DISTRIBUTED VARIATIONAL SPARSE BAYESIAN LEARNING FOR SENSOR NETWORKS

Thomas Buchgraber* and Dmitriy Shutin†

* Signal Processing and Speech Comm. Lab., Graz University of Technology, Austria
† German Aerospace Center, DLR Oberpfaffenhofen, Germany

ABSTRACT

In this work we present a distributed sparse Bayesian learning (dSBL) regression algorithm. It can be used for collaborative sparse estimation of spatial functions in wireless sensor networks (WSNs). The sensor measurements are modeled as a weighted superposition of basis functions. When kernels are used, the algorithm forms a distributed version of the relevance vector machine. The proposed method is based on a combination of variational inference and loopy belief propagation, where data is only communicated between neighboring nodes without the need for a fusion center. We show that for tree structured networks, under certain parameterization, dSBL coincides with centralized sparse Bayesian learning (cSBL). For general loopy networks, dSBL and cSBL are different, yet simulations show that dSBL compared to other sparse distributed regression methods, our method achieves similar sparsity and mean squared error performance. Furthermore, compared to other sparse distributed regression methods, our method does not require any cross-tuning of sparsity parameters.

Index Terms— Sparse Bayesian, distributed, collaborative learning, loopy belief propagation, sensor networks

1. INTRODUCTION

The research of sparse signal representation has been very intensive in recent years (see e.g. [1, 2, 3]). One of the approaches for finding sparse models, which lays down the foundation for this work, is based on sparse Bayesian learning (SBL) [4, 5, 6], exemplified by relevance vector machines (RVMs). Apart from that, advances in electronics and digital communications have made wireless sensor networks (WSNs) a very promising tool for efficiently solving large-scale decision and information-processing tasks [7]. Due to energy constraints and often limited communication capabilities, the operation of WSNs relies on distributed processing, when the aim of the whole network is achieved through synergy of individual sensors, able to sense, compute, and communicate data. In the context of distributed learning [8], which is an essential task for WSNs, we want to make inferences about the environment based on information gained from sensor observations. From this perspective, finding sparse representations in WSNs through collaborations is very important to enhance the energy and communication efficiency for such algorithms.

In this work, we propose a distributed sparse Bayesian learning (dSBL) algorithm for sparse regression in WSNs based on loopy belief propagation (LBP). As opposed to a previous approach [9], we now derive dSBL directly from centralized SBL (cSBL), where all data is available at one point, and show its equivalence for tree structured networks. Note that this equivalence does not hold for the previous model in [9]. Other related work on sparse distributed regression can be found, e.g. in [10, 11, 12]; these methods however require some cross-tuning of sparsity parameters, which is not necessary for dSBL.

The paper is organized as follows. In the Sections 2-4, we introduce cSBL and describe all relevant steps needed to obtain the proposed dSBL algorithm. In Section 5, we compare the simulation results of dSBL with its centralized counterpart and finally conclude in Section 6.

2. BAYESIAN MODEL DEFINITION

Consider a network with K sensors, where each sensor \( k = 1, \ldots, K \) observes a measurement \( t_k \), which we denote as the target signals. We assume that the measurements are noisy observations of some unknown field function \( f(x) \), spatially sampled at the sensor positions \( x_1, \ldots, x_K \), where \( x_k \in \mathbb{R}^d \) and \( d \) is the coordinate system dimension (e.g., \( d = 2 \) for planar deployment). Since we assume a static field function, the set \( D = \{x_k, t_k\}_{k=1}^{K} \) can be considered as distributed training data with inputs \( x_k \) and targets \( t_k \) available at sensor \( k \).

We now start to define the cSBL model, where \( D \) is assumed to be locally available at some point, and return to our distributed considerations in a later section.

