ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9414126

NETWORK CLASSIFIERS BASED ON SOCIAL LEARNING

Virginia Bordignon[⋆] *Stefan Vlaski*[⋆] *Vincenzo Matta*[†] *Ali H. Sayed*[⋆]

[⋆] School of Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL)
[†]DIEM, University of Salerno

ABSTRACT

This work proposes a new way of combining independently trained classifiers over space and time. Combination over space means that the outputs of *spatially distributed* classifiers are aggregated. Combination over time means that the classifiers respond to streaming data during testing and continue to improve their performance even during this phase. By doing so, the proposed architecture is able to improve prediction performance over time with unlabeled data. Inspired by social learning algorithms, which require prior knowledge of the observations distribution, we propose a Social Machine Learning (SML) paradigm that is able to exploit the imperfect models generated during the learning phase. We show that this strategy results in consistent learning with high probability, and it yields a robust structure against poorly trained classifiers. Simulations with an ensemble of feedforward neural networks are provided to illustrate the theoretical results.

Index Terms— Distributed classification, social learning, combination of classifiers, neural networks.

1. INTRODUCTION AND RELATED WORK

Social learning strategies allow the classification of unlabeled features by a *heterogeneous* network of agents [1–8]. The heterogeneity of the network is twofold: first, agents may be observing different (possibly non-overlapping) sets of attributes of the same underlying phenomenon; second, their statistical models need not be the same, e.g., two agents may be observing the same attribute from different perspectives. Neighboring agents share statistics about the observed features and diffuse this information across the network to arrive at a conclusion on the nature of the observed phenomenon.

Many social learning approaches exist in the literature that have been shown to yield correct asymptotic learning of the true state of nature under mild identifiability assumptions [1, 3–5, 7, 8]. These results, however, come at a cost: the strategies require prior knowledge of the true underlying distributions for the features. In practice we often have access to feature data only, or to some approximate models for the distributions. For example, uncertain likelihoods in social learning have been considered in [9], albeit only for multinomial distributions. In this work, we will allow for a fairly broad class of distributions.

Another distinguishing aspect is that we will consider cooperation among *spatially distributed* classifiers, and aggregation over *time* of the inference produced from streaming data. An ensemble of classifiers is known to be a more robust structure than an isolated, perhaps poorly trained, classifier [10]. Examples of ensemble approaches are Bagging [11] and Boosting [12], in which classifiers combine weighted decisions across *space*. Boosting requires labeled samples to tune the combinations weights. Both bagging and boosting methods do not address the streaming data case. Other examples include localized Gaussian Process Regression (GPR) methods [13–15], which require labeled samples for online training and focus on kernel-based classifiers. In our work, we are interested in more general classifier structures.

We therefore propose the Social Machine Learning (SML) approach: a decentralized algorithm for combining the outputs of a heterogeneous network of classifiers over space and time, based on the adaptive diffusion algorithm proposed in [7, 16, 17]. The SML structure inherits the following qualities from social learning: the ability to combine classifiers with different dimensions and statistical models, while providing asymptotic performance guarantees in addition to continuous performance improvement even during the prediction phase. We show that i) with high probability, consistent learning occurs despite the imperfectly trained models; and ii) poorly trained classifiers can leverage the networked setup to improve their performance. We exploit these results particularly for the setup of a network of feedforward neural networks (FNN).

Notation: Random variables are written in bold font and deterministic variables in normal font. $\mathbb{E}_x(\cdot)$ and $\mathbb{P}_x(\cdot)$ respectively denote the expectation and probability measure computed with respect to the single random variable $\boldsymbol{x}$.

2. THE DECISION-MAKING PROBLEM

A network of K agents is engaged to accomplish the following decision-making task. There is a *true* underlying binary state of nature represented by an equiprobable binary random variable $\boldsymbol{\gamma} \in \{-1, +1\} \triangleq \Gamma$. As time progresses, each agent collects streaming data arising from the true state of nature. More specifically, agent $k = 1, 2, \ldots, K$ observes at times $i = 1, 2, \ldots$, the random *feature vectors* $\boldsymbol{h}_{k,i} \in \mathcal{H}_k$, which are independent and identically distributed (i.i.d.) over time (but *not* necessarily across the agents). The features $\boldsymbol{h}_{k,i}$ at agent k, given the underlying true hypothesis γ, form a sequence of i.i.d. random vectors distributed according to some conditional distribution (or likelihood):

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

We allow the feature vectors to have different dimensions and attributes across the agents. The goal of the decision learning task is to let each agent learn, as $i \to \infty$, the right hypothesis γ.

