Learning From Heterogeneous Data Based on Social Interactions Over Graphs

Virginia Bordignon, *Member, IEEE*, Stefan Vlaski, *Member, IEEE*, Vincenzo Matta, *Senior Member, IEEE*, and Ali H. Sayed, *Fellow, IEEE*

***Abstract*—This work proposes a decentralized architecture, where individual agents aim at solving a classification problem while observing streaming features of different dimensions and arising from possibly different distributions. In the context of social learning, several useful strategies have been developed, which solve decision making problems through local cooperation across distributed agents and allow them to learn from streaming data. However, traditional social learning strategies rely on the fundamental assumption that each agent has significant prior knowledge of the underlying distribution of the observations. In this work we overcome this issue by introducing a machine learning framework that exploits social interactions over a graph, leading to a fully data-driven solution to the distributed classification problem. In the proposed social machine learning (SML) strategy, two phases are present: in the training phase, classifiers are independently trained to generate a belief over a set of hypotheses using a finite number of training samples; in the prediction phase, classifiers evaluate streaming unlabeled observations and share their instantaneous beliefs with neighboring classifiers. We show that the SML strategy enables the agents to learn consistently under this highly-heterogeneous setting and allows the network to continue learning even during the prediction phase when it is deciding on unlabeled samples. The prediction decisions are used to continually improve performance thereafter in a manner that is markedly different from most existing static classification schemes where, following training, the decisions on unlabeled data are not re-used to improve future performance.**

***Index Terms*—Distributed classification, social learning, neural networks, diffusion strategies.**

I. Introduction

SOCIAL learning strategies allow for the classification of unlabeled streaming observations by a network of agents [2], [3], [4], [5], [6], [7], [8], [9]. These agents are heterogeneous in two main aspects: first, they may be observing different (possibly non-overlapping) sets of attributes of the same observed scene; second, their statistical models need not be the same, e.g., agents may be observing the same attribute from different perspectives, which allows for a *distribution diversity* across agents. In a collaborative solution, neighboring agents share beliefs about their observations and diffuse this information across the network to arrive at a conclusion on the true underlying class explaining the common observed scene.

Under a Bayesian social learning paradigm, the agents would need to build the full posterior distribution of the data by exploiting sophisticated dependencies across their individual models. From a design perspective, this approach is seldom viable since it would require exploiting mutual dependencies across a (possibly large) set of agents. From a behavioral perspective, it has been shown [10] that in many distributed systems, agents learn in a non-Bayesian way in the following sense. They first manage to construct their belief according to a *local* Bayesian update rule, and then these locally updated beliefs are shared among neighbors to guarantee percolation of information across the network. In this work, we focus accordingly on the non-Bayesian paradigm. Several non-Bayesian social learning strategies exist in the literature. A common feature is that they allow for correct asymptotic learning of the truth [2], [3], [5], [7], [9]. This desirable learning property requires however prior knowledge of the true probability distributions characterizing agents' observations (usually referred to as *likelihoods*), which are in general not available in real-world applications. In practice, these models are only approximate, oftentimes the result of a previous *training* stage, where, from limited data, a parameterized model is learned.

This issue is recognized in the works [11], [12], where the authors propose a framework for incorporating uncertainty into non-Bayesian social learning. While [11] focuses only on sets of Gaussian distributions, their proposed strategy in [12] broadens the approach, but requires nonetheless the selection of some suitable family of distributions that is amenable to closed-form mathematical manipulations, such as determination of a conjugate prior. While relevant for numerically generated data, in practical applications there is generally little *a priori* evidence regarding the structure of likelihoods, e.g., in the distributed classification of images or videos.

In this work, we propose the Social Machine Learning (SML) strategy: a decentralized algorithm for combining the outputs of a heterogeneous network of classifiers over space

Manuscript received 17 December 2021; revised 18 November 2022; accepted 12 December 2022. Date of publication 26 December 2022; date of current version 21 April 2023. This work was supported in part by the Swiss National Science Foundation under Grant 205121-184999. An earlier version of this paper was presented in part at ICASSP 2021 [DOI: 10.1109/ICASSP39728.2021.9414126]. *(Corresponding author: Virginia Bordignon.)*

Virginia Bordignon and Ali H. Sayed are with the School of Engineering, École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland (e-mail: virginia.bordignon@epfl.ch; ali.sayed@epfl.ch).

Stefan Vlaski was with the Adaptive Systems Laboratory, École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland. He is now with the Department of Electrical and Electronic Engineering, Imperial College London, SW7 2BT London, U.K. (e-mail: s.vlaski@imperial.ac.uk).

Vincenzo Matta is with the Department of Information and Electrical Engineering and Applied Mathematics (DIEM), University of Salerno, 84084 Fisciano, Italy, and also with the National Inter-University Consortium for Telecommunications (CNIT), Italy (e-mail: vmatta@unisa.it).

Communicated by R. Venkataramanan, Associate Editor for Machine Learning.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3232368.

Digital Object Identifier 10.1109/TIT.2022.3232368

and time, based on the social learning algorithms proposed in [7], [8], [9], and [13]. Our approach has the advantage of allowing the use of a fairly general class of distributions, which is relevant in practical machine learning tasks. The SML strategy consists of two independent phases: a *training phase*, in which the classifiers are independently trained given a finite set of labeled data samples, and a *prediction phase*, in which the trained classifiers are deployed in a local collaborative structure while observing streaming unlabeled samples. An early version of this strategy was presented in [1].

In our proposed strategy, agents (or classifiers) cooperate with neighbors to overcome *local* spatial limitations. They also aggregate their *instantaneous* opinions from streaming observations, strengthening their decision-making capabilities over time. These two aspects, i.e., information aggregation over space and time, are common topics of research in the fields of ensemble [14] and multi-view learning [15].

Popular examples of ensemble approaches are bagging [16] and boosting [17], in which classifiers combine weighted decisions across *space*. However, such combination takes place in a *centralized* manner, namely, it is assumed that all agents communicate their decisions to a fusion center. This mechanism is fundamentally different from the *fully decentralized* setting addressed here, where only local cooperation between neighboring agents is permitted, and each individual agent is eventually able to learn the correct class. Moreover, both bagging and boosting methods do not address the streaming data case, i.e., they do not leverage the temporal quality of the online observations.

In multi-view learning, multiple views of the same data are available, which are jointly used to improve generalization performance. Multi-view co-training approaches [18] are notably suitable for semi-supervised learning, where a substantial number of unlabeled samples are available. In these approaches, distinct classifiers are trained on different views, and one classifier's predictions on new unlabeled examples are used to enlarge the labeled training set of the other. The procedure is repeated over the unlabeled samples, improving their accuracy over successive iterations. However, multi-view learning does not address the *distributed* and *streaming-data* aspects. Regarding the former aspect, in multi-view learning the classifiers are not spatially distributed or, if they are, it is simply assumed that they can share their beliefs without any constraints (i.e., as if they were co-located). Regarding the latter aspect, multi-view learning does not assume that streaming data are available for prediction.

The main contributions of this work are as follows. We propose a social machine learning strategy that inherits the following qualities from social learning: the ability to combine classifiers with different dimensions and statistical models, to adapt in view of changing real-time measurements, while providing asymptotic performance guarantees, in addition to continuous performance improvement during the prediction phase. We show that with high probability, consistent learning occurs despite the imperfectly trained models. We also show that poorly trained classifiers can leverage the networked setup to improve their performance, and that agents can use temporal memory to improve their generalization performance, in the form of classification accuracy. This property is distinct from existing static prediction phases for traditional classifiers, where classification decisions are instantaneous and do not take advantage from aggregation of increasing amounts of data that become available as time elapses. We illustrate these results using an application to the MNIST [19] dataset for image classification in a distributed setup. A comparison with AdaBoost shows how, contrary to AdaBoost, SML enables increased accuracy over time. We furthermore include a discussion in Appendix A comparing SML with the uncertain-likelihood approach introduced in [12].

Notation: We use boldface fonts to denote random variables, and normal fonts for their realizations. $\mathbb{E}$ and $\mathbb{P}$ denote expectation and probability operators, respectively. When necessary, suitable subscripts will be appended to denote the specific random variables the operators refer to.

II. Background

A. Inference Problem

We consider a network of K agents or classifiers, indexed by $k \in \{1, 2, \dots, K\}$, trying to identify the true state of nature γ_0 out of a binary set of hypotheses or classes $\Gamma = \{-1, +1\}$. The true state characterizes the *scene* all agents are observing. To make a decision on the true state, each agent relies on the observation of streaming private data, which relate to the observed scene. Data are qualified as private due to the implicit assumption that raw observations cannot be shared among agents in order to, for example, minimize communication costs or preserve secrecy.

More specifically, each agent k observes at each instant i the feature vector $\boldsymbol{h}_{k,i} \in \mathcal{H}_k$. The feature vectors are assumed to be independent and identically distributed (i.i.d.) over time. Moreover, the features $\boldsymbol{h}_{k,i}$ at agent k given the state γ_0 form a sequence of i.i.d. random vectors distributed according to some conditional distribution (or likelihood):

$$\boldsymbol{h}_{k,i} \sim L_k(h|\gamma_0), \quad h \in \mathcal{H}_k, \gamma_0 \in \Gamma. \tag{1}$$

Notably, the model allows the features to be dependent across agents (i.e., over space). The feature set $\mathcal{H}_k$ is particular to agent k, allowing agents to be observing different features from the same scene; in particular, the dimension of $\mathcal{H}_k$ can be different across the agents. For example, an agent might be observing RGB video frames while another might be receiving infrared imagery taken both from the same street scene. Another source of heterogeneity is the likelihood model $L_k(h|\gamma)$, which differs across agents and reflects their individual perceptions. Within the previous street scene example, agents might observe frames captured under different, possibly non-overlapping, fields of view.

We can treat the true state of nature as a random variable $\boldsymbol{\gamma}_0$ and furthermore establish that the pair $(\boldsymbol{h}_{k,i}, \boldsymbol{\gamma}_0)$ is distributed according to the following joint distribution:

$$(\boldsymbol{h}_{k,i}, \boldsymbol{\gamma}_0) \sim p_k(h, \gamma) = L_k(h|\gamma)p_k(\gamma), \tag{2}$$

with $h \in \mathcal{H}_k,\ \gamma \in \Gamma$, for every $i = 1, 2, \dots$ due to the i.i.d. assumption over time. Here, the notation $p_k(\gamma)$ corresponds to the prior distribution at agent k for $\boldsymbol{\gamma}_0$ over the discrete set of hypotheses Γ.

If the likelihood and prior distributions are perfectly known to agent k, different strategies can be deployed to enable truth learning. In a noncooperative framework, where each agent has enough information to solve the problem on their own, we can resort to the *Bayes classifier*. Alternatively, to leverage the data spread across different agents of the network, a cooperative strategy can be used, such as one of the existing *social learning* methods. We discuss each of the two strategies in more detail in the next paragraphs.

B. Bayes Classifier

When the Kullback-Leibler (KL) divergence [20] between likelihoods $L_k(h|+1)$ and $L_k(h|-1)$ is strictly positive, we say that agent k possesses *informative* likelihoods and can thus distinguish classes $+1$ and -1. Therefore, if the likelihood and prior distributions are known to agent k and its likelihoods are informative, the agent can employ the Bayes classifier to solve the following maximum-a-posteriori (MAP) problem given an observed sequence of features $\{\boldsymbol{h}_{k,j}\}$ with $j = 1, 2, \ldots, i$:

$$\boldsymbol{\gamma}_{k,i}^{\text{Bayes}} = \arg\max_{\gamma\in\Gamma} p_k(\gamma|\boldsymbol{h}_{k,1}, \boldsymbol{h}_{k,2}, \ldots, \boldsymbol{h}_{k,i}), \tag{3}$$

where $p_k(\gamma|\boldsymbol{h}_{k,1}, \boldsymbol{h}_{k,2}, \ldots, \boldsymbol{h}_{k,i})$ indicates the posterior probability of the event $\{\boldsymbol{\gamma} = \gamma\}$ given the sequence $\{\boldsymbol{h}_{k,j}\}$ with $j = 1, 2, \ldots, i$. Using Bayes' rule and the property of conditional independence of the features over time, and taking the logarithm of the posterior probability, we obtain

$$\begin{aligned}&\log p_k(\gamma|\boldsymbol{h}_{k,1}, \boldsymbol{h}_{k,2}, \ldots, \boldsymbol{h}_{k,i})\\ &= \sum_{j=1}^{i}\log L_k(\boldsymbol{h}_{k,j}|\gamma) + \log p_k(\gamma) - \log p_k(\boldsymbol{h}_{k,1}, \ldots, \boldsymbol{h}_{k,i}),\end{aligned} \tag{4}$$

where the joint distribution over the feature vectors, namely $p_k(\boldsymbol{h}_{k,1}, \ldots, \boldsymbol{h}_{k,i})$, does not depend on γ. For binary labels, solving (3) is equivalent to producing a binary decision $\boldsymbol{\gamma}_{k,i}^{\text{Bayes}}$ using the following test:

$$\sum_{j=1}^{i}\log\frac{L_k(\boldsymbol{h}_{k,j}|+1)}{L_k(\boldsymbol{h}_{k,j}|-1)} + \log\frac{p_k(+1)}{p_k(-1)} \underset{-1}{\overset{+1}{\gtrless}} 0. \tag{5}$$

Proposition 1 characterizes the consistency of Bayes classifier, also known as recursive Bayesian forecasting given streaming observations. It is a known result [21] and is therefore stated without proof.

Proposition 1 (Bayes Classifier for a Sequence of Observations): Assume that $p_k(\gamma) > 0$ for $\gamma \in \Gamma$, and that the KL divergences between likelihoods satisfy:

$$0 < \mathsf{D}_{\mathsf{KL}}\left[L_k(+1)||L_k(-1)\right] < \infty, \tag{6}$$

$$0 < \mathsf{D}_{\mathsf{KL}}\left[L_k(-1)||L_k(+1)\right] < \infty, \tag{7}$$

where $\mathsf{D}_{\mathsf{KL}}\left[L_k(\gamma)||L_k(\gamma')\right]$ denotes the KL divergence between likelihoods $L_k(h|\gamma)$ and $L_k(h|\gamma')$. Then the Bayes classifier $\boldsymbol{\gamma}_{k,i}^{\text{Bayes}}$ learns the truth with probability 1 as i goes to infinity. ■

While this result ensures consistent learning, it requires each individual agent to have informative likelihoods and thus to be able to distinguish both hypotheses. This restriction motivates the pursuit of *collaborative* social learning schemes, where agents exchange information to resolve ambiguities arising from incomplete information.

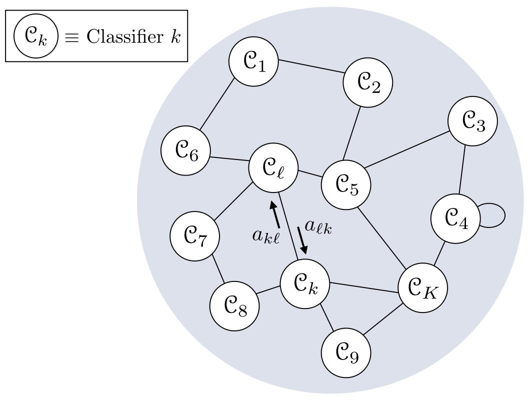

Fig. 1. Diagram of the network of classifiers.

C. Social Learning

In a multi-agent setup, a network of K agents is modeled as a strongly-connected graph, i.e., where there is a path in both directions linking any two agents and at least one self-loop (a diagram can be seen in Fig. 1). The graph possesses a left-stochastic combination matrix A, whose elements $a_{\ell k}$ satisfy:

$$\sum_{\ell=1}^{K} a_{\ell k} = 1, \quad a_{\ell k} \geq 0, \quad a_{\ell k} = 0 \text{ if } \ell \notin \mathcal{N}_k, \tag{8}$$

where $\mathcal{N}_k$ indicates the set of neighbors of k (including k itself). Each $a_{\ell k}$ is a combination weight scaling information arriving at agent k from agent ℓ. From the strong-connectivity assumption, A is a primitive matrix and we can therefore define its Perron eigenvector π as [22]:

$$A\pi = \pi, \quad \sum_{k=1}^{K}\pi_k = 1, \quad \pi_k > 0 \text{ for } k = 1, 2, \ldots, K. \tag{9}$$

If the $K-$agent network is collectively observing streaming features $\{\boldsymbol{h}_{1,i}, \boldsymbol{h}_{2,i}, \ldots, \boldsymbol{h}_{K,i}\}$, it is beneficial for agents to cooperate in order to solve the inference problem. One intuitive benefit of cooperation comes from the fact that the network is observing K times more data than each individual alone. Another fundamental benefit is that network cooperation can enable successful learning even when the decision problem is *not identifiable* for the individual agents, but is *globally* identifiable at the network level. Among the existing cooperative protocols, decentralized processing is particularly appealing for not only improving learning performance, but also for increasing robustness and ensuring data privacy [22]. Moreover, in a decentralized setup, agents share processed information, i.e., they do not share raw data, with their immediate neighbors.

Inspired by the advantages of decentralized processing, a large family of non-Bayesian social learning works has been proposed with the purpose of solving the aforementioned inference problem [2], [3], [5], [6], [7], [9]. Resorting to different update and combination mechanisms, many of them have the feature of yielding asymptotic truth learning [2], [3], [5], [6], [7]. In the following development, we detail two of these strategies.

1) Traditional Social Learning: We first consider the social learning (SL) strategy discussed in [5], [7], [8], and [10]. At every instant i, each agent k updates its *belief* (opinion) $\boldsymbol{\varphi}_{k,i}(\gamma)$ given its private observations and its neighbors' beliefs. The belief vector $\boldsymbol{\varphi}_{k,i}$ is a probability mass function over the set of hypotheses Γ, where each element $\boldsymbol{\varphi}_{k,i}(\gamma)$ represents the confidence that hypothesis γ corresponds to the true state.

The belief is updated according to a two-step protocol:

$$\boldsymbol{\psi}_{k,i}(\gamma) = \frac{\boldsymbol{\varphi}_{k,i-1}(\gamma) L_k(\boldsymbol{h}_{k,i}|\gamma)}{\sum_{\gamma' \in \Gamma} \boldsymbol{\varphi}_{k,i-1}(\gamma') L_k(\boldsymbol{h}_{k,i}|\gamma')}, \tag{10}$$

$$\boldsymbol{\varphi}_{k,i}(\gamma) = \frac{\exp\left\{\sum_{\ell \in \mathcal{N}_k} a_{\ell k} \log \boldsymbol{\psi}_{\ell,i}(\gamma)\right\}}{\sum_{\gamma' \in \Gamma} \exp\left\{\sum_{\ell \in \mathcal{N}_k} a_{\ell k} \log \boldsymbol{\psi}_{\ell,i}(\gamma')\right\}}, \tag{11}$$

where in the first step (Eq. (10)) agent k updates its *intermediate belief* $\boldsymbol{\psi}_{k,i}$ using the observed feature vector $\boldsymbol{h}_{k,i}$. Then in the second step (Eq. (11)), agents share their intermediate beliefs with neighboring agents and update their beliefs using a geometric averaging rule.

An equivalent linear way of representing (10) and (11) is in the form of the *diffusion strategy* [22], [23]:

$$\boldsymbol{\eta}_{k,i} = \boldsymbol{\lambda}_{k,i-1} + c_k(\boldsymbol{h}_{k,i}), \tag{12}$$

$$\boldsymbol{\lambda}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\eta}_{\ell,i}, \tag{13}$$

in terms of the following scalar quantities:

$$\boldsymbol{\lambda}_{k,i} \triangleq \log \frac{\boldsymbol{\varphi}_{k,i}(+1)}{\boldsymbol{\varphi}_{k,i}(-1)}, \quad \boldsymbol{\eta}_{k,i} \triangleq \log \frac{\boldsymbol{\psi}_{k,i}(+1)}{\boldsymbol{\psi}_{k,i}(-1)}, \tag{14}$$

$$c_k(\boldsymbol{h}_{k,i}) \triangleq \log \frac{L_k(\boldsymbol{h}_{k,i}|+1)}{L_k(\boldsymbol{h}_{k,i}|-1)}. \tag{15}$$

Equations (12) and (13) can be joined into a single equation as:

$$\boldsymbol{\lambda}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \Big(\boldsymbol{\lambda}_{\ell,i-1} + c_\ell(\boldsymbol{h}_{\ell,i})\Big). \tag{16}$$

The representation in (16) highlights that, in the decision variable $\boldsymbol{\lambda}_{k,i}$, agent k aggregates past (in the form of $\boldsymbol{\lambda}_{\ell,i-1}$) and present (in the form of $c_\ell(\boldsymbol{h}_{\ell,i})$) information relative to its neighbors $\ell \in \mathcal{N}_k$.

