21 May 2024

ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Distributed Learning from Interactions in Social Networks

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Distributed Learning from Interactions in Social Networks / Sasso, Francesco; Coluccia, Angelo; Notarstefano, Giuseppe. - ELETTRONICO. - (2018), pp. 8550320.2200-8550320.2205. (Intervento presentato al convegno 16th European Control Conference, ECC 2018 tenutosi a Cyprus nel 12-15 June 2018) [10.23919/ECC.2018.8550320].

This version is available at: https://hdl.handle.net/11585/678735 since: 2019-02-28

Published:

DOI: http://doi.org/10.23919/ECC.2018.8550320

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

(Article begins on next page)

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

This is the final peer-reviewed accepted manuscript of:

F. Sasso, A. Coluccia and G. Notarstefano, "Distributed Learning from Interactions in Social Networks," 2018 European Control Conference (ECC), Limassol, 2018, pp. 2200-2205,

The final published version is available online at: **http://dx.doi.org/10.23919/ECC.2018.8550320**

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Distributed Learning from Interactions in Social Networks

Francesco Sasso, Angelo Coluccia, *Senior Member, IEEE* and Giuseppe Notarstefano, *Member, IEEE*

***Abstract*— We consider a network scenario in which agents can evaluate each other according to a score graph that models some interactions. The goal is to design a distributed protocol, run by the agents, that allows them to learn their unknown state among a finite set of possible values. We propose a Bayesian framework in which scores and states are associated to probabilistic events with unknown parameters and hyperparameters, respectively. We show that each agent can learn its state by means of a local Bayesian classifier and a (centralized) Maximum-Likelihood (ML) estimator of parameter-hyperparameter that combines plain ML and Empirical Bayes approaches. By using tools from graphical models, which allow us to gain insight on conditional dependencies of scores and states, we provide a relaxed probabilistic model that ultimately leads to a parameter-hyperparameter estimator amenable to distributed computation. To highlight the appropriateness of the proposed relaxation, we demonstrate the distributed estimators on a social interaction set-up for user profiling.**

I. Introduction

A common feature of online social networks (OSNs) is the possibility of individuals to continuously interact among themselves, by sharing contents and expressing opinions or ratings on different topics [1], [2]. We address such a context by considering a network scenario in which nodes can mutually rate, i.e., can give/receive a score to/from other "neighboring" nodes, and aim at learning their own (or their neighbors') state. The state may indicate a social orientation, influencing level, or the belonging to a thematic community. Due to the large-scale nature of OSNs, centralized solutions exhibit limitations both in terms of computation burden and privacy preservation, hence distributed solutions are needed.

In recent years, a great interest has been devoted to distributed schemes in which nodes aim at estimating a common parameter, e.g., by means of Maximum Likelihood (ML) approaches, [3]–[5], or performing simultaneous estimation and classification, [5], [6]. In [7]–[9] a more general Bayesian framework is considered, in which nodes estimate local parameters, rather than reaching consensus on a common one. In particular, an Empirical Bayes approach is proposed in which the parameters of the prior distribution, called *hyperparameters*, are estimated through a distributed algorithm. The estimated hyperparameters are then combined with local measurements to obtain the Minimum Mean Square Error (MMSE) estimator of the local parameters.

In the recent literature on distributed social learning, agents aim at estimating a *common* unobservable state from noisy observations through non-Bayesian schemes in which each agent processes its own and its neighbors' beliefs [10]–[14], see also [15] for a tutorial. A different batch of references investigates interpersonal influences in groups of individuals and the emerging of asymptotic opinions, [16]–[18], see [2], [19] for a tutorial on opinion formation in social networks. The problem of self-rating in a social environment is discussed in [20], where agents can perform a predefined task, but with different abilities.

In the present paper, we set up a learning problem in a network context in which each node needs to classify its own local state based on observations coming from the interaction with other nodes. Interactions among nodes are expressed by evaluations that a node performs on other ones, modeled through a weighted digraph that we will be referred to as *score graph*. This general scenario captures a wide variety of contexts arising from social relationships, where nodes have only a partial knowledge of the world. Specifically, in Section II we devise a Bayesian probabilistic framework wherein, however, both the parameters of the observation model and the hyperparameters of the prior distribution are allowed to be unknown. In order to solve this interaction-based learning problem, we propose in Section III a learning approach combining a local Bayesian classifier with a joint parameter-hyperparameter Maximum Likelihood estimation approach. Since the ML estimator is computationally intractable even for moderately small networks, we resort to the conceptual tool of graphical models to identify a relaxation of the probabilistic model that leads to a distributed estimator. In Section IV we validate the performance of the proposed distributed estimator via Monte Carlo simulations.

II. Bayesian framework for interaction-based learning

In this section, we set up the interaction-based learning problem in which agents of a network interact with each others according to a score graph. To learn its own state each node can use observations associated to incoming or outcoming edges. We propose a Bayesian probabilistic model with unknown parameters, which need to be estimated to solve the learning problem.