If agent k knows the true joint distribution of features and label, it can then apply the paradigm of *Bayes classifiers* [18]. The Bayes classifier is the solution to a maximum-a-posteriori (MAP) problem, where the label estimated by classifier k is the label γ that maximizes $p_k(\gamma|\boldsymbol{h}_{k,1}, \boldsymbol{h}_{k,2}, \ldots, \boldsymbol{h}_{k,i})$, i.e., the posterior probability of γ given the sequence of features $\{\boldsymbol{h}_{k,j}\}_{j=1}^{i}$. However, agents might not have enough information to solve this classification problem alone, e.g., if the signals at agent k are not informative enough (for example, it may be the case that $L_k(h|-1) = L_k(h|+1)$ for all $h \in \mathcal{H}_k$). If however the network as a whole possesses enough information, under the weaker assumption of global identifiability, then a *social learning* scheme can be used and allows agents to learn the truth [1, 4, 5, 7, 8].

This work was supported in part by the Swiss National Science Foundation grant 205121-184999. E-mails: virginia.bordignon@epfl.ch, stefan.vlaski@epfl.ch, vmatta@unisa.it, ali.sayed@epfl.ch.

In practice, the statistical characterization (1) of the features and/or labels is often unknown. We will see next how the individual classifiers can be trained to approximate the unknown distributions.

3. LOCAL INSTANTANEOUS CLASSIFIERS

The fundamental assumption of this work is that the likelihoods $L_k(h|\gamma)$ are *unknown*. To circumvent this lack of knowledge, we assume that each agent is able to train locally some standard binary classifier during a *training* phase. In order to avoid confusion, the random variables pertaining to the training set are topped with a sign $\sim$. Whenever we are dealing with the training phase of the classifiers, feature vectors and labels are indexed with the time subscript n. For the *prediction* (i.e., testing) phase, we use the time subscript i.

Agent k is trained by collecting N_k examples constituted by pairs $\{\widetilde{\boldsymbol{h}}_{k,n}, \widetilde{\boldsymbol{\gamma}}_n\}_{n=1}^{N_k}$. Labels $\widetilde{\boldsymbol{\gamma}}_n$ are uniformly distributed over $\Gamma = \{+1, -1\}$ so the pair $(\widetilde{\boldsymbol{h}}_{k,n}, \widetilde{\boldsymbol{\gamma}}_n)$ is distributed according to the joint distribution:

$$p_k(h, \gamma) = p_k(\gamma) L_k(h|\gamma), \;\; h \in \mathcal{H}_k, \; \gamma \in \Gamma, \tag{2}$$

with the uniform prior $p_k(\gamma) = 1/2$ for $\gamma \in \Gamma$. We are interested in the following statistic:

$$\log \frac{p_k(+1|\boldsymbol{h}_{k,i})}{p_k(-1|\boldsymbol{h}_{k,i})} \overset{\text{(a)}}{=} \log \frac{L_k(\boldsymbol{h}_{k,i}|+1)}{L_k(\boldsymbol{h}_{k,i}|-1)} \tag{3}$$

where the equality in (a) follows from the Bayes rule and the uniform priors assumption. The log-likelihood ratio on the RHS of (3) is positive whenever the observation $\boldsymbol{h}_{k,i}$ is more likely to have come from class $+1$ and negative when it is more likely to have originated from class -1. This is the same sufficient statistic aggregated over space and time in social learning [7, 17] and in signal detection schemes [16, 19, 20].

After training, the classifier will generate approximate posterior models $\widehat{p}_k(\gamma|h)$. Thus, instead of (3), we will rely on the following *logit* statistic:

$$\log \frac{\widehat{p}_k(+1|\boldsymbol{h}_{k,i})}{\widehat{p}_k(-1|\boldsymbol{h}_{k,i})} = \log \frac{\widehat{p}_k(+1|\boldsymbol{h}_{k,i})}{1 - \widehat{p}_k(+1|\boldsymbol{h}_{k,i})} \triangleq f_k(\boldsymbol{h}_{k,i}). \tag{4}$$

The function f_k belongs to a specific class of functions $\mathcal{F}_k : \mathcal{H}_k \mapsto \mathbb{R}$ that depends on the choice of classifier. For example, in 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 $f_k(h; w) = w^\top h$ [18]. Since the logit in (4) operates on the feature vector collected by an *individual* agent in a *single time instant*, we will refer to (4) as a *local instantaneous* logit.

