DECENTRALIZED LEARNING OF DECISION MODELS FOR CLASSIFICATION WITH DEPENDENT AGENTS

Felice Scala[⋆] *Marco Carpentiero*[⋆] *Vincenzo Matta*[⋆] *Ali H. Sayed*[†]

[⋆] DIEM, University of Salerno, Fisciano, Italy
[†] EPFL, Lausanne, Switzerland

ABSTRACT

Under distributed decision-making (a.k.a. social learning), several agents cooperate according to a communication graph to solve a classification task. A traditional assumption in social learning is that the spatial dependence across the agents' data is not available, while each individual agent is assumed to know a *local* model encoding the *marginal* distribution of its personal data. Recent work shows that, when some joint parametric model is available to describe the spatial dependence, the theory of copulas can be exploited to learn the joint distribution of the data and improve the decision-making performance. However, in most cases a model for the joint distribution is not available. This work addresses this challenging scenario. We propose a *fully decentralized* strategy where the network of agents reproduces a multilayer perceptron to build a *discriminative model capturing the spatial dependence*. It is shown that this strategy sensibly outperforms the existing strategies.

Index Terms— Distributed decision-making, consensus, neural networks, social learning.

1. INTRODUCTION

In this work we study distributed decision-making. Several agents, connected by a graph, cooperate with their neighbors to solve a classification task. In recent years, this research problem has been popularized within the framework of *social learning (SL)* [1, 2, 3, 4, 5, 6]. A well-known fact from Bayesian decision theory is that the optimal manner to solve this problem is to compute the posterior probability of each hypothesis (given the data collected over time by all the agents) and decide in favor of the hypothesis that yields the maximum a posteriori probability (MAP).

A common obstacle encountered in social learning is that the agents are usually equipped with only *local* statistical models, i.e., with the marginal likelihoods that describe their own personal data, while no joint distribution across the agents is available [1]. As a matter of fact, the marginal distributions are local to each agent, while the dependence across the agents is *non-local*. For this reason, building the joint distribution (i.e., the optimal Bayesian solution) in a decentralized manner is challenging, which motivated the theory of *non-Bayesian* social learning [1, 3, 4, 5, 6]. This theory led to several useful algorithms that, ignoring spatial dependence, guarantee the convergence to the correct hypothesis for sufficiently long observation windows, for all agents, provided that the network is strongly connected and a global identifiability condition holds [1, 3, 4, 5, 6].

The work of M. Carpentiero and V. Matta was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

However, implementing a non-Bayesian strategy entails a loss in performance with respect to the optimal Bayesian solution. This is why it would be important for the agents to capture the spatial dependence, so as to be able to approximate the optimal Bayesian posterior. Capturing the dependence among the agents might lead to a sensible improvement in the classification performance, especially in scenarios where the discriminative power lies in the dependence itself. An example of the latter type is anomaly detection over IoT networks whose devices transmit data that are always not anomalous if examined individually. The anomalous behavior arises when some compromised devices transmit highly correlated patterns of data.

The problem of decentralized estimation of the joint distribution has been recently addressed in [7]. It is shown that, when a parametric copula model for spatial dependence is available, a decentralized algorithm can be designed to learn the copula parameters and build a decision strategy outperforming traditional social learning. Unfortunately, in many practical classification problems, a parametric statistical model for the joint likelihood is missing. Under the centralized setting, recent work addresses this issue from a generative perspective to learn a joint distribution described by an unknown copula model [8]. However, in classification problems, it is not necessary to learn the joint likelihood, and one can rely instead on a discriminative decision model (like a logistic model). Our main contribution is to show that this objective can be attained in a *fully decentralized* manner. We establish that it is possible *to learn the same decision model as a centralized multilayer perceptron by allowing only local information exchange between neighbors and preserving the privacy of the agents' data*. To this end, we exploit *finite-time consensus* algorithms [9, 10, 11, 12, 13]. Simulations reveal significant advantages with respect to the existing strategies.