A Bayesian network for cSBL [5, 6] is depicted in Figure 1, where hidden random variables are shown as shaded nodes, and known parameters and fixed basis functions as dots. The targets are modeled as

\[
t_k = \sum_{m=1}^{M} w_m \psi_m(x_k) + \epsilon_k,
\]

(1)
a superpositions of \( M \) weighted basis functions \( \psi_m(\cdot) \) with additive perturbation \( \epsilon_k \sim \mathcal{N}(0, \tau^{-1}) \), where we assume the precision

\[
\tau
\]

Note that although we use \( x_k \) as sensor coordinates here, it could also be any other kind of input vector available at sensor \( k \).
parameter τ to be known. Note that since we do not know the underlying true function \( f(x) \), and our model (1) may not be rich enough to perfectly represent any \( f \), the perturbation term \( \epsilon \) can be seen not only as including sensor measurement noise, but also any model mismatches. It is typically desired that (1) is as compact as possible. Thus we would like to find the sparest representation with the smallest possible number of basis functions. In a nutshell, we would like to find many weights to be zero. Inference on the cSBL model naturally leads to such sparse representations by assuming a hierarchical prior \( p(m|\alpha_m)p(\alpha_m) \) on the weights with \( p(m|\alpha_m) = \mathcal{N}(m|0, \alpha_m^{-1}) \) and \( p(\alpha_m) = \mathcal{Ga}(\alpha_m|a, b) \). Typically the parameters \( a, b \) are defined as \( a = b = 0 \), which leads to an improper non-informative Jeffreys prior and is the key to automatic relevance determination [5].

### 3. Variational Approximation

In SBL, we are interested in making predictions for a target \( t^* \) given an arbitrary input \( x^* \) and the training data \( D \). That is, we need to evaluate the predictive distribution

\[
p(t^*|t) = \int p(t^*|w)p(w|\mu, \Sigma) \, dw \, d\alpha,
\]

where in our notation we left out all deterministic parts (including \( x^* \)) and define \( w = [w_1, \ldots, w_M]^T \) and \( t = [t_1, \ldots, t_K]^T \).

Unfortunately, the posterior over all hidden variables inside the integral (2), defined as

\[
p(w, \alpha|t) = \frac{p(t|w)p(w|\alpha)p(\alpha)}{p(t)}
\]

cannot be computed in closed form due to the intractable model evidence \( p(t) \). Thus, also (2) cannot be computed in closed form and we need some effective approximation.

There are several approaches for cSBL approximate inference in the literature [5, 6, 13], where in our approach we follow the variational Bayesian (VB) approximation presented in [6].

In variational SBL, also known as the variational relevance vector machine when kernel basis functions are used, the intractable posterior (3) is approximated by

\[
q(w) = \mathcal{N}(w|\hat{\mu}, \hat{\Sigma}),
\]

where \( \hat{\mu} \) and \( \hat{\Sigma} \) are expressions of \( \mu \) and \( \Sigma \) that better represent the posterior, which cannot be obtained. By solving (5) and (6), we find \( q(w) = \mathcal{N}(w|\hat{\mu}, \hat{\Sigma}) \) and \( q(\alpha) = \prod_{m=1}^{M} \mathcal{Ga}(\alpha_m|\tilde{a}_m, \tilde{b}_m) \), cf. [6], with

\[
\begin{align*}
\hat{\Sigma} &= (\hat{\Phi}^T \Phi + \text{diag}(\alpha))^{-1}, \\
\hat{\mu} &= \hat{\Sigma} \hat{\Phi}^T \tau.
\end{align*}
\]

### 4. Distributed Sparse Bayesian Learning

Let us now extend the variational approximative cSBL model in such a way that we can derive the dSBL algorithm step by step. The fundamental problem is how to obtain a distributed versions of the variational updates (5) and (6).

First, consider the update expression (6). If we knew \( q(w) \) at each sensor in the network, i.e. if we knew its sufficient statistics \( \hat{\mu} \) and \( \hat{\Sigma} \), we could compute \( \alpha \), which is a sufficient statistic of \( q(\alpha) \) since \( \hat{\alpha}_a \) is constant and \( \hat{\alpha}_b = \hat{\alpha}_a / \hat{\alpha}_m \). That means, updating \( q(\alpha) \) is only a local computation at each sensor in the network if \( q(w) \) is known everywhere.

Secondly, we modify the update expression (5) for \( q(w) \) slightly to obtain an intermediate model with \( K \) hyperparameter vectors \( \alpha_1, \ldots, \alpha_K \) instead of one \( \alpha \) for each sensor. For this, we introduce a factor graph representation and show that we can obtain \( q(w) \) when \( \alpha_1 = \ldots = \alpha_K = \hat{\alpha} \) by using message passing inference schemes.

Thirdly, we tackle the remaining question. Namely, how to obtain \( q(w) \) at each sensor when our data \( D \) is distributed across the network. Since we are not interested in collecting all the data \( \{x_i, t_i\} \) for each sensor \( k \), we propose a further modification of the update expression (5) that better resembles the distributed nature of our problem. The resulting factor graph representation includes multiple distributions \( q(w_k) \) which can be obtained by using message passing as communication between neighboring sensors.