The local instantaneous classifiers are trained by choosing the function f within $\mathcal{F}_k$ that minimizes a suitable risk function $R_k(f)$. We define this optimal function as the *target model*:

$$f_k^o \triangleq \arg\min_{f \in \mathcal{F}_k} R_k(f). \tag{5}$$

In this work we focus on the logistic risk:

$$R_k(f) = \mathbb{E}_{h_k, \gamma} \log \left(1 + e^{-\widetilde{\gamma}_n f(\widetilde{\boldsymbol{h}}_{k,n})}\right), \tag{6}$$

which is commonly used for binary classification tasks for traditional classifiers such as logistic regression or more complex structures such as neural networks with softmax output layers. Note that the expectation is computed under the (unknown) joint distribution of the pair $(\widetilde{\boldsymbol{h}}_{k,n}, \widetilde{\boldsymbol{\gamma}})$. Since all agents rely on a finite set of training samples, they will solve instead an empirical optimization problem:

$$\widetilde{\boldsymbol{f}}_k^N \triangleq \arg\min_{f \in \mathcal{F}_k} \widetilde{\boldsymbol{R}}_k^N(f), \tag{7}$$

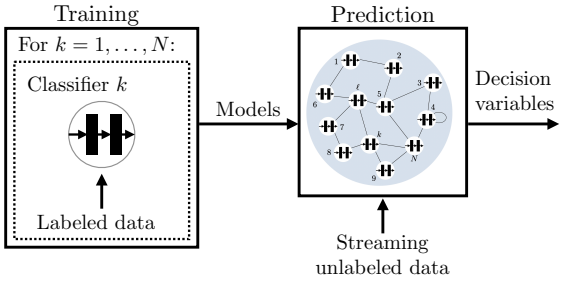

Fig. 1. Social Machine Learning (SML) diagram.

where the empirical risk is in the form of an empirical logistic risk:

$$\widetilde{\boldsymbol{R}}_k^N(f) = \frac{1}{N_k} \sum_{n=1}^{N_k} \log \left(1 + e^{-\widetilde{\gamma}_n f(\widetilde{\boldsymbol{h}}_{k,n})}\right), \tag{8}$$

which is computed over the training set. Since agents solve (7) until convergence, we assume they reach an empirical minimizer $\widetilde{\boldsymbol{f}}_k^N$ that is close enough to the target f_k^o for sufficiently large N_k under ergodicity assumptions. Next, we introduce the algorithm that allows these classifiers to be combined within a network.

4. SOCIAL MACHINE LEARNING

The network is modeled as a strongly connected graph (i.e., where there is always a path in both directions between any two agents and at least one self-loop) with a left-stochastic combination matrix A, whose elements $a_{\ell k}$ are nonnegative, and $a_{\ell k} = 0$ if agent $\ell \notin \mathcal{N}_k$, where $\mathcal{N}_k$ denotes the neighborhood of agent k. Under this condition, we define the Perron eigenvector π as [21]:

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

During the prediction phase, agents are observing *unlabeled* streaming private features $\boldsymbol{h}_{k,i}$. In Fig. 1, we show a diagram depicting the SML approach. After the training phase, the posterior models are used in the prediction phase to form the individual agent's decisions in a social learning setup.

