1, +1\}$ that generates the *local prediction* $s_i^n \triangleq f_i^n(\mathbf{x}_i^n)$ at time slot n. The specific structure of the local classifiers (e.g., pre-trained static classifiers vs. incremental classifiers [24], Naive Bayes vs. Decision Trees, etc.) has an impact on the final performance of our scheme. However, our scheme can be applied with any type of classifier, and different learners can also adopt different local classifiers. For this reason, throughout the manuscript we assume that the form of the local classifiers are given so that their design is not the focus of this work; instead, our emphasis will be on how to aggregate the local predictions made by the classifiers.

We define an *event* as a tuple,

$$e_\ell = \{t_\ell, t_{\ell+1}, y_\ell, s_{1,\ell}, \dots, s_{K,\ell}\},$$

where t_ℓ is the time instant at which the event begins, $t_{\ell+1}$ is the time instant at which the event ends (which corresponds to the time instant at which the subsequent event begins), $y_\ell, s_{1,\ell}, \dots, s_{K,\ell}$ are the label and the local predictions associated to the event. The label and the local predictions are fixed for the whole duration of the event, i.e., $y^n = y_\ell$ and $s_i^n = s_{i,\ell}$, $\forall\, i \in \mathcal{K}$ and $t_\ell \le n < t_{\ell+1}$.[2] We denote by $T_\ell = t_{\ell+1} - t_\ell$ the *duration* of the event. Fig. 1 illustrates these notations and the task of the learners: they must *detect* the events in a *timely* manner following the changes in the labels – we will formally define it in Section IV-A.

The learners are connected via an exogenously determined network $\mathcal{G}$, which is defined as the set of *links* among pairs of learners. We say that there is a *link* (i, j) between learner i and j if they can communicate directly. In this case, we say that learners i and j are *neighbors*. We consider only connected networks, i.e., all the learners can communicate with each other through a multi-hop communication in which intermediate learners forward the information they receive to the next learner in the path. A *path* between learners i and j is a sequence of d_{path} adjacent links that connect i to j, d_{path} is called the *length* of the path. The *distance* $d_{ij}(\mathcal{G})$ between learners i and j is the minimum among the lengths of all the paths that connect i to j, and a path with a length equal to the distance is called *shortest path*. The *eccentricity* $\epsilon_i(\mathcal{G}) \triangleq \max_j d_{ij}$ of learner i is the maximum distance between i and any other learner in the network. We denote by $\mathcal{K}_{i,d}(\mathcal{G})$ the set of learners whose distance from i is at most d, including learner i itself, and by $K_{i,d}(\mathcal{G})$ the cardinality of $\mathcal{K}_{i,d}(\mathcal{G})$. We refer to $\mathcal{K}_{i,d}(\mathcal{G})$ as learner i's *cluster* of level d. To simplify the notations, we will often omit the dependency from the network $\mathcal{G}$, i.e., we will write d_{ij} instead of $d_{ij}(\mathcal{G})$, and so on for the other network-dependent parameters.

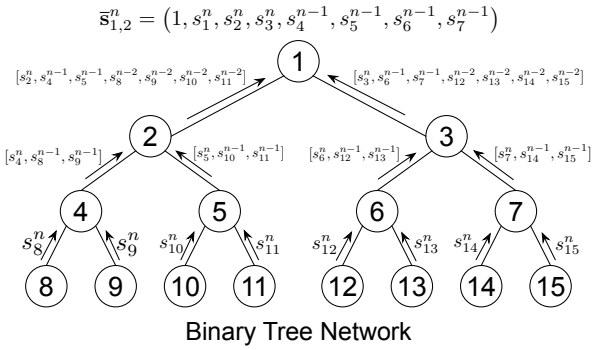

Fig. 2: Flow of information toward learner 1 at time slot n for a binary tree network, $\bar{\mathbf{s}}_{1,2}^n$ is learner 1's level-2 local prediction vector at time instant n.

We allow the distributed learners to exchange and aggregate their local predictions through multi-hop communications; however, within one time slot a learner can send only a single transmission to each of its neighbors. We denote by $\bar{s}_{ij}^n$ learner j's local prediction possessed by learner i before the aggregation at time instant n. The information is disseminated in the network as follows. First, each learner i observes $\mathbf{x}_i^n$ and updates $\bar{s}_{ii}^n = s_i^n = f_i^n(\mathbf{x}_i^n)$. Next, learner i transmits to each neighbor j the local prediction s_i^n and the local predictions $\bar{s}_{ik}^{n-1}$, for each learner $k \neq i$ such that the link (i, j) belongs to the shortest path between k and j. We assume that transmissions are always correctly received (we deal with erroneous communication in Subsection V-C), hence we obtain

$$\bar{s}_{ii}^n = s_i^n \quad ; \quad \bar{s}_{ij}^n = s_j^{n-d_{ij}+1} \;\; \forall\, j \neq i\,. \tag{1}$$

For instance, Fig. 2 represents the flow of information toward learner 1 for a binary tree network. Notice that some learners are required to transmit several local predictions, but each local prediction is simply encoded by one bit.

After having updated its local predictions $\bar{s}_{ij}^n$, $\forall\, j \in \mathcal{K}$, learner i aggregates all the local predictions to compute a final prediction $\hat{y}_i^n$. Similarly to most ensemble techniques, such as [9], [10], [12], [13], we consider a *weighted majority* aggregation rule. Learner i maintains in memory ϵ_i weight vectors,

$$\mathbf{w}_{i,d}^n \triangleq (w_{i0,d}^n, w_{ij,d}^n, \forall\, j \in \mathcal{K}_{i,d}) \quad \in \Re^{K_{i,d}+1}\,,$$

$d = 1, \dots, \epsilon_i$. Adopting the *level-d weight vector* $\mathbf{w}_{i,d}^n$, learner i can compute the aggregate prediction $\hat{y}_{i,d}^n$ with the weighted

[1] The proposed framework and scheme can be extended to multi-class labels.

[2] We remark that a binary classifier divides the input space into two regions: one region includes all the instances that are classified as $+1$, whereas the other region includes all the instances that are classified as -1. Hence, the assumption that the local predictions of a classifier are fixed during the whole duration of an event is equivalent to the assumption that the variations of the observed instances do not cause it to cross the separation boundary among the regions. For example, in the network intrusion example that we describe at the end of this section, a network attack can represent a situation in which some routers observe an increase in their input traffic. Though the input traffic can slightly vary in time, during such an attack these routers will consistently observe a traffic rate that is higher than a given threshold, and hence they will consistently generate the local prediction $+1$ during the whole duration of the attack. Finally, we remark that this assumption is only required for the performance analysis (Section V), and the proposed scheme can be adopted also in scenarios in which such an assumption does not hold.

majority aggregation rule

$$\hat{y}_{i,d}^n \triangleq \text{sgn}\left(\mathbf{w}_{i,d}^n \cdot \bar{\mathbf{s}}_{i,d}^n\right) \triangleq \begin{cases} +1 & \text{if } \mathbf{w}_{i,d}^n \cdot \bar{\mathbf{s}}_{i,d}^n \geq 0 \\ -1 & \text{otherwise} \end{cases} ,$$

where $\bar{\mathbf{s}}_{i,d}^n \triangleq (1, \bar{s}_{ij}^n, \forall\, j \in \mathcal{K}_{i,d}) \in \{-1,+1\}^{K_{i,d}+1}$ is learner i's *level-d local prediction vector*, $\mathbf{w}_{i,d}^n \cdot \bar{\mathbf{s}}_{i,d}^n \triangleq w_{i0,d}^n + \sum_{j\in\mathcal{K}_{i,d}} w_{ij,d}^n \bar{s}_{ij}^n$ is the inner product between $\mathbf{w}_{i,d}^n$ and $\bar{\mathbf{s}}_{i,d}^n$, and $\text{sgn}(\cdot)$ is the sign function.

In each time instant n, learner i selects a specific *cluster level* $\underline{d}_i^n$, $\underline{d}_i^n \in \{1,\ldots,\epsilon_i\}$, and outputs the *final prediction*

$$\hat{y}_i^n \triangleq \hat{y}_{i,\underline{d}_i^n}^n = \text{sgn}\left(\mathbf{w}_{i,\underline{d}_i^n}^n \cdot \bar{\mathbf{s}}_{i,\underline{d}_i^n}^n\right) . \tag{2}$$

In the above construction, learner i first selects $\underline{d}_i^n$, then it adopts $\mathbf{w}_{i,\underline{d}_i^n}^n$ to aggregate the local predictions $\{\bar{s}_{ij}^n, j \in \mathcal{K}_{i,\underline{d}_i^n}\}$ of the learners belonging to the cluster $\mathcal{K}_{i,\underline{d}_i^n}$, and finally it uses the sign of the aggregated information to output its final classification $\hat{y}_i^n$. According to (2), the equation $\mathbf{w}_{i,\underline{d}_i^n}^n \cdot \bar{\mathbf{s}}_{i,\underline{d}_i^n}^n = 0$ defines a hyperplane in $\Re^{K_{i,\underline{d}_i^n}}$ which separates the positive predictions (i.e., $\hat{y}_i^n = +1$) from the negative ones (i.e., $\hat{y}_i^n = -1$).

While weighted majority aggregation rules have been considered before in the ensemble learning literature [9], [10], [12], [13], there are some important distinctions that are worth emphasizing: 1) since we are limiting the learners to exchange information only via links, learners receive information from other learners with delay – see Eq. (1). For this reason, we allow the learners to select the cluster level $\underline{d}_i^n$, i.e., to trade-off the quantity and the currency of the information to exploit; and 2) we introduce the weight $w_{i0,d}^n$ that can be thought as the weight associated to a "virtual learner" that always sends the local prediction $+1$, this means that we do not constrain the separating hyperplane to be homogeneous.

In traditional *online learning settings* [25] a learner is assumed to receive a *feedback*, which consists of the real value of the label, after it outputs the final prediction; this information is exploited by the learner to update its configuration in order to improve its classification accuracy. In our context the feedback includes the information about an event, i.e., the starting time t_ℓ, the ending time $t_{\ell+1}$, the label y_ℓ, and the local predictions $s_{1,\ell},\ldots,s_{K,\ell}$.[3] However, we allow for the feedback to be received with delay. Specifically, the feedback that refers to the event e_ℓ is received by learner i at time instant $t_{\ell+1}+Z_{i,\ell}$, where $Z_{i,\ell}$ is a random variable bounded by $\overline{Z}$ – see Fig. 1. We assume that the feedbacks are received at the end of the time slot, after the learners have computed their final predictions for the current time slot. If at the end of the n-th time slot learner i receives a feedback, it exploits this information to update its aggregation weight vectors $\mathbf{w}_{i,d}^n$, $\forall\, d \in \{1,\ldots,\epsilon_i\}$, and to select the cluster level $\underline{d}_i^n$ in the subsequent time instants.

To summarize, a generic time slot n can be subdivided into the following four phases:

1. Observation: each learner i observes the instance $\mathbf{x}_i^n$;

2. Local Prediction Exchange: each learner i computes its current local prediction s_i^n, sends to its neighbors the local predictions it possesses, receives the local predictions sent by its neighbors, and updates the state of the local predictions it possesses;

3. Final Prediction: each learner i selects the cluster level $\underline{d}_i^n$ and computes its final prediction $\hat{y}_i^n$ using (2);

4. Update: if learner i receives a feedback it updates its current weight vectors $\mathbf{w}_{i,d}^n$, $\forall\, d \in \{1,\ldots,\epsilon_i\}$, and its current estimate $\underline{d}_i^n$ of the optimal cluster level to adopt.