2. BACKGROUND

We consider a network of K agents that wish to discover a hypothesis $\theta \in \Theta = \{1, \ldots, H\}$ after observing a stream of data. We will denote the data at agent k and time t by $\boldsymbol{x}_k(t) \in \mathbb{R}$,[1] where bold font is used to denote random quantities. The observations $\boldsymbol{x}_k(t)$ are independent and identically distributed over time, but can be dependent across the agents. They are conveniently collected into a vector $\boldsymbol{x}(t) = [\boldsymbol{x}_1(t), \ldots, \boldsymbol{x}_K(t)]$. For any fixed $\theta \in \Theta$, the joint generative model of the data is denoted by $\ell(x|\theta)$ (time dependence omitted due to the identical distribution over time). The observation vector $\boldsymbol{x}(t)$ generated at each time t is governed by the model $\ell(x|\theta^\star)$, where $\theta^\star$ is the *true* hypothesis in force.

[1] Scalar data are assumed to keep the notation simpler and the presentation compatible with the space limitations. From the analysis in the following, it is seen that our results easily extend to handle data with arbitrary dimension.

The decision strategy minimizing the error probability is the maximum a posteriori probability (MAP) rule, which, applied after t time instants, corresponds to choosing the hypothesis that maximizes the posterior

$$p\left(\theta|\{x(\tau)\}_{\tau=1}^{t}\right) = \frac{\prod_{\tau=1}^{t}\ell(x(\tau)|\theta)}{\sum_{\theta'\in\Theta}\prod_{\tau=1}^{t}\ell(x(\tau)|\theta')}, \tag{1}$$

where we assumed for simplicity that the hypotheses have uniform prior distribution. Note that (1) can be rewritten as

$$p\left(\theta|\{x(\tau)\}_{\tau=1}^{t}\right) = \frac{\prod_{\tau=1}^{t}e^{h(x(\tau)|\theta)}}{\sum_{\theta'\in\Theta}\prod_{\tau=1}^{t}e^{h(x(\tau)|\theta')}}, \tag{2}$$

where we introduced the *decision statistic* $h(x|\theta) \triangleq \log\frac{\ell(x|\theta)}{\ell(x|H)}$. Note that $h(x|H) = 0$ by definition and that the decision statistic can be defined with any hypothesis $\theta \in \Theta$ at the denominator, not necessarily $\theta = H$. Unfortunately, in most decision problems the exact form of the log likelihood ratio $h(x|\theta)$ is not available and must be learned from some training data. More specifically, an approximation of the optimal decision statistic $h(x|\theta)$ is chosen from a family of parametric functions $h_w(x|\theta)$, whose parameter w need to be learned. This procedure leads to the *approximate* posterior

$$p_w\left(\theta|\{x(\tau)\}_{\tau=1}^{t}\right) = \frac{\prod_{\tau=1}^{t}e^{h_w(x(\tau)|\theta)}}{\sum_{\theta'\in\Theta}\prod_{\tau=1}^{t}e^{h_w(x(\tau)|\theta')}}. \tag{3}$$

The problem of learning the decision statistics in social learning has been tackled before, e.g., in the context of social machine learning [1, 14, 15] and social learning under uncertain models [16]. However, all these works refer to learning the *marginal* models, and not the joint model that accounts for the dependence across the agents. This introduces a critical difference, since in [1, 14, 15, 16] each agent learns *individually* its own marginal model, while in the present work the agents need to cooperate to learn the joint model.

Summing up, the decision-making procedure requires: i) a *training stage*, where the agents learn in a decentralized manner the parameter w to select a decision statistic h_w; and ii) a *prediction stage*, where the agents build the posterior (3) by computing in a decentralized manner the learned statistic h_w. Thanks to (3), the posterior is computed sequentially as streaming observations arrive [1].