In [7,17], an adaptive version of social learning was introduced, where agents update their beliefs (or opinions) $\boldsymbol{\varphi}_{k,i}(\gamma)$ as[1]:

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

$$\boldsymbol{\varphi}_{k,i}(\gamma) = \frac{\exp\left\{\sum_{\ell=1}^K a_{\ell k} \log \boldsymbol{\psi}_{\ell,i}(\gamma)\right\}}{\sum_{\gamma' \in \Gamma} \exp\left\{\sum_{\ell=1}^K a_{\ell k} \log \boldsymbol{\psi}_{\ell,i}(\gamma')\right\}} \tag{11}$$

where $0 < \delta \ll 1$ is a small step-size parameter. In (10), the agent uses its private observation $\boldsymbol{h}_{k,i}$ to update its belief into an intermediate belief $\boldsymbol{\psi}_{k,i}(\gamma)$. In (11), the agent combines the intermediate beliefs coming from neighbors into its updated belief $\boldsymbol{\varphi}_{k,i}(\gamma)$. Note that these relations rely on knowledge of the exact likelihood functions $L_k(h|\gamma)$, whereas in this work these likelihoods will be estimated during the training phase. The objective is to show that with minimal pre-training, the estimated likelihoods will enable the social learning algorithm to classify unlabeled data correctly with high probability and, moreover, the confidence of the classifier in its decisions will continually grow over time in response to streaming data. This property is fundamentally different from existing static testing phases for traditional classifiers, where classification decisions are instantaneous and are not exploited to improve performance.

[1]The belief $\boldsymbol{\varphi}_{k,i}(\gamma)$ quantifies the confidence of agent k at instant i that γ is the true state of nature.

An equivalent way of representing (10) and (11) is in the form of an *adaptive diffusion strategy*:

$$\boldsymbol{\lambda}_{k,i} = (1-\delta)\sum_{\ell=1}^{K} a_{\ell k}\boldsymbol{\lambda}_{\ell,i-1} + \delta\sum_{\ell=1}^{K} a_{\ell k}\boldsymbol{c}_{\ell,i}, \tag{12}$$

where we defined $\boldsymbol{\lambda}_{k,i} \triangleq \log[\boldsymbol{\varphi}_{k,i}(+1)/\boldsymbol{\varphi}_{k,i}(-1)]$ and $\boldsymbol{c}_{k,i}$ is taken as the log-likelihood ratio seen in the RHS of (3). Eq. (12) has moreover the form of a distributed stochastic gradient algorithm with step-size δ and with a quadratic cost function – see [16, 21]. The algorithm in (12) constructs the aggregate classification variable $\boldsymbol{\lambda}_{k,i}$ from the past information, in the shape of $\boldsymbol{\lambda}_{k,i-1}$, and the present information $\boldsymbol{c}_{k,i}$ received from neighboring classifiers. The structure in (12) will enable cooperation over space and time.

In our approach, in the place of $\boldsymbol{c}_{k,i}$, we will consider the local instantaneous logit statistic $f_k(\boldsymbol{h}_{k,i})$. We also assume that each agent performs a *debiasing* operation before sharing the statistic $f_k(\boldsymbol{h}_{k,i})$, by discounting its empirical mean over the training dataset. We define this *empirical training mean* as:

$$\widetilde{\boldsymbol{\mu}}_k^N(f_k) = \frac{1}{N_k}\sum_{n=1}^{N_k} f_k(\widetilde{\boldsymbol{h}}_{k,n}). \tag{13}$$

The diffusion strategy in (12) is then run with the choice:

$$\boldsymbol{c}_{k,i} = f_k(\boldsymbol{h}_{k,i}) - \widetilde{\boldsymbol{\mu}}_k^N(f_k) \tag{14}$$

Note that $\boldsymbol{c}_{k,i}$ contains two independent sources of randomness. The first, introduced by $f_k(\boldsymbol{h}_{k,i})$, contains the randomness from the prediction sample $\boldsymbol{h}_{k,i}$. The second source of randomness comes from the training samples $\widetilde{\boldsymbol{h}}_{k,n}$, which are introduced in the term $\widetilde{\boldsymbol{\mu}}_k^N(f_k)$. The prediction and training feature vector samples are independent of each other.

The instantaneous decision of agent k, namely $\widehat{\boldsymbol{\gamma}}_{k,i}$, is taken according to the rule:

$$\widehat{\boldsymbol{\gamma}}_{k,i} = \operatorname{sign}(\boldsymbol{\lambda}_{k,i}), \tag{15}$$

where $\operatorname{sign}(x) = +1$, if $x \geq 0$ and $\operatorname{sign}(x) = -1$ otherwise. This choice is motivated by the fact that the logarithmic ratios in (4) are positive whenever the fiducial posterior probability $\widehat{p}_k(+1|h)$ exceeds $1/2$, and are negative otherwise.