A. Interaction network model

We consider a *network of agents* able to perform evaluations of other agents. The result of each evaluation is a score given by the evaluating agent on the evaluated one. Such an interaction is described by a *score graph*. Formally, we let $\{1, \dots, N\}$ be the set of agent identifiers and $G_S = (\{1, \dots, N\}, E_S)$ a digraph such that $(i, j) \in E_S$ if agent i

Francesco Sasso, Angelo Coluccia and Giuseppe Notarstefano are with the Department of Engineering, Università del Salento, via Monteroni, 73100, Lecce, Italy, {name.lastname}@unisalento.it.

This result is part of a project that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 638992 - OPT4SMART).

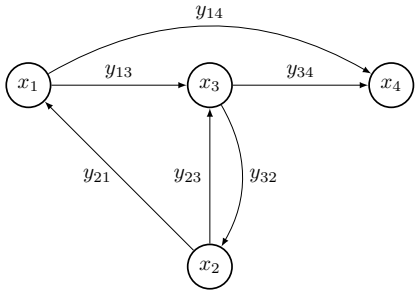

Fig. 1. Example of a score graph G_S.

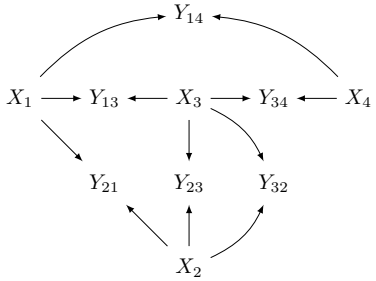

Fig. 2. The score Bayesian network related to the score graph in Fig. 1.

evaluates agent j. We denote by n the total number of edges in the graph, and assume that each node has at least one incoming edge in the score graph, that is, there is at least one agent evaluating it.

Let $\mathcal{C}$ and $\mathcal{R}$ be the set of possible state and score values, respectively. Being finite sets, we can assume $\mathcal{C} = \{c_1, \dots, c_C\}$ and $\mathcal{R} = \{r_1, \dots, r_R\}$, where C and R are the cardinality of the two sets, respectively. Consistently, in the network we consider the following quantities:

- $x_i \in \mathcal{C}$, unobservable *state* (or community) of agent i;
- $y_{ij} \in \mathcal{R}$, *score* (or evaluation result) of the evaluation performed by agent i on agent j.

An example of score graph with associated state and score values is shown in Fig. 1.

Besides the evaluation capability, the agents have also *communication* and *computation* functionalities. Agents communicate according to a time-dependent directed *communication graph* $t \mapsto G_{\text{cmm}}(t) = (\{1, \dots, N\}, E_{\text{cmm}}(t))$, where the edge set $E_{\text{cmm}}(t)$ describes the communication among agents: $(i, j) \in E_{\text{cmm}}(t)$ if agent i communicates to j at time $t \in \mathbb{Z}_{\geq 0}$. We introduce the notation $N^I_{\text{cmm},i}(t)$ and $N^O_{\text{cmm},i}(t)$ for the in- and out-neighborhoods of node i at time t in the communication graph. We will require these neighborhoods to include the node i itself; formally, we have

$$N^I_{\text{cmm},i}(t) = \{j : (j, i) \in E_{\text{cmm}}(t)\} \cup \{i\},$$
$$N^O_{\text{cmm},i}(t) = \{j : (i, j) \in E_{\text{cmm}}(t)\} \cup \{i\}$$

For the communication graph we assume the following:

Assumption 2.1: There exists an integer $Q \geq 1$ such that the graph $\bigcup_{\tau=tQ}^{(t+1)Q-1} G_{\text{cmm}}(\tau)$ is strongly connected $\forall\, t \geq 0$.

We point out that in general the (time-dependent) communication graph, modeling the distributed computation, is not necessarily related to the (fixed) score graph. We just assume that when the distributed algorithm starts each node i knows the scores received by in-neighbors in the score graph.

B. Bayesian probabilistic model

We consider the score $y_{ij}, (i, j) \in E_S$, as the (observed) realization of a random variable denoted by Y_{ij}; likewise, each state value $x_i, i \in \{1, \dots, N\}$, is the (unobserved) realization of a random variable X_i. To highlight the conditional dependencies among the random variables involved in the score graph, we resort to the tool of graphical models and in particular of Bayesian networks [21]. Specifically, we introduce the *Score Bayesian Network* with $N + n$ nodes X_i, $i = 1, \dots, N$, and Y_{ij}, $(i, j) \in E_S$ and $2n$ (conditional dependency) arrows defined as follows. For each $(i, j) \in E_S$, we have $X_i \to Y_{ij} \leftarrow X_j$ indicating that Y_{ij} conditionally depends on X_i and X_j. In Fig. 2 we represent the Score Bayesian Network related to the score graph in Fig. 1.