In Subsection IV-C we will propose a specific *update rule* to adapt $\mathbf{w}_{i,d}^n$, whereas in Subsection IV-B we describe a specific *selection rule* to choose the cluster level $\underline{d}_i^n$.

Before concluding this section, we discuss a possible application of our model.

Example: Network Intrusion. Consider a situation in which K learners (e.g., routers) are monitoring the traffic of a network and have to distinguish between *normal* connections and *attacks*. Since the traffic volume can be huge[4], it is costly and sometimes unfeasible to exchange the raw information observed by the learners. Hence, the learners can generate local predictions based on the local traffic they observe, exchange only the local predictions, and aggregate them to detect an attack in a more accurate and timely manner. In this context, an event corresponds to a specific attack or to a normal connection. In the KDDCUP'99 network intrusion data set [13], a data set widely used in the stream mining literature, multiple TCP connections are acquired every second, whereas network attacks last tenths of minutes. As a consequence, a long sequence of time slots with no attacks (i.e., the label is -1) is followed by a long sequence of time slots with attacks (i.e., the label is $+1$). This shows that in this type of application the event duration is typically much longer than the time slot.

IV. The Proposed Learning Scheme

In this section we first define a performance metric (Subsection V-A), then we propose a *selection rule* to choose the cluster level $\underline{d}_i^n$ (Subsection IV-B) and a *learning rule* to adapt the weight vectors $\mathbf{w}_{i,d}^n$ (Subsection IV-B).

A. Performance metric

We say that the event e_ℓ is *detected* by learner i if there exists a time instant $n_\ell \in \{t_\ell,\ldots,t_{\ell+1}\}$ such that

$$\begin{aligned} &\hat{y}_i^n \neq y_\ell \;\;,\;\; \forall\, n = t_\ell,\ldots,n_\ell - 1\,, \\ &\hat{y}_i^n = y_\ell \;\;,\;\; \forall\, n = n_\ell,\ldots,t_{\ell+1}\,. \end{aligned} \tag{3}$$

Notice that (3) includes also the case in which the final prediction $\hat{y}_i^n$ agrees with the label for the whole duration of the event, in this case $n_\ell = t_\ell$. For a detected event, we define the *detection delay* $T_{i,\ell}^{det} \triangleq n_\ell - t_\ell$.

Given a sequence of N events, $\mathbf{e}_N \triangleq (e_1, e_2, \ldots, e_N)$, we denote by $E_i^{det}(\mathbf{e}_N)$ and N_i^{det} the set and number of events detected by learner i, respectively. We define the *mis-detection probability* $P_i(\mathbf{e}_N)$ of learner i as the average number of undetected events,

$$P_i(\mathbf{e}_N) \triangleq \frac{N - N_i^{det}}{N}\,, \tag{4}$$

and we define the *delay* $D_i(\mathbf{e}_N)$ of learner i as the average detection delay,

$$D_i(\mathbf{e}_N) \triangleq \frac{\sum_{\ell\in E_i^{det}} T_{i,\ell}^{det}}{N_i^{det}}\,. \tag{5}$$

[3] Notice that after having observed t_ℓ learner i can compute the local predictions $s_{j,\ell} = \bar{s}_{ij}^{t_\ell + d_{ij} - 1}$, $\forall\, j$.

[4] A single Internet Service Provider can receive thousands of DNS queries each second [26].

In order to incorporate these two measures in a single performance metric, we consider the following *cost function*,

$$C_i(\mathbf{e}_N) \triangleq P_i(\mathbf{e}_N) + \alpha D_i(\mathbf{e}_N)\,. \tag{6}$$

The *trade-off parameter* α is a design parameter which is used to trade-off between detection accuracy and delay. The value of α must be small if the considered application does not require timely detections, whereas it must be large if it is essential to detect an event in a timely manner.

In the next subsections we design the cluster level selection rule and the weight vector update rule in order to minimize the cost $C_i(\mathbf{e}_N)$. Note that there is no a priori knowledge about the (stochastic) process that generates the labels and the instances, and about the accuracies of the local classifiers. Hence, it is not possible to write an optimization problem to minimize the (expected) cost function (6). The selection and update rules must be designed in order to minimize estimates of the real cost $C_i(\mathbf{e}_N)$, and such estimates can be computed thanks to the received feedback.

B. Cluster level selection rule

Before describing our approach for the cluster level selection rule, we consider a simple example to highlight the importance of the choice of $\underline{d}_i^n$. Let us consider the binary tree network illustrative example in Fig. 2 and let us focus on learner 1 and on a time instant n in which the event e_ℓ begins, such that the label y_ℓ is different from the label of the preceding event $e_{\ell-1}$. We assume that the probability $P[s_{i,\ell} = y_\ell]$ that the local prediction $s_{i,\ell}$ agrees with the label y_ℓ is $3/4$, $\forall\, i$, and the event $s_{i,\ell} = y_\ell$ is independent from the event $s_{j,\ell} = y_\ell$, $\forall i \neq j$. Finally, we assume that $w^n_{10,d} = 0$ and $w^n_{1i,d} = 1$, $\forall\, d, i$, i.e., the local predictions of different learners are weighted equally (with this choice the detection accuracy is maximized). Fig. 3 shows the probability P that learner 1 detects the event e_ℓ, using different cluster levels and in different time instants. On one hand, after two time slots (i.e., at time instant $n+2$) the highest probability to detect an event is achieved when learner 1 adopts $\underline{d}_i^n = 3$, i.e., when it aggregates all the local predictions. On the other hand, the probability to detect the event immediately is maximized when learner 1 adopts $\underline{d}_i^n = 1$, i.e., when it aggregates only the prediction from its neighbors. This example shows that the choice of $\underline{d}_i^n$ has opposite effects on the accuracy and the delay of learner i, hence it must reflect the trade-off defined in Eq. (6) between accuracy and delay. To reach this objective, we empirically estimate the cost function (6) for each possible value of the cluster level, and in time instant n we adopt the cluster level $\underline{d}_i^n$ that minimizes such empirical costs.[5]

Specifically, learner i keeps in memory the following $2\epsilon_i + 1$ parameters: 1) $N_i^{fed,n}$ is the number of feedbacks received by learner i by time instant n; 2) $N_{i,d}^{det,n}$ is the number of times learner i would have detected an event adopting the cluster level d, among all events whose feedbacks have been received by time instant n, $d = 1, \ldots, \epsilon_i$; 3) $T_{i,d}^n$ is the sum of the detection delays of all the events learner i would have detected adopting the cluster level d, among all events whose feedbacks have been received by time instant n, $d = 1, \ldots, \epsilon_i$. Exploiting these parameters learner i can compute

$$\hat{C}_{i,d}^n \triangleq \frac{N_i^{fed,n} - N_{i,d}^{det,n}}{N_i^{fed,n}} + \alpha \frac{T_{i,d}^n}{N_{i,d}^{det,n}}\,,$$

that represents an *estimate* of the cost function that learner i obtains selecting the cluster level d, $d = 1, \ldots, \epsilon_i$.

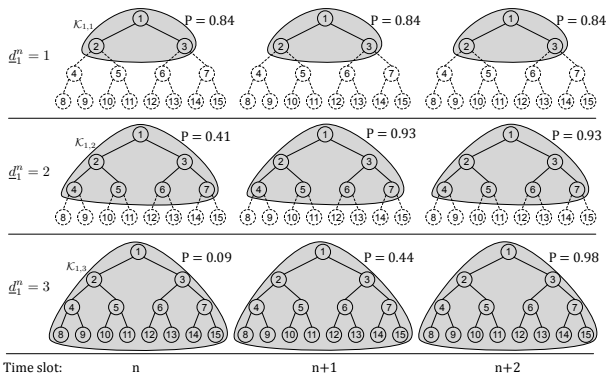

Fig. 3: Probability that learner 1 detects the event e_ℓ in a given time instant. Different rows refer to different cluster level values.

At time slots n learner i selects the cluster level that minimizes $\hat{C}_{i,d}^n$,

$$\underline{d}_i^n \triangleq \operatorname*{argmin}_d \frac{N_i^{fed,n} - N_{i,d}^{det,n}}{N_i^{fed,n}} + \alpha \frac{T_{i,d}^n}{N_{i,d}^{det,n}}\,. \tag{7}$$

The parameters $N_i^{fed,n}$, $N_{i,d}^{det,n}$ and $T_{i,d}^n$ are initialized to 0 and are updated whenever a new feedback is received. Assume that at time instant n the feedback that refers to the event e_ℓ is received. We write $I_{i,d,\ell} = 1$ if the event e_ℓ would have been detected by learner i adopting the current weight vector $\mathbf{w}_{i,d}^n$, and in this case $T_{i,d,\ell}$ is the corresponding detection delay; otherwise $I_{i,d,\ell} = T_{i,d,\ell} = 0$. The parameters $N_i^{fed,n}$, $N_{i,d}^{det,n}$ and $T_{i,d}^n$ are updated as follows,

$$\begin{aligned} N_i^{fed,n} &\triangleq N_i^{fed,n-1} + 1\,, \\ N_{i,d}^{det,n} &\triangleq N_{i,d}^{det,n-1} + I_{i,d,\ell}\,, \\ T_{i,d}^n &\triangleq T_{i,d}^{n-1} + T_{i,d,\ell}\,. \end{aligned} \tag{8}$$

C. Weight vectors update rule

To update the weight vector $\mathbf{w}_{i,d}^n$, we consider an empirical cost function that depends only on the probability to detect an event. The main reason why we do not consider the delay in this phase is that the values of the weights do not have a significant impact on the delay. Indeed, if the cluster level is d, after a maximum of $d-1$ time slots learner i receives all the information from the learners belonging to the level-d cluster; hence, the delay cannot be larger than $d-1$ independently of the values of the weights. Moreover, given the total cost defined in Eq. (6), it is not trivial to derive a suitable cost function for the individual events that includes also the delay, because the detection delay is undefined for non-detected events.

Let $n = t_\ell + Z_{i,\ell}$ the time instant at which learner i receives the feedback which refers to the event e_ℓ. We define the set

[5] In this work we focus on a scenario in which the cluster level is continuously adapted in real-time; as a consequence, each learner has to update the local predictions for all the other leaners and has to estimate the cost function for each possible value of the cluster level. This can be problematic and expensive if the number of learners is very large. A variation of the considered approach, in which the cluster learning phase is performed only once at the beginning (i.e., during a training phase) may be used for large scale stream-mining applications. Indeed, once the cluster for each node is learned, the real-time exchange of information is limited only to the nodes belonging to the same clusters, and each node has to receive and update only the local predictions of the nodes that belong to its cluster.

