Energy Efficient Sparse Bayesian Learning using Learned Approximate Message Passing

Christo Kurisummoottil Thomas[‡], Rakesh Mundlamuri[*], Chandra R Murthy[†] and Marios Kountouris[*]
[*]Communication Systems Department
EURECOM, Sophia-Antipolis, France
Email: {rakesh.mundlamuri,marios.kountouris}@eurecom.fr
[†] Indian Institute of Science, Bangalore, India
Email: cmurthy@iisc.ac.in

***Abstract*—Sparse Bayesian learning (SBL) is a well-studied framework for sparse signal recovery, with numerous applications in wireless communications, including wideband (millimeter wave) channel estimation and user activity detection. SBL is known to be more sparsity-inducing than other priors (e.g., Laplacian prior), and is better able to handle ill-conditioned measurement matrices, hence providing superior sparse recovery performance. However, the complexity of SBL does not scale well with the dimensionality of the problem due to the computational complexity associated with the matrix inversion step in the EM iterations. A computationally efficient version of SBL can be obtained by exploiting approximate message passing (AMP) for the inversion, coined AMP-SBL. However, this algorithm still requires a large number of iterations and careful hand-tuning to guarantee convergence for arbitrary measurement matrices. In this work, we revisit AMP-SBL from an energy-efficiency perspective. We propose a fast version of AMP-SBL leveraging deep neural networks (DNN). The main idea is to use deep learning to unfold the iterations in the AMP-SBL algorithm using very few, no more than 10, neural network layers. The sparse vector estimation is performed using DNN, and hyperparameters are learned using the EM algorithm, making it robust to different measurement matrix models. Our results show a reduction in energy consumption, primarily due to lower complexity and faster convergence rate. Moreover, the training of the neural network is simple since the number of parameters to be learned is relatively small.**

***Index terms*—** Sparse Bayesian learning, approximate message passing, deep unfolding, energy efficiency.

I. Introduction

Sparse signal reconstruction and compressed sensing have received significant attention over the last years and have been applied to various problems including massive multi-input multi-output (MIMO) channel estimation, biomagnetic imaging, image restoration, and echo cancellation. The compressed sensing problem can be formulated as

$$\boldsymbol{y} = \mathbf{A}\boldsymbol{x} + \boldsymbol{w} \tag{1}$$

where $\boldsymbol{y}$ is the observations or data, $\mathbf{A}$ is called the measurement or the sensing matrix, which is known and is of dimension $M \times N$ with $M < N$, $\boldsymbol{x}$ is the N-dimensional sparse signal, and $\boldsymbol{w}$ is the additive noise. Signal $\boldsymbol{x}$ contains only K non-zero entries, with $K \ll N$. $\boldsymbol{w}$ is assumed to be a white Gaussian noise, $\boldsymbol{w} \sim \mathcal{N}(0, \gamma^{-1}\mathbf{I})$, with γ representing the inverse variance (or precision) parameter. To address this problem, a variety of algorithms, such as orthogonal matching pursuit [1], basis pursuit method [2], and iterative re-weighted l_1 and l_2 algorithms [3] have been proposed in the literature. Bayesian algorithms, such as sparse Bayesian learning (SBL), impose a hierarchical prior on $\boldsymbol{x}$ and calculate the so-called hyperparameters of the prior using the maximum likelihood (ML) principle, and infer the posterior distribution of the sparse signal given some observations (data). The SBL algorithm was first introduced in [4], [5] for linear regression and sparse signal recovery applications, respectively.

Compared to other state-of-the-art techniques, the critical point about SBL is the hierarchical prior modeling, which results in a sparsifying prior on $\boldsymbol{x}$. In the empirical Bayesian approach, the hyperparameters (γ, the inverse noise variance, and $\boldsymbol{\alpha}$, the variance of the components of $\boldsymbol{x}$) are estimated first using an evidence maximization, referred to as the Type II maximum likelihood (ML) method [5]. With the estimate of $\boldsymbol{\alpha}, \gamma$ in hand, the posterior of $\boldsymbol{x}$ is formulated as $p(\boldsymbol{x}|\boldsymbol{y}, \widehat{\boldsymbol{\alpha}}, \widehat{\gamma})$ and the mean of this posterior distribution is used as a point estimate of $\boldsymbol{x}$. Recently, approximate message passing (AMP) [6], generalized AMP, and vector AMP [7], [8] have been introduced to compute the posterior distributions in a message passing (MP) framework with lower computational complexity. AMP uses the central limit theorem and Taylor series approximations to represent all the messages in a factor graph in belief propagation (BP) as Gaussian random variables.