Denoting by $\boldsymbol{Y}_{E_S}$ the vector of all the random variables $Y_{ij}, (i, j) \in E_S$, the joint distribution factorizes as

$$\mathbb{P}(\boldsymbol{Y}_{E_S}, X_1, \dots, X_N) = \Big(\prod_{(i,j) \in E_S} \mathbb{P}(Y_{ij} | X_i, X_j) \Big) \Big(\prod_{i=1}^{N} \mathbb{P}(X_i) \Big).$$

We assume Y_{ij}, $(i, j) \in E_S$ are ruled by a conditional probability distribution $\mathbb{P}(Y_{ij} | X_i, X_j; \boldsymbol{\theta})$, depending on a *parameter* vector $\boldsymbol{\theta}$ whose components take values in a given set Θ. For notational purposes, we define the tensor

$$p_{h|\ell,m}(\boldsymbol{\theta}) := \mathbb{P}(Y_{ij} = r_h | X_i = c_\ell, X_j = c_m; \boldsymbol{\theta}), \quad (1)$$

where $r_h \in \mathcal{R}$ and $c_\ell, c_m \in \mathcal{C}$. From the definition of probability distribution, we have the constraint $\boldsymbol{\theta} \in \mathcal{S}_\Theta$ with

$$\mathcal{S}_\Theta := \Big\{ \boldsymbol{\theta} \in \Theta : p_{h|\ell,m}(\boldsymbol{\theta}) \in [0, 1], \sum_{h=1}^{R} p_{h|\ell,m}(\boldsymbol{\theta}) = 1 \Big\}.$$

We model X_i, $i = 1, \dots, N$, as identically distributed random variables ruled by a probability distribution $\mathbb{P}(X_i; \boldsymbol{\gamma})$, depending on a *hyperparameter* vector $\boldsymbol{\gamma}$ whose components take values in a given set Γ. Again, we introduce the notation

$$p_\ell(\boldsymbol{\gamma}) := \mathbb{P}(X_i = c_\ell; \boldsymbol{\gamma}). \quad (2)$$

and, analogously to $\boldsymbol{\theta}$, we have the constraint $\boldsymbol{\gamma} \in \mathcal{S}_\Gamma$ with

$$\mathcal{S}_\Gamma := \Big\{ \boldsymbol{\gamma} \in \Gamma : p_\ell(\boldsymbol{\gamma}) \in [0, 1], \sum_{\ell=1}^{C} p_\ell(\boldsymbol{\gamma}) = 1 \Big\}.$$

We assume that $p_{h|\ell,m}$ and p_ℓ are continuous functions, and that each node knows $p_{h|\ell,m}, p_\ell$ and the scores received from its in-neighbors and given to its out-neighbors in G_S.

An example is discussed in the next subsection, while the problem of jointly estimating the *parameter-hyperparameter* $(\boldsymbol{\theta}, \boldsymbol{\gamma})$ will be addressed in Section III; the latter will be then a building block of the (distributed) learning scheme.

C. Example: social ranking scenario

A relevant scenario is user profiling in OSNs. In social relationships, in fact, people naturally tend to aggregate into groups based on some affinity; this is found also in OSN contexts. For instance, consider a thread on a dedicated subject, wherein each member can express her/his preferences by assigning to other members/colleagues' posts a score from 1 to R indicating an increasing level of appreciation for that post. To model the distribution of scores, we propose the following variant of the so-called Mallow's ϕ-model [22]:

$$p_{h|\ell,m}(\theta) = \frac{1}{\psi_{\ell,m}(\theta)} e^{-\left(\frac{(r_R - r_h)/r_R - d(c_\ell, c_m)/c_C}{\theta}\right)^2}, \quad (3)$$

where $r_h = h$ $(h = 1, \ldots, R)$, $c_\ell = \ell$ $(\ell = 1, \ldots, C)$, $\theta \in \mathbb{R}_{>0}$ is a dispersion parameter, $\psi_{\ell,m}(\theta)$ is a normalizing constant, and d is a semi-distance, i.e., $d \geq 0$ and $d(c_\ell, c_m) = 0$ if and only if $c_\ell = c_m$. Informally, the "farther" a given community c_ℓ is from another community c_m, the higher will be the distance $d(c_\ell, c_m)$, and thus the lower the score.

In many cases the resulting subgroups reflect some hierarchy in the population: basic examples are forums or working teams. Thus, we consider a scenario in which each person belongs to a community reflecting some degree of expertise about a given topic or field. In particular, we have C ordered communities, with ℓth community given by $c_\ell = \ell$. That is, for example, a person in the community c_1 is a *newbie*, while a person in c_C is a *master*. Since climbing in the hierarchy can be regarded as the result of several "promotion" events, a possible probabilistic model for the communities is a binomial distribution $\mathcal{B}(C-1, \gamma)$, where $\gamma \in [0, 1]$ represents the probability of being promoted, i.e.,

$$p_\ell(\gamma) = \binom{C-1}{c_\ell - 1} \gamma^{c_\ell - 1} (1-\gamma)^{C-1-(c_\ell - 1)}.$$

We will refer to this set-up as *social-ranking model*.

III. Interaction-based distributed learning

In this section we describe the proposed distributed learning scheme. Without loss of generality, we focus on a set-up in which a node wants to self-classify. The same scheme also applies to a scenario in which a node wants to classify its neighbors, provided it knows their given and received scores.

