# DISTRIBUTED BAYESIAN LEARNING WITH A BERNOULLI MODEL

*Zhe Shen and Petar M. Djurić*

Department of Electrical and Computer Engineering
Stony Brook University, Stony Brook, NY 11794
Email: {zhe.shen, petar.djuric}@stonybrook.edu

## ABSTRACT

In this paper, we study multi-agent systems where the agents learn not only from their own private observations, but also from the ones of other agents. We build on a recent work, where a Bayesian learning method proposed for a linear Gaussian model was studied. According to the method, the agents iteratively exchange information with their neighbors, and they update the summary of their information using the signals received from the neighbors. The agents aim at obtaining the global posterior distribution of the unknown parameters in as short time as possible in a distributed way. In this paper, the posteriors are modeled by Beta distributions. We address two settings, one where the private signals are observed without errors and another where they are contaminated with errors. Finally, we provide and discuss an example and show results from computer simulations.

***Index Terms***— Bernoulli model, Bayesian learning, distributed processing

## 1. INTRODUCTION

We study a network of Bayesian cooperative agents that observe Bernoulli signals with unknown parameters. Their objective is to obtain the global posterior distribution of these parameters without the presence of a fusion center in as short time as possible. The network of the agents is connected and it can be represented by an undirected graph. Every agent has its own initial private signals which are independent from the signals of the remaining agents. From these signals the agents obtain information, which they share with their neighbors. Thereby the agents learn not only from their private signals, but also from the signals of their neighbors. In each iteration, each agent broadcasts its signal once, and receives its neighbors' signals. When they receive the signals from the neighbors, they store them if they contain new information and generate new signals for their neighbors. After a finite number of iterations, each agent is guaranteed to have complete information needed to obtain the global posterior of the unknown parameter. In other words, all the agents achieve a consensus on the posterior of a fictitious fusion center.

In the past decade or so, within the signal processing community, the problem of distributed learning has been of very high interest due to its applications. A more complete survey on applications is provided in [1]. Previous work on Bayesian learning in networks can be found in [2, 3, 4, 5, 6, 7]. In [7], a similar Bayesian learning method is studied, but in that model, each agent only uses the received signals in the current iteration. Other types of learning approaches are based on the consensus [1, 8, 9, 10] and the gossip methods [11, 12, 13, 14].

In [15], an efficient Bayesian learning method with Gaussian distributions was addressed. In this paper, we exploit the underlying idea from [15] to solve problems in more general settings. We use the results of our derivation on a Bernoulli model in two scenarios. In both cases, we show by simulations how the agents can reach a consensus in a finite number of iterations and much faster than consensus-based methods. In practice, this translates to less communication and energy consumption.

Consensus-based and gossip methods usually do not need to know the structure of the network. However, if the agents know the topology, for example, as in static man-made networks which are designed for a long term or permanent use, one can apply methods for much quicker learning. In this paper, as in [15], we assume that each agent has knowledge of the network topology. If in case the agents do not know it, they can still implement our learning method and achieve the same result as if they knew the topology. The learning, however, will require additional communication. But once it is completed, it can readily be used in subsequent estimation tasks.

The paper is organized as follows. In Section 2, we formulate the problems that we address. In Section 3, we present the solution to the first problem and in Section 4, to the second problem. Simulation results are shown in Section 5, and conclusions are provided in Section 6.

## 2. PROBLEM FORMULATION

We address two problems. In the first one, the agents observe Bernoulli outcomes without errors and in the second, with errors. In the latter case, each agent knows only its own probability of error. The agents form a connected network described by an undirected graph $G = (\mathcal{N}_A, E)$, where $\mathcal{N}_A = \{1, 2, \cdots, n\}$ is the set of agents and $E$ is the set of edges in the network. In other words, if agents $A_i$ and $A_j$ are connected, the edge $(i, j)$ represents the connection between these agents. The topology of the network is time invariant. Agents only communicate with their neighbors and the communications occur in stages. In each stage, all the agents broadcast their new information to their neighbors.