2.1. Learning the decision statistic

We wish to build suitable decision functions $h_w(x|\theta)$ by learning a function through a multilayer perceptron (MLP) [17, 18, 19, 20, 21]. An MLP consists of L layers: i) the layer $l = 1$ is the input layer, fed by the data $x = [x_1, \ldots, x_K]$, i.e., by the vector collecting the agents' observations; ii) the layer $l = L$ is the output layer yielding the $H - 1$ approximate decision statistic values; iii) the remaining layers are hidden layers. Each layer has N_l nodes and is characterized by a weight matrix $W^{(l)}$, whose entries are denoted by $w_{n'n}^{(l)}$, with $n' = 1, \ldots, N_{l-1}$ and $n = 1, \ldots, N_l$. Since the input x has K entries, $N_0 = K$. In the following, we denote by w the *global parameter* that collects all the weights of the MLP. For $l = 1$, each node $n = 1, \ldots, N_1$ computes

$$g_n^{(1)}(x) = \sum_{k=1}^{K} w_{kn}^{(1)} x_k, \tag{4}$$

while each node n of layers $l = 2, \ldots, L$ computes

$$g_n^{(l)}(x) = \sum_{n'=1}^{N_l-1} w_{n'n}^{(l)}\sigma\left(g_n^{(l-1)}(x)\right), \tag{5}$$

where σ is a nonlinear activation function. The last layer L produces the $H - 1$ approximate decision statistic values, namely,

$$g_\theta^{(L)}(x) = h_w(x|\theta), \quad \theta = 1, \ldots, H-1. \tag{6}$$

We compactly represent the final output computed by the MLP as $f_w(x) \triangleq [h_w(x|1), \ldots, h_w(x|H-1)]$. A decision statistic $f_w(x)$ is selected by minimizing with respect to w a suitable cost function. One standard criterion is to minimize the (conditional) cross-entropy between the true posterior distribution in (2) and the approximate one in (3), which corresponds to minimizing the cost function [1, 17]

$$J(w) = \mathbb{E}\log\frac{\sum_{\theta'\in\Theta}e^{h_w(\boldsymbol{x}|\theta')}}{e^{h_w(\boldsymbol{x}|\boldsymbol{\theta})}}. \tag{7}$$

However, the expectation in (7) is taken over the true joint distribution of $(\boldsymbol{x}, \boldsymbol{\theta})$, which is of course unknown. For this reason, $J(w)$ is in practice replaced by an empirical cost computed over training data, as we now describe. The data set used to train the MLP is composed by pair of entries $(\boldsymbol{x}(1), \boldsymbol{\theta}(1))$, $(\boldsymbol{x}(2), \boldsymbol{\theta}(2)), \ldots$, where, for $i = 1, 2, \ldots$, the hypothesis $\boldsymbol{\theta}(i)$ is uniformly distributed in Θ, whereas $\boldsymbol{x}(i) = [\boldsymbol{x}_1(i), \ldots, \boldsymbol{x}_K(i)]$ is distributed according to the true joint model $\ell(\boldsymbol{x}(i)|\boldsymbol{\theta}(i))$. Different optimization routines can be used for training. One example that will be useful next is the popular Stochastic Gradient Descent (SGD) algorithm, which produces the sequence of estimated parameters $\boldsymbol{w}(i)$ defined by the following recursion, for some small step-size $\mu > 0$:

$$\boldsymbol{w}(i) = \boldsymbol{w}(i-1) - \mu\,\nabla\widehat{J}(\boldsymbol{w}(i-1); \boldsymbol{x}(i), \boldsymbol{\theta}(i)), \tag{8}$$

where

$$\widehat{J}(w; \boldsymbol{x}(i), \boldsymbol{\theta}(i)) \triangleq \log\frac{\sum_{\theta'\in\Theta}e^{h_w(\boldsymbol{x}(i)|\theta')}}{e^{h_w(\boldsymbol{x}(i)|\boldsymbol{\theta}(i))}} \tag{9}$$

is the *instantaneous approximation* for the cost function [17].

2.2. Finite-Time Consensus Algorithm

The decentralized strategy we are going to present aims at implementing the MLP training and prediction stages in a decentralized manner. To this end, we will see that the main ingredient is the decentralized evaluation of the first-layer linear combinations (4). To accomplish this task, in the present work we focus on the *finite-time consensus* (FTC) paradigm, which we summarize as follows.

We introduce a weighted graph to describe the communication structure linking the agents. This graph is conveniently represented by a *combination matrix* $A = [a_{jk}] \in \mathbb{R}^{K\times K}$. The condition $a_{jk} = 0$ means that agent k cannot receive information from agent j, whereas the condition $a_{jk} > 0$ means that it can. Accordingly, the neighborhood of agent k is defined as $\mathcal{N}_k \triangleq \{j : a_{jk} > 0\}$. Following the standard requirements adopted in the theory of decentralized optimization and learning, the combination matrix A will be assumed *primitive and doubly stochastic* [17].