However, a drawback of the algorithms mentioned above is that, in large dimensional settings, they require a large number of iterations to converge. In recent years, to circumvent the above issue, deep learning has been used to reduce the computational complexity of sparse signal recovery, for example, [9]–[11]. The basic principle in all these algorithms is to unfold the iterations using a deep neural network (DNN). The weights of the DNN are trained using a large set of training examples to minimize the reconstruction mean-squared error (MSE). Once the weights have been optimized, the DNN can be used to estimate the sparse vector $\boldsymbol{x}$ for a new observation $\boldsymbol{y}$.

[‡]Christo was with Communication Systems Department of EURECOM, during the course of this work. He is currently working at Qualcomm Inc., Espoo, Finland, email: ckurisum@qti.qualcomm.com

A. Contributions

- We propose a *learned* AMP-SBL with Expectation-Maximization (EM) for hyperparameter estimation (denoted as EM-LAMP-SBL) by combining EM-SBL, AMP, and deep learning. This enables our algorithm to converge in very few iterations compared to the original AMP-SBL [12], thanks to the introduction of learned weights in the AMP iterations. This results in fewer computations overall and hence achieves energy savings compared to iterative algorithms based on AMP, while achieving the same or better sparse signal recovery performance.
- We also observe in our simulations that as the measurement matrix deviates from the i.i.d. Gaussian model, the assumption under which AMP is derived, the EM-LAMP-SBL still converges to a possibly local optimum and offers good sparse recovery performance. However, the AMP-SBL algorithm diverges in such cases, thus illustrating the applicability of EM-LAMP-SBL to more general $\mathbf{A}$ matrices compared to previous approaches.

The key novelty in our work is in unfolding the AMP-SBL as a DNN (see Algorithm 1), identifying the part of the algorithm that can be made learnable (See Fig. 2), and developing a scheme for learning the weights (See Algorithm 2). We support our DNN-based architecture with numerical results which show that the EM-LAMP-SBL algorithm offers a low complexity and therefore energy-efficient approach to sparse signal recovery, while retaining or improving on the performance of their parent algorithms (See Fig. 3).

II. SBL System Model

We start with the signal model in (1). In sparse Bayesian learning, a two-layer hierarchical prior is assumed for the $\boldsymbol{x}$ as in [4]. The prior is chosen such that the resulting marginalized prior obtained for $\boldsymbol{x}$ encourages sparsity in the posterior estimate of $\boldsymbol{x}$. Specifically, $\boldsymbol{x}$ is assumed to be Gaussian distributed and parameterized by an unknown diagonal covariance with elements $\boldsymbol{\alpha} = [\alpha_1 \ \alpha_2 \ ... \ \alpha_N]$, where α_i represents the inverse variance or the precision parameter of x_i and is assumed to be Gamma distributed with parameters a and b. Let $\Gamma(x)$ denote the Gamma function. Thus,

$$p(\boldsymbol{x}|\boldsymbol{\alpha}) = \prod_{i=1}^{N} p(x_i|\alpha_i) = \prod_{i=1}^{N} \mathcal{N}(0, \alpha_i^{-1}), \tag{2}$$

and

$$p(\boldsymbol{\alpha}) = \prod_{i=1}^{N} p(\alpha_i|a,b) = \prod_{i=1}^{N} \frac{b^a \alpha_i^{a-1} e^{-b\alpha_i}}{\Gamma(a)}. \tag{3}$$