### 2.1. Model without errors

Let observations $x$ follow a Bernoulli distribution with a parameter $\theta$, where $\theta \in [0, 1]$. More precisely, $p(x = 1) = \theta$ and $p(x = 0) = 1 - \theta$. Suppose there is a connected network of agents which observe $x_j \in \mathbb{R}^{k_j \times 1}$ without errors, where $x_j$ is a vector of $k_j$ zeros and ones. The subscript $j$ refers to the $j$th agent and $k_j$ to the length of the observed vector. We refer to these observations as private

This work was supported by NSF under Award CCF-1320626.

signals, and we denote them as $y_j \in \mathbb{R}^{k_j \times 1}$. Since the signal vector $x_j$ is observed without errors, we have $y_j = x_j$. The value of $\theta$ is not known.

The agents assume a conjugate prior for the probability $\theta$, and it is a Beta density with parameters $\alpha_{j,0}$ and $\beta_{j,0}$, which is represented as $Beta(\alpha_{j,0}, \beta_{j,0})$. They use their private signals, to update this prior to the posterior $p(\theta|y_j, \alpha_{j,0}, \beta_{j,0})$, $j = 1, 2, \cdots, n$. Thus, at this stage, every agent has its own posterior. Now the agents start repeatedly exchanging information with their neighbors. After every exchange, they modify their posteriors of $\theta$. Their objective is achieving the same posterior as that of a fictitious fusion center, i.e., the posterior $p(\theta|y_{1:n}, \alpha_{1:n,0}, \beta_{1:n,0})$. Furthermore, the objective is to reach the desired posterior in a finite number of exchanges.

### 2.2. Model with errors

In the second model everything is the same as in the first one except that now the agents observe $x$ with errors. More specifically, we model the observation $y$ by

$$p(y = 1|x = 0) \quad = \quad p(y = 0|x = 1) \; = \; \epsilon, \tag{1}$$

where the probability of error $\epsilon \in [0, 0.5)$. The model is depicted in Fig. 1. As pointed out earlier, each agent has in general a different probability of error, and the agent knows it. An agent, however, does not know the probabilities of errors of the other agents.

The objectives are the same as stated in the previous subsection. Again, the agents have their own priors about $\theta$, $Beta(\alpha_{j,0}, \beta_{j,0})$ and they aim at estimating $p(\theta|y_{1:n}, \alpha_{1:n,0}, \beta_{1:n,0}, \epsilon_{1:n})$.

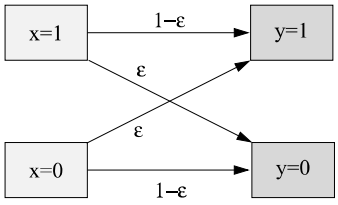


**Fig. 1**: The model with observation errors.

## 3. SOLUTION TO THE PROBLEM WITHOUT ERRORS

First we find the posterior distribution of $\theta$ of a fictitious fusion center. We write

$$p(\theta|y_{1:n}, \alpha_{1:n,0}, \beta_{1:n,0}) \propto \prod_{j=1}^{n} p(y_j|\theta) p(\theta|\alpha_{j,0}, \beta_{j,0}), \tag{2}$$

where

$$p(y_j|\theta) = \theta^{s_j} \, (1-\theta)^{k_j - s_j} \tag{3}$$

with $s_j$ being defined by $s_j = \sum_{i=1}^{k_j} y_{j,i}$ and

$$p(\theta|\alpha_{j,0}, \beta_{j,0}) \propto \theta^{\alpha_{j,0}-1} \, (1-\theta)^{\beta_{j,0}-1}. \tag{4}$$

We can readily show that

$$p(\theta|y_{1:n}, \alpha_{1:n,0}, \beta_{1:n,0}) \propto \theta^{\alpha_{fc}-1} \, (1-\theta)^{\beta_{fc}-1}, \tag{5}$$

where

$$\begin{aligned} \alpha_{fc} &= \sum_{j=1}^{n} (s_j + \alpha_{j,0}) - n + 1 = \sum_{j=1}^{n} \alpha_j(0) - n + 1, \\ \beta_{fc} &= \sum_{j=1}^{n} (k_j - s_j + \beta_{j,0}) - n + 1 = \sum_{j=1}^{n} \beta_j(0) - n + 1. \end{aligned} \tag{6}$$

Here we define the private signals $\alpha_j(0) = s_j + \alpha_{j,0}$ and $\beta_j(0) = k_j - s_j + \beta_{j,0}$, where $j = 1, 2, \cdots, n$. In summary, the posterior of the fusion center is a Beta density with parameters $\alpha_{fc}$ and $\beta_{fc}$.