The section is structured as follows. First, we derive a local Bayesian classifier provided that an estimation of parameter-hyperparameter $(\boldsymbol{\theta}, \boldsymbol{\gamma})$ is available. Then, based on a combination of plain ML and Empirical Bayes estimation approaches, we derive a joint parameter-hyperparameter estimator. Finally, we propose a suitable relaxation of the Score Bayesian Network which leads to a distributed estimator, based on proper distributed optimization algorithms.

A. Bayesian classifiers (given parameter-hyperparameter)

Each node can self-classify (i.e., learn its own state) if an estimate $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}})$ of parameter-hyperparameter $(\boldsymbol{\theta}, \boldsymbol{\gamma})$ is available. Before discussing in details how this estimate can be obtained in a distributed way, we develop a *decentralized* MAP self-classifier which uses only single-hop information, i.e., the scores it gives to and receives from neighbors.

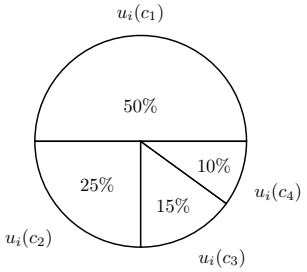

Fig. 3. Example of outcome of the soft classifier of an agent i, for $C = 4$: $\boldsymbol{u}_i = (0.5, 0.25, 0.15, 0.1)$.

Formally, let $\boldsymbol{y}_{N_i}$ be the vector of (observed) scores that agent i obtains by in-neighbors and provides to out-neighbors, i.e., the stack vector of y_{ji} with $(j, i) \in E_S$ and y_{ij} with $(i, j) \in E_S$. Consistently, let $\boldsymbol{Y}_{N_i}$ be the corresponding random vector. For each agent $i = 1, \ldots, N$, we define

$$u_i(c_\ell) := \mathbb{P}(X_i = c_\ell | \boldsymbol{Y}_{N_i} = \boldsymbol{y}_{N_i}; \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\theta}}), \quad \ell = 1, \ldots, C.$$

The *soft classifier* of i is the probability vector $\boldsymbol{u}_i := (u_i(c_1), \ldots, u_i(c_C))$ (whose components are nonnegative and sum to 1). In Fig. 3 we depict a pie-chart representation of an example vector $\boldsymbol{u}_i$.

From the soft classifier we can define the classical *Maximum A-Posteriori probability (MAP) classifier* as the argument corresponding to the maximum component of u_i, i.e.,

$$\hat{x}_i := \operatorname*{argmax}_{c_\ell \in \mathcal{C}} u_i(c_\ell).$$

The main result here is to show how to efficiently compute the MAP classifiers. First, we define

$$N_i^{\leftrightarrow} := \{j : (j, i) \in E_S, (i, j) \in E_S\},$$
$$N_i^{\leftarrow} := \{j : (j, i) \in E_S, (i, j) \notin E_S\},$$
$$N_i^{\rightarrow} := \{j : (i, j) \in E_S, (j, i) \notin E_S\},$$

and for each $h, k = 1, \ldots, R$ we introduce the quantities:

$$n_i^{\leftrightarrow}(h, k) := |\{j \in N_i^{\leftrightarrow} : y_{ij} = r_h, y_{ji} = r_k\}|,$$
$$n_i^{\leftarrow}(h) := |\{j \in N_i^{\leftarrow} : y_{ji} = r_h\}|,$$
$$n_i^{\rightarrow}(h) := |\{j \in N_i^{\rightarrow} : y_{ij} = r_h\}|.$$

Theorem 3.1: Let $i \in \{1, \ldots, N\}$ be an agent of the score graph. Then, the components of the vector u_i are given by

$$u_i(c_\ell) = \frac{v_i(c_\ell)}{Z_i}$$

where $Z_i = \sum_{\ell=1}^{C} v_i(c_\ell)$ is a normalizing constant, and $v_i(c_\ell) = p_\ell(\hat{\boldsymbol{\gamma}}) \pi_i^{\leftrightarrow}(c_\ell) \pi_i^{\leftarrow}(c_\ell) \pi_i^{\rightarrow}(c_\ell)$ with

$$\pi_i^{\leftrightarrow}(c_\ell) = \prod_{h,k=1}^{C} \Big(\sum_{m=1}^{C} p_{k|m,\ell}(\hat{\boldsymbol{\theta}}) p_{h|\ell,m}(\hat{\boldsymbol{\theta}}) p_m(\hat{\boldsymbol{\gamma}}) \Big)^{n_i^{\leftrightarrow}(h,k)},$$

$$\pi_i^{\leftarrow}(c_\ell) = \prod_{h=1}^{R} \Big(\sum_{m=1}^{C} p_{h|m,\ell}(\hat{\boldsymbol{\theta}}) p_m(\hat{\boldsymbol{\gamma}}) \Big)^{n_i^{\leftarrow}(h)},$$

$$\pi_i^{\rightarrow}(c_\ell) = \prod_{h=1}^{R} \Big(\sum_{m=1}^{C} p_{h|\ell,m}(\hat{\boldsymbol{\theta}}) p_m(\hat{\boldsymbol{\gamma}}) \Big)^{n_i^{\rightarrow}(h)}.$$

□

The proof is given in [23].