$E_i^{fed,n} \triangleq \{e_{\overline{\ell}} : t_{\overline{\ell}+1} + Z_{i,\overline{\ell}} \leq n\}$ of events whose feedbacks have been received by i by time instant n. For each event $e_{\overline{\ell}} \in E_i^{fed,n}$ and for each value of $d \in \{1, \ldots, \epsilon_i\}$, we consider the one-shot *hinge loss function* [27],

$$L_{i,\overline{\ell}}(\mathbf{w}_{i,d}) \triangleq \max\left(0, -y_{\overline{\ell}}\mathbf{w}_{i,d} \cdot \mathbf{s}_{i,d,\overline{\ell}}\right) ,$$

where $\mathbf{s}_{i,d,\overline{\ell}} \triangleq (1, s^n_{j,\overline{\ell}}, \forall\, j \in \mathcal{K}_{i,d})$ is the level-d local prediction vector associated to the event $e_{\overline{\ell}}$.

The hinge loss function $L_{i,\overline{\ell}}(\mathbf{w}_{i,d})$ is equal to 0 if the weight vector $\mathbf{w}_{i,d}$ allows to predict correctly the label $y_{\overline{\ell}}$ given the local prediction vector $\mathbf{s}_{i,d,\overline{\ell}}$ associated to the event $e_{\overline{\ell}}$, otherwise $L_{i,\overline{\ell}}(\mathbf{w}_{i,d})$ is proportional to the distance of the local prediction vector $\mathbf{s}_{i,d,\overline{\ell}}$ from the separating hyperplane defined by $\mathbf{w}_{i,d}$.

A possible approach to update the weight vector $\mathbf{w}^n_{i,d}$ is to compute $\mathbf{w}^{n+1}_{i,d}$ in order to minimize the *overall loss* of all the observations available at the end of time slot n,

$$\mathbf{w}^{n+1}_{i,d} = \underset{\mathbf{w}_i}{\operatorname{argmin}} \sum_{\ell \in E_i^{fed,n}} L_{i,\ell}(\mathbf{w}_{i,d}) . \tag{9}$$

However, to solve (9) learner i must store all the past labels and all the past local predictions of all the learners in the system, which is impractical in real stream mining systems in which the volume of the incoming data is high and the number of learners is large. Hence, we adopt a stochastic gradient descent algorithm to incrementally approach the solution of (9) using only the most recent feedback. The weight $\mathbf{w}^{n+1}_{i,d}$ is updated is the following manner:

$$\mathbf{w}^{n+1}_{i,d} = \mathbf{w}^n_{i,d} - \beta \nabla L_{i,\ell}(\mathbf{w}_{i,d}) ,$$

where $\nabla L_{i,\ell}(\mathbf{w}_{i,d})$ is the gradient of $L_{i,\ell}(\mathbf{w}_{i,d})$, and $\beta > 0$ is the step size. We have

$$\nabla L_{i,\ell}(\mathbf{w}_{i,d}) = \begin{cases} 0 & \text{if } \operatorname{sgn}(\mathbf{w}_{i,d} \cdot \mathbf{s}_{i,d,\ell}) = y_\ell \\ -y_\ell \mathbf{s}_{i,d,\ell} & \text{otherwise} \end{cases} .$$

Therefore, if in time instant n learner i observes the event e_ℓ, the resulting update rule for the weight vector $\mathbf{w}^n_{i,d}$ is

$$\mathbf{w}^{n+1}_{i,d} = \begin{cases} \mathbf{w}^n_{i,d} & \text{if } \operatorname{sgn}(\mathbf{w}^n_i \cdot \mathbf{s}_{i,d,\ell}) = y_\ell \\ \mathbf{w}^n_{i,d} + \beta y_\ell \mathbf{s}_{i,d,\ell} & \text{otherwise} \end{cases} . \tag{10}$$

This construction allows a meaningful interpretation. It shows that learner i should maintain its level of confidence in its level-d weight vector when it does not observe any event or when the label of the observed event y_ℓ agrees with the decision it would have taken with the current weight vector $\mathbf{w}^n_{i,d}$. If disagreement occurs, then learner i needs to assess which local predictions would have led to the mis-classification: the weight $w^n_{ij,d}$ that learner i adopts to scale the local predictions it receives from learner j is increased by β unit if $s_{j,\ell}$ agrees with the label y_ℓ, otherwise $w^n_{ij,d}$ is decreased by β unit.

Notice that if the weights are initialized to 0 then the choice of β does not influence the performance of the scheme. To see this, let denote by $\underline{\mathbf{w}}^n_{i,d}$ the weight vector we obtain using $\underline{\beta}$, and by $\overline{\mathbf{w}}^n_{i,d}$ the weight vector we obtain using $\overline{\beta}$. At each time instant n we have $\overline{\mathbf{w}}^n_{i,d} = \underline{\mathbf{w}}^n_{i,d}\overline{\beta}/\underline{\beta}$. Since $\overline{\beta}/\underline{\beta} > 0$, the predictions generated by Eq. (2) using $\overline{\mathbf{w}}^n_{i,d}$ are equal to the predictions generated using $\underline{\mathbf{w}}^n_{i,d}$. Hence, we can simply set $\beta = 1$.

We name the proposed distributed learning scheme Cluster Weighted Majority (CWM). The parameters N_i^{fed}, $N_{i,d}^{det}$, $T_{i,d}$, and $w_{ij,d}$ are initialized to 0 and adapted following the update rules (8) and (10), the cluster level is chosen with (7), and the final prediction is computed adopting the aggregation rule (2).

Algorithm Cluster Weighted Majority (CWM)

Initialization:
$N_i^{fed} = N_{i,d}^{det} = T_{i,d} = w_{ij,d} = \overline{s}_{ij} = 0, \quad \forall\, i, j, d$
For each learner i and time slot n
  Observe $\mathbf{x}_i$ and compute $s_i = f_i(\mathbf{x}_i)$
  Exchange and update $\overline{s}_{ij}$, $\forall\, j \in \mathcal{K}$
  Select $\underline{d} \leftarrow \operatorname{argmin}_d \frac{N_i^{fed} - N_{i,d}^{det}}{N_i^{fed}} + \alpha \frac{T_{i,d}}{N_i^{det}}$
  Predict $\hat{y}_i \leftarrow \operatorname{sgn}(\mathbf{w}_{i,\underline{d}} \cdot \overline{\mathbf{s}}_{i,\underline{d}})$
  If feedback y_ℓ is received
    $N_i^{fed} \leftarrow N_i^{fed} + 1$
    For each cluster level d
      Compute $\hat{y}_{i,d} \leftarrow \operatorname{sgn}(\mathbf{w}_{i,d} \cdot \mathbf{s}_{i,d,\ell})$
      If $\hat{y}_{i,d} = y_\ell$
        Update $N_{i,d}^{det} \leftarrow N_{i,d}^{det} + 1$
        Update $T_{i,d} \leftarrow T_{i,d} + T_{i,d,\ell}$
      Else
        Update $\mathbf{w}_{i,d} \leftarrow \mathbf{w}_{i,d} + y_\ell \mathbf{s}_{i,d,\ell}$

V. Performance Guarantees for CWM

In this section we analytically quantify the performance of the proposed scheme in the realizable scenario, in terms of mis-detection probability (4) and delay (5).

A. Realizable case and information eccentricity definitions

In this section we focus on the *realizable case*, i.e., we assume that there exists an unknown and non-empty class of weight vectors $\mathcal{W}$ such that for each possible event e_ℓ and for each weight vector $\mathbf{w} \in \mathcal{W}$ we have $\operatorname{sgn}\left(\mathbf{w} \cdot \mathbf{s}_{\overline{\ell}}\right) = y_\ell$, where $\mathbf{s}_{\overline{\ell}} = (1, s_{1,\ell}, \ldots, s_{K,\ell})$. This assumption asserts that there exist unknown weight vectors that would allow the learners to detect all the events *if* they had available all the local predictions. We remark that the validity of the realizability assumption assumption depends on the instances observed and local classifiers adopted by the learners; however, it does not require a specific structure on the local classifiers (e.g., some of them can be pre-trained static classifiers and others can be incremental classifiers).

Assume for the moment that an event e_ℓ has a duration equal to or larger than ϵ_i, such that learner i receives all the local predictions $s_{1,\ell}, \ldots, s_{K,\ell}$ before the event ends. This implies that learner i is potentially able to detect the event e_ℓ. However, learner i might be able to detect e_ℓ before it receives all the local predictions from all the learners in the network. For this reason, we define a new eccentricity metric, the *information eccentricity* $\epsilon_i^I(\mathcal{G})$, which is the distance of learner i from the minimum amount of local predictions it requires to detect all the possible events. Formally,

$$\epsilon_i^I(\mathcal{G}) = \min\{d : \exists\, \mathbf{w} \in \mathcal{W} \text{ such that } w_j = 0,\, \forall\, j \notin \mathcal{K}_{i,d}(\mathcal{G})\} .$$

The information eccentricity depends on the specific position of a learner in the network, and we have $\epsilon_i^I(\mathcal{G}) \leq \epsilon_i(\mathcal{G}) < K$. At most after $\epsilon_i^I(\mathcal{G}) - 1$ time slots from the beginning of an event, learner i possesses an amount of information that potentially allows him to detect the event.

Fig. 4 illustrates two simple examples of information eccentricity for the learners 1 and 9 in the binary tree network.

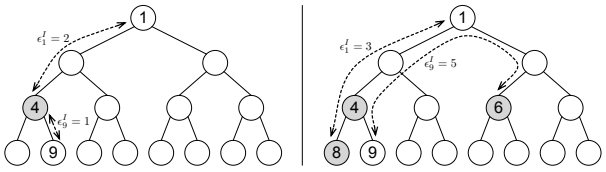

Fig. 4: An illustrative representation of the information eccentricity. If learner 4's local prediction is enough to detect all the events, then $\epsilon_1 = 2$ and $\epsilon_9 = 1$ (left side). If the local predictions of the learners 4, 6, and 8 are required to detect all the events, then $\epsilon_1 = 3$ and $\epsilon_9 = 5$.

In the first example (left side) the local prediction of learner 4 always agrees with the label, hence $\epsilon_1 = d_{14} = 2$ and $\epsilon_9 = d_{94} = 1$. In the second example (right side) an event is predicted correctly by either learner 4, learner 6, or learner 8, hence $\epsilon_1 = d_{18} = 3$ and $\epsilon_9 = d_{96} = 5$.[6] We remark that this is a simple illustrative example, in general the events may depend on the local predictions following more complex rules.

B. Mis-detection probability and delay bounds

Let $\mathbf{e}_N$ be a sequence of N events. We denote by N_d^{dur} the number of events in $\mathbf{e}_N$ that have a duration smaller than d, and we denote by $N_{i,d}^{mis}$ the number of mis-detections that learner i obtains if it always selects the cluster level $\underline{d}_i^n = d$, for each time instant n. Lemma 1 shows that $N_{i,d}^{mis}$ is upper bounded by a number which depends only on the number of aggregated local predictions $K_{i,d}$, on the number N_d^{dur} of events that have a duration smaller than d, on the maximum feedback delay $\overline{Z}$, and on the cluster level d. Notice that $N_{i,d}^{mis}$ is not the number of mis-detections of the proposed scheme, since the proposed scheme select $\underline{d}_i^n$ adopting (7) instead of using a fixed $\underline{d}_i^n = d$. However, the bound defined by Lemma 1 will be useful to derive the results in Theorems 1 and 2 that refer to our scheme.

Lemma 1. *For every d such that $\epsilon_i^I(\mathcal{G}) \leq d \leq \epsilon_i(\mathcal{G})$ we have*

$$N_{i,d}^{mis} \leq K_{i,d}\left(K_{i,d}+1\right) + N_d^{dur} + \left\lfloor \frac{\overline{Z}-d}{d} \right\rfloor , \qquad (11)$$

where $\lfloor x \rfloor$ is the largest integer lower than or equal to x.