Previous works [5], [7], [8], [10] show that, under the strategy in (10) and (11), the belief component $\boldsymbol{\varphi}_{k,i}(\gamma_0)$ converges almost surely to 1 as $i \to \infty$, i.e., the belief is maximized at the true hypothesis. Specifically, by developing the recursion in (16) it is possible to show that, as $i \to \infty$, the belief function is maximized at the true hypothesis, provided that a weighted combination (through the Perron eigenvector weights) of the detection statistics $c_\ell(\boldsymbol{h}_{\ell,i})$ has positive expectation under hypothesis $+1$ and negative expectation under -1, namely,[1]

$$\boxed{\sum_{\ell=1}^{K} \pi_\ell \mathbb{E}_{L_\ell(+1)} c_\ell(\boldsymbol{h}_{\ell,i}) > 0, \quad \sum_{\ell=1}^{K} \pi_\ell \mathbb{E}_{L_\ell(-1)} c_\ell(\boldsymbol{h}_{\ell,i}) < 0} \tag{18}$$

where $\mathbb{E}_{L_\ell(\gamma)}$ indicates that the expectation is computed with respect to the distribution $L_\ell(h|\gamma)$. We remark that the condition for consistency in (18) would apply to general detection statistics $c_\ell(\cdot)$, and not only to log-likelihood ratios as in (15). For example, detection statistics different from (15) may arise because the agents compute *mismatched* log-likelihood ratios due to imperfect knowledge. This observation is particularly relevant in our work since, when we will examine the social machine learning setting (where the likelihoods are unknown) we will need to work with general detection statistics learned from a training set. On the other hand, for the specific case where the likelihoods are known and (15) is employed, the conditions in (18) are satisfied whenever the network satisfies the *global identifiability* assumption [7], i.e., at least one agent in the network is able to distinguish the hypotheses. In this case, for at least one agent k, it follows that

$$\mathbb{E}_{L_k(+1)} c_k(\boldsymbol{h}_{k,i}) = \mathsf{D}_{\mathsf{KL}}[L_k(+1)||L_k(-1)] > 0 \tag{19}$$

$$\mathbb{E}_{L_k(-1)} c_k(\boldsymbol{h}_{k,i}) = -\mathsf{D}_{\mathsf{KL}}[L_k(-1)||L_k(+1)] < 0. \tag{20}$$

From the positivity of the Perron eigenvector π, (19) and (20) imply that the conditions in (18) are satisfied.

Therefore, the strategy in (10) and (11) allows agents to learn the truth asymptotically, as i tends to infinity, with probability 1 [5], [7]. Almost sure consistent learning is a desirable feature in any inference algorithm. This remarkable accuracy in learning the truth comes however at the expense of a reduced ability by the algorithm to *adapt* to drifting non-stationary conditions, e.g., to changes in the true state γ_0, to missing observations, and to changes in the statistical characteristics of the data. As discussed in [9], agents behave stubbornly in face of environment changes, displaying an unreasonably large time to adapt to new conditions. To address non-stationary applications, we introduce a second SL algorithm.

2) Adaptive Social Learning: In a real-time application, we expect the environment conditions to change with time, and the learning strategy should be able to track the drifting conditions within a reasonable response time. In [9] and [13], an *adaptive social learning* strategy was proposed to overcome the lack of adaptation in traditional social learning.

In one of the formulations proposed in [9], the first step of the update rule in (21) is replaced by an *adaptive* update as

[1]The sufficient condition for consistent learning in (18) can be reached by following similar arguments as in [8, Appendix A]. Developing the recursion in (16) and dividing by i, we can conclude that

$$\frac{1}{i}\boldsymbol{\lambda}_{k,i} \xrightarrow{\text{a.s.}} \sum_{\ell=1}^{K} \pi_\ell \mathbb{E}_{L_\ell(\gamma_0)} c_\ell(\boldsymbol{h}_{\ell,i}). \tag{17}$$

If $\sum_{\ell=1}^{K} \pi_\ell \mathbb{E}_{L_\ell(\gamma_0)} c_\ell(\boldsymbol{h}_{\ell,i}) > 0$, then $\boldsymbol{\lambda}_{k,i}$ goes to $+\infty$, and thus agents decide for class $+1$. Otherwise if $\sum_{\ell=1}^{K} \pi_\ell \mathbb{E}_{L_\ell(\gamma_0)} c_\ell(\boldsymbol{h}_{\ell,i}) < 0$, then $\boldsymbol{\lambda}_{k,i}$ goes to $-\infty$, and thus agents decide for class -1.

follows:

$$\boldsymbol{\psi}_{k,i}(\gamma) = \frac{\boldsymbol{\varphi}_{k,i-1}^{1-\delta}(\gamma) L_k(\boldsymbol{h}_{k,i}|\gamma)}{\sum_{\gamma' \in \Gamma} \boldsymbol{\varphi}_{k,i-1}^{1-\delta}(\gamma') L_k(\boldsymbol{h}_{k,i}|\gamma')}, \tag{21}$$

where $0 < \delta \ll 1$ is a small step-size (or learning) parameter.[2] The combination step, in which the intermediate beliefs are shared across neighbors, is the same as the one in (11). The introduction of a step-size to the local update in (21) infuses the algorithm with the ability to adapt in face of non-stationary conditions with an *adaptation time* that scales as $\mathcal{O}(1/\delta)$ [9]. In the limit case, when $\delta \to 0$, we recover the Bayesian update in (10).

Similarly to (12)–(13), we can represent (11) and (21) in the form of an *adaptive diffusion strategy* [24]:

$$\boldsymbol{\eta}_{k,i} = (1-\delta)\boldsymbol{\lambda}_{k,i-1} + c_k(\boldsymbol{h}_{k,i}), \tag{23}$$

$$\boldsymbol{\lambda}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{\eta}_{\ell,i}, \tag{24}$$

which yields:

$$\boldsymbol{\lambda}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k}\Big((1-\delta)\boldsymbol{\lambda}_{\ell,i-1} + c_\ell(\boldsymbol{h}_{\ell,i})\Big). \tag{25}$$

From (25), we see clearly that the step-size δ attenuates the influence of past data, embodied by $\boldsymbol{\lambda}_{\ell,i-1}$. As long as δ is strictly greater than zero and smaller than one, the recursion in (25) can be shown to be stable, i.e., $\boldsymbol{\lambda}_{k,i}$ does not degenerate to $\pm\infty$ as $i \to \infty$. This non-degenerate behavior is the reason why the adaptive social learning algorithm can quickly recover from a previous state when faced with changes in the environment [9].

The price for this improved adaptation is reflected on the learning accuracy. In contrast with the almost sure convergence found in traditional social learning, consistent learning now occurs asymptotically (as $i \to \infty$) with high probability in the regime of small step-sizes (as $\delta \to 0$) [9], [13]. The same sufficient condition for attaining consistent truth learning enunciated in (18), applies for the adaptive social learning algorithm as well, even for general detection statistics $c_\ell(\cdot)$ [24].

Similarly to most social learning algorithms in the literature, the family of likelihoods $L_k(h|\gamma)$ with $\gamma \in \Gamma$ is assumed to be *perfectly known* to each agent k. As we can see, in both of the aforementioned social learning strategies, one important statistic diffused across agents and over time is the log-ratio of likelihoods, $c_\ell(\cdot)$ as in (15), which is classically employed as the basic building block to design other types of distributed detection strategies [25], [26]. In real-world applications, these likelihood models are generally unavailable. Instead, they are obtained as the result of a prior training step in which (parameterized) models are trained using a finite set of data examples.

[2]In [9], it is shown that the update in (21) yields a learning performance that is similar to the alternative update:

$$\boldsymbol{\psi}_{k,i}(\gamma) = \frac{\boldsymbol{\varphi}_{k,i-1}^{1-\delta}(\gamma) L_k^{\delta}(\boldsymbol{h}_{k,i}|\gamma)}{\sum_{\gamma' \in \Gamma} \boldsymbol{\varphi}_{k,i-1}^{1-\delta}(\gamma') L_k^{\delta}(\boldsymbol{h}_{k,i}|\gamma')}, \tag{22}$$

in which δ also weights the new information contained in $L_k(\boldsymbol{h}_{k,i}|\gamma)$.

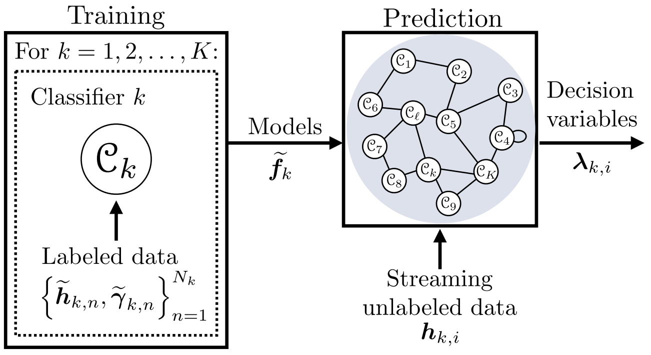
Training
For $k = 1, 2, \ldots, K$:
Classifier k
$\mathcal{C}_k$
Labeled data
$\left\{\widetilde{\boldsymbol{h}}_{k,n}, \widetilde{\boldsymbol{\gamma}}_{k,n}\right\}_{n=1}^{N_k}$
Models
$\widetilde{\boldsymbol{f}}_k$
Prediction
Streaming unlabeled data
$\boldsymbol{h}_{k,i}$
Decision variables
$\boldsymbol{\lambda}_{k,i}$

Fig. 2. Social Machine Learning (SML) diagram.

In view of this practical limitation of social learning, we propose in this work a two-phase learning strategy, which we refer to as Social Machine Learning (SML). In this strategy, the likelihood models are assumed to be *unknown*.

III. Social Machine Learning

The SML strategy is designed as a two-step approach. In the *training phase*, the classifiers are trained individually given private finite datasets. In the *prediction phase*, classifiers are deployed in a cooperative social learning structure. In Fig. 2, we show a diagram depicting the SML approach. These distinct *learning phases* are detailed in the following sections.

To avoid confusion, random variables related to the training phase are topped with a symbol $\sim$. Furthermore, data samples pertaining to the training datasets are indexed by n, whereas, in the prediction phase, they are indexed by i. For example, $\widetilde{\boldsymbol{h}}_{k,n} \in \mathcal{H}_k$ represents the $n-$th feature vector available in the training dataset of agent k, whereas $\boldsymbol{h}_{k,i} \in \mathcal{H}_k$ represents the feature vector observed by agent k at instant i during the prediction phase. We assume that the random variables are independent between different learning phases.

A. Training Phase

During training, each agent k has access to N_k examples consisting of pairs $\{\widetilde{\boldsymbol{h}}_{k,n}, \widetilde{\boldsymbol{\gamma}}_{k,n}\}_{n=1}^{N_k}$. We assume that the training set is balanced so that both classes are sufficiently explored, namely, we assume that, *during training*, labels $\widetilde{\boldsymbol{\gamma}}_{k,n}$ are uniformly distributed over $\Gamma = \{-1, +1\}$ for all agents. This is a standard technical assumption that will be useful to obtain readable bounds for the social machine learning strategy. However, we remark that the assumption of a balanced dataset during the training phase does not impose any constraint on the behavior of the true hypothesis during the *prediction* phase. In particular, the data observed during prediction are all coming from a certain true state of nature γ_0, and we will establish that the SML strategy achieves vanishing error regardless of the particular hypothesis being in force.

The pair $(\widetilde{\boldsymbol{h}}_{k,n}, \widetilde{\boldsymbol{\gamma}}_{k,n})$ is distributed according to the joint distribution:

$$(\widetilde{\boldsymbol{h}}_{k,n}, \widetilde{\boldsymbol{\gamma}}_{k,n}) \sim \widetilde{p}_k(h, \gamma) = L_k(h|\gamma)\widetilde{p}_k(\gamma), \tag{26}$$

where $h \in \mathcal{H}_k$, $\gamma \in \Gamma$ and $\widetilde{p}_k(\gamma) = 1/2$ for $\gamma \in \Gamma$. Note that, given a label γ, the corresponding feature vector $\widetilde{\boldsymbol{h}}_{k,n}$ is distributed according to

$$\widetilde{\boldsymbol{h}}_{k,n} \sim L_k(h|\gamma), \quad h \in \mathcal{H}_k, \gamma \in \Gamma. \tag{27}$$

Using these training samples, we wish to deploy a fully data-driven solution inspired by the social learning algorithms presented in Sec. II-C. To accomplish this, we first need to approximate the key unknown function c_k used in (16) and (25) by a quantity resulting from some statistical learning method. The choice of method and the characteristics of the problem at hand, e.g., training samples and class of models considered, will heavily determine the quality of the resulting approximation. These decisive factors and their impact will be examined later in this work.

Let us delve into the details of our training setup. First, note that, under the assumption of uniform priors during training, and using Bayes' rule, we can write:

$$c_k(h) = \log \frac{L_k(h|+1)}{L_k(h|-1)} = \log \frac{\widetilde{p}_k(+1|h)}{\widetilde{p}_k(-1|h)}, \tag{28}$$

where $\widetilde{p}_k(\gamma|h)$ represents the posterior probability of $\{\widetilde{\boldsymbol{\gamma}}_{k,n} = \gamma\}$ given $\{\widetilde{\boldsymbol{h}}_{k,n} = h\}$ using the joint model seen in (26). The significance of the log-ratio of posterior probabilities on the RHS of (28) can be interpreted in an intuitive manner: the log-ratio is positive whenever class $+1$ is more likely to be the true state of nature given the observation of h and negative when class -1 is more likely to explain the same data evidence. It is therefore reasonable that we seek an approximation for the log-ratio of posteriors in (28) during the training phase.

One relevant machine learning paradigm to approximate the posterior distribution is the *discriminative* paradigm (which includes, e.g., logistic regression and neural networks), where the output of the classifier is in the form of *approximate* posterior probabilities for each class, namely, $\widehat{p}_k(+1|h)$ and $\widehat{p}_k(-1|h)$. In order to illustrate this paradigm, it is convenient to introduce the *logit* statistic:

$$\log \frac{\widehat{p}_k(+1|h)}{\widehat{p}_k(-1|h)} = \log \frac{\widehat{p}_k(+1|h)}{1-\widehat{p}_k(+1|h)} \triangleq f_k(h), \tag{29}$$

where the function f_k can be chosen from an admissible class $\mathcal{F}_k$, namely,

$$f_k \in \mathcal{F}_k : \mathcal{H}_k \mapsto \mathbb{R}. \tag{30}$$

The choice of the class $\mathcal{F}_k$ depends on the choice of classifier. For example, in linear logistic regression with $h \in \mathbb{R}^M$, $\mathcal{F}_k$ is parameterized by a vector $w \in \mathbb{R}^M$, and we have the linear logit function [27]:

$$f_k(h; w) = w^\top h. \tag{31}$$

Another example is to consider MultiLayer Perceptrons (MLPs) with L hidden layers and a softmax output layer, whose weight matrices are given by $\{W_\ell\}$ over layers $\ell = 1, 2, \dots, L$. In the binary classification case, the network outputs two approximate posterior quantities [28], namely $\widehat{p}_k(+1|h; W)$ and $\widehat{p}_k(-1|h; W)$, where W represents the parameterization of the classifier w.r.t. matrices $\{W_\ell\}$. In this case, the logit function is given by the expression:

$$f_k(h; W) = \log \frac{\widehat{p}_k(+1|h; W)}{\widehat{p}_k(-1|h; W)}, \tag{32}$$

where the class of functions $\mathcal{F}_k$ is parameterized by matrices $\{W_\ell\}$ in a nonlinear manner. Note that the forthcoming analysis does not assume a specific model for the logit function, and applies instead to general classes $\mathcal{F}_k$.

The logit functions f_k are trained by each classifier $k = 1, 2, \dots, K$ by finding the function f_k within $\mathcal{F}_k$ that minimizes a suitable risk function $R_k(f_k)$. For example, in the already mentioned logistic regression and MLP cases, the training process results in optimal parameters w and W_ℓ for $\ell = 1, 2, \dots, L$, respectively.

One common risk function adopted in binary classification is the logistic risk:

$$R_k(f_k) = \mathbb{E}_{\tilde{h}_k, \tilde{\gamma}_k} \log\left(1 + e^{-\widetilde{\boldsymbol{\gamma}}_{k,n} f_k(\widetilde{\boldsymbol{h}}_{k,n})}\right), \tag{33}$$

where $\mathbb{E}_{\tilde{h}_k, \tilde{\gamma}_k}$ corresponds to the expectation computed under the (unknown) joint distribution $\widetilde{p}_k(h, \gamma)$ seen in (26). We remark that the logistic risk can be used either in association with the linear model in (31) or with more complex structures such as neural networks with softmax output layers. The logistic risk can be shown to be equivalent in the binary case to the cross-entropy risk function [29].

We define the *target risk* at every agent and the weighted network average according to:

$$\mathsf{R}_k^o \triangleq \inf_{f_k \in \mathcal{F}_k} R_k(f_k), \qquad \mathsf{R}^o \triangleq \sum_{k=1}^K \pi_k \mathsf{R}_k^o. \tag{34}$$

Unfortunately, in practice the expectation in (33) cannot be computed since the underlying feature/label distribution is unknown. The agents rely instead on a finite set of training samples to minimize an *empirical* risk:

$$\widetilde{\boldsymbol{f}}_k \triangleq \arg\min_{f_k \in \mathcal{F}_k} \widetilde{\boldsymbol{R}}_k(f_k), \tag{35}$$

given by

$$\widetilde{\boldsymbol{R}}_k(f_k) = \frac{1}{N_k} \sum_{n=1}^{N_k} \log\left(1 + e^{-\widetilde{\boldsymbol{\gamma}}_{k,n} f_k(\widetilde{\boldsymbol{h}}_{k,n})}\right), \tag{36}$$

which is computed over the training set. The resulting function $\widetilde{\boldsymbol{f}}_k$ can then be used by the agents to approximate the logit statistic in (29). For future use, we also define the *network average* for the expected risk and the empirical risk expressions:

$$R(f) \triangleq \sum_{k=1}^K \pi_k R_k(f_k), \qquad \widetilde{\boldsymbol{R}}(f) \triangleq \sum_{k=1}^K \pi_k \widetilde{\boldsymbol{R}}_k(f_k), \tag{37}$$

where the argument f represents the dependence of the risk expressions on the collection of functions $\{f_k\}$, i.e., $R(f) = R(f_1, f_2, \dots, f_K)$. This concise notation will be used whenever we are dealing with network-averaged quantities.

We will detail in the next section how the trained models can be deployed in the prediction phase, when agents are faced with streaming unlabeled feature vectors.

B. Prediction Phase

In the prediction phase, agents find themselves in the setup described in Sec. II-A. They aim at solving the inference problem of determining the true state $\gamma_0 \in \Gamma$, given streaming *unlabeled* private features $\boldsymbol{h}_{k,i}$, $i = 1, 2, \ldots$. The difference now is that they are equipped with the trained models $\{\widetilde{\boldsymbol{f}}_k\}$, which are constructed so as to provide a reasonable approximation for the log-ratio of posterior probabilities (see Fig. 2 for an illustrative diagram of this process).

During prediction, agents deploy one of the SL algorithms enunciated in Sec. II-C using the following approximation for the function c_k:

$$\widetilde{\boldsymbol{c}}_k(h) = \widetilde{\boldsymbol{f}}_k(h) - \widetilde{\boldsymbol{\mu}}_k(\widetilde{\boldsymbol{f}}_k), \tag{38}$$

where the second term on the RHS of (38) is called the *empirical training mean* and is defined for any function $f_k \in \mathcal{F}_k$ as:

$$\widetilde{\boldsymbol{\mu}}_k(f_k) = \frac{1}{N_k}\sum_{n=1}^{N_k} f_k(\widetilde{\boldsymbol{h}}_{k,n}), \tag{39}$$

i.e., it is defined as the average of function f_k over the training samples. Discounting the empirical training mean in (38) prevents the logit statistic from being biased towards one class or another. This is relevant considering that the decision of each agent is taken according to the rule

$$\boldsymbol{\gamma}^{\mathsf{SML}}_{k,i} \triangleq \text{sign}(\boldsymbol{\lambda}_{k,i}), \tag{40}$$

where $\text{sign}(x) = +1$, if $x \geq 0$ and $\text{sign}(x) = -1$ otherwise, i.e., the decision threshold is zero. Note that $\widetilde{\boldsymbol{c}}_k$ is a random function, whose randomness stems from the training phase.

Next, we illustrate how the debiasing operation used in (38) helps preventing biased decisions, but first let us define for any function $f_k \in \mathcal{F}_k$ the following conditional means:

$$\mu^+_k(f_k) \triangleq \mathbb{E}_{L_k(+1)} f_k(\boldsymbol{h}_{k,i}), \quad \mu^-_k(f_k) \triangleq \mathbb{E}_{L_k(-1)} f_k(\boldsymbol{h}_{k,i}). \tag{41}$$

Assume that f_k is fixed, i.e., $\widetilde{\boldsymbol{c}}_k(h) = f_k(h) - \widetilde{\boldsymbol{\mu}}_k(f_k)$, and that N_k is sufficiently large. Then, the empirical mean $\widetilde{\boldsymbol{\mu}}_k(f_k)$ approximates the expected value of $f_k(\widetilde{\boldsymbol{h}}_{k,n})$ in the training phase, namely, $[\mu^+_k(f_k) + \mu^-_k(f_k)]/2$ (see Eq. (96) in Appendix B). In this case, function $\widetilde{\boldsymbol{c}}_k$ is deterministic, and we can write:

$$\widetilde{c}_k(h) = f_k(h) - \frac{\mu^+_k(f_k) + \mu^-_k(f_k)}{2}. \tag{42}$$

Taking the conditional expectation of $\widetilde{c}_k(\boldsymbol{h}_{k,i})$, computed w.r.t. the prediction samples $\boldsymbol{h}_{k,i}$ given classes $+1$ and -1, yields:

$$\mathbb{E}_{L_k(+1)}\widetilde{c}_k(\boldsymbol{h}_{k,i}) = \frac{\mu^+_k(f_k) - \mu^-_k(f_k)}{2}, \tag{43}$$

$$\mathbb{E}_{L_k(-1)}\widetilde{c}_k(\boldsymbol{h}_{k,i}) = -\frac{\mu^+_k(f_k) - \mu^-_k(f_k)}{2}. \tag{44}$$

The approximation $\widetilde{c}_k$ satisfies the conditions for consistent learning in (18) if (43) is strictly positive and if (44) is strictly negative. Note that the debiasing operation introduces a symmetry to (43) and (44). Therefore both consistent learning conditions in (18) are satisfied by ensuring that the weaker condition $\mu^+_k(f_k) > \mu^-_k(f_k)$ holds, regardless of the sign of the individual terms $\mu^+_k(f_k)$ and $\mu^-_k(f_k)$. Thus, even in the biased case, in which $\mu^+_k(f_k) > \mu^-_k(f_k) > 0$, consistent learning using $\widetilde{c}_k$ can be achieved.