Now we return to the distributed processing that takes places in the network. First, the agents process their own data and obtain their own posteriors, which are also Beta densities with private signals $\alpha_j(0)$ and $\beta_j(0)$. It is clear that if the agents know the sums $\sum_{j=1}^{n} \alpha_j(0)$ and $\sum_{j=1}^{n} \beta_j(0)$, the agents will be able to construct the same posterior as the fictitious fusion center.

Once the agents form their initial posteriors, they begin communicating with their neighbors. Now, the goal of the agents (obtaining the two sums) is equivalent to finding the average values of all the $\alpha_j(0)$s and $\beta_j(0)$s, respectively. With the method described in the sequel, the agents exchange information in stages where at each stage $t$, the agent $A_j$ broadcasts to its neighbors $\bar{\alpha}_j(t)$ and $\bar{\beta}_j(t)$, i.e, its estimates of $\sum_{j=1}^{n} \alpha_j(0)/n$ and $\sum_{j=1}^{n} \beta_j(0)/n$, respectively. This information is possibly used by the neighbors to update their estimates which they broadcast at the next stage. It can be shown that with this strategy all the agents can reach the optimal posterior in at most $2nd$ stages, where $d$ is the diameter of the graph [15]. Here we assume that the agents know the topology of the network. Later, we will relax this assumption.

In the sequel, we describe the updating process of $\bar{\alpha}_j(t)$. The process of $\bar{\beta}_j(t)$ is analogous. Let the estimate $\bar{\alpha}_j(t)$ be a linear combination of all the $\alpha_i(0)$s, that is

$$\bar{\alpha}_j(t) = \sum_{i=1}^{n} \omega_{j,i}(t) \alpha_i(0), \tag{7}$$

where $t > 0$, $\omega_{j,i}(t) \geq 0$, $j \in \mathcal{N}_A$ and $\sum_{i=1}^{n} \omega_{j,i}(t) = 1$. How the agents choose the coefficients $\omega_{j,i}(t)$ is explained below.

In the rest of this section, we focus on $A_j$, and therefore we will omit in the notation of the subscript $j$. Also, for simplicity, in the rest of this section, we assume $k_j = k$, i.e., all the agents have the same number of observations. Each agent keeps a memory which contains selected received signals $\bar{\alpha}(t)$s and their coefficient vectors $\omega(t) = [\omega_1(t), \cdots, \omega_n(t)]$s. Since here $\omega(t)$ is the coefficient vector of $A_j$ at $t$, $\omega_i(t)$ is the coefficient that this agent assigns to the private signal of $A_i$ at $t$. The selected signals and the coefficients are mapped to $r_*$ and $h_*$, respectively, where $*$ denotes the next available index in the memory (because it is possible that an agent adds one or more signals in one iteration or no signals at all). Let the memory of the agent at time $t$ be given by $r(t) = [r_1, \cdots, r_{l(t)}]^\top \in \mathbb{R}^{l(t)\times 1}$ and $H(t) = [h_1, \cdots, h_{l(t)}]^\top \in \mathbb{R}^{l(t)\times n}$. The vector $r(t)$ is composed of the received and linearly independent signals, and its size $l(t)$ is the number of signals in the memory by time $t$. The matrix $H(t)$ is a coefficient matrix whose $i$th row is of the form $h_i = [h_{i,1}, h_{i,2}, \cdots, h_{i,n}]$. The agent's memory can succinctly be represented by

$$r(t) = H(t)\alpha, \tag{8}$$

where $\alpha = [\alpha_1(0), \cdots, \alpha_n(0)]^\top$ is the vector of all private signals.