From previous work [7] we know that, for sufficiently small values of the step-size δ, the adaptive diffusion strategy in (12) with decision rule in (15) is able to learn consistently[2] the true hypothesis under the following condition. Let

$$\mu_k^+(f_k) \triangleq \mathbb{E}_{L_k(h|+1)} f_k(\boldsymbol{h}_{k,i}), \quad \mu_k^-(f_k) \triangleq \mathbb{E}_{L_k(h|-1)} f_k(\boldsymbol{h}_{k,i}), \tag{16}$$

where the notation $\mathbb{E}_{L_k(h|\gamma)}$ indicates that the expectation is computed under the distribution $L_k(h|\gamma)$. Let also

$$\mu^+(f) \triangleq \sum_{k=1}^{K}\pi_k\mu_k^+(f_k), \quad \mu^-(f) \triangleq \sum_{k=1}^{K}\pi_k\mu_k^-(f_k), \tag{17}$$

where we use the compact notation f to indicate the dependency of the above averages on the group of functions $f_1, f_2, \ldots, f_K$. Then, consistent learning is achieved if:

$$\mu^+(f) > \widetilde{\boldsymbol{\mu}}^N(f) \quad \text{and} \quad \mu^-(f) < \widetilde{\boldsymbol{\mu}}^N(f). \tag{18}$$

For each agent, the result of the training phase is the optimal empirical classifier function $\widetilde{\boldsymbol{f}}_k^N$. Therefore, we are interested in determining if both events described in (18) are likely to simultaneously occur when the classifier functions are given by $\widetilde{\boldsymbol{f}}^N$, i.e., by the group of functions $\widetilde{\boldsymbol{f}}_1^N, \widetilde{\boldsymbol{f}}_2^N, \ldots, \widetilde{\boldsymbol{f}}_K^N$.

[2]In our setting, consistent learning means that the classification error probability can be made arbitrarily small by suitably reducing the value of the step-size δ.

4.1. SML Consistency

In Theorem 1, we will show that the SML strategy consistently learns the truth with high probability, as the number of training samples grows and for a moderately complex 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 average* (initially introduced as Rademacher penalty in [22]). We follow the definition in [23] and introduce, for a class of functions $\mathcal{F}$ and N samples $x_1, x_2 \ldots, x_N \in \mathcal{X}$, the set of vectors $\mathcal{F}\left(x_1^N\right)$ as $(f(x_1), f(x_2), \ldots, f(x_N))$ with $f \in \mathcal{F}$. Then, the (empirical) Rademacher average associated with $\mathcal{F}\left(x_1^N\right)$ is:

$$\mathcal{R}\left(\mathcal{F}\left(x_1^N\right)\right) \triangleq \mathbb{E}_r\left|\sup_{f\in\mathcal{F}}\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{r}_n f(x_n)\right|, \tag{19}$$

where $\boldsymbol{r}_n$ are independent and identically distributed Rademacher random variables, i.e., with $\mathbb{P}(\boldsymbol{r}_n = 1) = \mathbb{P}(\boldsymbol{r}_n = -1) = 1/2$.

Theorem 1 (SML Consistency). *For the logistic loss, assume that $R(f^o) < \log 2$ and that $f_k(h_k) < B$ for every $h_k \in \mathcal{H}_k$ and $k = 1, 2, \ldots, K$, with $B > 0$. For any $d \in (0, -\log(e^{R(f^o)} - 1))$, we have the following bound for the probability of consistent learning:*

$$\begin{aligned}
&\mathbb{P}\left(\mu^+(\widetilde{\boldsymbol{f}}^N) > \widetilde{\boldsymbol{\mu}}^N(\widetilde{\boldsymbol{f}}^N),\, \mu^-(\widetilde{\boldsymbol{f}}^N) < \widetilde{\boldsymbol{\mu}}^N(\widetilde{\boldsymbol{f}}^N)\right) \\
&\geq 1 - 2\sum_{k=1}^{K}\exp\left\{\frac{-\left(d-\rho_N^{(k)}\right)^2 N_k}{2B^2}\right\} \\
&\quad - \sum_{k=1}^{K}\exp\left\{\frac{-\left(\frac{\Delta-R(f^o)}{2}-\rho_N^{(k)}\right)^2 N_k}{2B^2}\right\},
\end{aligned} \tag{20}$$

with $\Delta \triangleq \log(1+e^{-d})$, $R(f^o) = \sum_{k=1}^{K}\pi_k R_k(f_k^o)$ and

$$\rho_N^{(k)} \triangleq 2\mathbb{E}_{h_k}\mathcal{R}(\mathcal{F}_k(\boldsymbol{h}_1^{N_k})). \tag{21}$$