We proceed now to formally translate the consistent learning conditions for the SL algorithms seen in (18), considering the approximation $\widetilde{\boldsymbol{c}}_k(\boldsymbol{h}_{k,i})$ in (38). First, we define the network average of the conditional means in (41):

$$\mu^+(f) \triangleq \sum_{k=1}^{K} \pi_k \mu^+_k(f_k), \quad \mu^-(f) \triangleq \sum_{k=1}^{K} \pi_k \mu^-_k(f_k), \tag{45}$$

and the network average of the *empirical* training mean:

$$\widetilde{\boldsymbol{\mu}}(f) = \sum_{k=1}^{K} \pi_k \widetilde{\boldsymbol{\mu}}_k(f_k). \tag{46}$$

The training phase will generate the set of models $\{\widetilde{\boldsymbol{f}}_k\}$, which are random with respect to the training datasets. Given a particular training setup, we can "freeze" the randomness of the training set and work conditionally on a particular realization of learned models $\{\widetilde{f}_k\}$.

We are now interested in ascertaining whether or not these particular learned models allow for consistent learning *during the prediction phase*. To this end, we can apply the condition for consistent learning seen in (18) to the functions $\{\widetilde{c}_k\}$ in (38), for a frozen set of trained models $\{\widetilde{f}_k\}$, resulting in the following two conditions:

$$\sum_{k=1}^{K} \pi_k \mathbb{E}_{L_k(+1)} \widetilde{f}_k(\boldsymbol{h}_{k,i}) > \sum_{k=1}^{K} \pi_k \widetilde{\mu}_k(\widetilde{f}_k), \tag{47}$$

$$\sum_{k=1}^{K} \pi_k \mathbb{E}_{L_k(-1)} \widetilde{f}_k(\boldsymbol{h}_{k,i}) < \sum_{k=1}^{K} \pi_k \widetilde{\mu}_k(\widetilde{f}_k). \tag{48}$$

where we recall that $\mathbb{E}_{L_k(\gamma)}$ is the expectation computed with respect to the *prediction* samples $\boldsymbol{h}_{k,i}$ under the distribution $L_k(h|\gamma)$, and the prediction samples are independent of any random variable generated in the training phase. Finally, substituting the definitions in (41) and (45) respectively into (47) and (48), yields the following necessary conditions for consistent learning within the SML paradigm, conditionally on a given set of trained models $\{\widetilde{f}_k\}$.

$$\boxed{\mu^+(\widetilde{f}) > \widetilde{\mu}(\widetilde{f}) \quad \text{and} \quad \mu^-(\widetilde{f}) < \widetilde{\mu}(\widetilde{f})} \tag{49}$$

Since, the above description is given conditioned on a set of trained models $\{\widetilde{f}_k\}$, the conditions in (49) depend on the randomness stemming from the training phase. Therefore, characterizing the consistency of learning requires characterizing probabilistically the occurrence of both events described in (49). More precisely, we can define the *probability of consistent learning*, namely,

$$P_c \triangleq \mathbb{P}\left(\mu^+(\widetilde{\boldsymbol{f}}) > \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}})\,,\, \mu^-(\widetilde{\boldsymbol{f}}) < \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}})\right), \tag{50}$$

where boldface fonts now highlight the randomness in the training set. In the next section, we provide the characterization of (50) for classifiers belonging to general classes of bounded real-valued functions $\mathcal{F}_k$. In this case, we assume that there exists some real value $\beta > 0$ such that:

$$|f_k(h)| \le \beta, \quad f_k \in \mathcal{F}_k,\, h \in \mathcal{H}_k. \tag{51}$$

For example, consider the linear logistic regression case seen in (31). In practical applications, features belong to a bounded set $\mathcal{H}_k$, and thus condition (51) would be satisfied if the vector of weights w is constrained according to $\|w\|_2 \le b$, where b is some positive real value. Similarly, in the multilayer perceptron example seen in (32), the condition in (51) is satisfied for norm-constrained neural networks [30], i.e., where the weight matrices are bounded in norm by a certain positive real value b.

IV. Consistency of Social Machine Learning

We will need to call upon well-established statistical learning paradigms (e.g., the Vapnik-Chervonenkis theory) and adapt them to the *distributed network* setting considered in this work [31], [32]. More specifically, we will move along the path summarized below.

- We will assume that the individual agents minimize an *empirical risk*, producing a collection of K learned models, namely, the functions $\{\widetilde{\boldsymbol{f}}_k\}$. As usual, these functions are random due to the randomness of the training samples.
- We will examine the prediction (i.e., classification) performance obtained with the learned models $\{\widetilde{\boldsymbol{f}}_k\}$. In particular, we will establish technical conditions for the social learning algorithm to predict reliably the correct label as the number of streaming data gathered during the *prediction phase* increases.
- Since the learned models inherit the randomness of the training set, the consistency guarantees must be formulated in a probabilistic manner — see (50). Specifically, we guarantee a high probability that the samples in the training set lead to models $\{\widetilde{\boldsymbol{f}}_k\}$ that enable correct classification.
- As it happens in classical statistical learning frameworks, the interplay between empirical and optimal risk will be critical to ascertain the learning and prediction ability of the classifiers. However, differently from what is obtained in classical statistical learning frameworks, our results will depend significantly on the *graph* properties. In particular, a major role will be played by *weighted combinations of the individual risk functions*. The combination weights are the entries of the Perron eigenvector reflecting the combination matrix that governs the social learning interactions among the agents. This property leads to novel and interesting phenomena, for example, consistent classification can be achieved even if some of the agents learn bad models, but the plurality of the agents is able to reach a satisfying *aggregate* risk value.

Under the framework described above, the nontrivial interplay between the training and prediction phases might lead to some confusion. Therefore, it is useful to clarify the main path followed in the forthcoming analysis. We will focus on the probability of consistent learning P_c in (50), namely, the probability that the training set produces, at the end of the *training* phase, a consistent classifier. By "consistent", we mean that the classifier is able to mark the unlabeled data observed during the *prediction* phase correctly as $i \to \infty$. The probability P_c will be shown to be close to 1 if the training set size is large enough, namely, we will show that consistent learning is achievable provided that *sufficient training* is allowed. As Theorem 1 will show, in order to quantify the qualification "sufficient", it is critical to introduce a formal way to characterize the classifier structure.

The complexity of the classifier structure is related to the complexity of the class of functions $\mathcal{F}_k$. The latter is quantified by using the concept of *Rademacher complexity* (initially introduced as Rademacher penalty in [33]). We follow the definition in [30] and [34] and consider a class of functions $\mathcal{F}$ and a set x with N training samples, namely, $x \triangleq \{x_1, x_2 \ldots, x_N\}$, where $x_n \in \mathcal{X}$ for all $n = 1, 2, \ldots, N$. We also introduce the set of vectors $\mathcal{F}(x)$ defined as:

$$\mathcal{F}(x) \triangleq \Big\{[f(x_1), f(x_2), \ldots, f(x_N)] \,\Big|\, x_n \in \mathcal{X}, f \in \mathcal{F}\Big\}. \tag{52}$$

Then, the (empirical) Rademacher complexity associated with $\mathcal{F}(x)$ is:

$$\mathcal{R}(\mathcal{F}(x)) \triangleq \mathbb{E}_r \sup_{f \in \mathcal{F}} \left| \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{r}_n f(x_n) \right|, \tag{53}$$

where $\boldsymbol{r}_n$ are independent and identically distributed Rademacher random variables, i.e., with $\mathbb{P}(\boldsymbol{r}_n = +1) = \mathbb{P}(\boldsymbol{r}_n = -1) = 1/2$. This quantity can be seen as a measure of *overfitting* during training over the class of functions $\mathcal{F}$ [35]. In general, to avoid overfitting during training, and to ensure an improved generalization performance, we choose models with small classifier complexity.

Applying the above definition to our multi-agent case, we define the individual empirical Rademacher complexity of agent k for samples $h^{(k)} \triangleq \{h_{k,1}, \ldots, h_{k,N_k}\}$ as

$$\mathcal{R}(\mathcal{F}_k(h^{(k)})) = \mathbb{E}_r \sup_{f_k \in \mathcal{F}_k} \left| \frac{1}{N_k} \sum_{n=1}^{N_k} \boldsymbol{r}_n f_k(h_{k,n}) \right|, \tag{54}$$

and its expected Rademacher complexity, for features $\boldsymbol{h}_{k,1}, \boldsymbol{h}_{k,2}, \ldots, \boldsymbol{h}_{k,N}$ as

$$\rho_k \triangleq \mathbb{E}_{h_k} \mathcal{R}(\mathcal{F}_k(\boldsymbol{h}^{(k)})), \tag{55}$$

which represents the Rademacher complexity of the k-th classifier structure, *averaged* over the feature distribution. We also define the (expected) *network Rademacher complexity* according to:

$$\rho \triangleq \sum_{k=1}^{K} \pi_k \rho_k, \tag{56}$$

which represents an average complexity across all agents in the network, weighted by their centrality scores (given by the elements of the Perron eigenvector π).

A. Learning Consistency

In Theorem 1, we show that the SML strategy consistently learns the truth during the prediction phase, with high probability as the number of training samples grows and for a moderately complex classifier structure. Before introducing the theorem, we define the following two quantities (we assume $N_k > 0$ for all k):

$$\alpha_k \triangleq \frac{N_{\mathsf{max}}}{N_k}, \qquad \alpha \triangleq \sum_{k=1}^{K} \pi_k \alpha_k, \tag{57}$$

with $N_{\mathsf{max}} \triangleq \max_k N_k$. The *individual imbalance penalty* α_k quantifies how distinct the number of training samples of agent k is compared with N_{max}. The *network imbalance penalty* α is the average of α_k over the network, and it quantifies how unequal the training samples are across different agents. For example, if all agents possess the same number of training samples, i.e., $N_k = N_{\mathsf{max}}$, for all $k = 1, 2, \ldots, K$, then α assumes minimal value with $\alpha = 1$. The value of α tends to grow when agents have very different number of training samples, e.g., when, for some k, $N_k \ll N_{\mathsf{max}}$.

Moreover, we assume that the target risk R^o is strictly smaller than $\log 2$. To understand the meaning of such assumption, we first consider a single agent k, for which $\mathsf{R}_k^o < \log 2$. This assumption eliminates the case where the classifier makes *uninformed* decisions of the form:

$$\widehat{p}_k(\gamma|h) = \frac{1}{2}, \text{ for any } h \in \mathcal{H}_k \text{ and } \gamma \in \Gamma, \tag{58}$$

i.e., where the classification decision is independent of the input feature vector. In this case, from (29), $f_k(h) = 0$ for any $h \in \mathcal{H}_k$, which in view of (33) implies $R_k(f_k) = \log 2$. This situation arises, for example, when the classifier structure is not complex enough to address the classification task at hand. Requiring $\mathsf{R}_k^o < \log 2$ guarantees that the classifier k performs better than a classifier that randomly assigns labels $+1$ and -1 with equal probability. Requiring that the *network* target risk satisfies $\mathsf{R}^o < \log 2$ is an even weaker assumption, since it establishes this bound to the risk values averaged over the graph. For example, suppose that, in a $K-$agent network, $K-1$ classifiers yield uninformed decisions like in (58), for which $\mathsf{R}_k^o = \log 2$. To satisfy $\mathsf{R}^o < \log 2$ on a network level, it suffices that one classifier performs better than the uninformed ones.

The next theorem characterizes the consistency of the SML strategy during the prediction phase in terms of an exponential lower bound on the probability of consistent learning in (50).

Theorem 1 (SML Consistency): For the logistic risk, assume that $\mathsf{R}^o < \log 2$ and that $f_k\big(h\big) \leq \beta$ for every $h \in \mathcal{H}_k$, $f_k \in \mathcal{F}_k$ and $k = 1, 2, \ldots, K$, with $\beta > 0$. Assume $\rho < \mathscr{E}(\mathsf{R}^o)$, where $\mathscr{E}(\mathsf{R}^o)$ is exactly computed in (122) and can be approximated as (see Fig. 11 in Appendix B):

$$\mathscr{E}(\mathsf{R}^o) \approx 0.2812\left(1 - \frac{\mathsf{R}^o}{\log 2}\right). \tag{59}$$

Then, we have the following bound for the probability of consistent learning, defined in (50):

$$P_c \geq 1 - 2\exp\left\{-\frac{8N_{\mathsf{max}}}{\alpha^2\beta^2}\Big(\mathscr{E}(\mathsf{R}^o) - \rho\Big)^2\right\}. \tag{60}$$

Proof: See Appendix B. ■

Theorem 1 has at least two important implications. First, if the network-average Rademacher complexity ρ is smaller than the function $\mathscr{E}(\mathsf{R}^o)$, then the probability of consistent learning is bounded in an exponential way. Now, the function $\mathscr{E}(\mathsf{R}^o)$ is an error exponent that determines how fast the probability of consistent learning approaches 1. It is a function of the optimal risk R^o — see the definition in (122). An excellent approximation for $\mathscr{E}(\mathsf{R}^o)$ is (59), showing that such exponent quantifies how close the target risk is to the $\log 2$ risk boundary. As already discussed, the $\log 2$ risk boundary corresponds to the risk associated with a binary classifier that randomly classifies samples with labels $+1$ and -1. The closer the target model is to the $\log 2$ risk, the smaller the value of $\mathscr{E}(\mathsf{R}^o)$. In other words, smaller values of $\mathscr{E}(\mathsf{R}^o)$ are symptomatic of more difficult classification problems. Therefore, Eq. (60) reveals a remarkable interplay between the inherent difficulty of the classification problem (quantified inversely by $\mathscr{E}(\mathsf{R}^o)$) and the complexity of the classifier structure (quantified by ρ). Ideally, we would like to have simple classification problems (i.e., higher values of $\mathscr{E}(\mathsf{R}^o)$) and low Rademacher complexity ρ. Notably, both indices are *network* indices that embody the network structure inside them.

Second, the exponent characterizing the bound in (60) depends on the size of the training sets at the individual agents (through the network imbalance penalty α and the maximum training-set size), and the bounding constant β. In particular, we see from (60) that the exponent (and, hence, the probability of consistent learning) increases if we have larger training sets (i.e., larger N_{max} and/or smaller α) and more constrained class of functions (i.e., smaller β).

Note that by considering an agent-dependent bound β_k on the logit function $f_k(h)$, under similar assumptions as in Theorem 1 and adjusting Theorem 3 in Appendix C, we can achieve the following alternative bound for the probability of consistent learning:

$$P_c \geq 1 - 2\exp\left\{-\frac{8N_{\mathsf{max}}}{\alpha_\beta^2}\Big(\mathscr{E}(\mathsf{R}^o) - \rho\Big)^2\right\}, \tag{61}$$

where we define the network average quantity

$$\alpha_\beta \triangleq \sum_{k=1}^{K} \pi_k \alpha_k \beta_k. \tag{62}$$

The expression in (61) is useful to encompass scenarios in which different agents have different bounding constants, in particular, the case in which some agent k' has a dominant bound on its logit function with respect to other agents, such that $\beta_{k'} \gg \beta_k$ for all $k \neq k'$. More generally, when $\pi_{k'}\alpha_{k'}\beta_{k'} \gg \pi_k\alpha_k\beta_k$ for some agent k', the probability of consistent learning is approximately lower bounded as:

$$P_c \geq 1 - 2\exp\left\{-\frac{8N_{\mathsf{max}}}{(\pi_{k'}\beta_{k'}\alpha_{k'})^2}\Big(\mathscr{E}(\mathsf{R}^o) - \rho\Big)^2\right\}. \tag{63}$$

In summary, the bound in (60) can be used to establish conditions under which the probability of consistent learning approaches 1 exponentially fast as the training-set sizes increase. To this end, we must observe that the quantity ρ itself depends on the training-set sizes. Accordingly, it is necessary to obtain an estimate (or a bound) for the network-average Rademacher complexity. Once this is done, we will be in the position of evaluating the sample complexity of the SML strategy, namely, of evaluating how many samples are necessary to achieve a target probability of consistency. This analysis will be pursued in the next section.

B. Sample Complexity

Under typical classifier structures, the Rademacher complexity scales as $C_k/\sqrt{N_k}$, where N_k is the number of training samples pertaining to agent k, and C_k is a constant quantifying the inherent complexity of the $k-$th classifier structure [30]. As an example, we will show in the next section how the Rademacher complexity behaves for the particular structure of multilayer perceptrons, and provide an upper bound for a given design of number of hidden layers and hidden units.

Now, assuming that the Rademacher complexity of each classifier k is bounded as $C_k/\sqrt{N_k}$, the *network* Rademacher complexity will be bounded as:

$$\rho \leq \sum_{\ell=1}^{K} \pi_k \frac{C_k}{\sqrt{N_k}} = \frac{1}{\sqrt{N_{\max}}} \underbrace{\sum_{k=1}^{K} \pi_k C_k \sqrt{\alpha_k}}_{\triangleq \mathsf{C}}, \tag{64}$$

where C is an average constant that mixes the individual complexity constants C_k, accounting for the Perron eigenvector entries π_k and the individual imbalance penalties α_k. In the case where (64) is satisfied, exploiting (60) we obtain the bound:

$$P_c \geq 1 - 2\exp\left\{-\frac{8N_{\max}}{\alpha^2\beta^2}\left(\mathscr{E}(\mathsf{R}^o) - \frac{\mathsf{C}}{\sqrt{N_{\max}}}\right)^2\right\}. \tag{65}$$

Equation (65) shows that when $N_{\max}$ scales to infinity (with the relative proportions between $N_{\max}$ and N_k kept fixed, i.e., α kept constant), the probability of consistent learning approaches 1 exponentially fast. Moreover, Eq. (65) can be used to carry out a sample-complexity analysis of the SML strategy, as stated in the forthcoming theorem.

Theorem 2 (SML Sample Complexity): Assume $\rho_k \leq C_k/\sqrt{N_k}$ for some constant $C_k > 0$ for all $k = 1, 2, \ldots, K$, and let

$$\mathsf{C} \triangleq \sum_{k=1}^{K} \pi_k C_k \sqrt{\alpha_k}. \tag{66}$$

Then, for the logistic risk, consistent learning takes place with probability at least $1-\varepsilon$, if the maximum number of training samples across the network satisfies:

$$N_{\max} > \left(\frac{\mathsf{C}}{\mathscr{E}(\mathsf{R}^o)}\right)^2 \left(1 + \frac{\alpha\beta}{2\mathsf{C}}\sqrt{\frac{1}{2}\log\left(\frac{2}{\varepsilon}\right)}\right)^2. \tag{67}$$

Proof: See Appendix D. ■

We now examine how the relevant system parameters appearing in (67) influence the sample complexity.

1) Target Performance: The desired probability of consistent learning, $1-\varepsilon$, influences the bound in (67) only logarithmically, and, hence, has a mild effect on the necessary number of training samples.

2) Term α: Term α quantifies how unequal the number of training samples is across agents. Larger values of α imply that agents have a more uneven number of samples, and thus require that $N_{\max}$ be increased to compensate for the lack of data at some agents in the network.

3) Term β: Term β corresponds to the bound of the output of the logit function f_k and, hence, increasing β corresponds to increasing the possible logit functions to choose from. Accordingly, from (67) we see that the larger the value of β, the larger the number of training samples necessary to result in highly probable consistent learning.

4) Term C: The constant C quantifies the complexity of the chosen classifier structure. The necessary number of training samples grows quadratically with an increase in the classifiers' complexity.

5) Term $\mathscr{E}(\mathsf{R}^o)$: As explained before, the term $\mathscr{E}(\mathsf{R}^o)$ quantifies (inversely) the difficulty of the classification problem. Smaller values of $\mathscr{E}(\mathsf{R}^o)$ are representative of more difficult classification problems, and accordingly correspond to the necessity of acquiring more training samples.

6) Role of the Network: Given the networked nature of our inference problem, described in the early Sec. II-A, and the fact that the conditions for consistent learning are given with respect to network average values as seen in (49), it is expected that the network structure plays a significant role in the results of Theorems 1 and 2. The network influence, as well as the graph topology, are captured by the presence of the Perron eigenvector π in the probability expression for consistent learning, through the network terms α, ρ and R^o, namely, the network imbalance penalty, the network Rademacher complexity and the network target risk.

The Perron eigenvector represents the centrality or influence of each agent in determining the values of the pertinent network terms, e.g., a more influential agent k has more power to steer the value of the network target risk R^o towards its own private target risk R_k^o. For doubly-stochastic combination matrices, the vector π is a vector with elements $1/K$ [22], thus influence is uniform across agents. While the dependence on the structure connecting the classifiers is not found in existing statistical bounds in the literature for ensembles of classifiers [34], [36], similar network average dependences are key quantities in distributed estimation and social learning [7], [9], [22]. For example, in social learning, convergence occurs around the hypothesis $\gamma \in \Gamma$ that minimizes the network average KL divergence, i.e., $\sum_{k=1}^{K} \pi_k D_{\mathsf{KL}}[L_k(\gamma^o)|L_k(\gamma)]$ [7], [8], [9].

In summary, Eq. (67) quantifies how the main system parameters act on the SML sample complexity. Specifically, we see that: $i)$ owing to the exponential bound, the dependence on the target error probability ε is mild; $ii)$ the number of samples to achieve a prescribed performance increases with the "size" of the class of functions (higher β), the heterogeneity among classifiers (higher α), the complexity of classifiers (higher C), and the difficulty of the learning problem (lower $\mathscr{E}(\mathsf{R}^o)$); and $iii)$ the network role is encoded in the Perron eigenvector that appears in the *network-averaged* values α, C, and R^o.