Now, we introduce the way of selecting the signals. At every iteration, if the coefficient vector $\omega(t)$ of the new received signal is linearly independent from all the coefficient vectors of the signals in its memory, the agent adds it to the memory. In other words, each agent keeps a received signal from a neighbor only if it is not in the space spanned by the set of signals already present in its memory. Thus, the memory of the agent keeps growing with time.

For example, at time instant $t = 1$, $A_j$ has a signal $r(1) = [r_1] = [\alpha_j(0)]$ and the matrix $H(1)$ has only one vector $h_1 =$

$[0, 0, \cdots, 0, 1, 0, \cdots, 0]$, where the entry one appears at the $j$th location. So, when at time $t$ the agent receives a signal from a neighbor $A_m$, $\bar{\alpha}_m(t) = \sum_{i=1}^{n} \omega_{m,i}(t)\alpha_i(0)$, and knows that it contains new information, this agent adds this signal by setting $r_{l(t-1)+1} = \bar{\alpha}(t)$ and $h_{l(t-1)+1} = \omega(t)$. This is the first update of the memory at time $t$. If more signals from neighbors with new information are received, more updates are made. The agent can calculate this information, because it knows the topology of the network. We point out that each agent needs to calculate the $H_j(t)$s of all the agents in the network in each iteration.

We enforce that the newly generated estimate $\bar{\alpha}(t+1)$ is a linear combination of the signals in its memory. That is the reason why each agent only needs to keep the linearly independent signals. At time $t$, the new estimate of each agent can be represented as

$$\bar{\alpha}(t+1) = \phi^{\top}(t)r(t), \tag{9}$$

where $\phi(t) = (\phi_1(t), \cdots, \phi_{l(t)}(t))^{\top}$, and $\phi^{\top}(t)1_{l(t)} = 1$ so that it is an unbiased estimate. The notation $1_{l(t)} \in \mathbb{R}^{l(t)\times 1}$ signifies a vector whose entries are all equal to one.

The variance of $\bar{\alpha}(t+1)$ is then

$$\begin{aligned} &var[\bar{\alpha}(t+1)] = \mathbb{E}[(\phi^{\top}(t)r(t) - \phi^{\top}(t)\mathbb{E}[r(t)])^2] \\ &= \phi^{\top}(t)\mathbb{E}\left[(r(t) - \mathbb{E}[r(t)])(r(t) - \mathbb{E}[r(t)])^{\top}\right]\phi(t) \\ &= \phi^{\top}(t)C_{r(t)}\phi(t), \end{aligned} \tag{10}$$

where $\mathbb{E}[\cdot]$ represents the expectation operator, and $var[\cdot]$ is the variance of the random variable inside the brackets. $C_{r(t)}$ is the covariance matrix of $r(t)$ which is given by

$$\begin{aligned} C_{r(t)} &= \mathbb{E}[(r(t) - \mathbb{E}[r(t)])(r(t) - \mathbb{E}[r(t)])^{\top}] \\ &= H(t)\mathbb{E}\left[(\alpha - \mathbb{E}(\alpha))(\alpha - \mathbb{E}(\alpha))^{\top}\right]H^{\top}(t) \\ &= H(t)C_{\alpha}H^{\top}(t), \end{aligned} \tag{11}$$

where $C_{\alpha} = \mathbb{E}\left[(\alpha - \mathbb{E}(\alpha))(\alpha - \mathbb{E}(\alpha))^{\top}\right]$. This covariance matrix is equal to $var[\alpha(0)]I$, because all the private signals are independent and have the same expectation and variance.

Now, using (10) and (11), we can write

$$var[\bar{\alpha}(t+1)] = var[\alpha(0)]\phi^{\top}(t)H(t)H^{\top}(t)\phi(t). \tag{12}$$

We want to choose $\phi(t)$ so that we minimize the variance $var[\bar{\alpha}(t+1)]$. Using the method of Lagrange multipliers, we can readily show that $\phi(t)$ is obtained from

$$\phi^{\top}(t) = \frac{1_{l(t)}^{\top}\left(H(t)H^{\top}(t)\right)^{-1}}{1_{l(t)}^{\top}\left(H(t)H^{\top}(t)\right)^{-1}1_{l(t)}}. \tag{13}$$

Once $\phi^{\top}(t)$ is found from (13), the agent computes $\bar{\alpha}(t+1)$ as in (9) and broadcasts the obtained value to its neighbors.