Given some values attached to each agent, the goal of a consensus algorithm is to allow all agents to evaluate the same linear combination of these values by exchanging information only with their neighbors. To illustrate the consensus algorithm, assume each agent k has a value $s_{k,0}$, and consider the iterative updates $s_{k,t} = \sum_{j\in\mathcal{N}_k} a_{jk}s_{j,t-1}$ (which, incidentally, correspond to the *asymptotic consensus algorithm* in the sense that $s_{k,t}$ converges as $t \to \infty$ to the *arithmetic average* of the initial states [17, 22, 23, 24]). In the *exact* or *finite-time* consensus framework that we consider here, it is shown that the same result can be attained in a *finite number of iterations* by using the following FTC routine.

We denote by $q_k(A) \triangleq \sum_{t=0}^{D_k+1} \alpha_{k,t} A^t$ the *minimal polynomial associated with the k-th network agent* corresponding to the combination matrix A [9]. Specifically, this is the *monic* polynomial (meaning that the coefficient α_{k,D_k+1} is equal to 1) that has the minimal degree D_k+1 among all polynomials satisfying the relation $e_k q_k(A) = 0$, with e_k being the $1 \times K$ basis vector with its k-th entry equal to 1 and all other entries equal to 0 [9]. By defining the coefficients $\beta_{k,t} \triangleq \sum_{\tau=0}^{D_k-t} \alpha_{k,D_k+1-\tau}$, it can be proved that $K \frac{\sum_{t=0}^{D_k} \beta_{k,t}\, s_{k,t}}{\sum_{t=0}^{D_k} \beta_{k,t}} = \sum_{k=1}^{K} s_{k,0}$. Thus, after D_k iterations required to compute the values $s_{k,t}$, agent k reaches consensus [9]. Since it is known that $D_k + 1 \leq K$, we conclude that all agents reach consensus in at most $K-1$ iterations [9]. [2]

3. DECENTRALIZED DEPENDENT DECISION-MAKING

To perform decision-making, all the network agents agree to reproduce a target centralized MLP defined by a certain desired structure. During the prediction stage, given some optimized parameter w obtained after training, it is necessary to evaluate the decision statistic $f_w(x)$. Examining the subsequent computations across the MLP layers described in Sec. 2.1, we observe that, for each parameter w and each pair (x, θ), the decision statistic $f_w(x)$ is a function of the linear combinations (4) and of the MLP weight matrices from the second layer onward. This means that $f_w(x)$ can be represented as

$$f_w(x) = \psi\left(\{g_n^{(1)}(x)\}_{n=1}^{N_1}\,,\, \{W^{(l)}\}_{l=2}^{L}\right) \tag{10}$$

for some function ψ. For $n = 1, \ldots, N_1$, the linear combinations $g_n^{(1)}(x)$ in (4) can be obtained in a decentralized manner by using the FTC algorithm where agent k starts with initial values $w_{kn}^{(1)}\, x_k$. Owing to the structure in (10), this means that all agents can compute the centralized MLP output, namely, the decision statistic $f_w(x)$, by the Decentralized Dependent Decision-making (D^3) prediction strategy summarized in Algorithm 1.

Algorithm 1 D^3 prediction

Require: Each agent k knows $W^{(l)}$ for $l > 1$ and $\{w_{kn}\}_{n=1}^{N_1}$
1: **for** $n = 1, \ldots, N_1$ and $k = 1, \ldots, K$ **do**
2: apply FTC with initial state $w_{kn}^{(1)}\, x_k$ to get $g_n^{(1)}(x)$ in (4)
3: traverse the MLP layers $l = 2, \ldots, L$ to compute (10)
4: **end for**

Observe that D^3 requires that agent k knows only the k-th row of $W^{(1)}$. Notably, since the value $w_{kn}^{(1)}$ is unknown to the other agents and since in the FTC step agent k shares the products $w_{kn}^{(1)}\, x_k$, the data x_k remains private. We have in fact proved the following result.