The inverse of noise variance γ is also assumed to be Gamma distributed, i.e., $p(\gamma) = (\Gamma(c))^{-1} d^c \alpha_i^{c-1} e^{-d\gamma}$, where c and d are known parameters. This prior distribution on x_i results in a Student-t distribution with parameters a and b, which promotes sparsity in the posterior estimates. Specifically, for small values a and b, the Student-t distribution has sharp peak at zero, which favors sparsity. Now, the likelihood of the observations $\boldsymbol{y}$ can be written as

$$p(\boldsymbol{y}|\boldsymbol{x}, \gamma) = (2\pi)^{-M/2} \gamma^{M/2} e^{\frac{-\gamma \|\boldsymbol{y}-\mathbf{A}\boldsymbol{x}\|^2}{2}}. \tag{4}$$

A. AMP-SBL Algorithm

Our point of departure is the algorithm derived in [12, Table I] for AMP-SBL. In this section, for any scalar variable a and any vector $\boldsymbol{a}$, $\widehat{a}$ and $\widehat{\boldsymbol{a}}$ represent their estimates, respectively.

Given the hyperparameter estimates, the posterior density of $\boldsymbol{x}$ is Gaussian [5]:

$$p(\boldsymbol{x}|\boldsymbol{y}, \widehat{\gamma}, \widehat{\boldsymbol{\alpha}}) = \mathcal{N}(\widehat{\boldsymbol{x}}, \boldsymbol{\Sigma}), \tag{5}$$

with $\widehat{\boldsymbol{x}} = \widehat{\gamma} \boldsymbol{\Sigma} \mathbf{A}^T \boldsymbol{y}$ and $\boldsymbol{\Sigma} = (\widehat{\gamma} \mathbf{A}^T \mathbf{A} + \text{Diag}(\widehat{\boldsymbol{\alpha}}))^{-1}$. To find $\widehat{\gamma}$ and $\widehat{\boldsymbol{\alpha}}$, we use the EM algorithm to maximize $p(\boldsymbol{y}|\gamma, \boldsymbol{\alpha})$. This is equivalent to maximizing

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}, \gamma) &= \log p(\boldsymbol{y}|\gamma, \boldsymbol{\alpha}) \\ &= -\frac{1}{2} \left[N \log 2\pi\right] + \log |\mathbf{C}| + \boldsymbol{y}^T \mathbf{C}^{-1} \boldsymbol{y}, \end{aligned} \tag{6}$$

with

$$\mathbf{C} = \gamma^{-1} \mathbf{I} + \mathbf{A} \text{Diag}(\boldsymbol{\alpha})^{-1} \mathbf{A}^T. \tag{7}$$

The EM algorithm proceeds by maximizing a lower bound to $\mathcal{L}(\boldsymbol{\alpha}, \gamma)$, obtained by treating $\boldsymbol{x}$ as hidden variables, namely,

$$\text{E}_{\boldsymbol{x}|\boldsymbol{y}, \widehat{\boldsymbol{\alpha}}_t, \widehat{\gamma}_t} p(\boldsymbol{y}, \boldsymbol{x}|\gamma, \boldsymbol{\alpha}). \tag{8}$$

Performing alternating optimization between γ and $\boldsymbol{\alpha}$ to maximize $\mathcal{L}(\boldsymbol{\alpha}, \gamma)$ leads to the following iterations

$$\text{E-Step: } Q(\boldsymbol{\alpha}, \gamma | \widehat{\boldsymbol{\alpha}}_t, \widehat{\gamma}_t) = \text{E}_{\boldsymbol{x}|\boldsymbol{y}, \widehat{\boldsymbol{\alpha}}_t, \widehat{\gamma}_t} \left[\log p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\alpha}, \gamma)\right], \tag{9}$$

$$\text{M-Step: } (\widehat{\gamma}_{t+1}, \widehat{\boldsymbol{\alpha}}_{t+1}) = \arg\max Q(\boldsymbol{\alpha}, \gamma | \widehat{\boldsymbol{\alpha}}_t, \widehat{\gamma}_t). \tag{10}$$

The above yields the hyperparameter updates in each iteration of Algorithm 1.