The reason for the need that each agent knows the structure of the network is that the structure determines the vector $\omega(t+1)$ for constructing the signal $\bar{\alpha}(t+1)$. With the knowledge of the structure, the agents calculate the $\omega(t+1)$s of all the other agents. Namely, from

$$\begin{aligned} \bar{\alpha}(t+1) &= \phi^{\top}(t)r(t) \\ &= \phi^{\top}(t)H(t)\alpha, \end{aligned} \tag{14}$$

they can find that $\omega(t+1) = \phi^{\top}(t)H(t)$. Therefore, when an agent receives a signal from a neighbor, it knows if there is new information in it. If there is new information, it stores the signal in its memory; otherwise it throws the signal away. At each iteration, the agent computes the matrices $H(t)$ of all the agents, where $H(t)$ is only determined by the topology.

We point out that the above derivation is general and does not rely on the assumption of Beta distributions.

In the case when the agents do not know the topology of the network, they can still obtain the sum of all the private signals. Then, they do not compute the matrices $H(t)$, but instead when they broadcast the estimates $\bar{\alpha}(t)$, they also broadcast the associated vectors $\omega(t)$. When this method is run for the first time, it may not be very efficient. However, once convergence is achieved, the agents can keep all the $\phi(t)$s for future use and then the method is as fast as the one with known topology. For a designed network, the pre-calculated $\phi(t)$s of each agent can be stored.

We can also optimize the number of communications. For example, when the agents do not receive any new information, they do not broadcast anything. This entails that the dimension of $r$ in the memory is not changed and that the estimate of this agent is the same as in the last iteration. Because the dimension of $r$ is $n$ at most, and if we define one broadcast as one communication, the upper bound of needed communications of all the agents is $n^2$.

## 4. SOLUTION TO THE PROBLEM WITH ERRORS

The model from subsection 2.2 is more difficult for processing than the model from the previous section. For the posterior of the fusion center we can write

$$p(\theta|y_{1:n}, \alpha_{1:n,0}, \beta_{1:n,0}, \epsilon_{1:n}) = \prod_{j=1}^{n} p(y_j|\theta, \epsilon_j)p_j(\theta), \tag{15}$$

where $p(y_j|\theta, \epsilon_j)$ is the likelihood of the $j$th agent and $p_j(\theta)$ is its prior. For the likelihood we write

$$p(y_j|\theta, \epsilon_j) = \prod_{i=1}^{k_j} \epsilon_j^{1-y_{j,i}}(1-\epsilon_j)^{y_{j,i}}\theta + \epsilon_j^{y_{j,i}}(1-\epsilon_j)^{1-y_{j,i}}(1-\theta). \tag{16}$$

Then for the local posterior of each agent $j$ we have

$$\begin{aligned} &p(\theta|y_j, \alpha_{j,0}, \beta_{j,0}, \epsilon_j) \\ &\propto C_1\theta^{k_j+\alpha_{j,0}-1}(1-\theta)^{\beta_{j,0}-1} + ... \\ &\quad + C_{k_j+1}\theta^{\alpha_{j,0}-1}(1-\theta)^{k_j+\beta_{j,0}-1}, \end{aligned} \tag{17}$$

where $C_i, i \in [1, k_j + 1]$ are coefficients which are determined by the observations $y_j$ and the error $\epsilon_j$. Clearly, the local posterior is a mixture of Beta densities with $k_j + 1$ components. Therefore, (15) and (17) imply that the posterior can also be written as a mixture of Beta densities but with many more components.