B. Joint Parameter-Hyperparameter ML estimation (JPH-ML)

Classification requires that at each node an estimate $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}})$ of parameter-hyperparameter $(\boldsymbol{\theta}, \boldsymbol{\gamma})$ is available.

On this regard, a few remarks about $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ are now in order. Depending on both the application and the network context, these parameters may be known, or (partially) unknown to the nodes. If both of them are known, we are in a pure Bayesian set-up in which, as just shown, each node can independently self-classify with no need of cooperation. The case of unknown $\boldsymbol{\theta}$ (and known $\boldsymbol{\gamma}$) falls into a Maximum-Likelihood framework, while the case of unknown $\boldsymbol{\gamma}$ (and known $\boldsymbol{\theta}$) can be addressed by an *Empirical Bayes* approach. In this paper we consider a general scenario in which both of them can be unknown. Our goal is then to compute, in a distributed way, an estimate of *parameter-hyperparameter* $(\boldsymbol{\theta}, \boldsymbol{\gamma})$ and use it for the classification at each node. In the following we show how to compute it in a distributed way by following a mixed Empirical Bayes and Maximum Likelihood approach. The *Joint Parameter-Hyperparameter Maximum Likelihood (JPH-ML) estimator* can be defined as

$$(\hat{\boldsymbol{\theta}}_{\text{ML}}, \hat{\boldsymbol{\gamma}}_{\text{ML}}) := \underset{(\boldsymbol{\theta},\boldsymbol{\gamma}) \in \mathcal{S}_\Theta \times \mathcal{S}_\Gamma}{\operatorname{argmax}} L(\boldsymbol{y}_{E_S}; \boldsymbol{\theta}, \boldsymbol{\gamma}) \tag{4}$$

where $\boldsymbol{y}_{E_S}$ is the vector of all scores $y_{ji}, (j,i) \in E_S$, and

$$L(\boldsymbol{y}_{E_S}; \boldsymbol{\theta}, \boldsymbol{\gamma}) = \mathbb{P}(\boldsymbol{Y}_{E_S} = \boldsymbol{y}_{E_S}\,;\, \boldsymbol{\theta}, \boldsymbol{\gamma}) \tag{5}$$

is the *likelihood function*.

Notice that $\boldsymbol{\theta}$ is directly linked to the observables $\boldsymbol{y}_{E_S}$; the hyperparameter $\boldsymbol{\gamma}$ is instead related to the unobservable states. While one could readily obtain the likelihood function for the sole estimation of $\boldsymbol{\theta}$ from the distribution of scores, the presence of $\boldsymbol{\gamma}$ requires to marginalize over all unobservable state (random) variables. By the law of total probability

$$L(\boldsymbol{y}_{E_S}; \boldsymbol{\theta}, \boldsymbol{\gamma}) = \sum_{\ell_1=1}^{C} \cdots \sum_{\ell_N=1}^{C} \mathbb{P}(\boldsymbol{Y}_{E_S} = \boldsymbol{y}_{E_S}, X_1 = c_{\ell_1}, \ldots, X_N = c_{\ell_N}). \tag{6}$$

Indicating with N_i^I the set of in-neighbors of agent i in the score graph (we are assuming that it is non-empty), the probability in (6) can be written as the product of the conditional probability of scores, i.e.,

$$\mathbb{P}(\boldsymbol{Y}_{E_S} = \boldsymbol{y}_{E_S} \,|\, X_1 = c_{\ell_1}, \ldots, X_N = c_{\ell_N}) = \prod_{i=1}^{N} \prod_{j \in N_i^I} \mathbb{P}(Y_{ji} = y_{ji} | X_j = c_{\ell_j}, X_i = c_{\ell_i})$$

multiplied by the prior probability of states, i.e.,

$$\mathbb{P}(X_1 = c_{\ell_1}, \ldots, X_N = c_{\ell_N}) = \prod_{i=1}^{N} \mathbb{P}(X_i = c_{\ell_i}).$$

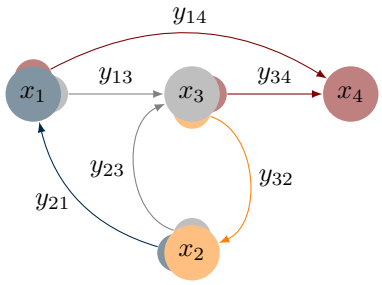

Fig. 4. Node-based relaxation of the score graph in Fig. 1, with virtual nodes indicating the virtual states of each node.