Proposition 1 (D^3 in the prediction phase). *Consider the centralized MLP described in Sec. 2.1 with weight matrices $W^{(l)}$, for $l = 1, \ldots, L$. Consider a network of K agents equipped with a primitive and doubly stochastic combination matrix and assume that each agent $k = 1, \ldots, K$ possesses the weight matrices $W^{(l)}$ for $l > 1$ and only the k-th row of $W^{(1)}$. Then, the D^3 strategy described by Algorithm 1 allows each agent k to compute, in a fully decentralized manner, the same decision function $f_w(x)$ as the centralized MLP.*

Proposition 1 implies that, during training, it suffices that each agent k learns the same weight matrices as the centralized MLP for layers $l > 1$, and the k-th row of the first-layer matrix. We now propose a decentralized strategy (D^3 training) that attains this goal. Note that in Proposition 1 all agents are assumed to know the matrices $W^{(l)}$ for $l > 1$. We will see that the D^3 training strategy guarantees this common knowledge in a *decentralized* manner, provided that the agents agree on the same initial matrices $W^{(l)}$ for $l > 1$.

It is convenient to start by examining the case where the optimization routine used for training is SGD. Exploiting (10), it is readily seen that the gradient of the cost function (9) has the following two properties. Regarding the weights corresponding to layer $l = 1$, in view of (4) we have, for $k = 1, \ldots, K$ and $n = 1, \ldots, N_1$,

$$\frac{\partial \widehat{J}(w; x, \theta)}{\partial w_{kn}^{(1)}} = x_k\, \varphi_{kn}^{(1)}\left(\{g_n^{(1)}(x)\}_{n=1}^{N_1}\,,\, \{W^{(l)}\}_{l=2}^{L}\,,\, \theta\right) \tag{11}$$

for some function $\varphi_{kn}^{(1)}$. For the other layers $l > 1$, for $n' = 1, \ldots, N_{l-1}$ and $n = 1, \ldots, N_l$, we have

$$\frac{\partial \widehat{J}(w; x, \theta)}{\partial w_{n'n}^{(l)}} = \varphi_{n'n}^{(l)}\left(\{g_n^{(1)}(x)\}_{n=1}^{N_1}\,,\, \{W^{(l)}\}_{l=2}^{L}\,,\, \theta\right) \tag{12}$$

for some function $\varphi_{n'n}^{(l)}$. Accordingly, by letting

$$\boldsymbol{g}_n^{(1)}(i) \triangleq \sum_{k=1}^{K} \boldsymbol{w}_{kn}^{(1)}(i-1)\boldsymbol{x}_k(i), \tag{13}$$

the SGD recursion (8) performed by the centralized MLP on the global parameter $\boldsymbol{w}(i)$ – with the corresponding weight matrices being denoted by $\boldsymbol{W}^{(l)}(i)$ – is equivalent, in terms of the individual weights, to the following recursions: for $k = 1, \ldots, K$ and $n = 1, \ldots, N_1$,

$$\begin{aligned}\boldsymbol{w}_{kn}^{(1)}(i) &= \boldsymbol{w}_{kn}^{(1)}(i-1) \\ &\quad -\mu\, \boldsymbol{x}_k(i)\, \varphi_{kn}^{(1)}\Big(\{\boldsymbol{g}_n^{(1)}(i)\}_{n=1}^{N_1}\,,\, \{\boldsymbol{W}^{(l)}(i-1)\}_{l=2}^{L}\,,\, \boldsymbol{\theta}(i)\Big)\,,\end{aligned} \tag{14}$$

while for $l > 1$, $n' = 1, \ldots, N_{l-1}$, and $n = 1, \ldots, N_l$,

$$\begin{aligned}\boldsymbol{w}_{n'n}^{(l)}(i) &= \boldsymbol{w}_{n'n}^{(l)}(i-1) \\ &\quad -\mu\, \varphi_{n'n}^{(l)}\left(\{\boldsymbol{g}_n^{(1)}(i)\}_{n=1}^{N_1}\,,\, \{\boldsymbol{W}^{(l)}(i-1)\}_{l=2}^{L}\,,\, \boldsymbol{\theta}(i)\right).\end{aligned} \tag{15}$$

Equations (14) and (15) can be used to design the sought decentralized training strategy. Actually, the result in Proposition 2 further ahead holds for a class of optimization routines more general than SGD, provided that the two conditions stated in the next assumption, which extend (14) and (15), are satisfied.