In the next section, we discuss in greater detail the expression of the classifier complexity ρ for feedforward neural networks as a function of the classifier structure, i.e., number of hidden layers (depth of the neural network) and weight of hidden units (width of the neural network), and the size of the training dataset.

C. Neural Network Complexity

In this section, we complement the result from Theorem 1 by showing that the term ρ in (56), which depends on the Rademacher complexity of the classifier, vanishes with an increasing number of training samples in the case of the MultiLayer Perceptron (MLP). Assume that one classifier has the structure of a MLP with L layers (excluding the input layer) and activation function σ. We drop index k as we are referring to a single MLP. Each layer ℓ consists of n_ℓ nodes, equivalently the size of layer ℓ is given by n_ℓ.

At each node $m = 1, 2, \dots, n_\ell$ of layers $\ell = 2, 3, \dots, L$, the following function $g_m^{(\ell)}$ is implemented:

$$g_m^{(\ell)}(h) = \sum_{j=1}^{n_{\ell-1}} w_{mj}^{(\ell)} \sigma\left(g_j^{(\ell-1)}(h)\right). \tag{68}$$

The parameters $w_{mj}^{(\ell)}$ correspond to the elements of the weight matrix W_ℓ of dimension $n_\ell \times n_{\ell-1}$. For the first layer, the function implemented at node m is of the form:

$$g_m^{(1)}(h) = \sum_{j=1}^{n_0} w_{mj}^{(1)} h_j, \tag{69}$$

where the input vector h has dimension n_0. A bias parameter can be incorporated in (69) by considering an additional input element $h_{n_0+1} = 1$.

For a MLP whose purpose is to solve a binary classification problem, we denote the output at layer L by $z \in \mathbb{R}^2$, where $z_m = g_m^{(L)}(h)$ for $m = 1, 2$. The final output is given by applying the softmax function to z, that is,

$$\widehat{p}(+1|h) = \frac{e^{z_1}}{e^{z_1} + e^{z_2}}, \quad \widehat{p}(-1|h) = \frac{e^{z_2}}{e^{z_1} + e^{z_2}}. \tag{70}$$

In this case, the logit function is given by:

$$f^{\mathsf{NN}}(h) = \log \frac{\widehat{p}(+1|h)}{\widehat{p}(-1|h)} = z_1 - z_2, \tag{71}$$

where we say that f^{NN} belongs to a class of functions $\mathcal{F}^{\mathsf{NN}}$, which is parameterized by matrices W_ℓ, for $\ell = 1, 2, \dots, L$, according to (68), (69) and (71).

The general evolution for the Rademacher complexity of class $\mathcal{F}^{\mathsf{NN}}$ described above is well known in the literature as scaling with $C/\sqrt{N}$ [30]. We would like nonetheless to obtain an expression for this complexity, which depends explicitly on the design choices for the MLP, i.e., depending on the depth and weights of the network. The objective is to provide the user with a general guideline on how to choose these parameters for a desired complexity value. With this purpose, a formal upper bound for this complexity is enunciated in Proposition 2 inspired by results from [30] and [37].

Proposition 2 (Rademacher Complexity of Norm-Constrained MLPs): Consider an L-layered multilayer perceptron, satisfying[3] $\|W_\ell\|_1 \leq b$, for every layer $\ell = 1, 2, \dots, L$. Assume that the input vector $x \in \mathbb{R}^{n_0}$ satisfies $\max_i |x_i| \leq c$, and that the activation function $\sigma(x)$ is Lipschitz with constant L_σ with $\sigma(0) = 0$. Then the Rademacher complexity for the set of vectors $\mathcal{F}^{\mathsf{NN}}(x)$ is bounded as:

$$\mathcal{R}(\mathcal{F}^{\mathsf{NN}}(x)) \leq \frac{4}{\sqrt{N}} \left[(2bL_\sigma)^{L-1} bc\sqrt{\log(2n_0)}\right]. \tag{72}$$

Proof: See Appendix E. ■

Assume we have a network of K classifiers, each with a MLP structure. Given Proposition 2, we can explicitly characterize the constant C_k found in (64) as:

$$C_k = 4\left[(2b^{(k)}L_\sigma^{(k)})^{(L^{(k)}-1)} b^{(k)} c^{(k)} \sqrt{\log(2n_0^{(k)})}\right], \tag{73}$$

where we introduce superscript (k) to indicate that the classifier structural parameters can change across different agents. This characterization in association with Theorem 2 can be used to design the MLP architecture, according to the available training samples, or yet to select the number of samples needed for a given set of previously fixed architectures.

V. Simulation Results

A. MNIST Dataset

In the simulations, we consider the MNIST dataset [19], building a binary classification problem aimed at distinguishing digits 0 and 1. We employ a network of 9 spatially distributed agents, where each agent observes only a part of the image (see Fig. 3). These agents wish to collaborate and discover which digit corresponds to the image they are collectively observing.

As we can see in Fig. 3, different agents will observe data with different levels of informativeness, e.g., agents 1 and 9 will dispose of little or no information within their attributed image patch. To overcome this lack of local information, agents are connected through a strongly-connected network, whose combination matrix was generated using an averaging rule [22]. In Fig. 4, we show the network topology.

In the training phase, each agent is provided with a balanced set of 200 labeled images. Using this set of examples, classifiers are independently trained using a MLP with 1 hidden layer with 64 hidden units and $\tanh$ activation function, over 30 training epochs. The updates are performed using a batch size of 10 with learning rate 0.001. The training is repeated 5 times for each agent. The empirical training risk for each classifier over the training epochs can be seen in Fig. 5, where the risk was averaged over the 5 training repetitions.

As expected, in Fig. 5 we see that classifiers 1 and 9 result in the least reliable training performances, i.e., their empirical risks exhibit the most variance across training. This could be

[3]Note that $\|W\|_1$ corresponds to the maximum column sum matrix norm of matrix W.

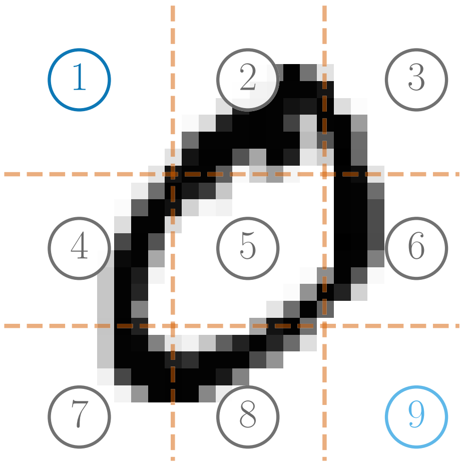

Fig. 3. Each fraction of the image is observed by a different agent. Agents 1 and 9, highlighted in blue, correspond to the least informed agents.

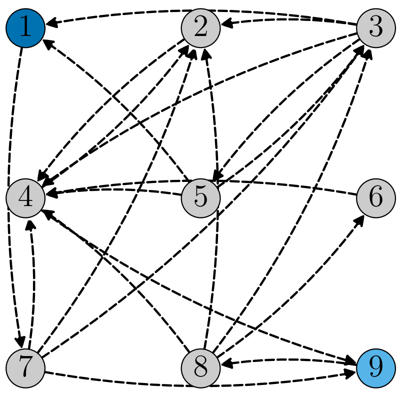

Fig. 4. Topology of the network of agents.

problematic if these agents were to solve the classification problem on their own, but we will see that their individual poor classification performance is mitigated when collaborating within the network.

In the prediction phase, agents observe unlabeled images over time. The nature of images switches at every *prediction cycle*: In the cycle corresponding to interval $i \in [0, 1000)$ agents start observing digits 0. In the following cycle, i.e., $i \in [1000, 2000)$, the nature of images changes to depict digits 1. Then, from instant $i = 2000$ it switches back to digits 0, and so on. We implement the SML strategy based on the adaptive social learning algorithm described in Sec. II-C. In Fig. 6, we see the evolution of the decision variable $\boldsymbol{\lambda}_{1,i}$ for agent 1 with $\delta = 0.01$.

In Fig. 6, we see how, despite the limited information available during training, agent 1 is able to clearly distinguish digits 0 and 1. The instantaneous decision of agent 1 is given by the sign of the decision variable $\boldsymbol{\lambda}_{1,i}$ at any given time, i.e., whether the decision variable lies above or below the decision threshold (the orange dashed line in Fig. 6). Moreover, within the same prediction cycle, we can see in Fig. 6 that the decision variable moves away from the decision threshold in the correct sense, i.e., it becomes more positive under digit 1 and more negative under digit 0.

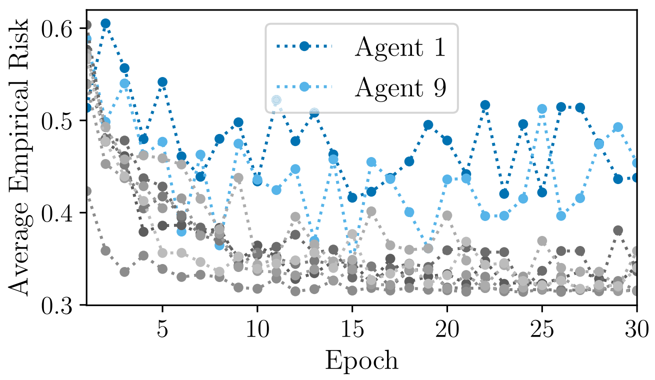

Fig. 5. Empirical training risk averaged over 5 repetitions. The risks corresponding to agents 1 and 9 are highlighted in blue.

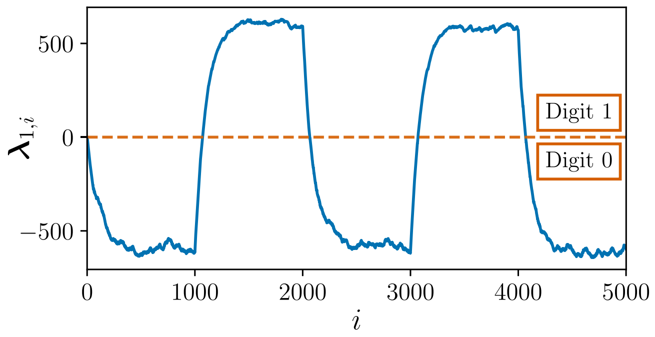

Fig. 6. Evolution of the decision variable for agent 1 over the prediction phase. The observed digit is 0 within interval [0,1000), then it switches every 1000 time instants.

B. Comparison With AdaBoost

We compare the performance of the Social Machine Learning strategy with the classical AdaBoost strategy, as presented in [38]. In Boosting strategies, agents are trained sequentially, yielding a logit statistic $\widetilde{\boldsymbol{f}}_k^{\mathsf{Boost}}$ for each classifier k. The agents in this case are neural network classifiers, with the same architecture as described in the previous example. Once each agent is trained, its performance on the training dataset is evaluated and results in a boosting weight a_k (see [38] for further details on the implementation of AdaBoost). Larger values of a_k indicate that agent k has a better accuracy in the training dataset and makes less mistakes.

During the prediction phase, as agents observe unlabeled data $\boldsymbol{h}_{k,i}$, the decision of an individual agent is given by:

$$\gamma_{k,i}^{\mathsf{Boost}} = \text{sign}\left(\widetilde{\boldsymbol{f}}_k^{\mathsf{Boost}}(\boldsymbol{h}_{k,i})\right), \tag{74}$$

and the collective decision is performed using the boosting weights determined during training, according to:

$$\gamma_i^{\mathsf{Boost}} = \text{sign}\left(\sum_{\ell=1}^{K} a_\ell \gamma_{\ell,i}^{\mathsf{Boost}}\right). \tag{75}$$

Note that computing $\gamma_i^{\mathsf{Boost}}$ requires centralized information, i.e., knowledge of the instantaneous decisions of all agents. We compare this centralized boosting decision with the individual instantaneous decision of agent 1 from the SML strategy, whose decision variable $\boldsymbol{\lambda}_{1,i}$ was seen in Fig. 6. In a strongly-connected network, when agents repeatedly interact

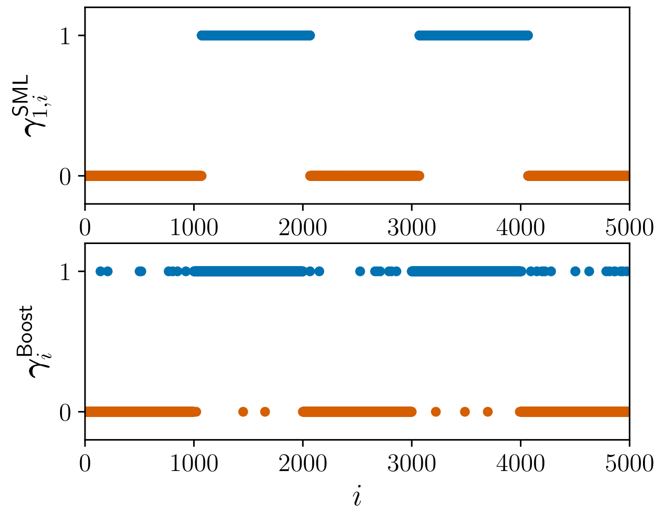

Fig. 7. Comparison of the individual decision of agent 1 within the SML framework and the collective AdaBoost decision. The observed digit is 0 within interval [0,1000), then it switches every 1000 time instants.

with their neighbors, information eventually propagates across all agents, and thus the decision variables $\boldsymbol{\lambda}_{k,i}$ tend to be similar over the network as time grows. The same holds for the decision $\gamma^{\mathsf{SML}}_{k,i}$, defined in (40). Therefore it suffices to compare the decisions under Adaboost with the decisions of agent 1 under the SML strategy.

The comparison can be seen in Fig. 7, for a similar prediction setup as previously described. As a result from training AdaBoost, the lowest boosting weights were obtained for agents $1, 3, 4, 9$. This result is expected since these agents are observing less relevant information (see Fig. 3) and can be regarded as the weakest agents.

In Fig. 7, we see how the SML strategy results in misclassified samples only during short periods after the state transitions occur, whereas the AdaBoost solution makes mistakes throughout the prediction phase. This difference can be explained by the fact that the SML strategy uses the ASL protocol in the prediction phase:

$$\boldsymbol{\lambda}_{k,i} = \underbrace{\sum_{\ell \in \mathcal{N}_k} a_{\ell k}}_{\text{network}} \Big((1-\delta) \underbrace{\boldsymbol{\lambda}_{\ell,i-1}}_{\text{past}} + \underbrace{\widetilde{\boldsymbol{c}}_\ell(\boldsymbol{h}_{\ell,i})}_{\text{new}} \Big), \tag{76}$$

which introduces a combination *over time*, where the past information is combined with new data, and a combination *over the network*, where information is exchanged locally with neighbors. The combination over time introduces an adaptation time necessary for the algorithm to reach steady-state performance. The adaptation time scales as $1/\delta$ — see [9]. Furthermore, due to the decentralized combination over the network, information takes some iterations to propagate throughout the network, as we will further discuss in the example shown in Fig. 8. In contrast, Adaboost does not perform any kind of combination over time (since it does not aggregate information sequentially) or over the network (since the solution is centralized) and therefore there is no adaptation time associated with its behavior. Note that the fact that Adaboost makes instantaneous decisions without aggregating information over time results into a worse performance in steady state (higher error probability) see Fig. 8. In other words, the proposed social machine learning strategy takes significant advantage from integrating information over time, at the price of a small adaptation delay.

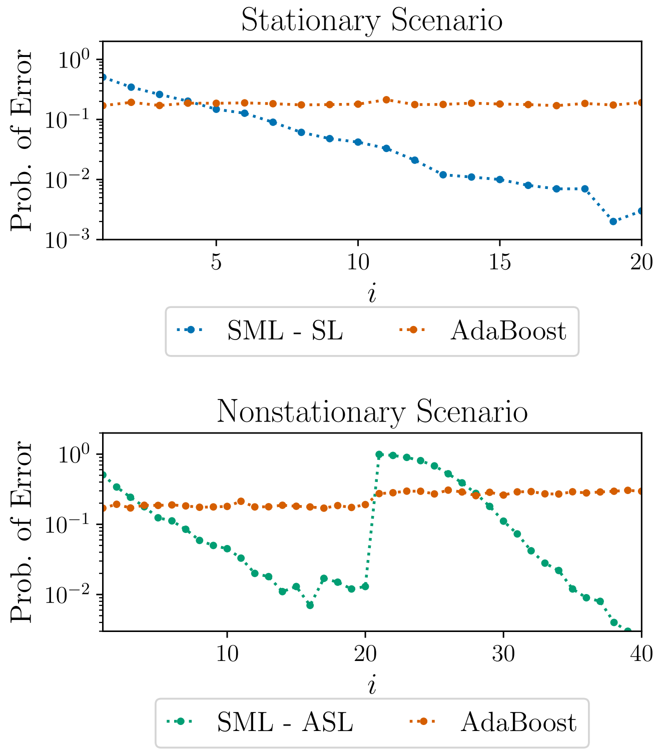

Fig. 8. Evolution of the probability of error for the SML strategy and AdaBoost (centralized decision) estimated from 1000 Monte Carlo runs. *Upper panel*: SML is run with the traditional social learning rule (SL). The true state corresponds to digit 0. *Bottom panel*: SML is run with the adaptive social learning rule (ASL). Until instant $i = 20$, the true state corresponds to digit 0, after which the true state is digit 1.

In the second simulation setup, we can observe how the SML strategy improves its learning performance (i.e., the error probability decreases) over time during the prediction phase. To emphasize this behavior, we reduce the number of hidden units in the neural network structure to 10, and the number of training samples available at each agent to 40. The SML strategy and AdaBoost are trained for the new setup, and deployed in two prediction scenarios: a *stationary* scenario, in which the true underlying class corresponds to digit 0 throughout the simulation period; and a *nonstationary* scenario, when the true underlying class starts at digit 0 and at instant $i = 20$ switches to digit 1.

The SML strategy is implemented considering distinct social learning approaches. In the stationary scenario, we use traditional social learning (SL), implemented with the Bayesian update in (10) and combination rule in (11). In the nonstationary scenario, we use adaptive social learning (ASL), implemented with the adaptive Bayesian update in (21) and combination rule in (11). In Fig. 8, we depict the probability of error of the centralized Boosting algorithm and the SML strategy at agent 1 (now with the choice of parameter $\delta = 0.1$) for the two scenarios. The probability is empirically estimated from 1000 Monte Carlo runs.

In the upper panel of Fig. 8, we note that the SML strategy, associated with the SL protocol, quickly surpasses AdaBoost's performance and attains a significantly improved accuracy over time. Notably this improvement in accuracy exhibits a linear behavior as times progresses. The traditional social learning strategy, although powerful, is not suitable to operate under nonstationary conditions, as discussed in [9]. This is why, in the bottom panel of Fig. 8, we consider instead the SML strategy with the ASL protocol. In this scenario, we can clearly distinguish two prediction cycles, corresponding to the period under different underlying classes of digits. Compared with AdaBoost, SML yields the best performance as time passes. It is able to adapt its predictive behavior in view of the change in the observed digit, and it eventually surpasses the performance of AdaBoost with an adaptation time that scales with $1/\delta$.

This improved performance can be explained by noticing the following aspect. The SML strategy leverages not only information distributed across agents, but also knowledge accumulated over time. We see that by considering, for example, the SML strategy associated with the SL protocol in (16), the evolution of the decision variable of agent k over the prediction phase is governed by the recursion:

$$\boldsymbol{\lambda}_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\Big(\boldsymbol{\lambda}_{\ell,i-1} + \widetilde{\boldsymbol{c}}_\ell(\boldsymbol{h}_{\ell,i})\Big). \tag{77}$$

As we can see in (77), at every instant i, $\boldsymbol{\lambda}_{k,i}$ aggregates information from the past through the term $\boldsymbol{\lambda}_{\ell,i-1}$. The aggregation of past decision variables leverages the fact that the underlying class changes slowly during the prediction phase, thus allowing the classifiers to grow in confidence over time.

We should also note that, in face of a single observation, i.e., at instant $i = 1$ in Fig. 8, AdaBoost outperforms SML in its classification accuracy. As already mentioned, this can be explained by the fact that SML is a decentralized algorithm, i.e., at each iteration, agent k communicates only with its one-hop neighbors. If the agent's neighbors happen to be poorly informed classifiers, then their $1-$iteration decision will also be unreliable. As the time passes, that is, as i grows, this challenge is overcome due to the strong-connectivity of the graph topology, which enables the diffusion of information across all agents. In the example above, this is accomplished around instant $i = 4$, when SML surpasses AdaBoost in performance.

C. Multiple-Hypothesis SML

Consider now the multiple-hypothesis setting, where instead of a binary set we have a set of M hypotheses $\Gamma \triangleq \{0, 1, \ldots, M-1\}$. If likelihood models $L_k(h|\gamma)$ are perfectly known, agents can use the social learning strategy (10)–(11) to update their beliefs. Similarly to (14) and (15), let us make the following change of variables:

$$\boldsymbol{\lambda}_{k,i}(\gamma) \triangleq \log\frac{\boldsymbol{\varphi}_{k,i}(0)}{\boldsymbol{\varphi}_{k,i}(\gamma)}, \quad c_k(\boldsymbol{h}_{k,i},\gamma) \triangleq \log\frac{L_k(\boldsymbol{h}_{k,i}|0)}{L_k(\boldsymbol{h}_{k,i}|\gamma)}, \tag{78}$$

for $\gamma \in \Gamma\setminus\{0\}$. Similarly to (16), we can use (78) to represent (10)–(11) as the following linear recursion:

$$\boldsymbol{\lambda}_{k,i}(\gamma) = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\Big(\boldsymbol{\lambda}_{\ell,i-1}(\gamma) + c_\ell(\boldsymbol{h}_{\ell,i},\gamma)\Big), \quad \gamma\in\Gamma\setminus\{0\}. \tag{79}$$

Likewise, we can write the adaptive social learning recursion, based on steps (21) and (11), as

$$\boldsymbol{\lambda}_{k,i}(\gamma) = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\Big((1-\delta)\boldsymbol{\lambda}_{\ell,i-1}(\gamma) + c_\ell(\boldsymbol{h}_{\ell,i},\gamma)\Big), \tag{80}$$

for $\gamma \in \Gamma\setminus\{0\}$. Note that, in both cases, from knowledge of $\boldsymbol{\lambda}_{k,i}(\gamma)$, we can recover the complete belief vector $\boldsymbol{\varphi}_{k,i}$ by considering that its entries must add up to 1.