Proof: See Appendix A. ∎

Among the three terms of the right hand side of (11), only N_d^{dur} increases in N. This means that the weight vector update rule defined in (10) converges after a finite number of updates to a weight vector $\mathbf{w}_{i,d}^n$ that allows learner i to detect all the events that are longer than d. The maximum number of required updates depends quadratically on the number $K_{i,d}$ of local predictions aggregated.

Theorem 1 asserts that there exists an upper bound for the mis-detection probability of a learner i that adopts our scheme. Such bound depends on many parameters of the system: the trade-off parameter α, the information eccentricity $\epsilon_i^I(\mathcal{G})$, the eccentricity $\epsilon_i(\mathcal{G})$, the number N_d^{dur} of events that have a duration smaller than d, the number of learners K, and the maximum feedback delay $\overline{Z}$. However, most contributions vanish to 0 as the number of events N increases.

[6]In this case a weight vector $\mathbf{w}$ that allows to detect all the events is for example $w_0 = 2$, $w_4 = w_6 = w_8 = 1$, and $w_i = 0$, $\forall\, i \neq 0, 4, 6, 8$. Notice that if we constrain $w_0 = 0$, as in [9], [10], [12], [13], we would not be able to find a weight vector $\mathbf{w}$ that allows to detect all the events.

Theorem 1. *The mis-detection probability $P_i(\mathbf{e}_N)$ of a learner i that adopts CWM is upper bounded by*

$$\mathbf{B}_i^{mis}(\mathbf{e}_N) = \alpha\left(\epsilon_i^I - 1\right)^2 + \epsilon_i \frac{N_{\epsilon_i}^{dur}}{N} + \frac{\epsilon_i^I - 1}{N} + \\ + \frac{\epsilon_i}{N}\left[K(K+1) + \overline{Z}\right] . \qquad (12)$$

Proof: See Appendix B. ∎

The bound $\mathbf{B}_i^{mis}(\mathbf{e}_N)$ increases in α, $\epsilon_i^I(\mathcal{G})$, $\epsilon_i(\mathcal{G})$, $N_{\epsilon_i}^{dur}$, K, and $\overline{Z}$, whereas it decreases in the number of events N. Importantly, notice that the bound (12) is valid for any number of events N and for any sequence $\mathbf{e}_N$. As a particular case, if $N \to +\infty$ the bound $\mathbf{B}_i^{mis}(\mathbf{e}_N)$ tends to $\alpha\left(\epsilon_i^I(\mathcal{G}) - 1\right)^2 + \epsilon_i P_{\epsilon_i}^{dur}$, where $P_{\epsilon_i}^{dur} \triangleq \lim_{N\to+\infty} N_{\epsilon_i}^{dur}/N$ can be interpreted as the probability that an event has a duration smaller than i's eccentricity $\epsilon_i(\mathcal{G})$. This is remarked in Corollary 1.

Corollary 1. $\lim_{N\to+\infty} \mathbf{B}_i^{mis}(\mathbf{e}_N) = \alpha\left(\epsilon_i^I - 1\right)^2 + \epsilon_i P_{\epsilon_i}^{dur}$.

Remark 1. Corollary 1 states that, adopting CWM, learner i's mis-detection probability is bounded by a number that depends only on the trade-off parameter α, on learner i's eccentricity ϵ_i and information eccentricity $\epsilon_i^I(\mathcal{G})$, and on the probability $P_{\epsilon_i}^{dur}$ that an event has a duration smaller than i's eccentricity $\epsilon_i(\mathcal{G})$. Since $\epsilon_i^I(\mathcal{G}) \leq \epsilon_i(\mathcal{G})$, if the designer knows an upper bound for $P_{\epsilon_i}^{dur}$ (e.g., in the KDDCUP'99 network intrusion data set [13] the typical duration of an attack is in the order of hundredths of time slots), then he can set the parameter α in order to guarantee a certain level of accuracy.

Notice that the lower α the lower the worst-case mis-detection probability, in the extreme case $\alpha = 0$ we have that $\mathbf{B}_i^{mis}(\mathbf{e}_N) \to P_{\epsilon_i}^{dur}$. However, α has the opposite impact on the delay, as remarked by Theorem 2 and Corollary 2.

Theorem 2. *The delay $D_i(\mathbf{e}_N)$ of a learner i that adopts CWM is upper bounded by*

$$\mathbf{B}_i^{del}(\mathbf{e}_N) = \epsilon_i^I - 1 + \frac{N_{i,\epsilon_i^I}^{mis}}{N\alpha}(\epsilon_i - \epsilon_i^I) . \qquad (13)$$

Proof: See Appendix C. ∎

Theorem 2 asserts that there exists an upper bound for the delay of a learner i that adopts our scheme. Such bound increases in $\epsilon_i^I(\mathcal{G})$, $\epsilon_i(\mathcal{G})$, and $N_{i,\epsilon_i^I}^{mis}$, whereas it decreases in α and N. Notice that when $\alpha \to \infty$ (i.e., we only care about the delay) the upper bound $\mathbf{B}_i^{del}(\mathbf{e}_N)$ approaches $\epsilon_i^I - 1$. Indeed, without any further assumption, the delay bound $\mathbf{B}_i^{del}(\mathbf{e}_N)$ can never be smaller than $\epsilon_i^I - 1$ for the following reason. The delay defined in Eq. (5) is the average of the detection delay, and the detection delay is defined only for detected events. We know that, by definition of ϵ_i^I, after a time equal to $\epsilon_i^I - 1$ learner i is potentially able to detect any event. However, it can also be that learner i is never able to detect any event before the time $\epsilon_i^I - 1$. In this case, which represents a worst case from the delay perspective, the delay cannot be lower than $\epsilon_i^I - 1$. As consequence, when $\alpha \to \infty$ the bound defined by Theorem 2 is strict.

Substituting (11) into (13) and computing the limit for $N \to +\infty$ we obtain the results stated in Corollary 2.

Corollary 2. $\lim_{N\to+\infty} \mathbf{B}_i^{del}(\mathbf{e}_N) = \epsilon_i^I - 1 + \frac{P_{\epsilon_i}^{dur}}{\alpha}(\epsilon_i - \epsilon_i^I)$.

Remark 2. Corollary 2 states that, adopting CWM, learner i's delay is bounded by a number that depends only on α, $\epsilon_i^I(\mathcal{G})$, and $P_{\epsilon_i}^{dur}$. By setting α the designer can decrease the delay to a desired value close enough to $\epsilon_i^I - 1$, but the worst case delay cannot be smaller than the time $\epsilon_i^I - 1$ needed to collect the local predictions required to detect all the events.

C. Robustness to communication errors

So far we have considered an idealized setting in which the learners *always* receive the transmission sent by their neighbors. In a distributed environment one cannot expect that this is always true: sometimes communications can fail, e.g., due to temporarily broken links. In this subsection we discuss how to deal with this issue and we extend the bounds derived in the preceding subsection to this scenario.

When communications are affected by errors it is important to define which information $\overline{s}_{ij}^n$ learner i uses whenever it does not receive learner j's local prediction in time instant n. It is natural to define $\overline{s}_{ij}^n$ as learner j's most recent local prediction possessed by learner i by time instant n. Formally, if learner i does not receive any local prediction from learner j at time instant n, it sets $\overline{s}_{ij}^n = \overline{s}_{ij}^{n-1}$. This means that Eq. (1) is not valid anymore, in the general case we have $\overline{s}_{ii}^n = s_i^n$ and $\overline{s}_{ij}^n = s_j^m$, for some $m \leq n - d_{ij} + 1$, $j \neq i$. In fact, when communications can fail, local predictions spread in the network with a rate lower than one link per time slot.

Learner i still uses the rules (8) and (10) to update its state, the rule (7) to select the cluster level, and the rule (2) to compute the final prediction.

In the following, we generalize Theorems 1 and 2 in case of failed communications, considering the *expected* mis-detection probability $P_i(\mathbf{e}_N)$ and delay $D_i(\mathbf{e}_N)$, where the expectation is taken with respect to the process that regulates the communication among learners. We consider the following communication model. We denote by p_{err} the probability that the transmission over a link fails.[7] We assume that the process that regulates the failure over a link is independent from the processes that regulate the failures over the other links and from the process that generates the events.

Theorem 3. *The expected mis-detection probability $P_i(\mathbf{e}_N)$ of a learner i that adopts CWM is upper bounded by*

$$\mathbf{B}_i^{mis}(\mathbf{e}_N) = \alpha\left(\epsilon_i^I - 1\right)^2 + \epsilon_i \frac{N_{\epsilon_i}^{dur}}{N} + \frac{\epsilon_i^I - 1}{N} + \\ + \frac{\epsilon_i}{N}\left[K(K+1) + \overline{Z}\right] + 1 - (1-p_{err})^{K-1} . \quad (14)$$

Proof: See Appendix D. ∎

Corollary 3.

$$\lim_{N\to+\infty} \mathbf{B}_i^{mis}(\mathbf{e}_N) = \alpha\left(\epsilon_i^I - 1\right)^2 + \epsilon_i P_{\epsilon_i}^{dur} + 1 - (1-p_{err})^{K-1} .$$

Remark 3. The term $1 - (1-p_{err})^{K-1}$ can be interpreted as the accuracy loss due to communication errors. This term tends to 0 if $p_{err} \to 0$, meaning that a little deviation from the ideal setting does not affect considerably the accuracy of the scheme. Notice that the bound (14) is independent from the durations of the events because we derived it in the worst-case scenario in which a single communication error in the first ϵ_i communication slots results in a mis-detection. However, if the durations of the events are typically much longer than ϵ_i then an event can be detected even if many communication errors occur, and the resulting expected mis-detection probability $P_i(\mathbf{e}_N)$ can be much smaller than $\mathbf{B}_i^{mis}(\mathbf{e}_N)$.

[7] Notice that the results in Theorems 3 and 4 can be generalized for link-dependent failure probabilities; however, this does not add any important insight.

Theorem 4. *The expected delay $D_i(\mathbf{e}_N)$ of a learner i that adopts CWM is upper bounded by*

$$\mathbf{B}_i^{del}(\mathbf{e}_N) = \epsilon_i^I - 1 + \frac{N_{i,\epsilon_i^I}^{mis}}{N\alpha}(\epsilon_i - \epsilon_i^I) + (K-1)\frac{p_{err}}{1-p_{err}} .$$

Proof: See Appendix E. ∎

Corollary 4.

$$\lim_{N\to+\infty} \mathbf{B}_i^{del}(\mathbf{e}_N) = \epsilon_i^I - 1 + \frac{P_{\epsilon_i}^{dur}}{\alpha}(\epsilon_i - \epsilon_i^I) + (K-1)\frac{p_{err}}{1-p_{err}} .$$

Remark 4. The term $(K-1)p_{err}/(1-p_{err})$ can be interpreted as the delay loss due to communication errors. This term tends to 0 if $p_{err} \to 0$, meaning that a little deviation from the ideal setting does not affect considerably the delay of the scheme.