We show that under certain assumptions, \( q(w_k) = q(w_k), \forall k \).

### 4.1. Factor graph representation

Factor graphs [15, 14] are probabilistic undirected bipartite graphs consisting of nodes (circles), representing random variables and factors (black boxes), representing functions of the variables they are connected to. Bayesian networks can be easily transformed into factor graphs, although one might obtain different independence perceptsibility in the transformed representation [14].

We now shown that the weight update expression (5), defined by its sufficient statistics (7), can be obtained through inference performed on the factor graph in Figure 2(a), where we now have multiple values \( \hat{\alpha}_1, \ldots, \hat{\alpha}_K \) instead of one \( \hat{\alpha} \). To show this, we define
The factors as
\[ f_i(\tilde{t}, w) = \exp\left\{ -\frac{\tau}{2} \left( \tilde{t} - \Phi w \right)^2 \right\}, \quad (9) \]
and
\[ f_{k\alpha}(\alpha_k, w) = \exp\left\{ -\frac{1}{2K} w^T A_k w \right\}, \quad \forall k, \quad (10) \]
where \( A_k = \text{diag}(\alpha_k) \). Note that each sensor needs to know \( K \), the total number of sensors in the network, for definition (10). Although we assume that \( K \) is known everywhere, it can be easily obtained via consensus algorithms like [16, 17], as pointed out in [18]. This can be performed fully decentralized without the need for routing protocols. Let us now apply the sum-product algorithm, as depicted in Figure 2(b), that has marginal distributions \( q(w_k) \), \( \forall k \), which are equal to the centralized \( q(w) \) distribution. That is, we can obtain \( q(w_k) = q(w) \) at each sensor in the network. The marginals \( q(w_k) \) can be efficiently obtained by again using the sum-product algorithm, where we pass messages from the leaves to an arbitrary root node and from there back to the leaves [14]. Let us define the new factors used in Figure 2(b), as
\[ f_{k\alpha}(\tilde{t}, w_k) = \exp\left\{ -\frac{\tau}{2} (\tilde{t} - \Phi_k^T w_k)^2 \right\}, \quad \forall k, \quad (12) \]
and
\[ f_{kl}(w_k, w_l) = \exp\left\{ -\frac{\beta}{2} \| w_k - w_l \|^2 \right\}, \quad \forall \{k, l\} \in \mathcal{E}, \quad (13) \]
where (12), as opposed to (9), now has only a single observation \( \tilde{t}_k \) and we denote (13) as the coupling factors. The factors \( f_{k\alpha} \) are defined as in (10), but are now connected to a local \( w_k \) instead of a global \( w \). Compared to Figure 2(a), the factors (13) are conceptually new and form a fundamental part in the dSBL model. When ever sensors \( k \) and \( l \) are connected in the WSN graph, a factor \( f_{kl} \) is also defined in the factor graph, i.e., the physical structure \( G \) coincides with the probabilistic structure between the weights \( q(w_k) \). That is, the sum-product algorithm can be performed by the WSN, where no routing scheme with any kind of multi-hop communication is needed. By inspecting (13) with \( \beta > 0 \), we have larger values of \( f_{kl} \) when \( \| w_k - w_l \| \) is small, which is the driving force to achieve consensus in the network since there is a larger probability that neighbors have similar weight distributions. This coupling is even stronger when the precision parameter \( \beta \), further denoted as coupling parameter, is large. When \( \beta \to \infty \), then \( f_{kl}(w_k, w_l) \propto \delta(w_k - w_l) \) and we obtain the same weight distributions everywhere.

4.2. Distributed computation of \( q(w) \)

So far we have assumed that \( q(w) \) is known to all sensors. Before we investigate on how to distribute \( q(w) \) across the network, let us define a WSN as an undirected connected graph \( G = (\mathcal{V}, \mathcal{E}) \) which consists of vertices (or sensors) \( \mathcal{V} \) and edges (or links) \( \mathcal{E} \) defined as a set of unordered pairs \( \{k, l\} \subset \mathcal{V} \) representing the communication links. Note that by defining an undirected graph, we implicitly assume that sensors can communicate in both directions along a communication link. We define the neighborhood of a sensor \( k \) as \( N(k) = \{l \mid \{k, l\} \in \mathcal{E}\} \).