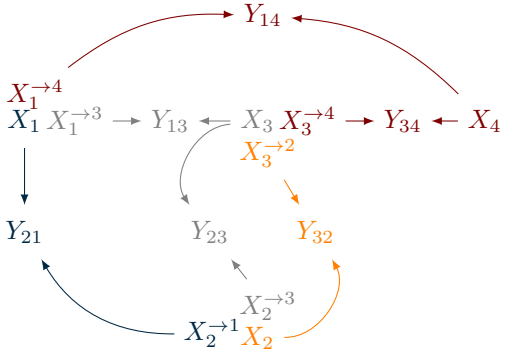

Fig. 5. Node-based relaxation of the score Bayesian network of Fig. 4.

Thus, the likelihood function turns out to be

$$L(\boldsymbol{y}_{E_S}; \boldsymbol{\theta}, \boldsymbol{\gamma}) = \sum_{\ell_1=1}^{C} \cdots \sum_{\ell_N=1}^{C} \prod_{i=1}^{N} p_{\ell_i}(\boldsymbol{\gamma}) \prod_{j \in N_i^I} p_{h_{ji}\,|\,\ell_i,\ell_j}(\boldsymbol{\theta})$$

where h_{ji} is the index of the score element $r_{h_{ji}} \in \mathcal{R} = \{r_1, \ldots, r_R\}$ associated to the score y_{ji}, i.e., $y_{ji} = r_{h_{ji}}$.

C. Distributed JPH Node-based Relaxed estimation (JPH-NR)

From the equations above it is apparent that the likelihood function couples the information at all nodes, so problem (4) is not amenable to distributed solution. To make it distributable, we propose a relaxation approach. To this aim we introduce, instead of $L(\boldsymbol{y}_{E_S}; \boldsymbol{\theta}, \boldsymbol{\gamma})$, a *Node-based Relaxed (NR) likelihood* $L_{NR}(\boldsymbol{y}_{E_S}; \boldsymbol{\theta}, \boldsymbol{\gamma})$. Let $\boldsymbol{y}_{N_i^I}$ be the vector of (observed) scores that agent i obtains by in-neighbors and $\boldsymbol{Y}_{N_i^I}$ the corresponding random vector. Then,

$$L_{NR}(\boldsymbol{y}_{E_S}; \boldsymbol{\theta}, \boldsymbol{\gamma}) := \prod_{i=1}^{N} \mathbb{P}(\boldsymbol{Y}_{N_i^I} = \boldsymbol{y}_{N_i^I}\,;\, \boldsymbol{\theta}, \boldsymbol{\gamma}). \tag{7}$$

This relaxation can be interpreted as follows. We imagine that each node has a virtual state, independent of its true state, every time it evaluates another node. Thus, in the Score Bayesian Network, besides the state variables X_i, $i = 1, \ldots, N$, there will be additional variables $X_i^{\rightarrow j}$ for each j with $(i,j) \in E_S$. To clarify this model, Figs. 4-5 depict the node-based relaxed graph and the corresponding graphical model for the same example given in Figs. 1-2.

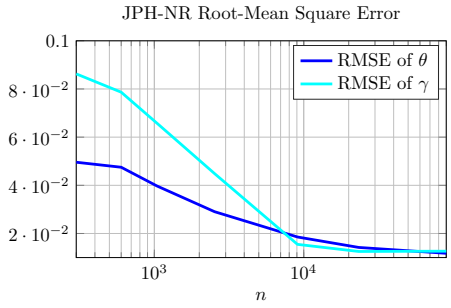

Fig. 6. JPH-NR RMSE of the estimates of (θ, γ) as a function of the number of edges n.

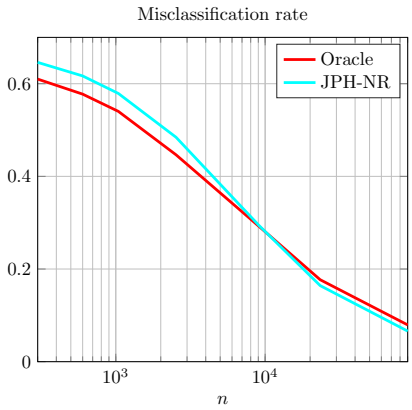

Fig. 7. Misclassification rate as function of the number of edges n increasing from N (cyclic graph) to $N^2 - N$ (complete graph), with $N = 300$, $\gamma = \frac{3}{10}$, $\theta = \frac{1}{5}$, $C = 6$ and $R = 3$.

Since $\boldsymbol{Y}_{N_i^I}$, $i = 1, \dots, N$, are not independent, then clearly $L \neq L_{NR}$. However, as it will appear from the numerical performance assessment reported in the Section IV, this choice yields reasonably small estimation errors.

Using this virtual independence between $\boldsymbol{Y}_{N_i^I}$, with $i = 1, \dots, N$, we define the *JPH-NR estimator* as

$$(\hat{\boldsymbol{\theta}}_{\text{NR}}, \hat{\boldsymbol{\gamma}}_{\text{NR}}) := \underset{(\boldsymbol{\theta},\boldsymbol{\gamma}) \in \mathcal{S}_\Theta \times \mathcal{S}_\Gamma}{\operatorname{argmax}} \; L_{NR}(\boldsymbol{y}_{E_S}; \boldsymbol{\theta}, \boldsymbol{\gamma}). \tag{8}$$

The next result characterizes the structure of JPH-NR (8).