We would like to use the same mechanism of learning as in the previous section. Because the posterior of the fusion center is complicated, we use a method to approximate the local posterior by a single Beta density by the moment matching method. Then, the approximated local posterior is a Beta density with parameters $\alpha_j(0)$ and $\beta_j(0)$. We note that these parameters do not have a direct relation with $k_j$. In order to obtain the two sums of $\alpha_j(0)$s and $\beta_j(0)$s, we assume that the expectations and variances of all $\alpha_j(0)$s and $\beta_j(0)$s are the same, respectively. Then in each iteration, in the exchanging of the two estimates $\bar{\alpha}_j(t)$ and $\bar{\beta}_j(t)$, we can use the proposed method as in the problem without errors. Because the coefficient matrices $H_j(t)$s in (8) are determined by the topology

of the network, they can share the same $H_j(t)$s. When all the signals converge, each agent can have the same posterior as that of a fictitious fusion center which is a Beta density with parameters $(\sum_{j=1}^{n} \alpha_j(0) - n + 1)$ and $(\sum_{j=1}^{n} \beta_j(0) - n + 1)$.

## 5. SIMULATIONS

Here we present simulation results that demonstrate the performance of the proposed method for the model with errors. Due to lack of space, we do not show results for the model without errors. We note that the convergence for the two models, given the same priors of $\theta$ and the same observations of the agents (but with $\epsilon = 0$ and $\epsilon \neq 0$) is exactly the same.

In the experiment, we had a network with 11 agents. The topology of the network was taken from [16] and is shown in Fig. 2. We set the unknown probability to $\theta = 0.76$. The errors and number of observations of all the agents were randomly chosen from the uniform distributions on $[0.01, 0.49]$ and $[1, 100]$, respectively. For comparison purposes, we also implemented the average consensus method from [1]. The value of the step-size of the consensus algorithm was set to 0.23. In Figs. 3 and 4, we

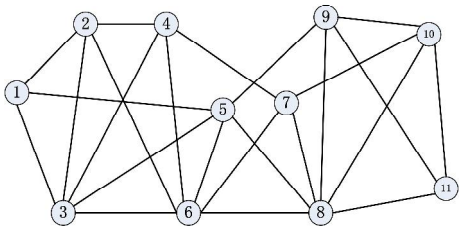


**Fig. 2**: The network of agents in the experiments.

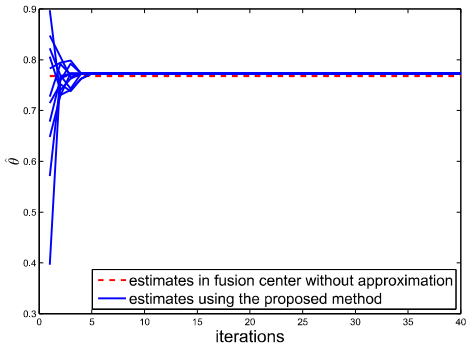


**Fig. 3**: The MAP estimates of all the agents as a function of iteration number obtained by the proposed method (solid lines). The MAP of the fusion center (the dashed line).

show one realization of the evolution of the agents' MAP estimates with iterations for the proposed and the consensus-based methods, respectively. We can see that the estimates of each agent of the proposed method converge to the estimate obtained from the exact average of the approximated sufficient statistics of all the agents. For this network, the proposed method always converges within four iterations, whereas the consensus-based method has a variable convergence time. Thus, with the proposed method, convergence is achieved much faster. We observe that this gain is paid by more

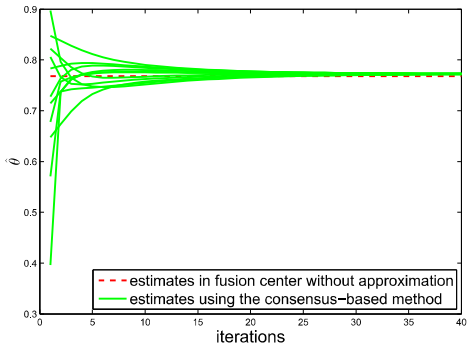


**Fig. 4**: The MAP estimates of all the agents as a function of iteration number obtained by the consensus-based method (solid lines). The MAP of the fusion center (the dashed line).