■

Sketch of proof: The proof cannot be included for space limitations, but we present some insights for it. First, define the average network risk as $R(f) \triangleq \sum_{k=1}^{K}\pi_k R_k(f_k)$. We have that:

$$\begin{aligned}
R(\widetilde{\boldsymbol{f}}^N) &\overset{(a)}{\geq} \sum_{k=1}^{K}\pi_k\log\left(1+\exp\left(-\mathbb{E}_{h_k,\gamma}\boldsymbol{\gamma}\widetilde{\boldsymbol{f}}_k^N(\boldsymbol{h}_{k,i})\right)\right) \\
&\overset{(b)}{\geq} \log\left(1+\exp\left(-\sum_{k=1}^{K}\pi_k\mathbb{E}_{h_k,\gamma}\boldsymbol{\gamma}\widetilde{\boldsymbol{f}}_k^N(\boldsymbol{h}_{k,i})\right)\right) \\
&= \log\left(1+\exp\left(-\frac{(\mu^+(\widetilde{\boldsymbol{f}}^N)-\mu^-(\widetilde{\boldsymbol{f}}^N))}{2}\right)\right),
\end{aligned} \tag{22}$$

where in (a) and (b) we used Jensen's inequality with the convexity of $\log(1+e^x)$. The inequality in (22) translates into:

$$R(\widetilde{\boldsymbol{f}}_k^N) \leq \log\left(1+e^{-d}\right) \Longrightarrow \frac{\mu^+(\widetilde{\boldsymbol{f}}^N)-\mu^-(\widetilde{\boldsymbol{f}}^N)}{2} \geq d. \tag{23}$$

If now the risk in (23) is sufficiently close to the risk $R(f^o) < \log 2$, we see that $d > 0$. In other words, a good generalization capability of the trained classifiers $\widetilde{\boldsymbol{f}}_k^N$, i.e., a lower risk, implies a larger gap between means μ^+ and μ^-. In view of (18), this gap

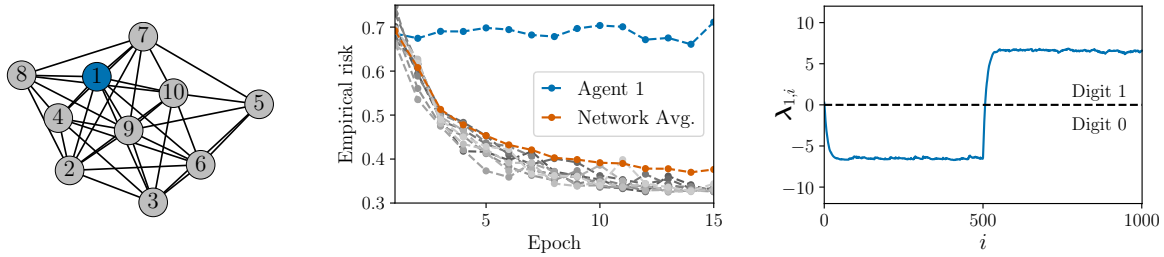

Fig. 2. Network classifiers for handwritten digits classification. *Leftmost panel*: Network topology. *Middle panel*: Empirical risk evolution for agent 1 (in blue), the rest of the agents (different shades of gray) and for the network average empirical risk (in red). *Rightmost panel*: Decision variable of agent 1 over the prediction phase, where the dashed line indicates the decision boundary between digits 1 and 0.

can be sufficient to achieve consistent learning, provided that the concentration error of the empirical training mean around the true mean has a maximum error of d.