An instantaneous classification decision can be made by taking the hypothesis that maximizes the belief $\boldsymbol{\varphi}_{k,i}$:

$$\boldsymbol{\gamma}^{\mathsf{SML}}_{k,i} \triangleq \arg\max_{\gamma\in\Gamma} \boldsymbol{\varphi}_{k,i}(\gamma). \tag{81}$$

The models $c_k(h,\gamma)$ are in practice unknown, therefore we extend next the SML approach to the multiple-hypothesis case.

1) Training Phase: Assume that each agent k possesses a balanced dataset $\{\widetilde{\boldsymbol{h}}_{k,n}, \widetilde{\boldsymbol{\gamma}}_{k,n}\}_{n=1}^{N_k}$ over the M hypotheses in Γ, then under the assumption of uniform priors over the classes, and using Bayes' rule, we can write:

$$c_k(h,\gamma) = \log\frac{L_k(h|0)}{L_k(h|\gamma)} = \log\frac{\widetilde{p}(0|h)}{\widetilde{p}(\gamma|h)}, \quad \gamma\in\Gamma\setminus\{0\}, \tag{82}$$

where $\widetilde{p}(\gamma|h)$ is the posterior probability of class γ given the observation h. In discriminative machine learning strategies, classifiers yield approximate posterior probabilities

$$\widehat{p}_k(\gamma|h;W), \quad \gamma\in\Gamma, \tag{83}$$

where the parameters W (e.g., $W = \{W_\ell\}$ can be the weight matrices of an L-layered neural network) are found by minimizing the cross-entropy risk:

$$R_k(W) \triangleq -\mathbb{E}_{\widetilde{h}_k,\widetilde{\gamma}_k} \log\widehat{p}_k(\widetilde{\boldsymbol{\gamma}}_{k,n}|\widetilde{\boldsymbol{h}}_{k,n};W). \tag{84}$$

Using its training dataset, each agent k can approximate the cross-entropy risk by its empirical counterpart:

$$\widetilde{\boldsymbol{R}}_k(W) \triangleq -\frac{1}{N_k}\sum_{n=1}^{N_k}\log\widehat{p}_k(\widetilde{\boldsymbol{\gamma}}_{k,n}|\widetilde{\boldsymbol{h}}_{k,n};W), \tag{85}$$

whose minimizer is given by:

$$\widetilde{\boldsymbol{W}}_k \triangleq \arg\min_{W\in\mathcal{W}_k} \widetilde{\boldsymbol{R}}_k(W), \tag{86}$$

where the parameter space $\mathcal{W}_k$ depends on the chosen machine architecture (e.g., for a neural network the parameter space depends on the number of layers and neurons per layer).

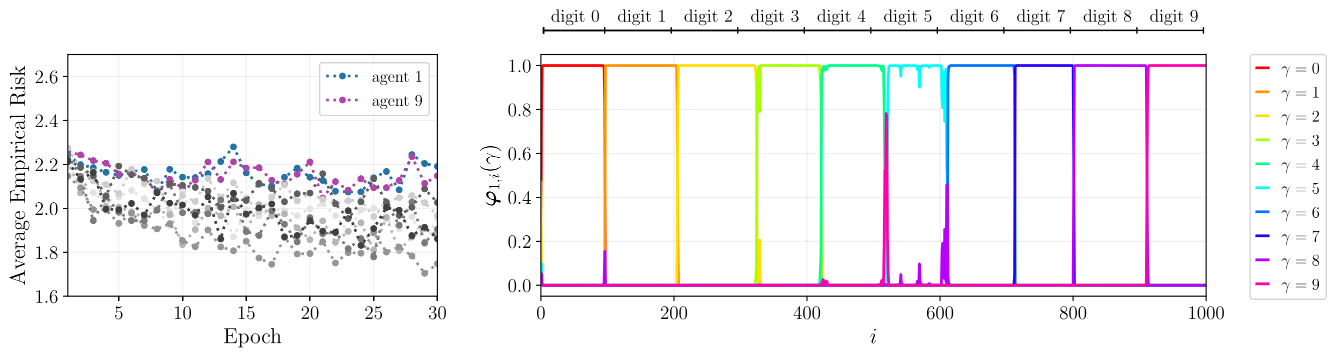

Fig. 9. Social machine learning applied to the MNIST example with classes $\Gamma \triangleq \{0, 1, \dots, 9\}$. (*Left*) Empirical training risk averaged over 5 repetitions for all agents over training epochs. The risk corresponding to agents 1 and 9, the least informed agents, are highlighted in color. (*Right*) Belief evolution over time for agent 1 as the underlying true state changes at every 100 time instants, i.e., $\gamma_0 = 0$ for $i \in [1, 100)$, then $\gamma_0 = 1$ for $i \in [100, 200)$, and so on.

2) Prediction Phase: Upon training the above classifiers, agents can use the learned models to generate the approximate posterior probabilities given an observation $\boldsymbol{h}_{k,i}$:

$$\widehat{p}_k(\gamma|\boldsymbol{h}_{k,i}; \widetilde{\boldsymbol{W}}_k), \quad \gamma \in \Gamma. \tag{87}$$

During prediction, agents can implement the social learning scheme in (79) by replacing the unknown statistic $c_k(\boldsymbol{h}_{k,i}, \gamma)$ by:

$$\widetilde{\boldsymbol{c}}_k(\boldsymbol{h}_{k,i}, \gamma) = \underbrace{\log \frac{\widehat{p}_k(0|\boldsymbol{h}_{k,i}; \widetilde{\boldsymbol{W}}_k)}{\widehat{p}_k(\gamma|\boldsymbol{h}_{k,i}; \widetilde{\boldsymbol{W}}_k)}}_{\triangleq \widetilde{\boldsymbol{f}}_k(\boldsymbol{h}_{k,i},\gamma)} - \widetilde{\boldsymbol{\mu}}_k(\widetilde{\boldsymbol{f}}_k, \gamma), \quad \gamma \in \Gamma \setminus \{0\}, \tag{88}$$

where the second term on the RHS is the empirical training mean conditioned on classes 0 and γ, defined as

$$\widetilde{\boldsymbol{\mu}}_k(\widetilde{\boldsymbol{f}}_k, \gamma) \triangleq \frac{1}{|\mathcal{N}_{k,\gamma}|} \sum_{n \in \mathcal{N}_{k,\gamma}} \widetilde{\boldsymbol{f}}_k(\widetilde{\boldsymbol{h}}_{k,n}, \gamma), \tag{89}$$

where $\mathcal{N}_{k,\gamma} \triangleq \{n : \widetilde{\boldsymbol{\gamma}}_{k,n} \in \{0, \gamma\}\}$, which corresponds to the training samples associated with labels 0 and γ. Note that if we reduce the multiple-hypothesis problem to a binary-hypothesis problem, i.e., $\Gamma = \{0, 1\}$, then (88) reduces back to (38), as we show next. When $\Gamma = \{0, 1\}$, we deal with a single statistic $\widetilde{\boldsymbol{c}}_k(\boldsymbol{h}_{k,i}, 1) \triangleq \widetilde{\boldsymbol{c}}_k(\boldsymbol{h}_{k,i})$ and a single logit model $\widetilde{\boldsymbol{f}}_k(\boldsymbol{h}_{k,i}, 1) \triangleq \widetilde{\boldsymbol{f}}_k(\boldsymbol{h}_{k,i})$, while the empirical training mean conditioned on classes 0 and 1 reduces to:

$$\widetilde{\boldsymbol{\mu}}_k(\widetilde{\boldsymbol{f}}_k, 1) \triangleq \widetilde{\boldsymbol{\mu}}_k(\widetilde{\boldsymbol{f}}_k) = \frac{1}{N_k} \sum_{n=1}^{N_k} \widetilde{\boldsymbol{f}}_k(\widetilde{\boldsymbol{h}}_{k,n}), \tag{90}$$

which is equivalent to the empirical training mean in (39) for the binary case.

We present next an experimental example, showing that the social machine learning strategy can be applied to the multiple-hypothesis classification problem, wherein a single machine is trained using data coming from multiple classes/hypotheses.

3) Multi-Class MNIST: Let us consider a similar setup as the one presented in Sec. V-A, except that now we take into account all classes contained in the MNIST dataset, that is, $M = 10$ where classes represent digits $0, 1, 2, \dots, 9$. We consider the same network shown in Fig. 4, where each agent sees a patch of the handwritten image according to Fig. 3.

In the training phase, each agent is given a balanced set of 1000 labeled images (100 images per digit). Classifiers are then independently trained using a MLP with 1 hidden layer containing 64 hidden units and using $\tanh$ activation function, over 30 training epochs. The updates are performed using a batch size of 10 with learning rate 0.001. The empirical training risk for each classifier over the training epochs can be seen in the left panel of Fig. 9, where the risk was averaged over 5 training repetitions.

In the prediction phase, agents observe unlabeled images over time. The nature of images changes at every 100 samples: In $i \in [0, 100)$ agents start observing digits 0; In $i \in [100, 200)$, agents observe digits 1; In $i \in [200, 300)$, agents observe digits 2, and so on. We implement the SML strategy based on the adaptive social learning algorithm in (80) with the choice $\delta = 0.1$, and with the statistic $\widetilde{\boldsymbol{c}}_k(h, \gamma)$ given in (88). In the right panel of Fig. 9, we see the evolution of the belief $\boldsymbol{\varphi}_{1,i}(\gamma)$ for agent 1 over time i.

Note that, as the digit being observed by the agent changes, the algorithm is able to track these changes in such a way that the belief is maximized at the true state. In this example, the belief is not only maximized at the changing true state, but it is also able to concentrate its full confidence around the truth during most of the prediction phase.

VI. Concluding Remarks

In this work, we focused on the following classification problem. A network of spatially distributed agents observes an event and all agents wish to determine the underlying class exploiting a growing number of streaming observations collected over time. Such problem has been thoroughly studied within the *social learning* literature, where agents possess a set of possible models to explain their observations. By cooperating with neighbors, these agents are able to overcome local limitations and achieve collective consistent learning of the true underlying class.

These methods, however powerful, depend on the prior knowledge of the set of possible models, or *likelihoods*, which characterize the distribution of observations given different underlying classes. In this work, we provided a fully data-driven solution to the aforementioned problem.

We introduced Social Machine Learning, which is a two-step framework that allows aggregating the information perceived by heterogeneous classifiers to improve their decision performance over time, as the classifiers observe streaming data. The classifiers are heterogeneous in the sense that their private observations originate from different distributions. In our approach, we introduce a training phase that, with a finite training dataset, results in approximate models for the unknown data logit statistics. These models are deployed in a prediction phase, where one of the available social learning algorithms can be used.

We show that consistent learning in the prediction phase can be achieved with high probability, and we describe how the number of training samples should scale to yield the desired consistency. Furthermore, the decentralized collaboration among agents results in an increased robustness in face of poorly informed agents, as seen in the simulation results. Simulations also show that our solution continually improves performance over time, leveraging past acquired knowledge to make better informed decisions in the present.

Appendix A
Comparison Between SML and [12]

Both [12] and our work address the issue of unspecified or unknown likelihood models, which arises naturally in practical applications where only data is available for inference. However, our approach is significantly different from the approach proposed in [12], providing completely original contributions. We highlight next the main differences between the two works.

Consider the approach in [12]. The first step of the approach consists in choosing a proper family of parametric distributions. These distributions should satisfy two requirements. On one hand, they must have an analytical form that makes the subsequent steps viable. On the other hand, to guarantee good performance, they must match the underlying physics of the observed phenomenon [12]. They should also satisfy a series of regularity conditions stated in Assumption 7 found in [12]. In contrast, in the SML strategy we make no assumption about the underlying distributions and rely instead on a model-independent and data-oriented approach by using a machine learning architecture, which can be as general as a neural network structure. One practical example is a residual neural network, e.g., the RESNET18 architecture [39] which has millions of parameters. These architectures are particularly suitable when the designer has very little preliminary knowledge concerning the nature of the dataset and would like to choose a structure that can *learn* the "shape" of the underlying distribution using only data.

In the SML strategy, we do not need to constrain our likelihood models to belong to any usual distribution family, such as Gaussian, multinomial, or Poisson. In contrast, the choice of the family of distributions is important in [12] to avoid the difficulties that arise in other steps of the procedure. For instance, in the second step the system designer must be able to derive a closed-form expression for the conjugate prior function, which is only well defined for a limited family of distributions [40]. Without a conjugate prior expression, we also cannot determine a function to update its hyperparameters. Finally, the authors in [12] explain that even after performing the aforementioned steps it is possible that the uncertain likelihood update does not have a closed-form solution, in which case agents must use numerical methods to compute the belief update.

The approach in [12] is valuable and is applicable in many scenarios of interest. However, our approach introduces significant novelty and is especially suited to strongly data-driven settings where little or no prior knowledge is available. In fact, the SML approach contemplates a wide variety of applications, including those envisioned in [12], due to the general architecture used in training. For example, consider a realistic learning problem where agents are trying to detect the underlying class of a stream of images consisting of handwritten digits. We have shown in this work that the SML is particularly suitable for such applications. In our example, each agent trains a multilayer perceptron using their training datasets, by solving an empirical minimization problem. The samples from the MNIST dataset consist of 786 pixels, which are partly observed by agents in such a way that each agent sees an image patch with ~ 90 pixels. It is unclear how the strategy proposed in [12] could be used in this case, first, due to the more diverse statistics of the data samples compared to the distribution families considered in [12], and second, due to the high-dimensionality of the samples.

We show next that, with reference to an example proposed in [12], where features follow a 2-dimensional Gaussian distribution, the SML strategy delivers better performance than the strategy in [12], with the advantage of not requiring knowledge about the parameterization of the true distributions.

A. 2-Dimensional Gaussian Example

Inspired by the 2-dimensional Gaussian example in [12], we consider the following simulation setup with binary classes, i.e., $\gamma \in \{-1, +1\}$. We consider the same 4-agent network used in [12], where the combination matrix is given by:

$$A = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0 & 0.5 \end{bmatrix}. \tag{91}$$

Define the following distributions:

$$g_1 \triangleq \mathcal{G}(\mathsf{m}_1, \Sigma_1), \qquad g_2 \triangleq \mathcal{G}(\mathsf{m}_2, \Sigma_2), \tag{92}$$

where $\mathcal{G}(\mathsf{m}, \Sigma)$ represents a multivariate Gaussian distribution with mean vector m and covariance matrix Σ. In this example, we choose

$$\mathsf{m}_1 = \mathsf{m}_2 \triangleq \begin{bmatrix} 0 & 0 \end{bmatrix}, \tag{93}$$

and

$$\Sigma_1 \triangleq \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \Sigma_2 \triangleq \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}. \tag{94}$$

Table I shows the correct likelihood models for each agent, i.e., the true distribution of their observations h given different hypotheses γ. Note that agent 2 is the only informative agent, whose likelihoods are different given different classes.

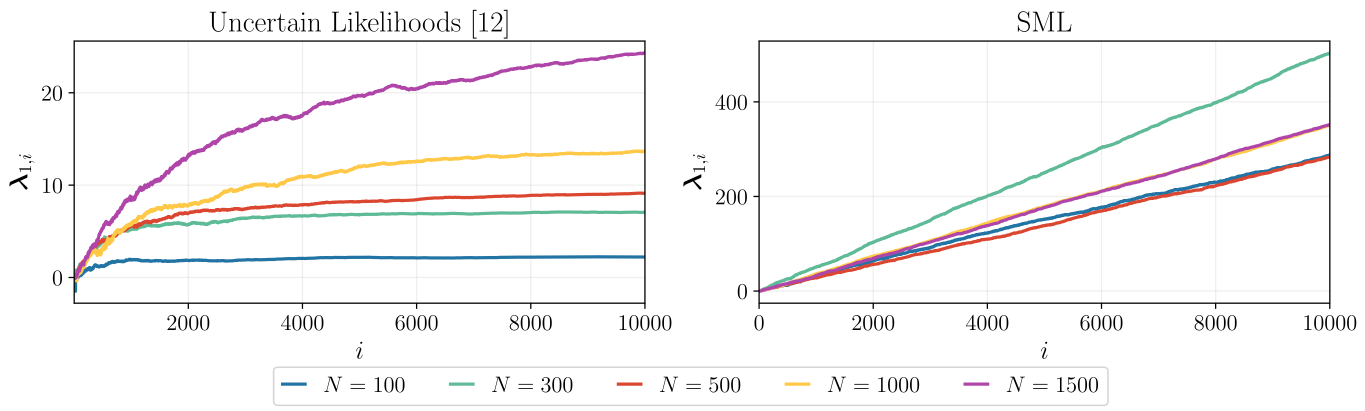

Fig. 10. Evolution over time of the log-ratio of beliefs for agent 1, averaged across 10 experiments regarding the 2-D Gaussian example using two different approaches. (*Left*) Social learning with uncertain models [12]. (*Right*) Social machine learning (SML).

TABLE I
LIKELIHOOD MODELS SETUP

	agent 1	agent 2	agent 3	agent 4
$L_k(h\|+1)$	g_1	g_1	g_1	g_1
$L_k(h\|-1)$	g_1	g_2	g_1	g_1

We implement the strategy with uncertain likelihoods proposed in [12], assuming that agents use multivariate Gaussian distributions as the parameterized family of distributions. We assume that all agents possess the same number of training samples, i.e., $N_1 = N_2 = N_3 = N_4 \triangleq 2N$, where N corresponds to the number of training samples per class. During prediction, we assume the true state is $+1$. The details in implementation follow the steps in [12].

For the sake of comparison with the SML strategy, we choose to display the evolution of the log-ratio of beliefs, namely

$$\boldsymbol{\lambda}_{k,i} \triangleq \log \frac{\boldsymbol{\varphi}_{k,i}(+1)}{\boldsymbol{\varphi}_{k,i}(-1)} \tag{95}$$

which can be seen in the left panel of Fig. 10 for agent 1 and for different sizes of training sets, i.e., for different N. Note that for all training scenarios, the curve tends to converge to a fixed positive value, which implies that, on one hand, the belief is maximized at the true hypothesis $+1$, i.e., $\boldsymbol{\varphi}_{1,i}(+1) > \boldsymbol{\varphi}_{1,i}(-1)$ for large i, and on the other hand, that the belief at the wrong hypothesis $\boldsymbol{\varphi}_{1,i}(-1)$ does not vanish, i.e., $\boldsymbol{\varphi}_{k,i}(-1) > 0$ for large i.

We apply the SML approach to the aforementioned setup. We consider that each agent can train a feedforward neural network, with 2 hidden layers with 10 hidden units each and tanh activation function. The training is performed over 1000 epochs with learning rate 0.0001. We see in the right panel of Fig. 10 the evolution of the log-ratio of beliefs over time for different sizes of training sets. For all training setups, the curves grow linearly with increasing time i. This implies not only that the belief is maximized at the true hypothesis, but also that for large i the belief corresponding to the wrong hypothesis -1 vanishes, i.e., $\boldsymbol{\varphi}_{k,i}(-1) \to 0$, which means that $\boldsymbol{\varphi}_{k,i}(+1) \to 1$ (since the entries of the belief add up to 1).

Note that the profound difference between the behavior of the log-ratio of beliefs implies a significant difference in performance between the two strategies. In the SML strategy, this diverges linearly, as it happens in traditional social learning with *known* likelihood models. This linear growth implies an exponentially fast convergence of the belief vector toward the right hypothesis, with a probability of correct learning that improves steadily (and exponentially) as time elapses. These benefits are lost with the approach in [12], since the log-ratio of beliefs is no longer diverging exponentially.

APPENDIX B
PROOF OF THEOREM 1

Before detailing the proof of Theorem 1, we provide a roadmap of the proof. The proof starts from Lemma 1 in this same Appendix, resulting in the lower bound (98). On the RHS of (98), there are two probability terms that should be upper bounded. The first one is bounded using Eq. (132) in Theorem 3 (found in Appendix C), while the second one is bounded using Lemma 2 (found in Appendix F) and then Eq. (131) in Theorem 3. After some algebraic manipulations, we reach the final result of Theorem 1.

We present next Lemma 1, which provides a lower bound on the probability of consistent learning. We denote the total expected value of $f_k(\widetilde{\boldsymbol{h}}_{k,n})$ by:

$$\mu_k(f_k) \triangleq \mathbb{E}_{\widetilde{h}_k} f_k(\widetilde{\boldsymbol{h}}_{k,n}) = \frac{\mu_k^+(f_k) + \mu_k^-(f_k)}{2}, \tag{96}$$

where we considered equal priors over the two classes $+1$ and -1. We also denote its average across the network by:

$$\mu(f) \triangleq \sum_{\ell=1}^{K} \pi_k \mu_k(f_k) \overset{\text{(a)}}{=} \frac{\mu^+(f) + \mu^-(f)}{2}, \tag{97}$$

where (a) follows from using (96) and the definition of $\mu^+(f)$ and $\mu^-(f)$ found in (45).