Assumption 1 (Optimization routine)**.** *The optimization routine used for MLP training satisfies the following conditions:*

$$\begin{aligned}&\boldsymbol{w}_{kn}^{(1)}(i) = \\ &\phi_{kn}^{(1)}\left(\{\boldsymbol{W}^{(l)}(0)\}_{l=2}^{L}\,,\, \Big\{\boldsymbol{x}_k(\tau)\,,\, \{\boldsymbol{g}_n^{(1)}(\tau)\}_{n=1}^{N_1}\,,\, \boldsymbol{\theta}(\tau)\Big\}_{\tau=1}^{i}\right)\end{aligned} \tag{16}$$

for some function $\phi_{kn}^{(1)}$. For $l > 1$, $n' = 1, \ldots, N_{l-1}$ and $n = 1, \ldots, N_l$,

$$\boldsymbol{w}_{n'n}^{(l)}(i) = \phi_{n'n}^{(l)}\left(\{\boldsymbol{W}^{(l)}(0)\}_{l=2}^{L}\,,\, \Big\{\{\boldsymbol{g}_n^{(1)}(\tau)\}_{n=1}^{N_1}\,,\, \boldsymbol{\theta}(\tau)\Big\}_{\tau=1}^{i}\right) \tag{17}$$

for some function $\phi_{n'n}^{(l)}$.

[2] When each agent k knows only its weights $\{a_{jk}\}_{j \in \mathcal{N}_k}$, there exist decentralized algorithms to compute $\beta_{k,t}$ [9, 10, 11, 12].

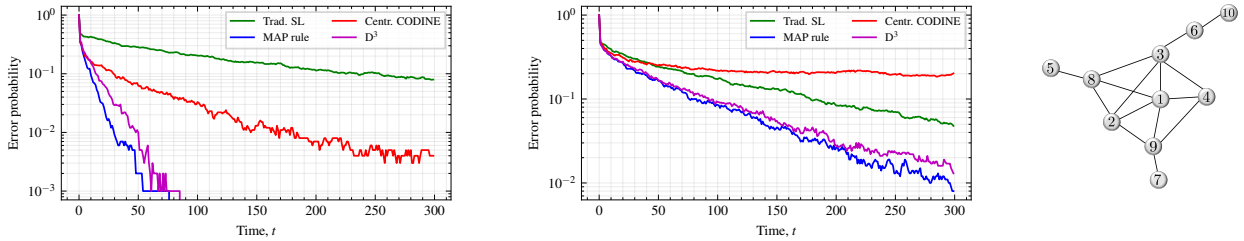

Fig. 1. Comparison between D^3 and existing strategies in terms of error probability. *Left*. First example in Sec. 4, where the spatial dependence is useful to discriminate. *Center*. Second example, where the spatial dependence is independent of the hypothesis. *Right*. Network topology.

We will now exploit (16) and (17) to design the D^3 strategy for training summarized in Algorithm 2.

Algorithm 2 D^3 training

Require: Each agent k starts with $\boldsymbol{W}^{(l)}(0)$ for $l > 1$ and $\{\boldsymbol{w}_{kn}^{(1)}(0)\}_{n=1}^{N_1}$
1: **for** $i = 1, 2, \ldots$ and $k = 1, \ldots, K$ **do**
2: apply FTC with initial state $\boldsymbol{w}_{kn}^{(1)}(i-1)\,\boldsymbol{x}_k(i)$ to get $\boldsymbol{g}_n^{(1)}(i)$ in (13)
3: update the weights by a routine satisfying Assumption 1
4: **end for**