VI. Simulations

In this section we present several illustrative results aimed to evaluate the efficacy of the proposed approach and to understand the impact of the network topology and of the system parameters in the learning problem. In Subsection VI-A we generate artificial data sets departing from a scenario that has been considered in some estimation works [17]. Artificial data sets allow us to control some system parameters (e.g., the intensity of the observed noise) and to understand the impact of such parameters in the learning problem. Finally, in Subsection VI-B we consider a real data set – the network intrusion data set used for the KDD Cup 1999 [13] – and we compare the performance of our scheme with the performance achieved by the weighted majority scheme [10] and by the combine then adapt scheme [15].

A. Artificial data set: label embedded in Gaussian noise

We consider the following artificial data set. We generate N_{eve} events such that 1) the duration of an event follows a geometric distribution with parameter p_{dur}; 2) the label y_ℓ of an event is generated with probabilities $P[y_\ell = -1] = P[y_\ell = 1] = 0.5$; 3) the feedback of an event is received with probability p_{fed} and (if received) with a delay from the end of the event that follows a geometric distribution with parameter p_{fed}. Similarly to some estimation works [17], we assume that each learner i observes the label embedded in an additive Gaussian noise with zero mean and variance equal to σ, i.e., $x_{i,\ell} = y_\ell + q_{i,\ell}$, $q_{i,\ell} \sim \mathcal{N}(0,\sigma)$, and the noise terms $q_{i,\ell}$ are independent across different learners and across different events.

For each learner i we consider a static local classifier that generates the following local prediction: $s_i^n = 1$ if $x_i^n \geq 0$, and $s_i^n = -1$ if $x_i^n < 0$. Notice that this represents the most accurate local prediction given the statistic of the considered noise. The use of static well-trained local classifiers allows us to focus on the advantages of our scheme and on the learning aspect of our scheme, instead of on the learning aspect of incremental classifiers.

Parameter description	Symbol	Value
Number of events simulated	N_{eve}	10^3
Probability label is 1	$P[y_\ell] = 1$	0.5
Geometric distribution parameter for the event duration	p_{dur}	0.1
Geometric distribution parameter for the feedback delay	p_{fed}	0.1
Standard deviation of the noise	σ	2
Tradeoff parameter	α	0.1
Probability that a communication fails	p_{err}	0
Probability that a feedback is not received	p_{fed}	0
Number of simulation runs	N_{sym}	50

TABLE I: Basic Simulation Settings.

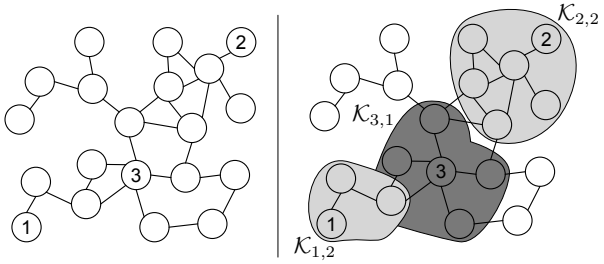

Fig. 5: Randomly generated network topology. The right side represents the clusters adopted by learners 1, 2 and 3 when $\sigma = 1$.

We adopt the basic simulation settings represented in Table I and we perform several series of simulations. In each simulation we vary the value of a single parameter within a specific range, and for each value of the parameter we average the results over $N_{sym} = 50$ simulation runs.

To get some insights on how the performance of the individual learners adopting the CWM scheme depend on their positions in the network, we first consider the randomly generated network topology represented in the left side of Fig. 5, consisting of $K = 20$ learners, and we focus our attention on the learners marked as 1, 2, and 3. Fig. 6 shows the cost associated to each learner (left side) and the cluster level selected by each learner at the end of the simulation (right side), varying the value σ of the standard deviation of the noise. When the noise is very low, all the three learners adopt a cluster level $\underline{d}_i^n = 1$, i.e., they aggregate only the local predictions of their neighbors, this is enough for them to detect almost all the events instantaneously. With the increase of σ learners 1 and 2, which are located at the border of the network and have only a single neighbor, need to increase the values of the cluster levels they select to be able to detect the events accurately, this results in larger delays and costs. In fact, for large values of σ learner 3, which is located in a central location and surrounded by many learners, achieves a substantially lower cost than learner 1 and 2. Notice that learner 2 has a single neighbor as learner 1, but 2's neighbor is better connected than 1's neighbor, and this allows learner 2 to achieve typically better performance than learner 1. An exception to this is encountered for the range of values $\sigma \in [0.4, 1.1]$, in which learner 1's cost is lower than learner 2's cost. The reason behind this counter-intuitive result is the following. When learner 2 adopts a cluster level $\underline{d}_2^n = 2$ – see for example the right side of Fig. 5 – four among the six local predictions it aggregates belong to learners that are two hops away. As a consequence, the final prediction is mainly driven by these learners and the prediction delay increases consistently. On the other hand, if learner 1 adopts a cluster level $\underline{d}_1^n = 2$ only one among the three local predictions that it aggregates belongs to a learner that

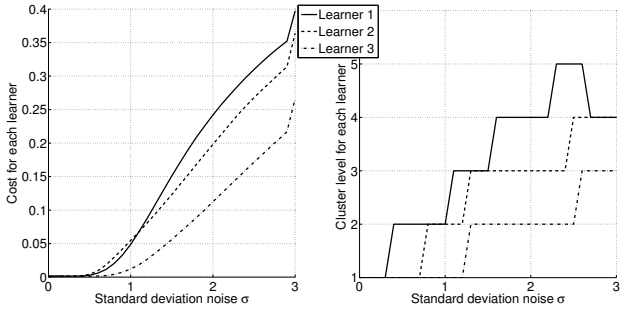

Fig. 6: Cost and cluster level for learners 1, 2, and 3 vs. the standard deviation of the noise σ for the random network topology.

is two hops away. This allows learner 1 to slightly increase its detection accuracy without increasing consistently its prediction delay. Indeed, this is also the reason why learner 1's optimal cluster level is $\underline{d}_1^n = 2$ starting from $\sigma = 0.4$, whereas learner 2 starts adopting $\underline{d}_2^n = 2$ only when $\sigma = 0.8$. Another interesting thing to remark is that $\underline{d}_1^n$ decreases by one unit – from $\underline{d}_1^n = 5$ to $\underline{d}_1^n = 4$ – when σ is larger than 2.6. This is due to the fact that, when the noise is very large, even with the cluster level of $\underline{d}_1^n = 5$ learner 1 is subject to many mis-detections, hence it is convenient for him to adopt a lower cluster level and improve its delay.

Next, we investigate how far are the worst case analytical bounds derived in Subsection V-B with respect to the empirical performance achieved by learner 3. For this type of analysis we increase the number of simulated events to $N_{eve} = 10^5$, we consider only the simulation runs satisfying the realizability assumption, and in order to have enough realizable cases we decrease the standard deviation of the noise associated to learner 1 to $\sigma = 0.2$ (i.e., the local predictions of learner 1 agree with the labels most of the times). As a consequence, in most of the simulation runs the information eccentricity of learner 3 is equal to the distance between learners 3 and 1, i.e., $\epsilon_3^I = 3$; whereas learner 3's eccentricity is $\epsilon_3 = 4$. Fig. 7 shows that the mis-detection probability bound is close to the real mis-detection probability experienced by learner 3 when $\alpha = 0$, i.e., when the only goal is to detect the events accurately. As soon as α increases, the mis-detection probability bound diverges from the real mis-detection probability. As for the prediction delay, for a small α the delay experienced by learner 3 is slightly larger than one time slot, whereas when α increases (i.e., detecting events in a timely manner becomes more important) the prediction delay improves at the expense of the detection accuracy. Instead, according to Eq. (13), the prediction delay bound is always larger than $\epsilon_3^I - 1 = 2$. It is not surprising that the bounds are in general not strict, indeed we remark that the bounds derived in Subsection V-B are worst-case bounds that are valid for any sequence of events and local predictions; hence, it is reasonable that, on average, the proposed scheme performs better that what the bounds suggest.

Now we investigate how the average cost of the random network – defined as the sum of the costs of all learners divided by the number of learners – varies with respect to σ. We compare the cost obtained by our scheme with the costs achieved by two benchmark schemes: in the *no aggregation* scheme the learners do not exchange and aggregate information, they only rely on their local predictions, whereas in the *fixed cluster* scheme the learners always aggregate all the local predictions, i.e., $\underline{d}_i^n = \epsilon_i$,

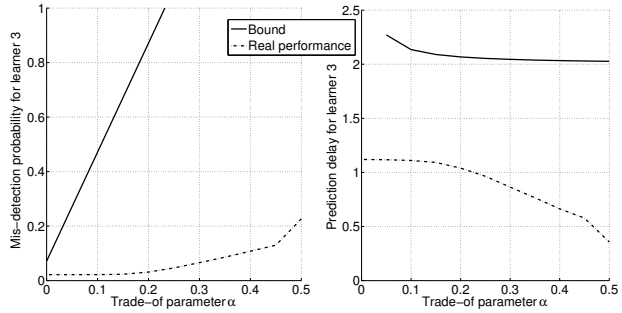

Fig. 7: Mis-detection probability and prediction delay for learner 3 vs. the trade-off parameter α.

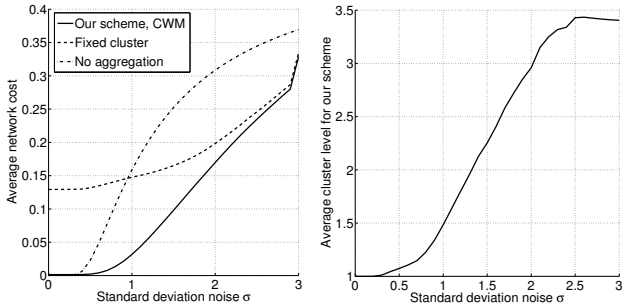

Fig. 8: Average network cost and cluster level vs. the standard deviation of the noise σ for the random network topology.

$\forall\, i, n$, and adopt (10) to adapt the unique weight vector $\mathbf{w}^n_{i,\epsilon_i}$. The left side of Fig. 8 shows that for very low values of σ the costs of CWM and of the no aggregation scheme are equal to 0. Indeed, in this range of values the learners can detect correctly all the events relying simply on their local predictions, and this allows them to detect the events instantaneously. The right side of Fig. 8, which represents the average cluster level adopted by the learners for the CWM scheme, shows that for very low values of σ the learners automatically learn that the optimal strategy is to aggregate only the local predictions of their neighbors. Differently from CWM and the no aggregation scheme, for very low σ the fixed cluster scheme has a non-zero cost because the learners are not always able to detect the events instantaneously. On the other hand, with the increase of σ it is necessary to aggregate many local predictions to detect accurately the events, and the costs of CWM and of the fixed aggregation scheme converge. In summary, the proposed scheme is able to successfully adapt to the unknown characteristics of the noise: when σ is low CWM gives the advantage of the no aggregation scheme in terms of prediction delay, when σ is large CWM provides the benefit of the fixed aggregation scheme in terms of detection accuracy, and for intermediate values of σ CWM outperforms both benchmarks. Finally, notice that for large values of σ the average cluster level of CWM stabilizes to a value of about 3.5, which is substantially lower than the average eccentricity of the network, which is 5.65. This implies that it is useless to propagate all the local predictions through the whole network, because local predictions from very far learners are never aggregated. Hence, to limit the complexity of the proposed scheme – in terms of number of clusters that each learner has to consider and of information amount to spread in the network – without incurring any cost in terms of performance, the local predictions can be forwarded only for a finite number of links.