Lemma 1 (Probability Bound for Consistent Learning): For any $d > 0$, we have that:

$$P_c \geq 1 - \mathbb{P}\left(\left|\widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}})\right| \geq d\right) - \mathbb{P}\left(R(\widetilde{\boldsymbol{f}}) \geq \Delta\right), \tag{98}$$

where $\Delta \triangleq \log(1+e^{-d})$ and P_c is the probability of consistent learning defined in (50).

Proof: Define the following events, which will be used in the proof:

$$\mathcal{A} \triangleq \left\{ \left| \mu(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \right| \geq \frac{\mu^+(\widetilde{\boldsymbol{f}}) - \mu^-(\widetilde{\boldsymbol{f}})}{2} \right\}, \tag{99}$$

$$\mathcal{B} \triangleq \left\{ \frac{\mu^+(\widetilde{\boldsymbol{f}}) - \mu^-(\widetilde{\boldsymbol{f}})}{2} > d \right\}. \tag{100}$$

First, in view of (50) and using de Morgan's law [41], we can write:

$$\begin{aligned}
1 - P_c &= \mathbb{P}\Big(\Big\{ \mu^+(\widetilde{\boldsymbol{f}}) \leq \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \Big\} \bigcup \Big\{ \mu^-(\widetilde{\boldsymbol{f}}) \geq \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \Big\} \Big) \\
&\overset{\text{(a)}}{=} \mathbb{P}\Big(\Big\{ \mu^+(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}}) \leq \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}}) \Big\} \\
&\qquad \bigcup \Big\{ \mu^-(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}}) \geq \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}}) \Big\} \Big) \\
&\overset{\text{(b)}}{=} \mathbb{P}\left(\left\{ \mu^+(\widetilde{\boldsymbol{f}}) - \frac{\mu^+(\widetilde{\boldsymbol{f}}) + \mu^-(\widetilde{\boldsymbol{f}})}{2} \leq \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}}) \right\} \right. \\
&\qquad \left. \bigcup \left\{ \mu^-(\widetilde{\boldsymbol{f}}) - \frac{\mu^+(\widetilde{\boldsymbol{f}}) + \mu^-(\widetilde{\boldsymbol{f}})}{2} \geq \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}}) \right\} \right) \\
&= \mathbb{P}\left(\left\{ \frac{\mu^+(\widetilde{\boldsymbol{f}}) - \mu^-(\widetilde{\boldsymbol{f}})}{2} \leq -\Big(\mu(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \Big) \right\} \right. \\
&\qquad \left. \bigcup \left\{ \frac{\mu^+(\widetilde{\boldsymbol{f}}) - \mu^-(\widetilde{\boldsymbol{f}})}{2} \leq \mu(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \right\} \right) \\
&\overset{\text{(c)}}{=} \mathbb{P}(\mathcal{A}) \\
&\overset{\text{(d)}}{=} \mathbb{P}\Big(\mathcal{A}, \mathcal{B}) + \mathbb{P}\Big(\mathcal{A}, \overline{\mathcal{B}} \Big) \\
&\overset{\text{(e)}}{\leq} \mathbb{P}\left(\left| \mu(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \right| \geq d \right) + \mathbb{P}\left(\frac{\mu^+(\widetilde{\boldsymbol{f}}) - \mu^-(\widetilde{\boldsymbol{f}})}{2} \leq d \right),
\end{aligned} \tag{101}$$

where in (a) we subtract $\mu(\widetilde{\boldsymbol{f}})$ from both terms within the probability operator, in (b) we replace $\mu(\widetilde{\boldsymbol{f}})$ with (97), and (c) follows from the following relation:

$$\begin{aligned}
&\left\{ \frac{\mu^+(\widetilde{\boldsymbol{f}}) - \mu^-(\widetilde{\boldsymbol{f}})}{2} \leq -\Big(\mu(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \Big) \right\} \\
&\bigcup \left\{ \frac{\mu^+(\widetilde{\boldsymbol{f}}) - \mu^-(\widetilde{\boldsymbol{f}})}{2} \leq \mu(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \right\} \\
&\Leftrightarrow \left\{ \left| \mu(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \right| \geq \frac{\mu^+(\widetilde{\boldsymbol{f}}) - \mu^-(\widetilde{\boldsymbol{f}})}{2} \right\} \triangleq \mathcal{A}.
\end{aligned} \tag{102}$$

In (d), we used the law of total probability, and (e) follows from:

$$\mathcal{A} \bigcap \mathcal{B} \Rightarrow \left\{ \left| \mu(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) \right| \geq d \right\}, \tag{103}$$

where $\mathcal{B}$ is defined in (100), and also from the fact that the probability of intersection of two events is upper bounded by the probability of one of the events.

Let us address the second probability term on the RHS of (101). Consider the average network risk evaluated on the training samples $(\widetilde{\boldsymbol{h}}_{k,n}, \widetilde{\gamma}_{k,n})$, computed for a given function $f_k \in \mathcal{F}_k$:

$$\begin{aligned}
&\sum_{k=1}^{K} \pi_k R_k(f_k) \\
&= \sum_{k=1}^{K} \pi_k \mathbb{E}_{\widetilde{h}_k, \widetilde{\gamma}_k} \log\left(1 + \exp\Big(-\widetilde{\gamma}_{k,n} f_k(\widetilde{\boldsymbol{h}}_{k,n}) \Big) \right) \\
&\overset{\text{(a)}}{\geq} \sum_{k=1}^{K} \pi_k \log\left(1 + \exp\Big(-\mathbb{E}_{\widetilde{h}_k, \widetilde{\gamma}_k} \widetilde{\gamma}_{k,n} f_k(\widetilde{\boldsymbol{h}}_{k,n}) \Big) \right) \\
&\overset{\text{(b)}}{\geq} \log\left(1 + \exp\left(-\sum_{k=1}^{K} \pi_k \mathbb{E}_{\widetilde{h}_k, \widetilde{\gamma}_k} \widetilde{\gamma}_{k,n} f_k(\widetilde{\boldsymbol{h}}_{k,n}) \right) \right) \\
&\overset{\text{(c)}}{\geq} \log\left(1 + \exp\left(\frac{1}{2} \sum_{k=1}^{K} \pi_k \mathbb{E}_{L_k(-1)} f_k(\widetilde{\boldsymbol{h}}_{k,n}) \right.\right. \\
&\qquad \left.\left. - \frac{1}{2} \sum_{k=1}^{K} \pi_k \mathbb{E}_{L_k(+1)} f_k(\widetilde{\boldsymbol{h}}_{k,n}) \right) \right) \\
&\overset{\text{(d)}}{=} \log\left(1 + \exp\left(-\frac{\mu^+(f) - \mu^-(f)}{2} \right) \right),
\end{aligned} \tag{104}$$

where in (a) and (b) we used Jensen's inequality with the convexity of function $\log(1+e^x)$. In (c), we used the assumption of uniform priors during training, and in (d) we used the definition of the conditional means averaged over the network found in (41) and (45). From (104), we have the following implication for a given $f_k \in \mathcal{F}_k$ for $k = 1, 2, \ldots, K$:

$$\frac{\mu^+(f) - \mu^-(f)}{2} \leq d \Rightarrow \sum_{k=1}^{N} \pi_k R_k(f_k) \geq \log\left(1 + e^{-d}\right). \tag{105}$$

Replace f_k in (105) by $\widetilde{\boldsymbol{f}}_k$ (i.e., the models obtained in the training phase). Then (105) results in:

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{k=1}^{N} \pi_k R_k(\widetilde{\boldsymbol{f}}_k) \geq \log\left(1 + e^{-d}\right) \Big) \\
&\quad \geq \mathbb{P}\left(\frac{\mu^+(\widetilde{\boldsymbol{f}}) - \mu^-(\widetilde{\boldsymbol{f}})}{2} \leq d \right) = \mathbb{P}\left(\overline{\mathcal{B}}\right).
\end{aligned} \tag{106}$$

Using (106) in (101) and defining $\Delta \triangleq \log(1 + e^{-d})$ yields the bound in (98). ■

Proof of Theorem 1: From Lemma 1, we obtain the lower bound in (98) for the probability of consistent learning. Next, we need to examine each of the terms on the RHS of (98).

Regarding the first term, we can write:

$$\left| \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}}) \right| \leq \sup_{f \in \mathcal{F}} \left| \widetilde{\boldsymbol{\mu}}(f) - \mu(f) \right|, \tag{107}$$

which implies that

$$\mathbb{P}\left(\left| \widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}}) \right| \geq d \right) \leq \mathbb{P}\left(\sup_{f \in \mathcal{F}} \left| \widetilde{\boldsymbol{\mu}}(f) - \mu(f) \right| \geq d \right), \tag{108}$$

providing us with a *uniform* bound for the first term on the RHS of (98). We will now call upon Theorem 3 (Appendix C) to obtain an exponential upper bound on the RHS of (108). Using (132) with the choice $x = d$ in (108) yields:

$$\mathbb{P}\left(\left|\widetilde{\boldsymbol{\mu}}(\widetilde{\boldsymbol{f}}) - \mu(\widetilde{\boldsymbol{f}})\right| \geq d\right) \leq \exp\left\{\frac{-(d-4\rho)^2 N_{\mathsf{max}}}{2\alpha^2\beta^2}\right\}, \tag{109}$$

for any positive d such that $d > 4\rho$.

Next, we examine the second term on the RHS of (98). Using Lemma 2 (Appendix F), with the choice $x = \Delta - \mathsf{R}^o$, we can derive the following uniform upper bound:

$$\mathbb{P}\left(R(\widetilde{\boldsymbol{f}}) \geq \Delta\right) \leq \mathbb{P}\left(\sup_{f\in\mathcal{F}}\left|\widetilde{\boldsymbol{R}}(f) - R(f)\right| \geq \frac{\Delta - \mathsf{R}^o}{2}\right), \tag{110}$$

for any positive d such that $\Delta = \log(1+e^{-d}) > \mathsf{R}^o$. Such d exists since, by assumption, $\mathsf{R}^o < \log 2$.

Next, consider Eq. (131) of Theorem 3 (Appendix C), with the choice $x = (\Delta - \mathsf{R}^o)/2$ and function $\phi(x) = \log(1+e^x)$, which is a function with Lipschitz constant $L_\phi = 1$. Replacing (131) with these choices into (110) results in the bound:

$$\mathbb{P}\left(R(\widetilde{\boldsymbol{f}}) \geq \Delta\right) \leq \exp\left\{\frac{-(\frac{\Delta-\mathsf{R}^o}{2} - 4\rho)^2 N_{\mathsf{max}}}{2\alpha^2\beta^2}\right\}, \tag{111}$$

for any d such that $(\Delta - \mathsf{R}^o)/2 > 4\rho$. Using (109) and (111) in (98) results in the following bound on the probability of consistent learning

$$\begin{aligned} P_c \geq 1 &- \exp\left\{\frac{-8(\frac{d}{4}-\rho)^2 N_{\mathsf{max}}}{\alpha^2\beta^2}\right\} \\ &- \exp\left\{\frac{-8(\frac{\Delta-\mathsf{R}^o}{8}-\rho)^2 N_{\mathsf{max}}}{\alpha^2\beta^2}\right\}, \end{aligned} \tag{112}$$

for any d satisfying

$$\frac{d}{4} - \rho > 0 \quad \text{and} \quad \frac{\Delta - \mathsf{R}^o}{8} - \rho > 0, \tag{113}$$

i.e., for any d contained in the following interval:

$$d \in (4\rho, -\log(e^{8\rho + \mathsf{R}^o} - 1)). \tag{114}$$

For simplicity, we can rewrite (112) in the following manner:

$$P_c \geq 1 - \exp\left\{\frac{-8E_1^2(x)N_{\mathsf{max}}}{\alpha^2\beta^2}\right\} - \exp\left\{\frac{-8E_2^2(x)N_{\mathsf{max}}}{\alpha^2\beta^2}\right\}, \tag{115}$$

where we introduced the auxiliary functions:

$$E_1(x) \triangleq x - \rho, \tag{116}$$

$$E_2(x) \triangleq \frac{\log(1+e^{-4x}) - \mathsf{R}^o}{8} - \rho, \tag{117}$$

and the free variable d was replaced by

$$x \triangleq \frac{d}{4}. \tag{118}$$

We can now maximize the minimum exponent, i.e., the slowest decay rate, over the free parameter x. To this end, let us consider the value $x^\star$ that solves the equation:

$$E_1(x^\star) = E_2(x^\star), \tag{119}$$

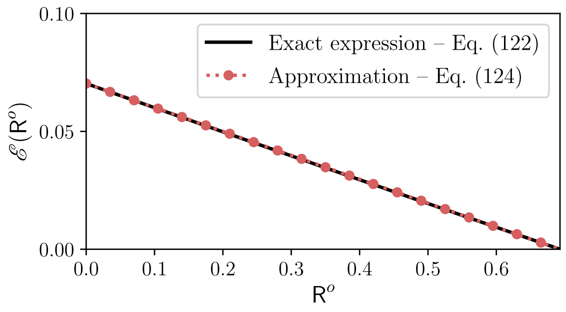

Fig. 11. Comparison between the exact expression in (122) and the approximation in (124).

which corresponds to:

$$x^\star = \frac{\log(1+e^{-4x^\star}) - \mathsf{R}^o}{8} \Leftrightarrow e^{\mathsf{R}^o} e^{12x^\star} - e^{4x^\star} - 1 = 0. \tag{120}$$

Setting $e^{4x^\star} = y$, we have to solve the third-order equation:

$$e^{\mathsf{R}^o} y^3 - y - 1 = 0, \tag{121}$$

whose unique real-valued solution $y^\star$ is available in closed form. Within the range $\mathsf{R}^o \in [0, \log 2]$, $y^\star$ is strictly greater than 1, yielding:

$$\begin{aligned} x^\star &= \frac{1}{4}\log(y^\star) \\ &= \frac{1}{4}\log\left(\frac{2\times 3^{\frac{1}{3}} + 2^{\frac{1}{3}} e^{-\mathsf{R}^o}[Z(\mathsf{R}^o)]^{\frac{2}{3}}}{6^{\frac{2}{3}}[Z(\mathsf{R}^o)]^{\frac{1}{3}}}\right) \triangleq \mathscr{E}(\mathsf{R}^o), \end{aligned} \tag{122}$$

where

$$Z(\mathsf{R}^o) = 9e^{2\mathsf{R}^o} + \sqrt{3e^{3\mathsf{R}^o}(-4 + 27e^{\mathsf{R}^o})}. \tag{123}$$

A good approximation for the function $\mathscr{E}(\mathsf{R}^o)$ is the linear fit — see Fig. 11:

$$\mathscr{E}(\mathsf{R}^o) \approx 4\mathscr{E}(0)\left(1 - \frac{\mathsf{R}^o}{\log 2}\right), \tag{124}$$

where the maximum allowed complexity corresponding to a zero risk is:

$$4\mathscr{E}(0) = 0.2812, \tag{125}$$

which is related to the solution of the third-order equation:

$$y^3 - y - 1 = 0. \tag{126}$$

Fig. 11 shows how accurate the linear approximation in (124) is with respect to the exact expression for $\mathscr{E}(\mathsf{R}^o)$ in (122) within the interval $\mathsf{R}^o \in [0, \log 2]$.

Now, since $E_1(x)$ is an increasing function of x, while $E_2(x)$ is a decreasing function of x, we conclude that if we choose a value $x \neq x^\star$ the minimum exponent necessarily decreases. Accordingly, the minimum exponent is maximized at the value $x^\star = \mathscr{E}(\mathsf{R}^o)$.

Finally, letting

$$\rho < \mathscr{E}(\mathsf{R}^o), \tag{127}$$

we end up with the following bound:

$$P_c \geq 1 - 2\exp\left\{-\frac{8N_{\mathsf{max}}}{\alpha^2\beta^2}\Big(\mathscr{E}(\mathsf{R}^o) - \rho\Big)^2\right\}, \tag{128}$$

and the proof is complete. ■

Appendix C Auxiliary Theorem

To develop the forthcoming result, we consider a L_ϕ-Lipschitz loss function $\phi : \mathbb{R} \mapsto \mathbb{R}_+$. The individual expected and empirical risks are written accordingly as:

$$R_k(f_k) = \mathbb{E}_{h_k,\gamma_k}\phi(-\boldsymbol{\gamma}_{k,n}f_k(\boldsymbol{h}_{k,n})), \tag{129}$$

$$\widetilde{\boldsymbol{R}}_k(f_k) = \frac{1}{N_k}\sum_{n=1}^{N_k}\phi(-\boldsymbol{\gamma}_{k,n}f_k(\boldsymbol{h}_{k,n})), \tag{130}$$

where we removed the symbol $\sim$ from the top of random variables $\boldsymbol{\gamma}_{k,n}$ and $\boldsymbol{h}_{k,n}$ for simplicity of notation. Their network averages $R(f)$ and $\widetilde{\boldsymbol{R}}(f)$ are defined as shown in (37).

Theorem 3 (Uniform Law of Large Numbers): Assume that the loss function $\phi : \mathbb{R} \mapsto \mathbb{R}_+$ is $L_\phi-$Lipschitz and that there exists $\beta > 0$ such that $f_k\big(h\big) \leq \beta$ for every $h \in \mathcal{H}_k$, and $f_k \in \mathcal{F}_k$ and $k = 1, 2, \ldots, K$. Then we have the following two results. First,

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}}\Big|\widetilde{\boldsymbol{R}}(f) - R(f)\Big| \geq x\right) \leq \exp\left\{\frac{-N_{\mathsf{max}}(x - 4L_\phi\rho)^2}{2\alpha^2L_\phi^2\beta^2}\right\}, \tag{131}$$

for any $x > 4L_\phi\rho$. Second,

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}}|\widetilde{\boldsymbol{\mu}}(f) - \mu(f)| \geq x\right) \leq \exp\left\{\frac{-N_{\mathsf{max}}\,(x - 4\rho)^2}{2\alpha^2\beta^2}\right\}, \tag{132}$$

for any $x > 4\rho$, where $N_{\mathsf{max}} \triangleq \max_k N_k$, ρ is the network Rademacher complexity defined in (56), and α is defined as (57).

Proof: In the proof, we use the known *independent bounded differences inequality*, which is also known as *McDiarmid's inequality* [42]. The inequality is reproduced here without proof to facilitate its reference in the forthcoming results.

McDiarmid's Inequality: Let $\boldsymbol{x}$ represent a sequence of independent random variables $\boldsymbol{x}_n$, with $n = 1, 2, \ldots, N$ and $\boldsymbol{x}_n \in \mathcal{X}_n$ for all n. Suppose that the function $g : \prod_{n=1}^N \mathcal{X}_n \mapsto \mathbb{R}$ satisfies for every $j = 1, 2, \ldots, N$:

$$|g(x) - g(\check{x})| \leq c_j \tag{133}$$

whenever the sequences x and $\check{x}$ differ only in the $j-$th component. Then we have for $t > 0$:

$$\mathbb{P}\Big(g(\boldsymbol{x}) - \mathbb{E}g(\boldsymbol{x}) \geq t\Big) \leq e^{-2t^2/\sum\limits_{j=1}^N c_j^2}, \tag{134}$$

$$\mathbb{P}\Big(g(\boldsymbol{x}) - \mathbb{E}g(\boldsymbol{x}) \leq -t\Big) \leq e^{-2t^2/\sum\limits_{j=1}^N c_j^2}. \tag{135}$$

■

Before introducing the proof for Theorem 3, we provide a roadmap of the results used in the proof. Both (131) and (132) are proven using McDiarmid's inequality, followed by the auxiliary Lemma 4 (found in Appendix F) and finally using Lemma 3 (also found in Appendix F).

We now develop the proof of (131) and (132) in Theorem 3 separately as follows.

1) Proof of (131)*:* Consider that the sequence of samples $\boldsymbol{x}_n$ is replaced by a sequence of random pairs $(\boldsymbol{h}_n, \boldsymbol{\gamma}_n)$, with $n = 1, 2, \ldots, N_{\mathsf{max}}$, where $N_{\mathsf{max}} \triangleq \max_k N_k$. The quantity $\boldsymbol{h}_n$ is a sequence collecting random variables (or vectors) $\boldsymbol{h}_{k,n}$ for $k = 1, 2, \ldots, K$:

$$\boldsymbol{h}_n \triangleq \{\boldsymbol{h}_{1,n}, \boldsymbol{h}_{2,n}, \ldots, \boldsymbol{h}_{K,n}\}, \tag{136}$$

and $\boldsymbol{\gamma}_n$ is a sequence of random variables $\boldsymbol{\gamma}_{k,n}$ for $k = 1, 2, \ldots, K$:

$$\boldsymbol{\gamma}_n \triangleq \{\boldsymbol{\gamma}_{1,n}, \boldsymbol{\gamma}_{2,n}, \ldots, \boldsymbol{\gamma}_{K,n}\}. \tag{137}$$

The pairs $(\boldsymbol{h}_n, \boldsymbol{\gamma}_n)$ are independent and identically distributed over time, i.e., for all n.