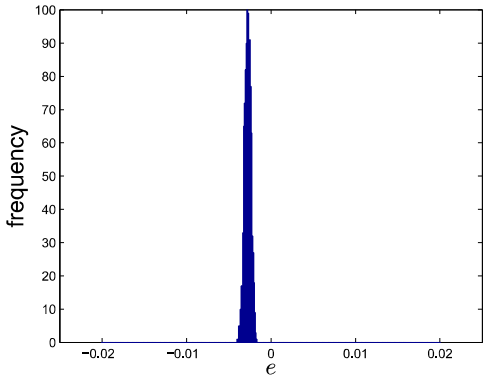


**Fig. 5**: A histogram of errors of 1000 agents from 1000 trials ($\theta = 0.76$).

intensive computations required by our method as opposed to the computations needed by the consensus-based method. In Fig. 5, we present a histogram of the error of the MAP estimates of the agents. We ran 1000 trials with different sets of observations and parameters. In each trial, we recorded the error of the agent's estimate defined by $e_i = \widehat{\theta}_i - \widetilde{\theta}_i$, where $\widehat{\theta}_i$ and $\widetilde{\theta}_i$ are the agents' and fusion center estimates in the $i$th trial, respectively. The results show a good performance of the method. However, they also suggest that the applied approximation introduces small bias in the obtained estimates.

## 6. CONCLUSION

In this paper, we presented a Bayesian learning method in a network of agents, where the agents aim at estimating the posterior distribution of a probability in a Bernoulli model. Each agent knows the structure of the network and stores the received signals from their neighbors only if they are linearly independent from the previously received signals. The agents use the newly received linearly independent signals to modify their signals and subsequently they broadcast them to their neighbors. We demonstrate the performance of the method with computer simulations and compare it with the performance of a consensus-based method.

## 7. REFERENCES


[1] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[2] D. Gale and S. Kariv, "Bayesian learning in social networks," *Games and Economic Behavior*, vol. 45, no. 2, pp. 329–346, 2003.

[3] D. Acemoglu, M. A. Dahleh, I. Lobel, and A. Ozdaglar, "Bayesian learning in social networks," *The Review of Economic Studies*, vol. 78, no. 4, pp. 1201–1236, 2011.

[4] P. M. Djurić and Y. Wang, "Distributed Bayesian learning in multiagent systems: Improving our understanding of its capabilities and limitations," *Signal Processing Magazine, IEEE*, vol. 29, no. 2, pp. 65–76, 2012.

[5] V. Krishnamurthy, "Quickest time herding and detection for optimal social learning," *arXiv preprint arXiv:1003.4972*, 2010.

[6] C. P. Chamley, *Rational Herds: Economic Models of Social Learning*, Cambridge University Press, 2003.

[7] E. Mossel and O. Tamuz, "Iterative maximum likelihood on networks," *Advances in Applied Mathematics*, vol. 45, no. 1, pp. 36–49, 2010.

[8] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proceedings of the 2003 American Controls Conference*, 2003.

[9] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[10] D. Li, Q. Liu, X. Wang, and . Lin, "Consensus seeking over directed networks with limited information communication," *Automatica*, vol. 49, no. 2, pp. 610–618, 2013.

[11] 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, 2010.

[12] Y. Wang and P. M. Djurić, "A gossip method for optimal consensus on a binary state from binary actions," 2013.

[13] H. Wang, X. Liao, and T. Huang, "Average consensus in sensor networks via broadcast multi-gossip algorithms," *Neurocomputing*, vol. 117, pp. 150–160, 2013.

[14] S. Wu and M. G Rabbat, "Broadcast gossip algorithms for consensus on strongly connected digraphs," *Signal Processing, IEEE Transactions on*, vol. 61, no. 16, pp. 3959–3971, 2013.

[15] E. Mossel and O. Tamuz, "Efficient Bayesian learning in social networks with Gaussian estimators," *arXiv preprint arXiv:1002.0747*, 2010.

[16] A. J. Bean and A. C. Singer, "Cooperative estimation in heterogeneous populations," in *Proceedings of the Conference on Signals, Systems and Computers (ASILOMAR)*, 2011, pp. 696–699.