4.2.1. Tree Networks

When we consider tree networks, i.e., networks without loops, we can define a factor graph, as depicted in Figure 2(b), that has marginal distributions \( q(w_k) \), \( \forall k \), which are equal to the centralized \( q(w) \) distribution. That is, we can obtain \( q(w_k) = q(w) \) at each sensor in the network. The marginals \( q(w_k) \) can be efficiently obtained by again using the sum-product algorithm, where we pass messages from the leaves to an arbitrary root node and from there back to the leaves [14]. Let us define the new factors used in Figure 2(b), as
\[ f_{k\alpha}(\tilde{t}, w_k) = \exp\left\{ -\frac{\tau}{2} (\tilde{t} - \Phi_k^T w_k)^2 \right\}, \quad \forall k, \quad (12) \]
and
\[ f_{kl}(w_k, w_l) = \exp\left\{ -\frac{\beta}{2} \| w_k - w_l \|^2 \right\}, \quad \forall \{k, l\} \in \mathcal{E}, \quad (13) \]
where (12), as opposed to (9), now has only a single observation \( \tilde{t}_k \) and we denote (13) as the coupling factors. The factors \( f_{k\alpha} \) are defined as in (10), but are now connected to a local \( w_k \) instead of a global \( w \). Compared to Figure 2(a), the factors (13) are conceptually new and form a fundamental part in the dSBL model. Whenever sensors \( k \) and \( l \) are connected in the WSN graph, a factor \( f_{kl} \) is also defined in the factor graph, i.e., the physical structure \( G \) coincides with the probabilistic structure between the weights \( q(w_k) \). That is, the sum-product algorithm can be performed by the WSN, where no routing scheme with any kind of multi-hop communication is needed. By inspecting (13) with \( \beta > 0 \), we have larger values of \( f_{kl} \) when \( \| w_k - w_l \| \) is small, which is the driving force to achieve consensus in the network since there is a larger probability that neighbors have similar weight distributions. This coupling is even stronger when the precision parameter \( \beta \), further denoted as coupling parameter, is large. When \( \beta \to \infty \), then \( f_{kl}(w_k, w_l) \propto \delta(w_k - w_l) \) and we obtain the same weight distributions everywhere.

3Note that factors need not to be normalized.
In the following, we generally derive the messages and show that as $\beta \to \infty$, for tree structured graphs $G$, the equivalence $q(w_k) = q(w)$ holds for all $k$. Similar as before the messages from the observations to $w_k$ can be determined as $m_{f_k \rightarrow w_k} = \delta (f_k - f_k)$ and $m_{f_k \rightarrow w_k} = f_k(t_k, u_k)$. In the same manner we obtain $m_{\alpha_k \rightarrow f_k} = \delta (\alpha_k - \alpha_k)$ and $m_{\alpha_k \rightarrow w_k} = f_k(\alpha_k, w_k)$ for all $k$. Since all factors in Figure 2(b) have Gaussian shape, also the messages between the sensors have Gaussian shape. Additionally, as normalization can be done at any stage during message passing, we further assume normalized messages for simplicity. Thus, we can generally define the incoming message at node $w_k$, coming from some factor $f_{k,l}$, as normalized Gaussian

$$m_{f_{k,l} \rightarrow w_k} \propto \exp \left\{ -\frac{1}{2} (w_k - \mu_{uk})^T \Lambda_{uk} (w_k - \mu_{uk}) \right\},$$
(14)

with mean vector $\mu_{uk}$ and precision matrix (inverse covariance matrix) $\Lambda_{uk}$.

We can determine the messages send from sensor $k$ to sensor $l$ as\(^4\)

$$m_{w_k \rightarrow f_{k,l}} \propto m_{f_{k,l} \rightarrow w_k} m_{f_{k,l} \rightarrow w_k} \prod_{u \in N(k) \setminus l} m_{f_{k,u} \rightarrow w_k},$$
(15)

and

$$m_{f_{k,l} \rightarrow w_l} \propto \int f_k(w_l, u_l) m_{f_{k,l} \rightarrow w_l} dw_k,$$
(16)

where the sufficient statistics of (16) are obtained as

$$\Lambda_{kl} = \left( \frac{A_k}{K} + \tau \phi_k \phi_k^T + \sum_{u \in N(k) \setminus l} \Lambda_{uk} \right)^{-1} + \beta^{-1} I,$$
(17)