Define the following auxiliary quantity:

$$\chi_k(f_k) \triangleq \mathbb{E}_{h_k,\gamma_k}\phi(-\boldsymbol{\gamma}_{k,n}f_k(\boldsymbol{h}_{k,n})), \tag{138}$$

where we recall that $\mathbb{E}_{h_k,\gamma_k}$ is the expectation computed according to the joint distribution of $\boldsymbol{h}_{k,n}$ and $\boldsymbol{\gamma}_{k,n}$. Our function of interest is the following:

$$g(h,\gamma) = \sup_{f\in\mathcal{F}}\left|\sum_{k=1}^K \pi_k\Big[\chi_k(f_k) - \frac{1}{N_k}\sum_{n=1}^{N_k}\phi(-\gamma_{k,n}f_k(h_{k,n}))\Big]\right|, \tag{139}$$

where, to keep a concise notation, the arguments h, γ indicate that the function $g(\cdot)$ depends on the collection of sequences h_n (defined in (136)) and γ_n (defined in (137)) for $n = 1, 2, \ldots, N_k$. The argument f represents the ensemble of functions $\{f_k\}$, where $f_k \in \mathcal{F}_k$, and we define the global space of functions:

$$\mathcal{F} \triangleq \mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \times \mathcal{F}_K. \tag{140}$$

From the collections h and γ, we can construct collections $\check{h}$ and $\check{\gamma}$, by replacing $h_{k,j}$ and $\gamma_{k,j}$ respectively with the distinct samples $\check{h}_{k,j}$ and $\check{\gamma}_{k,j}$ for all $k = 1, 2, \ldots, K$. If $j > N_k$, the inner summand in (139) is not altered, then, using the indicator function, we can write

$$\begin{aligned}
&g(\check{h},\check{\gamma})\\
&= \sup_{f\in\mathcal{F}}\left|\sum_{k=1}^K \pi_k\Big[\chi_k(f_k) - \frac{1}{N_k}\sum_{\substack{n=1\\n\neq j}}^{N_k}\phi(-\gamma_{k,n}f_k(h_{k,n}))\right.\\
&\quad - \frac{\mathbb{1}[j \leq N_k]}{N_k}\phi(-\check{\gamma}_{k,j}f_k(\check{h}_{k,j}))\\
&\quad \left. - \frac{\mathbb{1}[j > N_k]}{N_k}\phi(-\gamma_{k,j}f_k(h_{k,j}))\Big]\right|\\
&= \sup_{f\in\mathcal{F}}\left|\sum_{k=1}^K \pi_k\left[\chi_k(f_k) - \frac{1}{N_k}\sum_{n=1}^{N_k}\phi(-\gamma_{k,n}f_k(h_{k,n}))\right.\right.\\
&\quad \left.\left. + \frac{\mathbb{1}[j \leq N_k]}{N_k}\Big(\phi(-\gamma_{k,j}f_k(h_{k,j})) - \phi(-\check{\gamma}_{k,j}f_k(\check{h}_{k,j}))\Big)\right]\right|,
\end{aligned} \tag{141}$$

where $\mathbb{1}[E]$ is indicator function defined as: $\mathbb{1}[E] = 1$, if event E takes place, $\mathbb{1}[E] = 0$ otherwise. It is convenient to introduce the following quantities:

$$u_k(f_k) \triangleq \chi_k(f_k) - \frac{1}{N_k}\sum_{n=1}^{N_k}\phi(-\gamma_{k,n}f_k(h_{k,n})), \tag{142}$$

$$v_k(f_k) \triangleq \frac{\mathbb{1}[j \le N_k]}{N_k}\Big[\phi(-\gamma_{k,j}f_k(h_{k,j})) - \phi(-\check{\gamma}_{k,j}f_k(\check{h}_{k,j}))\Big], \tag{143}$$

where the dependence of $u_k(\cdot)$ upon (h,γ) and of $v_k(\cdot)$ upon $(\check{h},\check{\gamma})$ has been skipped for ease of notation. In view of the definitions in (142) and (143), we can rewrite (139) and (141) as:

$$g(h,\gamma) = \sup_{f\in\mathcal{F}}\left|\sum_{k=1}^{K}\pi_k u_k(f_k)\right| \tag{144}$$

$$g(\check{h},\check{\gamma}) = \sup_{f\in\mathcal{F}}\left|\sum_{k=1}^{K}\pi_k u_k(f_k) + \sum_{k=1}^{K}\pi_k v_k(f_k)\right|. \tag{145}$$

Applying Lemma 4 (Appendix F) with the choices $s_1 = g(h,\gamma)$, $s_2 = g(\check{h},\check{\gamma})$, and

$$S(f) = \sum_{k=1}^{K}\pi_k u_k(f_k), \quad T(f) = \sum_{k=1}^{K}\pi_k v_k(f_k), \tag{146}$$

we obtain:

$$|g(h,\gamma) - g(\check{h},\check{\gamma})| \le \sup_{f\in\mathcal{F}}\left|\sum_{k=1}^{K}\pi_k v_k(f_k)\right| \overset{\text{(a)}}{\le} \sum_{k=1}^{K}\pi_k \sup_{f_k\in\mathcal{F}_k}\Big|v_k(f_k)\Big|, \tag{147}$$

where (a) follows from the triangle inequality and the sub-additive property of the supremum operator. Replacing (143) into (147) yields

$$\begin{aligned}
&|g(h,\gamma) - g(\check{h},\check{\gamma})| \\
&\le \sum_{k=1}^{K}\pi_k \sup_{f_k\in\mathcal{F}_k}\left|\frac{\mathbb{1}[j\le N_k]}{N_k}\Big[\phi(-\gamma_{k,j}f_k(h_{k,j})) - \phi(-\check{\gamma}_{k,j}f_k(\check{h}_{k,j}))\Big]\right| \\
&\le \sum_{k=1}^{K}\pi_k \sup_{f_k\in\mathcal{F}_k}\left|\frac{1}{N_k}\Big[\phi(-\gamma_{k,j}f_k(h_{k,j})) - \phi(-\check{\gamma}_{k,j}f_k(\check{h}_{k,j}))\Big]\right| \\
&\overset{\text{(a)}}{\le} L_\phi \sum_{k=1}^{K}\frac{\pi_k}{N_k}\sup_{f_k\in\mathcal{F}_k}\Big|\gamma_{k,j}f_k(h_{k,j}) - \check{\gamma}_{k,j}f_k(\check{h}_{k,j})\Big| \\
&\overset{\text{(b)}}{\le} L_\phi \sum_{k=1}^{K}\frac{\pi_k}{N_k}\sup_{f_k\in\mathcal{F}_k}\Big\{\Big|\gamma_{k,j}\Big|\Big|f_k(h_{k,j})\Big| + \Big|\check{\gamma}_{k,j}\Big|\Big|f_k(\check{h}_{k,j})\Big|\Big\} \\
&\overset{\text{(c)}}{\le} 2L_\phi\beta\sum_{k=1}^{K}\frac{\pi_k}{N_k} \overset{\text{(d)}}{=} \frac{2\alpha L_\phi\beta}{N_{\max}}.
\end{aligned} \tag{148}$$

where (a) follows from the Lipschitz property of ϕ, (b) follows from the triangle inequality, (c) follows from the boundedness assumption $f_k(h) \le \beta$ and the fact that $|\gamma_{k,n}| = 1$ for all k and i. Finally, in (d) we used the definition in (57), namely,

$$\alpha \triangleq \sum_{k=1}^{K}\pi_k\frac{N_{\max}}{N_k}. \tag{149}$$

Applying McDiarmid's Inequality in (134) with $c_j = 2\alpha L_\phi\beta/N_{\max}$, we obtain the following deviation bound:

$$\mathbb{P}\Big(\sup_{f\in\mathcal{F}}|R(f) - \widetilde{\boldsymbol{R}}(f)| - \mathbb{E}\sup_{f\in\mathcal{F}}|R(f) - \widetilde{\boldsymbol{R}}(f)| \ge t\Big) \le e^{-t^2N_{\max}/(2\alpha^2L_\phi^2\beta^2)}, \tag{150}$$

holding for all $t > 0$. To conclude the proof, we seek to upper bound the second term inside the probability operator in (150). The result from Lemma 3 (Appendix F) can be directly employed to conclude that:

$$\mathbb{E}\sup_{f\in\mathcal{F}}|R(f) - \widetilde{\boldsymbol{R}}(f)| \le 4L_\phi\rho. \tag{151}$$

In view of (151), we have that

$$\begin{aligned}
&\sup_{f\in\mathcal{F}}|R(f) - \widetilde{\boldsymbol{R}}(f)| \ge t + 4L_\phi\rho \\
&\Rightarrow \sup_{f\in\mathcal{F}}|R(f) - \widetilde{\boldsymbol{R}}(f)| - \mathbb{E}\sup_{f\in\mathcal{F}}|R(f) - \widetilde{\boldsymbol{R}}(f)|) \ge t.
\end{aligned} \tag{152}$$

From (152) and (150), we can conclude that

$$\mathbb{P}\Big(\sup_{f\in\mathcal{F}}|R(f) - \widetilde{\boldsymbol{R}}(f)| \ge t + 4L_\phi\rho\Big) \le e^{-t^2N_{\max}/(2\alpha^2L_\phi^2\beta^2)}. \tag{153}$$

Defining $x = t + 4L_\phi\rho$, and noting that $x > 4L_\phi\rho$ since $t > 0$, completes the proof of (131).

2) Proof of (132)*:* The proof for the uniform bound in (132) follows similar arguments and will be thus presented in a concise manner. We start by using McDiarmid's Inequality with the following choice of function g:

$$g(h) = \sup_{f\in\mathcal{F}}\left|\sum_{k=1}^{K}\pi_k\Big[\nu_k(f_k) - \frac{1}{N_k}\sum_{n=1}^{N_k}f_k(h_{k,n})\Big]\right|, \tag{154}$$

where we define the auxiliary quantity:

$$\nu_k(f_k) \triangleq \mathbb{E}_{h_k}f_k(\boldsymbol{h}_{k,n}). \tag{155}$$

We follow similar steps as the ones used to prove (131), which results in the following bound:

$$\mathbb{P}\Big(\sup_{f\in\mathcal{F}}|\mu(f) - \widetilde{\boldsymbol{\mu}}(f)| - \mathbb{E}\sup_{f\in\mathcal{F}}|\mu(f) - \widetilde{\boldsymbol{\mu}}(f)|) \ge t\Big) \le e^{-t^2N_{\max}/(2\alpha^2\beta^2)}. \tag{156}$$

We use again Lemma 3 (Appendix F) to bound the second term inside the probability operation in (156). For this we take $L_\phi = 1$ and we take $\boldsymbol{\gamma}_n = 1$ as a deterministic variable, which allows us to derive the result:

$$\mathbb{E}\sup_{f\in\mathcal{F}}|\mu(f) - \widetilde{\boldsymbol{\mu}}(f)| \le 4\rho. \tag{157}$$

Replacing this bound in (156), defining $x = t + 4\rho$, with $x > 4\rho$, yields the final result. ■

APPENDIX D
PROOF OF THEOREM 2

Assuming $\rho_k \leq C_k/\sqrt{N_k}$, it follows that (64) holds, i.e.,

$$\rho \leq \frac{\mathsf{C}}{\sqrt{N_{\mathsf{max}}}}. \tag{158}$$

For the bound in Theorem 1 to hold, the Rademacher complexity must satisfy

$$\rho \leq \mathscr{E}(\mathsf{R}^o). \tag{159}$$

In view of (158), (159) is met if we choose:

$$\frac{\mathsf{C}}{\sqrt{N_{\mathsf{max}}}} < \mathscr{E}(\mathsf{R}^o) \Leftarrow N_{\mathsf{max}} > \left(\frac{\mathsf{C}}{\mathscr{E}(\mathsf{R}^o)}\right)^2. \tag{160}$$

Next, for a desired minimum probability of consistent learning we should consider the bound found in (65). We have that:

$$\begin{aligned} &P_c \geq 1-\varepsilon \\ &\Leftrightarrow 2\exp\left\{-\frac{8N_{\mathsf{max}}}{\alpha^2\beta^2}\left(\mathscr{E}(\mathsf{R}^o)-\frac{\mathsf{C}}{\sqrt{N_{\mathsf{max}}}}\right)^2\right\} \leq \varepsilon \\ &\Leftrightarrow N_{\mathsf{max}}\left(\mathscr{E}(\mathsf{R}^o)-\frac{\mathsf{C}}{\sqrt{N_{\mathsf{max}}}}\right)^2 \geq \frac{\alpha^2\beta^2}{8}\log\left(\frac{2}{\varepsilon}\right). \end{aligned} \tag{161}$$

We can develop the quadratic term in the LHS of (161) as

$$\begin{aligned} &N_{\mathsf{max}}\left(\mathscr{E}(\mathsf{R}^o)-\frac{\mathsf{C}}{\sqrt{N_{\mathsf{max}}}}\right)^2 \\ &\quad = N_{\mathsf{max}}\left[\mathscr{E}(\mathsf{R}^o)\right]^2 - 2\sqrt{N_{\mathsf{max}}}\,\mathsf{C}\,\mathscr{E}(\mathsf{R}^o) + \mathsf{C}^2. \end{aligned} \tag{162}$$

Let

$$z = \sqrt{N_{\mathsf{max}}}\,\mathscr{E}(\mathsf{R}^o), \quad b = \mathsf{C}^2 - \frac{\alpha^2\beta^2}{8}\log\left(\frac{2}{\varepsilon}\right). \tag{163}$$

To solve the inequality in (161), we must study the following quadratic equality:

$$z^2 - 2\mathsf{C}z + b = 0, \tag{164}$$

whose positive solution is:

$$z = \mathsf{C} + \sqrt{\frac{\alpha^2\beta^2}{8}\log\left(\frac{2}{\varepsilon}\right)}. \tag{165}$$

Thus the inequality in (161) is satisfied whenever:

$$\sqrt{N_{\mathsf{max}}} > \frac{1}{\mathscr{E}(\mathsf{R}^o)}\left(\mathsf{C} + \sqrt{\frac{\alpha^2\beta^2}{8}\log\left(\frac{2}{\varepsilon}\right)}\right), \tag{166}$$

or yet when:

$$N_{\mathsf{max}} > \left(\frac{\mathsf{C}}{\mathscr{E}(\mathsf{R}^o)}\right)^2\left(1 + \frac{\alpha\beta}{2\mathsf{C}}\sqrt{\frac{1}{2}\log\left(\frac{2}{\varepsilon}\right)}\right)^2. \tag{167}$$

The final result of the theorem is established, since the bound in (167) is more stringent than (160).

APPENDIX E
PROOF OF PROPOSITION 2

Before introducing the proof, in order to establish the complexity of class $\mathcal{F}^{\mathsf{NN}}$, we will resort to a set of known inequalities involving the Rademacher complexity operator [37], [43], summarized in Property 1 (Appendix F). The proof follows an inductive argument similar to the one used in [30], where we establish an upper bound for the Rademacher complexity of the output of one layer with respect to the output of the previous layer, then this bound is iterated over the depth of the Multilayer Perceptron (MLP).

We wish to analyze the complexity of the class of functions $\mathcal{F}^{\mathsf{NN}}$, which is defined in (71) as the difference between the outputs of the neural network z_1 and z_2, for an input vector $x \in \mathbb{R}^{n_0}$. That is, function f^{NN} has the following form (as seen in (71)):

$$f^{\mathsf{NN}}(x) = \log\frac{\widehat{p}(+1|x)}{\widehat{p}(-1|x)} = z_1 - z_2, \tag{168}$$

where z_1, z_2 implement functions $g^{(L)} \in \mathcal{G}^{(L)}$ as defined in (68) for $\ell = L$. We thus say that $f^{\mathsf{NN}} \in \mathcal{F}^{\mathsf{NN}}$, with

$$f^{\mathsf{NN}}(x) = g_1^{(L)}(x) - g_2^{(L)}(x), \tag{169}$$

where $g_1^{(L)}, g_2^{(L)} \in \mathcal{G}^{(L)}$.

From items 1 and 2 in Property 1 (Appendix F). choosing $c = -1$, the empirical Rademacher complexity of $\mathcal{F}^{\mathsf{NN}}(x)$ will satisfy:

$$\begin{aligned} \mathcal{R}\left(\mathcal{F}^{\mathsf{NN}}(x)\right) &\leq \mathcal{R}\left(\mathcal{G}^{(L)}(x)\right) + \mathcal{R}\left(\mathcal{G}^{(L)}(x)\right) \\ &= 2\mathcal{R}\left(\mathcal{G}^{(L)}(x)\right). \end{aligned} \tag{170}$$

From (68), the Rademacher complexity of $\mathcal{G}^{(\ell)}(x)$ can be expressed as:

$$\mathcal{R}\left(\mathcal{G}^{(\ell)}(x)\right) = \mathbb{E}_r \sup_{w_j, g_j^{(\ell-1)}}\left|\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{r}_i\sum_{j=1}^{m}w_j\sigma\left(g_j^{(\ell-1)}(x_i)\right)\right|, \tag{171}$$

where $\boldsymbol{r}_i$ are independent and identically distributed Rademacher random variables, with $\mathbb{P}(\boldsymbol{r}_i = +1) = \mathbb{P}(\boldsymbol{r}_i = -1) = 1/2$. The term on the RHS of (171) can be rewritten as:

$$\begin{aligned} &\mathbb{E}_r \sup_{w, g^{(\ell-1)}}\left|\frac{1}{N}\sum_{j=1}^{m}w_j\sum_{i=1}^{N}\boldsymbol{r}_i\sigma\left(g_j^{(\ell-1)}(x_i)\right)\right| \\ &\overset{(a)}{\leq} \mathbb{E}_r \sup_{w, g^{(\ell-1)}}\|w\|_1 \max_j\left|\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{r}_i\sigma\left(g_j^{(\ell-1)}(x_i)\right)\right| \\ &\overset{(b)}{\leq} b\,\mathbb{E}_r \sup_{g^{(\ell-1)}}\max_j\left|\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{r}_i\sigma\left(g_j^{(\ell-1)}(x_i)\right)\right| \\ &\overset{(c)}{=} b\,\mathcal{R}\left(\sigma\circ\mathcal{G}^{(\ell-1)}(x)\right) \overset{(d)}{\leq} 2bL_\sigma\mathcal{R}\left(\mathcal{G}^{(\ell-1)}(x)\right), \end{aligned} \tag{172}$$

where (a) follows from triangle inequality and taking the maximum w.r.t. j, (b) follows from the assumption that $\|w\|_1 \leq b$, (c) follows from the fact that $g_j^{(\ell-1)} \in \mathcal{G}^{(\ell-1)}$ for all $j =$

$1, 2, \ldots, m$, (c) and (d) follows from the contraction principle (item 3 in Property 1, Appendix F) in association with the assumption that σ is a Lipschitz function with constant L_σ.

Replacing (172) into (171), we have the following recursion:

$$\mathcal{R}\left(\mathcal{G}^{(\ell)}(x)\right) \leq 2bL_\sigma \mathcal{R}\left(\mathcal{G}^{(\ell-1)}(x)\right). \tag{173}$$

We can develop the recursion above across all layers up to ℓ:

$$\mathcal{R}\left(\mathcal{G}^{(\ell)}(x)\right) \leq (2bL_\sigma)^{\ell-1}\mathcal{R}\left(\mathcal{G}^{(1)}(x)\right). \tag{174}$$

It remains to bound the Rademacher complexity relative to $\mathcal{G}^{(1)}(x)$ of the first layer, whose functions have the form of $g_m^{(1)}$ defined in (69). For this purpose, we can directly use the result in Lemma 15 of [30], which bounds the Rademacher complexity of a linear separator with bounded ℓ_p norm. Applying this lemma with $p = 1$, $\gamma = b$ and $\|x\|_\infty = \max_i |x_i| \leq c$, we have:

$$\begin{aligned}\mathcal{R}\left(\mathcal{G}^{(1)}(x)\right) &= \mathbb{E}_r \sup_{w_j}\left|\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{r}_i\sum_{j=1}^{d} w_j x_{i,j}\right| \\ &\leq \frac{2bc\sqrt{\log(2n_0)}}{\sqrt{N}}.\end{aligned} \tag{175}$$

Replacing (174) with $\ell = L$ and (175) into (170) yields the final result.

Appendix F
Auxiliary Results

We list three key properties of Rademacher complexity, which are used in some of our results. These properties are well known and therefore are reported here without proof, which can be found in [37] and [43].

Property 1 (Inequalities involving Rademacher Complexity [37]): Let $\mathcal{F}, \mathcal{F}_1, \ldots, \mathcal{F}_K$ be classes of real-valued functions, and x a sequence of samples $\{x_1, x_2, \ldots, x_N\}$. The Rademacher complexity defined in (53) satisfies the following properties:

1) Subadditivity:

$$\mathcal{R}\left(\textstyle\sum_{k=1}^K \mathcal{F}_k(x)\right) \leq \textstyle\sum_{k=1}^K \mathcal{R}(\mathcal{F}_k(x)), \tag{176}$$

with $\mathcal{F}_1(x) + \mathcal{F}_2(x) \triangleq \{[f_1(x_1) + f_2(x_1), f_1(x_2) + f_2(x_2), \ldots, f_1(x_N) + f_2(x_N)] : f_1 \in \mathcal{F}_1, f_2 \in \mathcal{F}_2\}$.

2) Scaling: For every $c \in \mathbb{R}$,

$$\mathcal{R}(c\,\mathcal{F}(x)) \leq |c|\mathcal{R}(\mathcal{F}(x)), \tag{177}$$

where $c\,\mathcal{F}(x) \triangleq \{[cf(x_1), cf(x_2), \ldots, cf(x_N)] : f \in \mathcal{F}\}$.

3) Contraction principle: Let $\phi : \mathbb{R} \mapsto \mathbb{R}_+$ be *Lipschitz* with constant L_ϕ and $\phi(0) = 0$. Then:

$$\mathcal{R}(\phi \circ \mathcal{F}(x)) \leq 2L_\phi \mathcal{R}(\mathcal{F}(x)), \tag{178}$$

with $\phi \circ \mathcal{F}(x) \triangleq \{[\phi(f(x_1)), \phi(f(x_2)), \ldots, \phi(f(x_N))] : f \in \mathcal{F}\}$.

■

The next three lemmas are important auxiliary results used in the proofs of Theorem 1 and Theorem 3.