According to these observations, in order to obtain (20), it is necessary to examine the statistical concentration properties of the risk and of the empirical training mean. This task is complicated by the fact that both quantities depend on random functions $\boldsymbol{f}_k^N$. For this reason, we must resort to *uniform* (w.r.t. the class of functions) laws of large numbers. These types of concentration results are based notably on McDiarmid's inequality [24]. □

In what concerns the expression in (20), the term $\rho_N^{(k)}$ contains the complexity of the chosen classifier structure, which depends itself on the training set size N_k. We will see in the next section that it evolves as $\mathcal{O}(1/\sqrt{N_k})$ for feedforward neural networks. Therefore as N_k grows, we can neglect $\rho_N^{(k)}$. Now, since by assumption $d \in (0, -\log(e^{R(f^o)}-1))$, both terms d and $\Delta - R(f^o)$ are strictly positive. This implies that both exponential terms in (20) vanish, which in turn implies that the probability of consistent learning for the proposed strategy approaches 1, as the training sets grow.

4.2. Neural Network Complexity

In this section, we complement the result from Theorem 1 by showing that the term $\rho_N^{(k)}$ in (21), which depends on the Rademacher complexity of the classifier, vanishes with an increasing number of training samples in the case of feedforward neural networks (FNN). Assume that one classifier has the structure of a FNN with L layers (excluding the input layer) and activation function σ. We drop index k as we are referring to a single FNN. Each layer ℓ consists of n_ℓ nodes, equivalently the size of layer ℓ is given by n_ℓ.

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

$$g_m^{(\ell)}(h) = \sum_{j=1}^{n_{\ell-1}} w_{mj}^{(\ell)} \sigma\left(g_j^{(\ell-1)}(h)\right) - \theta_m^{(\ell)}. \tag{24}$$

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

$$g_m^{(1)}(h) = \sum_{j=1}^{n_0} w_{mj}^{(1)} h_j - \theta_m^{(1)}, \tag{25}$$

where the input vector h has dimension n_0.

For a FNN 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. In this case the logit function is given by:

$$f^{\mathsf{NN}}(h) = \log \frac{\widehat{p}(+1|h)}{\widehat{p}(-1|h)} = z_1 - z_2 \tag{26}$$

where we say that f^{NN} belongs to a class of functions $\mathcal{F}^{\mathsf{NN}}$, which is parameterized by matrices W_ℓ and bias vectors $\theta^{(\ell)}$, for $\ell = 1, 2, \ldots, L$, according to (24), (25) and (26).

We are interested in finding an expression for the Rademacher complexity of class $\mathcal{F}^{\mathsf{NN}}$ described above. An upper bound for this complexity can be found in Lemma 1 inspired by results from [25] (proof is omitted due to space limitations).

Lemma 1 (Rademacher Complexity of FNNs). *Consider an L-layered feedforward neural network, satisfying $\|w_m^{(\ell)}\|_1 \leq b$, $|\theta^{(\ell)}(m)| \leq a$, for every node $m = 1, 2, \ldots, n_\ell$ and every layer $\ell = 1, 2, \ldots, L$. Assume that the input vector $h \in \mathbb{R}^{n_0}$ satisfies $\|h\|_\infty \leq c$[3], that the activation function $\sigma(x)$ is Lipschitz with constant L_σ and that $\sigma(0) = 0$. Then the Rademacher average for the set of vectors $\mathcal{F}^{\mathsf{NN}}(h_1^N)$ is bounded by:*

$$\mathcal{R}(\mathcal{F}^{\mathsf{NN}}(h_1^N)) \leq \frac{2}{\sqrt{N}} \left[(2bL_\sigma)^{L-1} bc\sqrt{2\log(2n_0)} + \sum_{\ell=0}^{L-1} (2bL_\sigma)^\ell a \right]. \tag{27}$$

■

5. SIMULATION RESULTS

To illustrate the proposed strategy, we consider a network of 10 agents, whose topology can be seen in Fig. 2. The combination matrix is generated using an averaging rule [21], and we ensure that at least one agent possesses a self-loop.

We consider the MNIST dataset [26], using digits 0 and 1 for a binary classification task. Feature vectors are the 784 pixels of each image of the handwritten digits. Each agent disposes of 98 training samples for each class of digits. With this dataset, each agent trains its own classifier, which has the structure of a feedforward neural network with one hidden layer with 64 nodes, and activation function $\arctan(\cdot)$. To illustrate the robustness of the network of classifiers, we purposely tamper with the dataset for Agent 1, highlighted in the leftmost panel of Fig. 2. To obtain a poor training performance, we provide agent 1 with only digits 1 during training and randomly assigned labels.