To reduce the computational complexity (in inverting the matrix $\boldsymbol{\Sigma}$) in the E-step, we use belief propagation (BP) [13]. In BP, all the messages passed between the nodes in the factor graph remain Gaussian (and hence can be parameterized by the means and variances). However, for a dense matrix $\mathbf{A}$, BP involves the exchange of MN messages (with $\min(M, N)$ computations per message) at every iteration [13]. The BP can be efficiently implemented using AMP [14]. The AMP iterations (with t denoting the iteration index) are as follows:

$$\boldsymbol{v}_t = \boldsymbol{y} - \mathbf{A}\widehat{\boldsymbol{x}}_t + \frac{\boldsymbol{v}_{t-1}}{M} \sum_{n=1}^{N} \eta_n'(r_{n,t}; \alpha_{n,t}), \tag{11}$$

and

$$\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}(\widehat{\boldsymbol{x}}_t + \mathbf{A}^T \boldsymbol{v}_t, \boldsymbol{\alpha}_t) \tag{12}$$

where $\boldsymbol{\eta}$ is a Lipschitz-continuous function and $\boldsymbol{\eta}'$ represents the derivative of $\boldsymbol{\eta}$ with respect to the first argument. The second term in $\boldsymbol{v}_t$, $\frac{\boldsymbol{v}_{t-1}}{M} \sum_{n=1}^{N} \eta_n'(r_{n,t}; \alpha_{n,t})$ represents the so-called Onsager correction term. When the entries of $\mathbf{A}$ are i.i.d sub-Gaussian with variance proportional to $1/M$, then the

Onsager correction term decouples the input to the shrinkage function $\boldsymbol{\eta}$, see [14] for details. That is,

$$\mathbf{r}_t = \widehat{\boldsymbol{x}}_t + \mathbf{A}^T \boldsymbol{v}_t \tag{13}$$

can be accurately (in the large system limit where $M, N \rightarrow \infty$) modeled as

$$\mathbf{r}_t = \boldsymbol{x} + \mathcal{N}(\mathbf{0}, c_t \mathbf{I}) \tag{14}$$

where c_t, is given in step 3 of the *Message Updates* part of Algorithm 1. Further, performing an minimum MSE (MMSE) estimate leads to the expression for $\widehat{\boldsymbol{x}}$ as

$$\begin{aligned} \widehat{\boldsymbol{x}} &= \boldsymbol{F}_t \mathbf{r}_t, \\ \text{where } \boldsymbol{F}_t &= \frac{1}{c_t}\left(\frac{1}{c_t}\mathbf{I} + \text{Diag}(\widehat{\boldsymbol{\alpha}}_t)\right)^{-1}. \end{aligned} \tag{15}$$

Note that no matrix inversion is required in the above step. However, AMP-SBL has several limitations such as divergence of the algorithm when $\mathbf{A}$ does not have i.i.d. entries. An intuitive workaround for this is to make $\mathbf{A}, \mathbf{A}^T$ in (11), (12) learnable parameters and optimize them using a DNN, which is the basis of our proposed approach described below.

III. Learned AMP-SBL (EM-LAMP-SBL)

In this section, we present our proposed EM-LAMP-SBL and highlight the differences between EM-LAMP-SBL and the state-of-the-art AMP-SBL. We also explain the benefits obtained from the DNN based solution. Finally, we describe the training procedure in detail.

A. Proposed EM-LAMP-SBL

In our proposed EM-LAMP-SBL approach, we consider unfolding the iterations of the AMP-SBL algorithm [12]. Our proposed solution is inspired by the learned AMP (LAMP) algorithm proposed in [15], where the iterations of an AMP algorithm are unfolded into a T layer DNN. In the LAMP version of the SBL, the MMSE estimate of the sparse vector $\boldsymbol{x}$ in SBL is replaced with the LAMP estimate and the block diagram is depicted in Fig 1. Unlike LAMP, which considers an identity covariance matrix for $\boldsymbol{x}$, in EM-LAMP-SBL, the covariance matrix is diagonal with non-identical entries. Due to this subtle but crucial difference, the LAMP iterations are not the same as that of AMP-SBL. Note that the LAMP algorithm involves replacing the matrices $(\mathbf{A},\ \mathbf{A}^T)$ with learnable parameters $(\mathbf{A}_t,\ \mathbf{B}_t)$ at iteration t. Another difference between EM-LAMP-SBL and LAMP is that the shrinkage function $\boldsymbol{\eta}(\mathbf{r}_t; \boldsymbol{\alpha}_t)$, which is a component-wise vector operation becomes different for each x_i. The n^{th} component of $\boldsymbol{\eta}(\mathbf{r}_t,\ \boldsymbol{\alpha}_t)$ is represented as $\eta_n(r_{n,t}; \alpha_{n,t})$. The only constraint on $\boldsymbol{\eta}$ is that it should be a Lipschitz-continuous function [15]. The hyperparameters $(\boldsymbol{\alpha},\ \gamma)$ are estimated using the M-step of the algorithm. The hyperparameter learning step can thus be viewed as a non-linear activation function, which is also a component-wise operation.