Consider some initialization weight matrices $\{\boldsymbol{W}^{(l)}(0)\}_{l=1}^{L}$. Assume that all agents agree about the matrices from the second layer onward, while, for $l = 1$, each agent k only knows the k-th row of $\boldsymbol{W}^{(1)}(0)$. In step 2 of Algorithm 2, for each MLP first-layer node, $n = 1, \ldots, N_1$, the agents implement FTC, where each agent k uses as initial state the scaled observation $\boldsymbol{w}_{kn}^{(1)}(0)\,\boldsymbol{x}_k(1)$. After FTC, each agent has computed the values $\boldsymbol{g}_n(1)$ in (13). Accordingly, each agent k can implement (16) and (17). Note that to evaluate (16) agent k needs $\boldsymbol{x}_k(1)$, which is actually known to agent k. The manner in which (16) and (17) are computed depends on the particular implementation of the neural network. For example, for the SGD case in (14)–(15), the backpropagation algorithm is the standard method to compute the necessary gradients [17]. Iterating the above reasoning for all i, we see that agent k recovers: i) the k-th row of the weight matrix $\boldsymbol{W}^{(1)}(i)$ computed by the centralized MLP; and ii) the entire matrices $\boldsymbol{W}^{(l)}(i)$ computed by the centralized MLP for $l > 1$. We have in fact established the following result.

Proposition 2 (D^3 in the training phase). *Consider the centralized MLP described in Sec. 2.1 and a network of K agents equipped with a primitive and doubly stochastic combination matrix. Then, by running the D^3 strategy in Algorithm 2 with an optimization routine satisfying Assumption 1, each agent k is able to compute, in a fully decentralized manner: for $l > 1$, the same weight matrices $\boldsymbol{W}^{(l)}(i)$ as the centralized MLP; and for $l = 1$, the k-th row of the weight matrices $\boldsymbol{W}^{(1)}(i)$ computed by the centralized MLP.*

4. ILLUSTRATIVE EXAMPLES

We consider the network graph displayed in the rightmost plot of Fig. 1, on top of which we design a combination matrix A by using the Metropolis policy [1]. The classification task to be solved by the agents is a binary hypothesis test with the following statistical characterization. The marginal distributions under $\theta = 1$ and $\theta = 2$ are Gaussian with unit variance and means equal to $m_1 = 0$ and $m_2 = 0.1$, respectively. Since the means are different, the decision problem is individually identifiable, i.e., the marginal distributions allow each agent to discriminate. The joint distribution is constructed by applying the Joe Copula [25], which is determined by a single coupling parameter, say κ. In the simulations, we consider two settings. In the first scenario, the parameter κ vary across the hypotheses (we set $\kappa_1 = 1.25$ and $\kappa_2 = 1.75$), which means that the spatial dependence is also useful to discriminate. In the second scenario, we set $\kappa_1 = \kappa_2 = 1.25$. In the training stage, we implemented Algorithm 2 using as the optimization routine ADAM [26], which can be verified to satisfy Assumption 1. In the prediction stage, the posterior (3) is evaluated with the decision statistics h_w computed by Algorithm 1, run with the parameter w estimated during training. We will compare the D^3 strategy against three schemes, namely, the centralized MAP with the exact posterior (which is computed assuming that the Joe copula and its parameters are known); the traditional SL algorithm [1] with geometric averaging (which only employs marginal models); the recent CODINE strategy [8], which assumes no knowledge regarding the copula function and aims at learning it from the data. The MLP used by D^3 has 3 layers, with 128 nodes in the first two layers. The leftmost plot of Fig. 1 refers to the scenario where spatial dependence helps discriminate. The error probability of traditional SL decreases with time, which is correct since the decision problem is marginally identifiable. However, since traditional SL ignores spatial dependence, we can do better. In fact, the D^3 strategy reduces the error probability by orders of magnitude, and also approaches the optimal centralized MAP. The CODINE scheme, which is also centralized and should ideally be able to learn the dependence structure, performs worse than D^3. This may come as no surprise, because CODINE attempts to learn a generative model, whereas D^3 reduces the learning complexity by seeking a discriminative decision model. Let us focus now on the middle plot of Fig. 1. D^3 continues to be close to the optimal MAP. In this case the gains are less pronounced, since the spatial dependence is not a discriminant factor (it is the same under both hypotheses). However, learning the spatial dependence allows to estimate better the posterior, which explains why D^3 stays close to MAP. Notably, in this second scenario the error made by the CODINE scheme in approximating the copula function has a more significant effect on the performance, also affecting the discriminative power that is embedded in the marginal distributions.