$$\mu_{kl} = (A_k^{-1} - \beta^{-1} I)(\tau \phi_k t_k + \sum_{u \in N(k) \setminus l} \Lambda_{uk} \mu_{uk}),$$
(18)

and $\hat{A}_k = \text{diag}(\hat{\alpha}_k)$. Passing messages (17) and (18) from the leave nodes to an arbitrary root node $w_k$ in the tree network, leads to the marginal distribution $q(w_k)$ with mean vector $\hat{\mu}_k$ and precision matrix $\hat{\Lambda}_k$ defined as

$$\Lambda_k = \hat{A}_k + \tau \phi_k \phi_k^T + \sum_{u \in N(k)} \Lambda_{uk},$$
(19)

and

$$\mu_k = \Lambda_k^{-1} (\tau \phi_k t_k + \sum_{u \in N(k)} \Lambda_{uk} \mu_{uk}).$$
(20)

Passing messages from the root node back to all other nodes $w_l$, for $l \neq k$, allows all other marginal distributions to be obtained. Now, inserting $\beta \to \infty$, for trees it is easy to verify that

$$\Lambda_k^{[\beta \to \infty]} = \frac{1}{K} \sum_{k=1}^K \Lambda_k + \tau \Phi \Phi^T,$$
(21)

and

$$\mu_k^{[\beta \to \infty]} = \tau (\Lambda_k^{[\beta \to \infty]} )^{-1} \Phi t_k,$$
(22)

which is equivalent to (7) if $\hat{A}_k = \hat{A}$, $\forall k$, as we intended to show. That is, $(\Lambda_k^{[\beta \to \infty]} )^{-1} = \Sigma$ and $\mu_k^{[\beta \to \infty]} = \mu$ for all $k$.

\(^4\)The notation $N(k) \setminus l$ denotes the set of all neighbors of $k$ except $l$.

### 4.2.2. General Loopy Networks

When we consider loopy graphs $G$, which is appropriate for almost any WSN, applying the sum-product algorithm leads to LBP, an iterative approximate inference method. Convergence of LBP cannot be guaranteed in general, but for Gaussian LBP, as used in our case, there exist convergence criteria [19, 20]. Unfortunately, if Gaussian LBP converges, only the mean can be guaranteed to be correct under the previously mentioned criteria. The latter is also the reason why consensus propagation [17], which has a Gaussian LBP interpretation and lays the foundation for our dSBL investigations, is able to converge to the true mean. The consensus parameter $\beta$, as described in [17], is critical to trade off fast convergence (small $\beta$) and accurate mean estimates (large $\beta$).

Furthermore, in loopy graphs, we need to define a schedule in which messages are updated until convergence. Algorithm 1 presents the resulting dSBL algorithm with synchronous message updates, where we have reformulated the messages as a function of intermediate marginals. Although convergence to the true marginal distributions cannot be guaranteed, our empirical investigations in Section 5 suggest performance close to cSBL at a much faster convergence rate over the variational inference iterations.

#### Algorithm 1 Distributed sparse Bayesian learning (dSBL)

Initialize: $\hat{\alpha}_k$ (e.g. $\hat{\alpha}_{k,m} = 10^{-1}$, $\forall m$), $\forall k$.

% Variational update loop
while (Not converged) do
  Initialize: $n = 0$, $\Lambda_k^{(0)} = 0$ and $\mu_{uk}^{(0)} = 0$, $\forall (u,k) \in E$

% Message passing update loop
while ($\text{LBP}$ not converged) do
  $n = n + 1$

% Compute intermediate marginals, $\forall k$
  $\Lambda_k^{(n)} = (K^{-1} \Lambda_k + \tau \phi_k \phi_k^T + \sum_{u \in N(k)} \Lambda_{uk})^{-1}$
  $\mu_{uk}^{(n)} = (\Lambda_{uk})^{-1} (\tau \phi_k t_k + \sum_{u \in N(k)} \Lambda_{uk} \mu_{uk})$