In order to study the impact of the network topology on the performance of CWM, in the following series of simulation

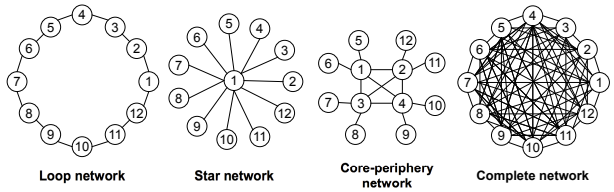

Fig. 9: Loop, star, core-periphery, and complete network topologies.

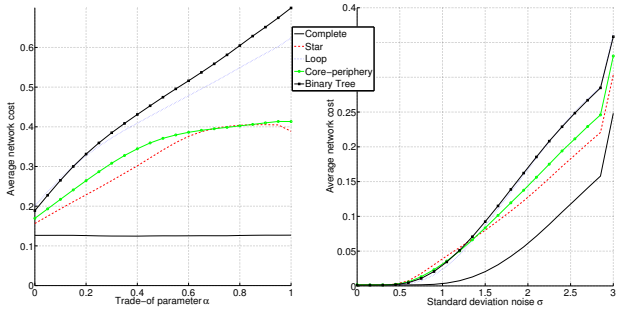

Fig. 10: Average network cost for different network topologies vs. the standard deviation of the noise σ and the trade-off parameter α.

we consider $K = 12$ learners that are connected through the loop, star, core-periphery, and complete network topologies represented in Fig. 9, and through the binary tree topology obtained by removing learners 13, 14 and 15 from Fig. 2.

We first analyze how the tradeoff parameter α affects the average network cost. We consider a range of α from 0 to 1. The left side of Fig. 10 shows that the mis-detection probability and the delay of the complete network topology are constant with respect to α. Indeed, in the complete topology all the information is available instantaneously to all learners. As a consequence, learners have a lot of information to combine to filter the noise and achieve large detection accuracies, and at the same time events are always detected immediately (if detected). The figure shows also that there exists an ordering among the networks in terms of performance: the networks that have small diameters (complete, star, and core-periphery) achieve lower cost than the networks that have large diameters (loop and binary tree). However, this is not always true and interesting trade-offs exist, as we see in the next simulations.

Now we investigate the impact of the standard deviation of the noise σ. The right side of Fig. 10 shows how the network performance vary from $\sigma = 0$ to $\sigma = 3$ for different network topologies. Interestingly, there is a range of values of σ such that the loop and tree topologies achieve lower costs than the core-periphery and star topologies. In this range the learners of the loop and binary tree topologies adopt often the cluster level $\underline{d}^n_i = 1$ because the noise is enough low to allow for the detection of almost all the events by aggregating a total of three local predictions (all learners of the loop topology have two neighbors and half of the learners of the binary tree topology have at least two neighbors). On the other hand, the noise is not low enough to allow the learners to detect accurately all the events by aggregating only two predictions. Hence, most of the learners in the core-periphery and star topologies either 1) select $\underline{d}^n_i = 1$ obtaining a lower accuracy than the learners in the loop and binary tree topologies; or 2) they select $\underline{d}^n_i > 1$ obtaining larger delays than the learners in the loop and binary tree topologies.

Next, we investigate how the average network cost varies with

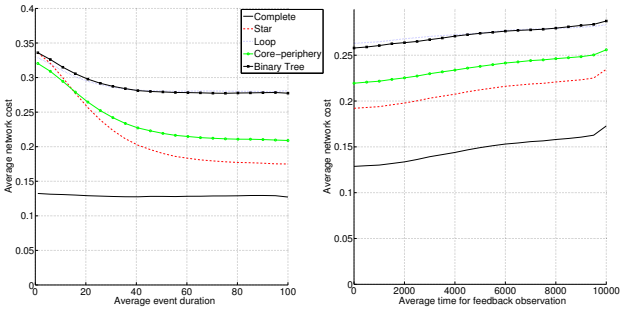

Fig. 11: Average network cost for different network topologies vs. the average event duration $1/p_{dur}$ and the average time to observe a feedback $1/p_{fed}$.

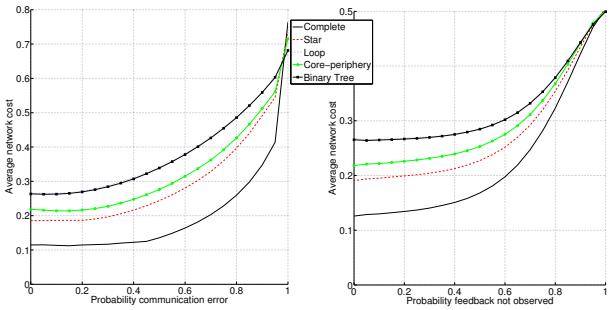

Fig. 12: Average network cost for different network topologies vs. the probability that a communication fails p_{err} and the probability that a feedback is not received p_{fed}.

respect to the average event duration $1/p_{dur}$ and to the average time to observe a feedback $1/p_{fed}$. The left side of Fig. 11 shows that it is difficult to detect very short events. Indeed, in this case the learners must use small cluster levels (otherwise the events end before the required local predictions are collected), and as a consequence they cannot accurately filter out the noise. However, the network costs stabilize starting from an event duration of some tenths of time slots, this means that the event duration is not an important factor as long as it is not extremely low. The right side of Fig. 11 shows that the performance of the network decreases very slowly as the average time to observe a feedback increases. This implies that the learners adopting the CWM scheme learn fast, they require few feedbacks to learn an accurate detection model.

Finally, we investigate the impact on the average network cost of the probability p_{err} that a communication fails and of the probability p_{fed} that a feedback is not received. The left side of Fig. 12 shows that p_{err} affects substantially the network costs if it is large, but the trends of the network costs are flat (for all the considered topologies) for a range of p_{err} from 0 to 0.2. In most applications, it is unreasonable to have larger communication error probabilities. Similar considerations are valid for the right side of Fig. 12, which shows that the network cost is very flat for a very large range of p_{fed}. This confirms that the proposed CWM scheme allows the learners to learn accurately even with few feedbacks.

B. Real data set: the KDDCUP'99 data set

In this subsection we exploit the network intrusion data set used for the KDD Cup 1999 and available in the UCI archive [28]. This data set is widely used in the stream mining literature [13], [29]. It consists of a series of TCP connection records, labeled either as normal connections or as attacks. For a more

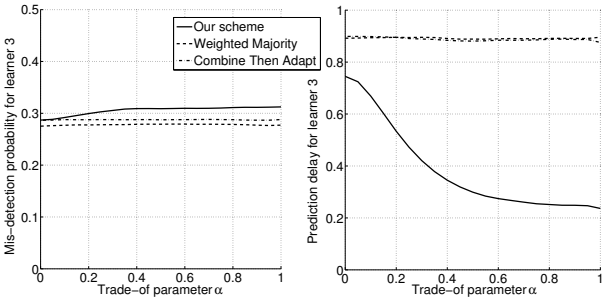

Fig. 13: Mis-detection probability and prediction delay for learner 3 vs. the trade-off parameter α.

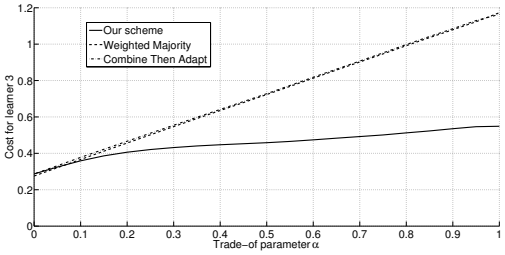

Fig. 14: Cost for learner 3 vs. the trade-off parameter α.

detailed description of the data set we refer the reader to [29].

As described in the example at the end of Section III, this data set refers to a local communication network and is very suitable for the framework considered in this manuscript. However, to the best of our knowledge, the KDDCUP'99 data set has been released without any information about the underlying network topology of the local communication network. For this reason we enrich such a data set with an associated network topology. Specifically, we consider again the randomly generated network topology represented in the left side of Fig. 5. We split each instance of the original data set into three non-overlapping sub-instances, these sub-instances represent the observations of learners 1, 2 and 3. These three learners adopt logistic regression classifiers to generate the local predictions, such classifiers are trained using the first $10,000$ instances of the KDDCUP'99 data set. The other learners generate random local predictions. As a consequence, only the local predictions of leaners 1, 2, and 3 contain useful information to detect network attacks.

We investigate the performance that learner 3 achieves adopting our scheme, the weighted majority scheme [10], and the combine then adapt scheme [15], varying the value of the trade-off parameter α. The left side of Fig. 13 shows that the three schemes achieve comparable accuracies for $\alpha = 0$, i.e., when the prediction delay is not important. The value of α does not have any impact on the mis-detection probability (left side of Fig. 13) and on the prediction delay (right side of Fig. 13) of the weighted majority and combine then adapt schemes; indeed, the goal of these two schemes is to estimate the optimal weights to detect the events, without taking into account the timeliness of the detection. Instead, Fig. 13 proves that our scheme is able to trade-of the prediction accuracy and the prediction delay: with the increase of α, the prediction accuracy of our scheme gets worse, whereas the prediction delay improves. To quantify the advantages of our scheme, Fig. 14 shows the cost incurred by learner 3 with the threes schemes. When α is 0 the costs of the three schemes are comparable, whereas when α increases (i.e., detecting events in a timely manner becomes more important) the cost associated to our scheme diverges from the cost associated

to the other two benchmark schemes. This proves the efficacy of the proposed approach in scenarios in which timely detections are required.

VII. Conclusions and Future Work

In this paper we analyze the problem of how a set of distributed learners that are interconnected over a network can cooperate to detect in a timely manner events of interests. We propose a scheme in which the learners exchange local predictions and each learner selects a cluster of surrounding learners and aggregates the local predictions belonging to that cluster using a weighted majority rule. We propose online learning rules to learn the weight vector for each cluster and the cluster to adopt. In the realizable case we determine performance guarantees for the proposed scheme. Numerical simulations show that the proposed scheme is able to successfully adapt to the unknown characteristics of the data streams and can achieve substantial performance gain with respect to a scheme in which the learners act individually and with respect to a scheme in which the learners always aggregate all available local predictions.

We conclude with a discussion on open research problems that are worth investigating. First, the scalability of the proposed scheme needs to be investigated and modifications may be required in order to cope with the scalability issues of large steam mining applications. For example, in order to limit the amount of exchanged and stored information, the cluster level may be learned during a training period instead of being adapted continuously. Second, in this work we did not investigate specific issues that arise when the network topology varies continuously in time. For example, the learners need to recompute the shortest paths that connect them to the other learners – which can be computationally expensive for large networks – and they need to learn again the optimal cluster levels and weights, either from scratch or exploiting the parameters learned in the old network. Third, in addition to the robustness to communication errors, other types of robustness can be analyzed. For example, the robustness of the scheme to imperfect feedback (i.e., erroneous feedback or not received feedback) represents an interesting area for future research.