The training phase is run using mini-batch iterates of 10 samples, over 15 epochs. The empirical risk evolution at the individual agents as training progresses is shown in Fig. 2 (middle panel), where we can see how the training performance of agent 1 is much worse than the performance of other agents. The average empirical risk, which is given by $\sum_{k=1}^K \pi_k \widehat{\boldsymbol{R}}^N(\boldsymbol{f}^N)$, is hardly affected by the deviating behavior of agent 1.

In the prediction phase, all agents are receiving streaming observations, i.e., images of digits. Agents are observing digits 0 until the time instant 500, from which they start observing digits 1. In the rightmost panel of Fig. 2, we see the classification variable $\boldsymbol{\lambda}_{1,i}$ of agent 1 over time, showing that, although agent 1 has a poorly trained model, it is able to learn consistently the true state.

[3] $\|x\|_\infty$ denotes the ℓ_∞-norm defined as $\|x\|_\infty \triangleq \max_i |x_i|$.

6. REFERENCES

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

[2] 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 Processing Magazine*, vol. 30, no. 3, pp. 43–57, 2013.

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

[4] H. Salami, B. Ying, and A. H. Sayed, "Social learning over weakly connected graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 2, pp. 222–238, 2017.

[5] A. Lalitha, T. Javidi, and A. D. Sarwate, "Social learning and distributed hypothesis testing," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6161–6179, 2018.

[6] V. Matta, V. Bordignon, A. Santos, and A. H. Sayed, "Interplay between topology and social learning over weak graphs," *IEEE Open Journal of Signal Processing*, vol. 1, pp. 99–119, 2020.

[7] V. Bordignon, V. Matta, and A. H. Sayed, "Adaptive social learning," *submitted for publication*, available at arXiv:2004.02494 [cs.MA], 2020.

[8] X. Zhao and A. H. Sayed, "Learning over social networks via diffusion adaptation," in *Proc. Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2012, pp. 709–713.

[9] J. Z. Hare, C. A. Uribe, L. Kaplan, and A. Jadbabaie, "Non-Bayesian social learning with uncertain models," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4178–4193, 2020.

[10] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

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

[12] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[13] V. Tresp, "A Bayesian committee machine," *Neural computation*, vol. 12, no. 11, pp. 2719–2741, 2000.

[14] D. Nguyen-Tuong, J. Peters, and M. Seeger, "Local Gaussian process regression for real time online model learning and control," in *Proc. International Conference on Neural Information Processing Systems*, 2008, pp. 1193–1200.

[15] A. Lederer, A. J. O. Conejo, K. Maier, W. Xiao, and S. Hirche, "Real-time regression with dividing local Gaussian processes," available at arXiv:2006.09446 [cs.LG], 2020.

[16] V. Matta and A. H. Sayed, *Estimation and detection over adaptive networks*, pp. 69–106, Elsevier, 2018.

[17] V. Bordignon, V. Matta, and A. H. Sayed, "Adaptation in online social learning," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2020, pp. 2170–2174.

[18] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, MIT press, 2018.

[19] Steven M Kay, *Fundamentals of Statistical Signal Processing: Detection Theory*, Prentice Hall PTR, 1993.

[20] H Vincent Poor, *An Introduction to Signal Detection and Estimation*, Springer Science & Business Media, 2013.

[21] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. ARTICLE, pp. 311–801, 2014.

[22] V. Koltchinskii, "Rademacher penalties and structural risk minimization," *IEEE Transactions on Information Theory*, vol. 47, no. 5, pp. 1902–1914, 2001.

[23] S. Boucheron, O. Bousquet, and G. Lugosi, "Theory of classification: A survey of some recent advances," *ESAIM: Probability and Statistics*, vol. 9, pp. 323–375, 2005.

[24] C. McDiarmid, "Concentration," in *Probabilistic methods for algorithmic discrete mathematics*, pp. 195–248. Springer, 1998.

[25] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 463–482, 2002.

[26] Y. LeCun, C. Cortes, and C. J. Burges, "Mnist handwritten digit database," 2010, ATT Labs [Online]. Available: `http://yann.lecun.com/exdb/mnist`.