% Compute messages, $\forall (k,l) \in E$
  $\Lambda_k^{(n)} = (\Lambda_k^{(n)} - \Lambda_{kl}^{(n-1)})^{-1} + \beta^{-1} I$
  $\mu_{kl}^{(n)} = (\Lambda_{kl}^{(n)} - \Lambda_{kl}^{(n-1)})^{-1} (\Lambda_{kl}^{(n)} \mu_{kl}^{(n)} - \Lambda_{kl}^{(n-1)} \mu_{kl}^{(n-1)})$

end while

% Compute the marginals, $\forall k$
  $\Sigma_k = (K^{-1} \Lambda_k + \tau \phi_k \phi_k^T + \sum_{u \in N(k)} \Lambda_{uk})^{-1}$
  $\mu_k = \Sigma_k (\tau \phi_k t_k + \sum_{u \in N(k)} \Lambda_{uk} \mu_{uk})$

% Update hyperparameters, $\hat{\alpha}_{k,m}^{(n)}$, $\forall m$, $\forall k$
  $\hat{\alpha}_{k,m} = (2n + 1)/(2n + \mu_{km}^{(n)} + \Sigma_{k,m})$

end while

Compute the final marginals with LBP as before

### 4.3. Exploiting sparsity for efficient communication

When a hyperparameter $\hat{\alpha}_{k,m}$ diverges, or practically reaches a large threshold $\delta_k$, the basis function $m$ can be pruned at node $k$. Also the messages, i.e. $\mu_k$ and $\Lambda_k$ can be transmitted to all neighbors $l \in N(k)$ without the irrelevant components. For more details about the implementation of this efficient communication and how to merge distinct sets of active basis functions between sender and receiver, we refer the reader to [9] due to space limitations.
5. SIMULATIONS

For our simulations we define the basis functions like in RVMs, i.e., a bias \( \psi_0(x) = 1 \) and Gaussian kernels \( \psi_m(x) = \kappa(x, x_{m-1}) \) for \( m = 2, \ldots, K + 1 \) centered on the sensor positions\(^5\), where 

\[
\kappa(x, x_{m-1}) = \exp\{-\theta_k\|x - x_{m-1}\|^2\}
\]

with kernel parameter \( \theta_k = 15 \). We randomly deploy \( K = 50 \) sensors in a \( d = 2 \) dimensional area, where both coordinates lie in the interval \([0,1] \). Figure 3 shows the sensor positions plus connections, where we generated the connections by increasing the transmission radius equally for each sensor until a connected graph was obtained. The sensor measurements (targets) are generated by adding zero-mean white Gaussian noise with variance \( \sigma^2 = 10^{-3} \) to the underlying spatial function \( f(x) = 0.5 \sin((5x_1 - 2.5)) + 0.5 + x_2 \). We assume \( \tau = \sigma^{-2} \) where this is only an approximation, since we neglect any possible model mismatches. Nonetheless, this is a good choice since we do not know how well our model fits the data in advance. Comments on possible implementations for distributed estimation of \( \tau \) are given in Section 6.

The hyperprior parameters \( \alpha \) and \( \beta \) were both set to zero.

Table 1 shows a comparison of dSBL and cSBL, where we compare the mean squared error (MSE), the no. of basis functions (#Bfs) and the no. of variational inference iterations (#varIter) until the algorithms converge. We define the MSE for the variational approximate predictor (cf. [6]) of (2) for cSBL as 

\[
\text{MSE} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \| f(x_{\text{test},i}) - \hat{\phi}_{\text{cSBL},i}(\hat{\mu}) \|^2
\]

where \( N_{\text{test}} = 10^4 \) is the number of test points \( x_{\text{test},i} \). We placed on a uniform 100 \times 100 grid on the sensor deployment area. For dSBL, we have multiple MSEs, one for each sensor, where we now use \( \mu_k \) instead of \( \hat{\mu} \). In this sense, we give average and maximum numbers for MSE and #Bfs over the set of sensors. The algorithms are considered to have converged when the maximum absolute change of all elements of the intermediate marginal parameters \( \mu_k^{(n)} \) and \( \Lambda_k^{(n)} \) for all sensors \( k \) over the message iterations \( n \) to be smaller than \( 10^{-3} \). We denote the final number \( n \), after convergence, as the number of message iterations (#msgIter). The average and maximum numbers for #msgIter are obtained over the variational iterations.