Appendix A
Proof of Lemma 1

For each $d \geq \epsilon_i^I$, assumptions **A1** and **A2** and the information eccentricity definition imply that there exists a non-empty class of weight vectors

$$\mathcal{W}_d = \{\mathbf{w}_{i,d} : \text{sgn}(\mathbf{w}_{i,d} \cdot \mathbf{s}_{i,d,\ell}) = y_\ell, \forall \text{ events } e_\ell\}\,.$$

The local predictions $s_{ij,\ell} \in \{-1,+1\}$, $\forall\, j \in \mathcal{K}_{i,d}$, represent a particular vertex of an hypercube in $\Re^{K_{i,d}}$. The class $\mathcal{W}_d$ defines a class of hyperplanes $\mathcal{H}(\mathcal{V}_{-1},\mathcal{V}_{+1})$ 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. Denote by $\gamma(H)$ the minimum distance between the hyperplane H and the vertexes of the hypercube. We want to compute

$$\gamma \triangleq \min_{\mathcal{V}_{-1},\mathcal{V}_{+1}} \max_{H \in \mathcal{H}(\mathcal{V}_{-1},\mathcal{V}_{+1})} \gamma(H)\,.$$

It is easy to see that the sets $\mathcal{V}_{-1}$ and $\mathcal{V}_{+1}$ that minimize the resulting γ are such that one vertex must be separated by all the vertexes it is connected with through an edge, and the separating hyperplane that maximizes γ 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 $\mathbf{v} \in 0, 2^{K_{i,d}}$, and we want to find the hyperplane defined by the parameters $(a_0,\ldots,a_{K_{i,d}})$ which separates with the largest margin the vertex $\mathbf{v}^{(0)} = (0,\ldots,0)$ from the vertexes $\mathbf{v}^{(j)} = (0,\ldots,0,2,0,\ldots,0)$ having 2 in position j, $j = 1,\ldots,K_{i,d}$. Imposing that the signed distance between $\mathbf{v}^{(0)}$ and the separating hyperplane is the opposite of the signed distance between $\mathbf{v}^{(j)}$ and the separating hyperplane, we obtain

$$\frac{a_0 + v_1^{(0)}a_1 + \ldots + v_{K_{i,d}}^{(0)}a_{K_{i,d}}}{\sqrt{\sum_{i=1}^{K_{i,d}} a_i^2}} = -\frac{a_0 + v_1^{(j)}a_1 + \ldots + v_{K_{i,d}}^{(j)}a_{K_{i,d}}}{\sqrt{\sum_{i=1}^{K_{i,d}} a_i^2}}$$
$$\rightarrow a_0 = -a_j\,.$$

Repeating the same procedure for every vertex $\mathbf{v}^{(j)}$, $j = 1,\ldots,K_{i,d}$, we obtain that the best separating hyperplane must satisfy $a_0 = -a_j$, $\forall\, j$, hence the distance between it and $\mathbf{v}^{(0)}$ is $\frac{|a_0|}{\sqrt{K_{i,d}a_0^2}} = \frac{1}{\sqrt{K_{i,d}}}$.

The above means that it is always possible to find a unit length weight vector $\mathbf{u}$ which separates the local prediction vectors which a margin of at least $\frac{1}{\sqrt{K_{i,d}}}$, i.e., such that $\|\mathbf{u}\| = 1$ and $y_\ell \mathbf{u} \cdot \mathbf{s}_{i,d,\ell} \geq \frac{1}{\sqrt{K_{i,d}}}$. Let $\mathbf{w}_{i,d}^{(n)}$ be learner i's level-d weight vector before the n-th update, and assume for the moment that the feedbacks are received at the end of an event, i.e., $Z_{i,\ell} = T_\ell$. We have

$$\mathbf{w}_{i,d}^{(n+1)} \cdot \mathbf{u} = \mathbf{w}_{i,d}^{(n)} \cdot \mathbf{u} + y_\ell \mathbf{s}_{i,d,\ell}\mathbf{u} \geq \mathbf{w}_{i,d}^{(n)} \cdot \mathbf{u} + \frac{1}{\sqrt{K_{i,d}}}\,.$$

By applying straightforward inductive argument we have

$$\mathbf{w}_{i,d}^{(n+1)} \cdot \mathbf{u} \geq \frac{n}{\sqrt{K_{i,d}}}\,. \tag{15}$$

We also have $\|\mathbf{w}_{i,d}^{(n+1)}\|^2 = \|\mathbf{w}_{i,d}^{(n)} + y_\ell \mathbf{s}_{i,d,\ell}\|^2 = \|\mathbf{w}_{i,d}^{(n)}\|^2 + \|\mathbf{s}_{i,d,\ell}\|^2 + 2y_\ell \mathbf{s}_{i,d,\ell} \cdot \mathbf{w}_{i,d}^{(n)} \leq \|\mathbf{w}_{i,d}^{(n)}\|^2 + \|\mathbf{s}_{i,d,\ell}\|^2 = \|\mathbf{w}_{i,d}^{(n)}\|^2 + K_{i,d} + 1$, where the inequality is valid because during an update we must have $\text{sgn}(y_\ell \mathbf{s}_{i,d,\ell}) \neq y_\ell$.

By applying straightforward inductive argument we have

$$\|\mathbf{w}_{i,d}^{(n+1)}\|^2 \leq n\,(K_{i,d}+1)\,. \tag{16}$$

Putting together (15) and (16) we have

$$\sqrt{n\,(K_{i,d}+1)} \geq \|\mathbf{w}_{i,d}^{(n+1)}\| \geq \mathbf{w}_{i,d}^{(n+1)} \cdot \mathbf{u} \geq \frac{n}{\sqrt{K_{i,d}}}\,.$$

which implies $n \leq K_{i,d}\,(K_{i,d}+1)$.

This means that after a maximum of $K_{i,d}\,(K_{i,d}+1)$ updates learner i's level-d weight vector reaches a state that allows learner i to detect all the events. Notice that the number of updates is equal to the number of mis-detections if all the events have a duration larger than d and if the feedbacks are received immediately after the events end. Since this is not valid, we have to add to the number of updates 1) N_d^{dur}, which represents the maximum number of additional errors that the system can make because of the events that are too short to be detected with the cluster level d, and 2) $\left\lfloor \frac{\overline{Z} - T_{min}}{T_{min}} \right\rfloor$, which represents the maximum

number of additional mis-detections from the $K_{i,d}\,(K_{i,d}+1)$-th mis-detection to the $K_{i,d}\,(K_{i,d}+1)$-th update. Finally, we obtain $N_{i,d}^{mis} \leq K_{i,d}\,(K_{i,d}+1) + N_d^{dur} + \left\lfloor \dfrac{\overline{Z}-d}{d} \right\rfloor$.

Appendix B
Proof of Theorem 1

Let $\tilde{N}_{i,d}^{mis}$ be the number of errors made using the cluster level d only in those time slots in which our scheme selects the cluster level d. Obviously, $\tilde{N}_{i,d}^{mis} \leq N_{i,d}^{mis}$, hence the right side of (11) is an upper bound of $\tilde{N}_{i,d}^{mis}$ for $d \geq \epsilon_i^I(\mathcal{G})$.

$\tilde{N}_{i,d}^{mis}$ is also upper bounded by the number of times $N_{i,d}^{sel}$ learner i selects the cluster level d. We must have

$$N_{i,d}^{sel} \leq N_{i,\epsilon_i^I}^{mis} + N\alpha\left(\epsilon_i^I - 1\right) + 1\,. \tag{17}$$

Indeed, assume for absurd that (17) is not valid. Let n denotes the last time instant the cluster level d is selected. Since (17) is not valid we must have

$$N_i^{fed,n} - N_{i,d}^{det,n} > N_i^{fed,n} - N_{i,\epsilon_i^I}^{det,n} + N\alpha\left(\epsilon_i^I - 1\right)\,. \tag{18}$$

However, since $T_{i,d}^n \geq 0$ and $T_{i,\epsilon_i^I}^n \leq N\left(\epsilon_i^I - 1\right)$, (18) implies that $\hat{C}_{i,\epsilon_i^I}^n < \hat{C}_{i,d}^n$, and this contradicts the initial assumption that d is selected as the cluster level. Hence, (17) is valid.

Finally, since $K_{i,d} \leq K$, $N_d^{dur} \leq N_{\epsilon_i}^{dur}$, and $\left\lfloor \frac{\overline{Z}-d}{d} \right\rfloor < \overline{Z}$, $\forall\, d$, we obtain

$$\begin{aligned}
\tilde{N}_i^{mis} &= \sum_{d=1}^{\epsilon_i^I - 1} \tilde{N}_{i,d}^{mis} + \sum_{d=\epsilon_i^I}^{\epsilon_i} \tilde{N}_{i,d}^{mis} \leq \\
&\leq \left(\epsilon_i^I - 1\right)\left[K\,(K+1) + N_{\epsilon_i}^{dur} + \overline{Z} + N\alpha\left(\epsilon_i^I - 1\right) + 1\right] + \\
&+ \left(\epsilon_i - \epsilon_i^I + 1\right)\left[K(K+1) + N_{\epsilon_i}^{dur} + \overline{Z}\right] = \\
&= N\alpha\left(\epsilon_i^I - 1\right)^2 + \epsilon_i N_{\epsilon_i}^{dur} + \epsilon_i^I - 1 + \epsilon_i\left[K(K+1) + \overline{Z}\right]\,.
\end{aligned}$$

Dividing the above by N we obtain (12).

Appendix C
Proof of Theorem 2

We denote by $\overline{T}_{i,d}^n$ learner i's average detection delay associated to the selection of the cluster level d until time instant n. We have $\overline{T}_{i,d}^n \leq d-1$, because after $d-1$ time slots learner i receives all the local predictions from the learners belonging to the cluster $\mathcal{K}_{i,d}(\mathcal{G})$.

Let n be a time instant in which i selects the cluster level $d > \epsilon_i^I$. This implies that

$$\frac{N_{i,d}^{mis}}{n} + \alpha\overline{T}_{i,d}^n \leq \frac{N_{i,\epsilon_i^I}^{mis}}{n} + \alpha\overline{T}_{i,\epsilon_i^I}^n\,.$$