5. CONCLUSION

We proposed D^3, a decentralized classification strategy exploiting dependence across agents. The strategy does not require parametric models for the dependence and allows each agent to approximate the posterior probability relevant to the decision problem by implementing an MLP in a distributed way. The analysis revealed significant advantages over existing strategies. Extensions include: vector data, adaptive strategies such as diffusion, and generative approaches.

6. REFERENCES

[1] V. Matta, V. Bordignon, and A. H. Sayed, *Social Learning: Opinion Formation and Decision-Making Over Graphs*. Now Publishers, 2025.

[2] C. Chamley, *Rational Herds: Economic Models of Social Learning*. Cambridge University Press, 2004.

[3] 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, Sep. 2012.

[4] X. Zhao and A. H. Sayed, "Learning over social networks via diffusion adaptation," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, 2012, pp. 709–713.

[5] A. Nedić, A. Olshevsky, and C. A. Uribe, "Fast Convergence Rates for Distributed Non-Bayesian Learning," *IEEE Trans. Automat. Control*, vol. 62, no. 11, pp. 5538–5553, Nov. 2017.

[6] 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, Sep. 2018.

[7] F. Scala, M. Carpentiero, V. Matta, and A. H. Sayed, "Social learning with dependent agents," in *Proc. International Conference on Digital Signal Processing (DSP)*, Pylos, Greece, Jun. 2025, pp. 1–5.

[8] N. A. Letizia, N. Novello and A. M. Tonello, "Copula density neural estimation," *IEEE Trans. on Neural Netw. Learn. Syst.*, vol. 36, no. 10, pp. 19452–19459, Oct. 2025.

[9] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *Proc. American Control Conference*, Jul. 2007, pp. 711–716.

[10] Y. Yuan, G.-B. Stan, L. Shi, and J. Gonçalves, "Decentralised final value theorem for discrete-time LTI systems with application to minimal-time distributed consensus," in *Proc. IEEE Conference on Decision and Control (CDC)*, Shanghai, China, Dec. 2009, pp. 2664–2669.

[11] Y. Yuan, G. B. Stan, L. Shi, M. Barahona, and J. Goncalves, "Decentralised minimum-time consensus," *Automatica*, vol. 49, no. 5, pp. 1227–1235, May 2013.

[12] T. Charalambous, Y. Yuan, T. Yang, W. Pan, C. N. Hadjicostis, and M. Johansson, "Distributed finite-time average consensus in digraphs in the presence of time delays," *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 4, pp. 370–381, Dec. 2015.

[13] A. Sandryhaila, S. Kar, and J. M. F. Moura, "Finite-time distributed consensus through graph filters," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 1080–1084.

[14] V. Bordignon, S. Vlaski, V. Matta and A. H. Sayed, "Learning from heterogeneous data based on social interactions over graphs," *IEEE Trans. on Inf. Theory*, vol. 69, no. 5, pp. 3347–3371, 2023.

[15] M. Carpentiero, V. Bordignon, V. Matta and A. H. Sayed, "Social learning with adaptive models," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Seoul, South Korea, Apr. 2024, pp. 1–5.

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

[17] A. H. Sayed, *Inference and Learning from Data*. Cambridge University Press, 2022.

[18] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*. Springer, 2006.

[19] A. Pinkus, "Approximation theory of the MLP model in neural networks," *Acta Numerica*, vol. 8, pp. 143–195, Jan. 1999.

[20] D. A. Sprecher, "A representation theorem for continuous functions of several variables," *Proceedings of the American Mathematical Society*, vol. 16, no. 2, pp. 200–203, Apr. 1965.

[21] R. C. Buck, "Approximate complexity and functional representation," *Journal of Mathematical Analysis and Applications*, vol. 70, no. 1, pp. 280–298, Jul. 1979.

[22] M. H. Degroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, Mar. 1974.

[23] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, Sep. 2004.

[24] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.

[25] R. B. Nelsen, *An Introduction to Copulas*. Springer, 2007.

[26] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 2015.