In Algorithm 1, we detail the iterations in our proposed EM-LAMP-SBL. We also depict the block diagram of the computations involved in one iteration of EM-LAMP-SBL in Fig 2. In Algorithm 1, we denote the posterior covariance at iteration t as $\boldsymbol{\Sigma}_t = \text{Diag}(\boldsymbol{\sigma}_t^2)$, $\boldsymbol{\sigma}_t^2 = [\sigma_{1,t}^2, ..., \sigma_{N,t}^2]$, where $\sigma_{i,t}^2$ is the posterior variance of x_i.

Algorithm 1 EM-LAMP-SBL (AMP-SBL) Algorithm. The steps in the brackets are the steps implemented in AMP-SBL in lieu of the corresponding steps in EM-LAMP-SBL.

Given: $\boldsymbol{y}, \mathbf{A}, M, N$.

Initialization: *a,b,c,d* are taken to be on the order of 10^{-10}. $\alpha_i^0 = a/b, \forall i, \gamma^0 = c/d$ and $\sigma_i^{2,0} = \frac{1}{\|\mathbf{A}_i\|^2 \gamma^0 + \alpha_i^0}, \boldsymbol{x}^0 = \mathbf{0}$.

At iteration t,

Definitions:

1) $\eta_n(r_{n,t}; \alpha_{n,t}) = r_{n,t} \frac{\frac{1}{c_t}}{\frac{1}{c_t} + \alpha_{n,t}}$, $\eta_n'(r_{n,t}; \alpha_{n,t}) = \frac{\frac{1}{c_t}}{(\frac{1}{c_t} + \alpha_{n,t})}$
2) $f_n(x_{n,t},\ \sigma_{n,t}^2) = |x_{n,t}|^2 + \sigma_{n,t}^2$

Message Updates:

1) $\mathbf{r}_t = \mathbf{B}_t \boldsymbol{v}_t + \widehat{\boldsymbol{x}}_t$, $(\mathbf{B}_t = \mathbf{A}^T)$
2) $\widehat{\boldsymbol{x}}_{t+1} = \boldsymbol{\eta}(\mathbf{r}_t; \boldsymbol{\alpha}_t), \quad \boldsymbol{\Sigma}_{t+1} = (\frac{1}{c_t}\mathbf{I} + \text{Diag}(\boldsymbol{\alpha}_t))^{-1}$
3) $c_{t+1} = \frac{1}{\gamma_t} + \frac{1}{M}\text{tr}\{\boldsymbol{\Sigma}_{t+1}\}$
4) $\boldsymbol{v}_{t+1} = \boldsymbol{y} - \mathbf{A}_t \widehat{\boldsymbol{x}}_{t+1} + \frac{\boldsymbol{v}_t}{M} \sum_{n=1}^{N} \eta_n'(r_{n,t}; \alpha_{n,t})$, ($\mathbf{A}_t = \mathbf{A}$)

Hyperparameter Updates:

1) $\boldsymbol{\alpha}_{t+1} = \frac{a + \frac{1}{2}}{\frac{\mathbf{f}(\widehat{\boldsymbol{x}}_{t+1}, \boldsymbol{\Sigma}_{t+1})}{2} + b}$
2) $\frac{1}{\gamma_{t+1}} = \frac{\frac{\|\boldsymbol{y} - \mathbf{A}_t \widehat{\boldsymbol{x}}_{t+1}\|^2 + \text{tr}(A^T A \boldsymbol{\Sigma}_{t+1})}{2} + d}{c + \frac{M}{2}}$.