Since $\frac{N_{i,d}^{mis}}{n} \geq 0$ and $\overline{T}_{i,\epsilon_i^I}^n \leq \epsilon_i^I - 1$, we have $\overline{T}_{i,d}^n \leq \frac{N_{i,\epsilon_i^I}^{mis}}{n} + \alpha(\epsilon_i^I - 1)$ that implies $n \leq \frac{N_{i,\epsilon_i^I}^{mis}}{\alpha(\overline{T}_{i,d}^n - \epsilon_i^I - 1)}$. As a consequence, denoting by $N_{i,d}^{sel}$ the number of times that the cluster level d is selected up to time instant N, we have that $N_{i,d}^{sel}$ is bounded by $\frac{N_{i,\epsilon_i^I}^{mis}}{\alpha(\overline{T}_{i,d}^N - \epsilon_i^I - 1)}$, $\forall\, d > \epsilon_i^I$. Finally, we obtain:

$$\begin{aligned}
D_i(\mathbf{e}_N) &\leq \frac{1}{N}\left[\sum_{d<\epsilon_i^I} N_{i,d}^{sel}(d-1) + \sum_{d>\epsilon_i^I} N_{i,d}^{sel}\overline{T}_{i,d}^n\right. \\
&\left. + \left(N - \sum_{d<\epsilon_i^I} N_{i,d}^{sel} - \sum_{d>\epsilon_i^I} N_{i,d}^{sel}\right)\left(\epsilon_i^I - 1\right)\right] \\
&\leq \frac{1}{N}\left[\sum_{d>\epsilon_i^I} N_{i,d}^{sel}\overline{T}_{i,d}^n + \left(N - \sum_{d>\epsilon_i^I} N_{i,d}^{sel}\right)\left(\epsilon_i^I - 1\right)\right] \\
&\leq \frac{1}{N}\left[\sum_{d>\epsilon_i^I} \frac{N_{i,\epsilon_i^I}^{mis}\overline{T}_{i,d}^n}{\alpha(\overline{T}_{i,d}^N - \epsilon_i^I - 1)} + \left(N - \sum_{d>\epsilon_i^I} \frac{N_{i,\epsilon_i^I}^{mis}}{\alpha(\overline{T}_{i,d}^N - \epsilon_i^I - 1)}\right)\left(\epsilon_i^I - 1\right)\right] \\
&\leq \frac{1}{N}\left[\sum_{d>\epsilon_i^I} \frac{N_{i,\epsilon_i^I}^{mis}}{\alpha(\epsilon_i - \epsilon_i^I)}(\epsilon_i - 1) + \left(N - \sum_{d>\epsilon_i^I} \frac{N_{i,\epsilon_i^I}^{mis}}{\alpha(\epsilon_i - \epsilon_i^I)}\right)\left(\epsilon_i^I - 1\right)\right] \\
&= \frac{N_{i,\epsilon_i^I}^{mis}}{N\alpha}(\epsilon_i - 1) + \left(1 - \frac{N_{i,\epsilon_i^I}^{mis}}{N\alpha}\right)\left(\epsilon_i^I - 1\right) = \epsilon_i^I - 1 + \frac{N_{i,\epsilon_i^I}^{mis}}{N\alpha}\left(\epsilon_i - \epsilon_i^I\right),
\end{aligned}$$

where the first inequality is valide because $\overline{T}_{i,d}^n \leq d-1$, the second inequality is valid because $\sum_{d<\epsilon_i^I} N_{i,d}^{sel}(d-1) \leq \sum_{d<\epsilon_i^I} N_{i,d}^{sel}(\epsilon_i - 1)$, the third inequality is valid because the right hand side increases in $N_{i,d}^{sel}$ and $N_{i,d}^{sel} \leq \frac{N_{i,\epsilon_i^I}^{mis}}{\alpha(\overline{T}_{i,d}^N - \epsilon_i^I - 1)}$, and the forth inequality is valid because the right hand side increases in $\overline{T}_{i,d}^N$ and $\overline{T}_{i,d}^N \leq (\epsilon_i - 1)$.

Appendix D
Proof of Theorem 3

Theorem 1 shows that in absence of communication errors we have that $P_i(\mathbf{e}_N)$ is bounded by

$$\begin{aligned}
\mathbf{B}_i^{mis}(\mathbf{e}_N) = \alpha\left(\epsilon_i^I - 1\right)^2 + \epsilon_i\frac{N_{\epsilon_i}^{dur}}{N} + \frac{\epsilon_i^I - 1}{N} + \\
+ \frac{\epsilon_i}{N}\left[K(K+1) + \overline{Z}\right]\,.
\end{aligned} \tag{19}$$

The communication errors represents an additional source of mis-detection. We can focus on an event e_ℓ whose duration T_ℓ is equal or larger than ϵ_i, because the term $\frac{N_{\epsilon_i}^{dur}}{N}$ takes already into account the mis-detections caused by events whose duration is smaller than ϵ_i. Let A be the additional expected mis-detection contribution caused by failed communications. We have

$$\begin{aligned}
A &= P\left[i \text{ does not receive all local predictions after } T_\ell \text{ slots}\right] \\
&\leq P\left[i \text{ does not receive all local predictions after } \epsilon_i \text{ slots}\right] \\
&\leq P\left[\text{at least 1 communication error in } K-1 \text{ transmissions}\right] \\
&= 1 - P\left[\text{no communication errors in } K-1 \text{ transmissions}\right] \\
&= 1 - (1 - p_{err})^{K-1}\,,
\end{aligned}$$

where the first inequality is valid because $T_\ell \geq \epsilon_i$ and the second inequality is valid because for the following argument. Consider a spanning tree $\mathcal{G}^{SP}$ of the network $\mathcal{G}$. The probability that i does not receive all local predictions after ϵ_i time slots in $\mathcal{G}$ is equal to or lower than the probability that i does not receive all local predictions after ϵ_i time slots in $\mathcal{G}^{SP}$, because $\mathcal{G}$ contains some

redundant links that can protect from communication errors. Learner i receives all local predictions after ϵ_i time slots in $\mathcal{G}^{SP}$ if and only if: 1) at the first time slot all learners whose distance from i is ϵ_i transmit correctly their information to the learners whose distance from i is ϵ_i; 2) 1) at the second time slot all learners whose distance from i is $\epsilon_i - 1$ transmit correctly their information to the learners whose distance from i is $\epsilon_i - 2$; ... at the ϵ_i-th time slot all learners whose distance from i is 1 transmit correctly their information to i. Since the network $\mathcal{G}^{SP}$ has $K-1$ links, learner i receives all local predictions after ϵ_i time slots in $\mathcal{G}^{SP}$ if and only no communication error occurs in $K-1$ transmissions.

We conclude the proof by adding the upper bound of A to the right hand side of (19).

Appendix E
Proof of Theorem 4

Theorem 2 shows that in absence of communication errors we have that $D_i(\mathbf{e}_N)$ is bounded by

$$\mathbf{B}_i^{del}(\mathbf{e}_N) = \epsilon_i^I - 1 + \frac{N_{i,\epsilon_i^I}^{mis}}{N\alpha}(\epsilon_i - \epsilon_i^I)\,. \tag{20}$$

The communication errors increase such delay bound because learner i may require more time to come into possession of the local predictions it needs to detect the event. To bound such additional expected time T^{com} due to the communication errors, let consider the worst network possible in terms of time required to receive the local predictions from all the learners. We consider the network in which learners are connected in line and learner i is located at an extreme. For simplicity of exposition, we rename learner i into learner 1 and we order the learner sequentially (i.e., learner 2 is neighbor of 1, 3 is neighbor of 2, and so on). In absence of communication errors learner 1 requires $K-1$ time slots to receive the local predictions from all the other learners. With communication errors the time required to receive all local prediction is equal to 1) the time required by K to send the information containing it local prediction to learner $K-1$; plus 2) the time required by $K-1$ to send the information containing K's local prediction (and, if not already sent, $K-1$'s local prediction) to learner $K-2$; plus ... plus the time required by 2 to send K's local prediction (and, if not already sent, the local prediction of other learners in the cascade) to 1. Such time is the sum of $K-1$ i.i.d. geometric random variable with parameter $(1-p_{err})$. Hence, the expected additional time is:

$$T^{com} \leq (K-1)\frac{1}{1-p_{err}} - (K-1) = (K-1)\frac{p_{err}}{1-p_{err}}\,.$$

We conclude the proof by adding the upper bound of T^{com} to the right hand side of (20).

References

[1] "IBM Smarter Planet," http://www.ibm.com/smarterplanet, retrieved October, 2012.

[2] L. Canzian and M. van der Schaar, "Real-time stream mining: online knowledge extraction using classifier networks," *to appear in IEEE Network Magazine - Special Issue on Networking for Big Data*, 2015.

[3] Y. Zhang, D. Sow, D. S. Turaga, and M. van der Schaar, "A fast online learning algorithm for distributed mining of bigdata," in *The Big Data Analytics Workshop at SIGMETRICS*, 2013.

[4] M. Antunes, M. A. Turkman, and K. F. Turkman, "A Bayesian approach to event prediction," *Journal of Time Series Analysis*, vol. 24, no. 6, pp. 631–646, 2003.

[5] G. Amodeo, R. Blanco, and U. Brefeld, "Hybrid models for future event prediction," in *Proc. ACM CIKM*, 2011, pp. 1981–1984.

[6] S. Asur and B. A. Huberman, "Predicting the future with social media," in *Proc. IEEE WI-IAT*, vol. 1, 2010, pp. 492–499.

[7] H. Choi and H. Varian, "Predicting the present with google trends," *Economic Record*, vol. 88, no. s1, pp. 2–9, 2012.

[8] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[9] 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.

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

[11] Z. Haipeng, S. R. Kulkarni, and H. V. Poor, "Attribute-distributed learning: Models, limits, and algorithms," *IEEE Trans. Signal Process.*, vol. 59, no. 1, pp. 386–398, 2011.

[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] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, 2010.

[15] Z. J. Towfic, J. Chen, and A. H. Sayed, "On distributed online classification in the midst of concept drifts," *Neurocomputing*, vol. 112, pp. 139–152, 2013.

[16] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, 2006.

[17] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, *Distributed detection and estimation in wireless sensor networks*, available at: http://arxiv.org/pdf/1307.1448v2.pdf, 2013.

[18] S. Barbarossa and G. Scutari, "Decentralized maximum-likelihood estimation for sensor networks composed of nonlinearly coupled dynamical systems," *IEEE Trans. Signal Process.*, vol. 55, no. 7, pp. 3456–3470, 2007.

[19] H. Zhang, J. Moura, and B. Krogh, "Dynamic field estimation using wireless sensor networks: Tradeoffs between estimation error and communication cost," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2383–2395, 2009.

[20] J. N. Tsitsiklis, "Decentralized detection," in *Advances in Statistical Signal Processing*, 1993, pp. 297–344.

[21] V. Veeravalli, "Decentralized quickest change detection," *IEEE Trans. Inform. Theory*, pp. 1657–1665, 2001.

[22] J.-F. Chamberland and V. Veeravalli, "Decentralized detection in sensor networks," *IEEE Trans. Signal Process.*, vol. 51, no. 2, pp. 407–416, Feb 2003.

[23] L. Canzian and M. van der Schaar, "A network of cooperative learners for data–driven stream mining," in *Proc. IEEE ICASSP*, vol. 3, May 2014, pp. 2908–2912.

[24] D. Shutin, S. R. Kulkarni, and H. V. Poor, "Incremental reformulated automatic relevance determination," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4977–4981, 2012.

[25] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, 2004.

[26] K. Ishibashi, T. Toyono, and K. Toyama, "Detecting mass-mailing worm infected hosts by mining DNS traffic data," in *Proc. SIGCOMM*, 2005.

[27] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?" *Neural Comput.*, vol. 16, no. 5, pp. 1063–1076, May 2004.

[28] 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.

[29] 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.

Luca Canzian [M'13] 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. From January 2013 to April 2014 he was a Postdoc at the Electrical Engineering Department of the University of California, Los Angeles (UCLA). Since April 2014 he has been a Postdoc at the Computer Science Department of the University of Birmingham. His research interests include Big Data, online learning, real-time stream mining, and game theory applied to wireless networks.

Mihaela van der Schaar [F'10] is Chancellor's Professor of Electrical Engineering at UCLA. She was Distinguished Lecturer of the Communications Society and Editor in Chief of IEEE Transactions on Multimedia. She received an NSF CAREER Award, 3 IBM Faculty Awards, and several Best Paper Awards. She holds 33 granted US patents. She is also the founding director of the UCLA Center for Engineering Economics, Learning, and Networks. Her research interests include engineering economics and game theory, network science, expert and social networks, online reputation and social media, dynamic multi-user networks and system designs, wireless networks, online and interactive learning, and real-time stream mining.