Lemma 2 (Upper Bound on the Estimation Error for the Empirical Risk): From the definitions in (34)–(35) and (37), for $x > 0$ we have that:

$$\mathbb{P}\left(R(\widetilde{\boldsymbol{f}}) - \mathsf{R}^o \geq x\right) \leq \mathbb{P}\left(\sup_{f\in\mathcal{F}}\left|\widetilde{\boldsymbol{R}}(f) - R(f)\right| \geq \frac{x}{2}\right). \tag{179}$$

Proof: From (34) and (35), we can verify that for all $k = 1, 2, \ldots, K$:

$$\widetilde{\boldsymbol{R}}_k(\widetilde{\boldsymbol{f}}_k) \leq \widetilde{\boldsymbol{R}}(f_k), \text{ for all } f_k \in \mathcal{F}_k, \tag{180}$$

which imply, from the definitions in (37), that

$$\widetilde{\boldsymbol{R}}(\widetilde{\boldsymbol{f}}) \leq \widetilde{\boldsymbol{R}}(f), \text{ for all } f \in \mathcal{F}, \tag{181}$$

where $\mathcal{F}$ is the global class of functions defined in (140). We can develop the expression of the estimation error to obtain the following uniform bound:

$$\begin{aligned}R(\widetilde{\boldsymbol{f}}) - \mathsf{R}^o &\overset{\text{(a)}}{=} R(\widetilde{\boldsymbol{f}}) - \inf_{f\in\mathcal{F}} R(f) \\ &= R(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{R}}(\widetilde{\boldsymbol{f}}) + \widetilde{\boldsymbol{R}}(\widetilde{\boldsymbol{f}}) - \inf_{f\in\mathcal{F}} R(f) \\ &= R(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{R}}(\widetilde{\boldsymbol{f}}) + \sup_{f\in\mathcal{F}}\left(\widetilde{\boldsymbol{R}}(\widetilde{\boldsymbol{f}}) - R(f)\right) \\ &\overset{\text{(b)}}{\leq} R(\widetilde{\boldsymbol{f}}) - \widetilde{\boldsymbol{R}}(\widetilde{\boldsymbol{f}}) + \sup_{f\in\mathcal{F}}\left(\widetilde{\boldsymbol{R}}(f) - R(f)\right) \\ &\leq 2\sup_{f\in\mathcal{F}}\left|\widetilde{\boldsymbol{R}}(f) - R(f)\right|,\end{aligned} \tag{182}$$

where (a) follows from the definition in (34) and (b) follows from (181).

Finally, from (182) and using the target risk notation in (34), we note that

$$R(\widetilde{\boldsymbol{f}}) - \mathsf{R}^o \geq x \Rightarrow \sup_{f\in\mathcal{F}}\left|\widetilde{\boldsymbol{R}}(f) - R(f)\right| \geq x/2, \tag{183}$$

for any $x > 0$, thus concluding the proof. ■

Lemma 3 (Uniform Upper Bound for Lipschitz Cost Functions): Assume that the pair of sequences $(\boldsymbol{h}_n, \boldsymbol{\gamma}_n)$ is sampled independently from the same joint distribution for all $n = 1, 2, \ldots, N_{\mathsf{max}}$. Let $f_k : \mathcal{H}_k \mapsto \mathbb{R}$ be a function belonging to class $\mathcal{F}_k$, and let $\phi : \mathbb{R} \mapsto \mathbb{R}_+$ be a L_ϕ−Lipschitz function. Then it follows that

$$\begin{aligned}&\mathbb{E}_{h,\gamma}\sup_{f\in\mathcal{F}}\left|\sum_{k=1}^K \pi_k\left[\chi_k(f_k) - \frac{1}{N_k}\sum_{n=1}^{N_k}\phi(-\boldsymbol{\gamma}_{k,n}f_k(\boldsymbol{h}_{k,n}))\right]\right| \\ &\leq 4L_\phi\rho,\end{aligned} \tag{184}$$

with

$$\chi_k(f_k) \triangleq \mathbb{E}_{h_k,\gamma_k}\phi(-\boldsymbol{\gamma}_{k,n}f_k(\boldsymbol{h}_{k,n})). \tag{185}$$

Proof: Introduce the artificial pair $\boldsymbol{h}'_n, \boldsymbol{\gamma}'_n$, sampled independently with the same joint distribution of $\boldsymbol{h}_n, \boldsymbol{\gamma}_n$. We develop the following symmetrization argument, inspired by the ones used in [34] and [37].

First, we use the triangle inequality and the subadditive property of the supremum operator:

$$\mathbb{E}_{h,\gamma}\sup_{f\in\mathcal{F}}\left|\sum_{k=1}^K \pi_k\left[\chi_k(f_k) - \frac{1}{N_k}\sum_{n=1}^{N_k}\phi(-\boldsymbol{\gamma}_{k,n}f_k(\boldsymbol{h}_{k,n}))\right]\right|$$

$$\leq \sum_{k=1}^{K} \pi_k \mathbb{E}_{h_k,\gamma_k} \sup_{f_k \in \mathcal{F}_k} \left| \chi_k(f_k) - \frac{1}{N_k} \sum_{n=1}^{N_k} \phi(-\boldsymbol{\gamma}_{k,n} f_k(\boldsymbol{h}_{k,n})) \right|, \tag{186}$$

where we recall that the argument f represents the ensemble of functions $\{f_k\}$, with $f_k \in \mathcal{F}_k$, and $\mathcal{F}$ denotes the global space of functions defined in (140).

We focus on the individual elements of the summation on the RHS of (186), that is, on each term indexed by k. We drop subscript k everywhere to simplify the notation.

$$\begin{aligned}
&\mathbb{E}_{h,\gamma} \sup_{f\in\mathcal{F}} \left| \chi(f) - \frac{1}{N}\sum_{n=1}^{N} \phi(-\boldsymbol{\gamma}_n f(\boldsymbol{h}_n)) \right| \\
&\overset{\text{(a)}}{=} \mathbb{E}_{h,\gamma} \sup_{f\in\mathcal{F}} \left| \mathbb{E}_{h',\gamma'} \frac{1}{N}\sum_{n=1}^{N} \Big[\phi(-\boldsymbol{\gamma}'_n f(\boldsymbol{h}'_n)) - \sum_{n=1}^{N} \phi(-\boldsymbol{\gamma}_n f(\boldsymbol{h}_n))\Big] \right| \\
&\overset{\text{(b)}}{\leq} \mathbb{E}_{h,\gamma}\mathbb{E}_{h',\gamma'} \sup_{f\in\mathcal{F}} \left| \frac{1}{N}\sum_{n=1}^{N} \Big[\phi(-\boldsymbol{\gamma}'_n f(\boldsymbol{h}'_n)) - \phi(-\boldsymbol{\gamma}_n f(\boldsymbol{h}_n))\Big] \right| \\
&\overset{\text{(c)}}{=} \mathbb{E}_{h,\gamma}\mathbb{E}_{h',\gamma'}\mathbb{E}_r \sup_{f\in\mathcal{F}} \left| \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{r}_n \Big[\phi(-\boldsymbol{\gamma}'_n f(\boldsymbol{h}'_n)) - \phi(-\boldsymbol{\gamma}_n f(\boldsymbol{h}_n))\Big] \right| \\
&\overset{\text{(d)}}{\leq} 2\mathbb{E}_{h,\gamma}\mathbb{E}_r \sup_{f\in\mathcal{F}} \left| \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{r}_n \phi(-\boldsymbol{\gamma}_n f(\boldsymbol{h}_n)) \right| \\
&\overset{\text{(e)}}{\leq} 4L_\phi \mathbb{E}_{h,\gamma}\mathbb{E}_r \sup_{f\in\mathcal{F}} \left| \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{r}_n \boldsymbol{\gamma}_n f(\boldsymbol{h}_n) \right| \\
&\overset{\text{(f)}}{\leq} 4L_\phi \mathbb{E}_{h}\mathbb{E}_r \sup_{f\in\mathcal{F}} \left| \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{r}_n f(\boldsymbol{h}_n) \right| \\
&= 4L_\phi \mathbb{E}_h \mathcal{R}\left(\mathcal{F}(\boldsymbol{h})\right).
\end{aligned} \tag{187}$$

We explain now each of the steps (a)–(f) performed in (187). In (a) we used the i.i.d. property of the artificial samples $(\boldsymbol{h}'_n, \boldsymbol{\gamma}'_n)$, (b) follows from the following two properties: $i)$ $|\mathbb{E}\boldsymbol{x}| \leq \mathbb{E}|\boldsymbol{x}|$; $ii)$ $\sup_{f\in\mathcal{F}} \mathbb{E}|\boldsymbol{y}(f)| \leq \mathbb{E}\sup_{f\in\mathcal{F}} |\boldsymbol{y}(f)|$.

In (c) we introduced the i.i.d. Rademacher random variables, i.e., $\boldsymbol{r}_n \in \{-1,+1\}$ with uniform probability, which are independent of samples $(\boldsymbol{h}_n, \boldsymbol{\gamma}_n)$ and $(\boldsymbol{h}'_n, \boldsymbol{\gamma}'_n)$. Since $(\boldsymbol{h}_n, \boldsymbol{\gamma}_n)$ and $(\boldsymbol{h}'_n, \boldsymbol{\gamma}'_n)$ are identically distributed and independently sampled, exchanging $(\boldsymbol{h}_n, \boldsymbol{\gamma}_n)$ and $(\boldsymbol{h}'_n, \boldsymbol{\gamma}'_n)$ is immaterial and therefore we can safely introduce the Rademacher random variables $\boldsymbol{r}_n$ in the summation.

In (d), we used the triangle inequality for the absolute value and the fact that $(\boldsymbol{h}_n, \boldsymbol{\gamma}_n)$ and $(\boldsymbol{h}'_n, \boldsymbol{\gamma}'_n)$ are identically distributed. In (e), we use the Lipschitz property of ϕ associated with the contraction principle of the Rademacher complexity (item 3 in Property 1) to conclude that:

$$\begin{aligned}
&\mathbb{E}_r \sup_{f\in\mathcal{F}} \left| \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{r}_n \phi(-\boldsymbol{\gamma}_n f(\boldsymbol{h}_n)) \right| \\
&\qquad \leq 2L_\phi \mathbb{E}_r \sup_{f\in\mathcal{F}} \left| \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{r}_n \boldsymbol{\gamma}_n f(\boldsymbol{h}_n) \right|.
\end{aligned} \tag{188}$$

Step (f) follows from similar symmetrization arguments, considering that $\boldsymbol{\gamma}_n$ assumes values ± 1 and that $\boldsymbol{r}_n$ and $-\boldsymbol{r}_n$ are equally distributed and independent form the samples and over i. Finally replacing (187) into (186) for each of the summands indexed by k, for all terms indexed by k, and recalling the definition of ρ in (56), we obtain (184). ■

Lemma 4 (Auxiliary Result for Bounded Differences): Assume $S(f)$ and $T(f)$ are operators dependent on a real-valued function $f \in \mathcal{F}$, and consider the following quantities:

$$s_1 = \sup_{f\in\mathcal{F}} \left| S(f) \right|, \quad s_2 = \sup_{f\in\mathcal{F}} \left| S(f) + T(f) \right|. \tag{189}$$

Then, we have that:

$$|s_1 - s_2| \leq \sup_{f\in\mathcal{F}} \left| T(f) \right|. \tag{190}$$

Proof: The proof is split in two cases.

a) Case $s_2 \geq s_1$:

$$\begin{aligned}
s_2 - s_1 &= \sup_{f\in\mathcal{F}} \left| S(f) + T(f) \right| - \sup_{f\in\mathcal{F}} \left| S(f) \right| \\
&\leq \sup_{f\in\mathcal{F}} \left| S(f) \right| + \sup_{f\in\mathcal{F}} \left| T(f) \right| - \sup_{f\in\mathcal{F}} \left| S(f) \right| \\
&= \sup_{f\in\mathcal{F}} \left| T(f) \right|,
\end{aligned} \tag{191}$$

where the inequality follows from the triangle inequality and the subadditive property of the supremum operator, i.e., $\sup_{f\in\mathcal{F}}[a(f) + b(f)] \leq \sup_{f\in\mathcal{F}} a(f) + \sup_{f\in\mathcal{F}} b(f)$.

b) Case $s_2 < s_1$:

$$\begin{aligned}
s_1 - s_2 &= \sup_{f\in\mathcal{F}} \left| S(f) \right| - \sup_{f\in\mathcal{F}} \left| S(f) + T(f) \right| \\
&= \sup_{f\in\mathcal{F}} \Big(\left| S(f) \right| - s_2 \Big) \\
&\overset{\text{(a)}}{\leq} \sup_{f\in\mathcal{F}} \Big(\left| S(f) \right| - \left| S(f) + T(f) \right| \Big) \\
&\leq \sup_{f\in\mathcal{F}} \Big| \left| S(f) \right| - \left| S(f) + T(f) \right| \Big| \\
&\overset{\text{(b)}}{\leq} \sup_{f\in\mathcal{F}} \left| S(f) - S(f) + T(f) \right| = \sup_{f\in\mathcal{F}} \left| T(f) \right|,
\end{aligned} \tag{192}$$

where (a) follows from the definition of s_2, and (b) from the reverse triangle inequality, i.e., $|a - b| \geq \big| |a| - |b| \big|$.

Using (191) and (192), we obtain the desired result in (190). ■

References

[1] V. Bordignon, S. Vlaski, V. Matta, and A. H. Sayed, "Network classifiers based on social learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 5185–5189.

[2] A. Jadbabaie, P. Molavi, A. Sandroni, and A. Tahbaz-Salehi, "Non-Bayesian social learning," *Games Econ. Behavior*, vol. 76, no. 1, pp. 210–225, 2012.

[3] X. Zhao and A. H. Sayed, "Learning over social networks via diffusion adaptation," in *Proc. Conf. Rec. Forty Sixth Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, Nov. 2012, pp. 709–713.

[4] V. Krishnamurthy and H. V. Poor, "Social learning and Bayesian games in multiagent signal processing: How do local and global decision makers interact?" *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 43–57, May 2013.

[5] A. Nedić, A. Olshevsky, and C. A. Uribe, "Fast convergence rates for distributed non-Bayesian learning," *IEEE Trans. Autom. Control*, vol. 62, no. 11, pp. 5538–5553, Nov. 2017.

[6] H. Salami, B. Ying, and A. H. Sayed, "Social learning over weakly connected graphs," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 3, no. 2, pp. 222–238, Jun. 2017.

[7] A. Lalitha, T. Javidi, and A. D. Sarwate, "Social learning and distributed hypothesis testing," *IEEE Trans. Inf. Theory*, vol. 64, no. 9, pp. 6161–6179, May 2018.
[8] V. Matta, V. Bordignon, A. Santos, and A. H. Sayed, "Interplay between topology and social learning over weak graphs," *IEEE Open J. Signal Process.*, vol. 1, pp. 99–119, 2020.
[9] V. Bordignon, V. Matta, and A. H. Sayed, "Adaptive social learning," *IEEE Trans. Inf. Theory*, vol. 67, no. 9, pp. 6053–6081, Sep. 2021.
[10] P. Molavi, A. Tahbaz-Salehi, and A. Jadbabaie, "A theory of non-Bayesian social learning," *Econometrica*, vol. 86, no. 2, pp. 445–490, 2018.
[11] J. Z. Hare, C. A. Uribe, L. Kaplan, and A. Jadbabaie, "Non-Bayesian social learning with uncertain models," *IEEE Trans. Signal Process.*, vol. 68, pp. 4178–4193, 2020.
[12] J. Z. Hare, C. A. Uribe, L. Kaplan, and A. Jadbabaie, "A general framework for distributed inference with uncertain models," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 7, pp. 392–405, 2021.
[13] V. Bordignon, V. Matta, and A. H. Sayed, "Adaptation in online social learning," in *Proc. 28th Eur. Signal Process. Conf. (EUSIPCO)*, Jan. 2021, pp. 2170–2174.
[14] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. Int. Workshop Multiple Classifier Syst.* Cham, Switzerland: Springer, 2000, pp. 1–15.
[15] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Inf. Fusion*, vol. 38, pp. 43–54, Nov. 2017.
[16] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[17] 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, pp. 119–139, Aug. 1995.
[18] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, 1998, pp. 92–100.
[19] Y. LeCun, C. Cortes, and C. J. Burges. (2010). *MNIST Handwritten Digit Database*. [Online]. Available: http://yann.lecun.com/exdb/mnist
[20] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 1991.
[21] L. G. Epstein et al., "Non-Bayesian learning," *BE J. Theor. Econ.*, vol. 10, no. 1, pp. 1–20, 2010.
[22] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, nos. 4–5, pp. 311–801, Jul. 2014.
[23] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, Apr. 2014.
[24] V. Matta and A. H. Sayed, "Estimation and detection over adaptive networks," in *Cooperative and Graph Signal Processing*. Amsterdam, The Netherlands: Elsevier, 2018, pp. 69–106.
[25] R. Viswanathan and P. K. Varshney, "Distributed detection with multiple sensors Part I. Fundamentals," *Proc. IEEE*, vol. 85, no. 1, pp. 54–63, Jan. 1997.
[26] R. S. Blum, S. A. Kassam, and H. V. Poor, "Distributed detection with multiple sensors II. Advanced topics," *Proc. IEEE*, vol. 85, no. 1, pp. 64–79, Jan. 1997.
[27] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2018.
[28] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.
[29] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
[30] B. Neyshabur, R. Tomioka, and N. Srebro, "Norm-based capacity control in neural networks," in *Proc. Conf. Learn. Theory*, 2015, pp. 1376–1401.
[31] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," in *Measures of Complexity*. Berlin, Germany: Springer, 2015, pp. 11–30.
[32] L. Devroye, L. Gyorfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, vol. 31. Berlin, Germany: Springer, 2013.
[33] V. Koltchinskii, "Rademacher penalties and structural risk minimization," *IEEE Trans. Inf. Theory*, vol. 47, no. 5, pp. 1902–1914, Jul. 2001.
[34] S. Boucheron, O. Bousquet, and G. Lugosi, "Theory of classification: A survey of some recent advances," *ESAIM Probab. Statist.*, vol. 9, pp. 323–375, Nov. 2005.
[35] P. L. Bartlett, S. Boucheron, and G. Lugosi, "Model selection and error estimation," *Mach. Learn.*, vol. 48, nos. 1–3, pp. 85–113, 2002.
[36] C. Cortes, M. Mohri, and U. Syed, "Deep boosting," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1179–1187.
[37] P. L. Bartlett and S. Mendelson, "Rademacher and Gaussian complexities: Risk bounds and structural results," *J. Mach. Learn. Res.*, vol. 3, pp. 463–482, Mar. 2003.
[38] Y. Freund, R. E. Schapire, and N. Abe, "A short introduction to boosting," *J.-Jpn. Soc. Artif. Intell.*, vol. 14, no. 5, pp. 771–780, Sep. 1999.
[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[40] M. H. DeGroot, *Optimal Statistical Decisions*. Hoboken, NJ, USA: Wiley, 2005.
[41] P. Billingsley, *Probability and Measure*. Hoboken, NJ, USA: Wiley, 2008.
[42] C. McDiarmid, "On the method of bounded differences," in *Surveys in Combinatorics*, vol. 141. Cambridge, U.K.: Cambridge Univ. Press, Aug. 1989, pp. 148–188.
[43] M. Ledoux and M. Talagrand, *Probability in Banach Spaces: Isoperimetry and Processes*. Berlin, Germany: Springer, 2013.

Virginia Bordignon (Member, IEEE) received the degrees in engineering from the École Centrale de Lyon, France, and the Federal University of Rio Grande do Sul (UFRGS), Brazil, the M.S. degree in electrical engineering from UFRGS, and the Ph.D. degree in electrical engineering from École Polytechnique Fédérale de Lausanne (EPFL), Switzerland. She is currently a Post-Doctoral Researcher with the Adaptive Systems Laboratory, EPFL. Her research interests include statistical inference, distributed learning, and information processing over networks.

Stefan Vlaski (Member, IEEE) received the B.Sc. degree in electrical engineering and information technology from Technical University Darmstadt, Darmstadt, Germany, in 2013, and the M.S. degree in electrical engineering and Ph.D. degree in electrical and computer engineering from the University of California at Los Angeles, Los Angeles, CA, USA, in 2014 and 2019, respectively. From 2019 to 2021, he was a Post-Doctoral Researcher with the Adaptive Systems Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. He is currently a Lecturer with Imperial College London, London, U.K., where he conducts research at the intersection of machine learning, network science, and optimization.

Vincenzo Matta (Senior Member, IEEE) is currently a Full Professor of telecommunications with the Department of Information and Electrical Engineering and Applied Mathematics (DIEM), University of Salerno, Italy. He is the author of more than 130 articles published on international journals and proceedings of international conferences. His research interests include adaptation and learning over networks, social learning, statistical inference on graphs, and security in communication networks. He is a member of the Sensor Array and Multichannel Technical Committee of the Signal Processing Society (SPS), and served as an IEEE SPS Representative for the IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS. He serves as an Associate Editor for the IEEE OPEN JOURNAL OF SIGNAL PROCESSING. He served as an Associate Editor for the IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS, the IEEE SIGNAL PROCESSING LETTERS, and the IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS, and a Senior Area Editor for the IEEE SIGNAL PROCESSING LETTERS.

Ali H. Sayed (Fellow, IEEE) worked before as a Distinguished Professor and the Chair of electrical engineering at UCLA. He is currently the Dean of Engineering with the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, where he also leads the Adaptive Systems Laboratory. He is a member of the U.S. National Academy of Engineering (NAE) and The World Academy of Sciences (TWAS). He is a fellow of EURASIP and AAAS. His work has been recognized with several awards, including more recently the 2022 IEEE Fourier Award, the 2020 IEEE Norbert Wiener Society Award, and several best paper awards. He served as the President for the IEEE Signal Processing Society in 2018 and 2019.