Proposition 3.2: The JPH-NR estimator based on the node-based relaxation of the score Bayesian network is given by

$$(\hat{\boldsymbol{\theta}}_{\text{NR}}, \hat{\boldsymbol{\gamma}}_{\text{NR}}) = \underset{(\boldsymbol{\theta},\boldsymbol{\gamma}) \in \mathcal{S}_\Theta \times \mathcal{S}_\Gamma}{\operatorname{argmax}} \sum_{i=1}^{N} g(\boldsymbol{\theta}, \boldsymbol{\gamma}; \boldsymbol{n}_i) \tag{9}$$

with $\boldsymbol{n}_i = [n_i^{(1)} \cdots n_i^{(R)}]^\top$, $n_i^{(h)} := |\{j \in N_i^I : y_{ji} = r_h\}|$, and

$$g(\boldsymbol{\theta}, \boldsymbol{\gamma}; \boldsymbol{n}_i) = \log \Big(\sum_{\ell=1}^{C} p_\ell(\boldsymbol{\gamma}) \prod_{h=1}^{R} \Big(\sum_{m=1}^{C} p_{h\,|\,m,\ell}(\boldsymbol{\theta}) p_m(\boldsymbol{\gamma}) \Big)^{n_i^{(h)}} \Big). \tag{10}$$

□

The proof is given in [23].

Proposition 3.2 ensures that the JPH-NR estimator can be computed by solving an optimization problem that has a separable cost (i.e., the sum of N local costs). Available distributed optimization algorithms for asynchronous networks can be adopted to this aim, e.g. [24], [25], [26].

IV. Distributed learning for social ranking

We report numerical results for the social ranking model described in Section II-C with $C = 6$, $R = 3$ and $N = 300$. We adopt in (3) the semi-distance $d(c_\ell, c_m) = |c_\ell - c_m| = |\ell - m|$. The true values of parameter-hyperparameter are $\theta = \frac{1}{5}$ and $\gamma = \frac{3}{10}$.

Monte Carlo simulations have been run to test the performance of the JPH-NR estimator, with 1000 trials for each point. Fig. 6 reports the RMSE for the estimation of (θ, γ) as a function of the number of edges. It is worth noting that the estimation errors decrease as the number of edges increases, since more data are available.

The impact of estimation errors on the learning performance is shown in Fig. 7: the curve clearly shows that the inferential relationship between scores and states is "weaker" hence more data are needed for a good learning. As a benchmark, the curve corresponding also to the "oracle" classifier that uses the true value of γ and θ is reported. Remarkably, the proposed estimator is very close to the performance of the benchmark.

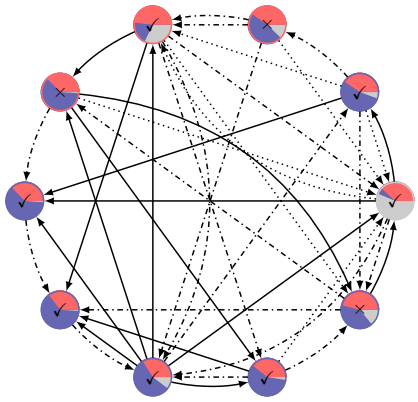

Fig. 8. Soft classifier representation of a particular score graph.

Finally, we report an additional case to highlight the usefulness of the soft classifier. We considered a network of $N = 10$ agents, divided in $C = 3$ communities, in which the maximum score is $R = 3$. The related score graph G_S is shown in Fig. 8. We drew the states and scores in the given score graph according to the previous distributions, and then used the social ranking model to solve the learning problem as before, by means of the JPH-NR estimator.

The contour of a node has a color which indicates the true state of the node. Inside the node we have represented the outcome of the soft classification, i.e., the output of the local self-classifier, as a pie-chart. The colors used are: red for state c_1, blue for state c_2, gray for state c_3. Moreover, each edge is depicted by a different pattern based on its evaluation result r_h: solid lines are related to scores equal to 3, dash dot lines are related to scores equal to 2, while dotted lines are related to scores equal to 1. We assigned to each node a symbol ✓ or × indicating if the MAP classifier

correctly decided for the true state or not.

Fig. 8 shows a realization with three misclassification errors; remarkably, all of them correspond to a lower confidence level given by the soft classifier, which is an important indicator of the lack of enough information to reasonably trust the decision. It can be observed that the edge patterns concur to determine the decision. Indeed, the only gray-state node is correctly classified thanks to the predominant number of dotted edges insisting on it, and similarly for the blue-state nodes which mostly have solid incoming edges. When a mix of scores are available, clearly there is more uncertainty and the learning may fail, as for two of the red-state nodes.