The results in Table 1 are given for different pruning thresholds \( \theta_0 \) and coupling parameters \( \beta \). Note that cSBL only depends on \( \theta_0 \) and not on \( \beta \). Boldface numbers highlight better MSE or sparsity performance. It is very interesting to see that when \( \beta \) is large compared to \( \theta_0 \), dSBL performs similar to cSBL but requires much less variational inference iterations. We note that variational cSBL is known for its slow convergence as compared to, e.g., an alternative that uses implicit hyperparameter updates [5]. During all simulations, no convergence problems occurred. As we see further in Table 1, the max. and avg. values of MSE and #Bfs are mostly equivalent in dSBL, which means that the network has achieved consensus. When \( \theta_0 \) is large compared to some fixed \( \beta \), no pruning happens and all dSBL results are equivalent. This is because all elements of \( \alpha_k \) quickly converge to finite values lower than \( \theta_0 \). As we empirically observed in our simulations, the largest values of \( \alpha_k \) after convergence are at a level around \( \beta N(k) \), where in cSBL they diverge towards infinity. Figure 4 shows an example of the hyperparameter evolution of one particular sensor \( k = 22 \) compared to that of cSBL, where \( \theta_0 = 10^6 \) and \( \beta = 10^4 \). Note that sensor 22 is also highlighted in Figure 3, where we see that \( N(22) = 1 \). Thus, the largest hyperparameters (Fig. 4(a)) converge to a level around \( \beta \).

6. CONCLUSIONS AND OUTLOOK

We have presented a distributed sparse Bayesian learning (dSBL) algorithm for sensor networks. When the network graph is a tree and the coupling parameter \( \alpha \rightarrow \infty \), we have shown that dSBL is consistent with standard centralized sparse Bayesian learning (cSBL), where all data is available at one central location. Based on this consistency, we have extended the algorithm for arbitrary networks with loops, which leads to an iterative Gaussian loopy belief propagation (LBP) subroutine. In this case, \( \beta \) becomes a crucial parameter to trade-off LBP’s convergence speed vs. its accuracy as described in [17]. Performance comparison of dSBL and cSBL for an example network led to the surprising observation that for large \( \beta \), dSBL performs similar to cSBL at a much faster convergence rate over the variational inference iterations. This convergence speed discrepancy even increases for higher pruning thresholds.

In a future work, it would be interesting to investigate on how to find the smallest reasonable \( \theta_0 \) for a given \( \beta \). Furthermore, intensive simulations should show if the actual superior dSBL performance in terms of MSE and the number of basis functions as shown in Table 1 is significant. Throughout this paper we have assumed \( \tau \) to be known. However, in cSBL it is possible to estimate \( \tau \) during the variational inference updates [6]. It would be straight forward to implement this also in a distributed way using consensus algorithms like [16, 17], since the estimator for \( \tau^{-1} \) basically is a distributed average of squared errors, cf. [6].

7. REFERENCES


<table>
<thead>
<tr>
<th>$\theta_0$</th>
<th>$\beta$</th>
<th>MSE</th>
<th>dSBL</th>
<th>cSBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>$10^3$</td>
<td>-20.72dB</td>
<td>-20.60dB</td>
<td>12</td>
</tr>
<tr>
<td>$10^4$</td>
<td>$10^6$</td>
<td>-22.30dB</td>
<td>-22.30dB</td>
<td>14</td>
</tr>
<tr>
<td>$10^4$</td>
<td>$10^8$</td>
<td>-22.32dB</td>
<td>-22.32dB</td>
<td>14</td>
</tr>
<tr>
<td>$10^6$</td>
<td>$10^3$</td>
<td>-20.57dB</td>
<td>-20.44dB</td>
<td>51</td>
</tr>
<tr>
<td>$10^6$</td>
<td>$10^6$</td>
<td>-22.30dB</td>
<td>-22.30dB</td>
<td>14</td>
</tr>
<tr>
<td>$10^6$</td>
<td>$10^8$</td>
<td>-22.32dB</td>
<td>-22.32dB</td>
<td>14</td>
</tr>
<tr>
<td>$10^8$</td>
<td>$10^3$</td>
<td>-20.57dB</td>
<td>-20.44dB</td>
<td>51</td>
</tr>
<tr>
<td>$10^8$</td>
<td>$10^6$</td>
<td>-22.30dB</td>
<td>-22.30dB</td>
<td>51</td>
</tr>
<tr>
<td>$10^8$</td>
<td>$10^8$</td>
<td>-22.32dB</td>
<td>-22.32dB</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 1. Performance comparison of dSBL and cSBL for the network shown in Figure 3. See the text for details.

Fig. 4. Hyperparameter evolution over the variational iterations.