V. Conclusion

In this paper we have proposed a novel probabilistic framework for distributed learning, which is particularly relevant to emerging contexts such as cyber-physical systems and social networks. In the proposed set-up, nodes of a network want to learn their (unknown) state; differently from a classical set-up, the information does not come from (noisy) measurements of the state but rather from observations produced by the interaction with other nodes. For this problem we have proposed a hierarchical (Bayesian) framework in which the parameters of the interaction model as well as hyperparameters of the prior distributions may be unknown. Node classification is performed by means of a local Bayesian classifier that uses parameter-hyperparameter estimates, obtained by combining the plain ML with the Empirical Bayes estimation approaches in a joint scheme. The resulting estimator is very general but, unfortunately, not amenable to distributed computation. Therefore, by relying on the conceptual tool of graphical models, we have proposed an approximated ML estimator that exploits a proper relaxation of the conditional dependencies among the involved random variables. Remarkably, the approximated likelihood function leads to distributed estimation algorithms. To demonstrate the application of the proposed schemes, we have addressed an example scenario from user profiling in social networks, for which Monte Carlo simulations are reported. Results show that the proposed distributed learning scheme, although based on relaxation of the exact likelihood function, exhibits performance very close to the ideal classifier that has perfect knowledge of all parameters.

References

[1] V. Amelkin, F. Bullo, and A. K. Singh, "Polar opinion dynamics in social networks," *IEEE Transactions on Automatic Control*, 2017.

[2] A. V. Proskurnikov and R. Tempo, "A tutorial on modeling and analysis of dynamic social networks. part i," *Annual Reviews in Control*, vol. 43, 2017.

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

[4] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links Part I: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2008.

[5] A. Chiuso, F. Fagnani, L. Schenato, and S. Zampieri, "Gossip algorithms for simultaneous distributed estimation and classification in sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 691–706, 2011.

[6] F. Fagnani, S. M. Fosson, and C. Ravazzi, "A distributed classification/estimation algorithm for sensor networks," *SIAM Journal on Control and Optimization*, vol. 52, no. 1, pp. 189–218, 2014.

[7] A. Coluccia and G. Notarstefano, "Distributed estimation of binary event probabilities via hierarchical bayes and dual decomposition," in *52nd IEEE Conference on Decision and Control*, 2013.

[8] A. Coluccia and G. Notarstefano, "A hierarchical bayes approach for distributed binary classification in cyber-physical and social networks," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7406–7411, 2014.

[9] A. Coluccia and G. Notarstefano, "A bayesian framework for distributed estimation of arrival rates in asynchronous networks," *IEEE Transactions on Signal Processing*, vol. 64, no. 15, pp. 3984–3996, 2016.

[10] 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.

[11] S. Shahrampour and A. Jadbabaie, "Exponentially fast parameter estimation in networks using distributed dual averaging," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 6196–6201.

[12] A. Lalitha, A. Sarwate, and T. Javidi, "Social learning and distributed hypothesis testing," in *Information Theory (ISIT), 2014 IEEE International Symposium on*, 2014, pp. 551–555.

[13] P. Molavi, A. Tahbaz-Salehi, and A. Jadbabaie, "Foundations of non-bayesian social learning," *report*, 2016.

[14] A. Nedic, A. Olshevsky, and C. A. Uribe, "Fast convergence rates for distributed non-bayesian learning," *IEEE Transactions on Automatic Control*, 2017.

[15] A. Nedić, A. Olshevsky, and C. A. Uribe, "A tutorial on distributed (non-bayesian) learning: Problem, algorithms and results," in *55th IEEE Conference on Decision and Control*, 2016, pp. 6795–6801.

[16] A. Mirtabatabaei and F. Bullo, "Opinion dynamics in heterogeneous networks: convergence conjectures and theorems," *SIAM Journal on Control and Optimization*, vol. 50, no. 5, pp. 2763–2785, 2012.

[17] A. Mirtabatabaei, P. Jia, N. E. Friedkin, and F. Bullo, "On the reflected appraisals dynamics of influence networks with stubborn agents," in *2014 American Control Conference*, 2014, pp. 3978–3983.

[18] N. E. Friedkin, A. V. Proskurnikov, R. Tempo, and S. E. Parsegov, "Network science on belief system dynamics under logic constraints," *Science*, vol. 354, no. 6310, pp. 321–326, 2016.

[19] P. Frasca, H. Ishii, C. Ravazzi, and R. Tempo, "Distributed randomized algorithms for opinion formation, centrality computation and power systems estimation: A tutorial overview," *European Journal of Control*, 2015.

[20] W. Li, F. Bassi, L. Galluccio, and M. Kieffer, "Self-rating in a community of peers," in *55th IEEE Conference on Decision and Control*, 2016, pp. 5888–5893.

[21] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[22] C. L. Mallows, "Non-null rankings models," *Biometrika*, vol. 44, pp. 114–130, 1957.

[23] F. Sasso, A. Coluccia, and G. Notarstefano, "Interaction-based distributed learning in cyber-physical and social networks," *arXiv preprint arXiv:1706.04081*, 2017.

[24] R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, "Analysis of newton-raphson consensus for multi-agent convex optimization under asynchronous and lossy communications," in *54th IEEE Conference on Decision and Control*, 2015, pp. 418–424.

[25] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[26] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.