To summarize our approach, we replace the AMP-SBL using a T-layer EM-LAMP-SBL with learnable parameters $(\mathbf{A}_t,\ \mathbf{B}_t)$. As we will show, this leads to faster convergence compared to the original AMP-SBL algorithm in [12]. Note that the resulting faster convergence is obtained at the expense of training the weights and a $T-$fold increase in memory complexity.

B. EM-LAMP-SBL Training

The training method for EM-LAMP-SBL is shown in Algorithm 2. Using the model in (1), we can artificially generate a training dataset in advance. Moreover, since the model is inspired by unfolding an existing high-performing algorithm, the number of learnable parameters is far fewer than a traditional neural network. Hence, a relatively small training dataset is sufficient for training the network. We start with the first layer and train each layer successively, freezing the weights of the previous layers. We use the MSE between the output of each layer and the true sparse vector as the loss function:

$$\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} \left\| \boldsymbol{x}^{(i)} - \eta(\mathbf{r}_T; \boldsymbol{\alpha}_T) \right\|^2 \tag{16}$$

where m denotes the minibatch size and $\boldsymbol{x}^{(i)}$ denotes the i^{th} training sample. In Fig 2, we depict a single layer of EM-LAMP-SBL. Each layer contains a learnable part which outputs the MMSE estimate of $\boldsymbol{x}$ given the hyperparameters,

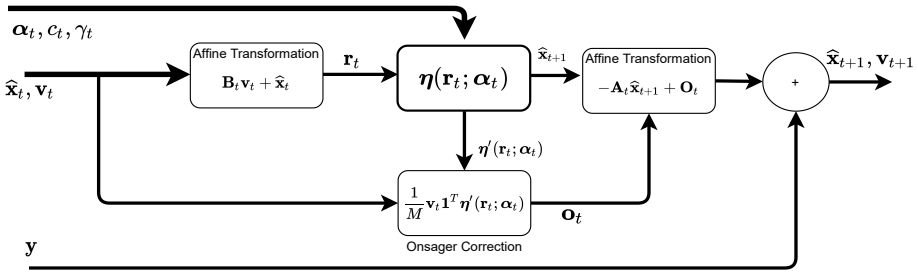

Fig. 1. Block diagram representing one iteration of LAMP-SBL.

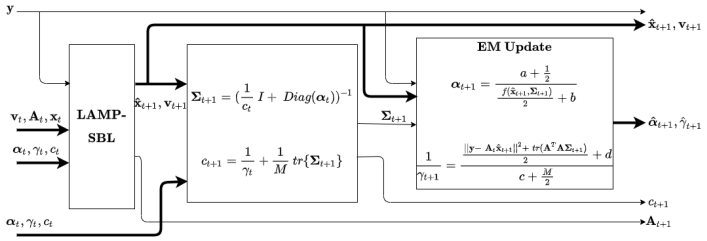

Fig. 2. A single layer of EM-LAMP-SBL. Thicker lines in the diagram represent multiple variables.

Algorithm 2 EM-LAMP-SBL

Given: $\boldsymbol{y}, \mathbf{A}, M, N$.
Initialization: *a,b,c,d* are taken to be very low, on the order of 10^{-10}. $\alpha_i^0 = a/b, \forall i, \gamma^0 = c/d$ and $\sigma_i^{2,0} = \frac{1}{\|\mathbf{A}_i\|^2\gamma^0 + \alpha_i^0}, \boldsymbol{x}^0 = \mathbf{0}, \boldsymbol{v}_0 = \boldsymbol{y}$.
Define $\boldsymbol{\Theta}_0 = \{\mathbf{B}, \mathbf{A}\}$
At iteration t,
  for $t = 1 : T$ **do**
    Initialize $\mathbf{B}_t = \mathbf{B}_{t-1}$, $\mathbf{A}_t = \mathbf{A}_{t-1}$. Specify the max iterations R and number of iterations to wait for the min NMSE S
    for R **do**
      Sample minibatch of m measurement vectors from the training set
      Learn $\boldsymbol{\Theta}_t$ using Adam optimizer
      if $\boldsymbol{\Theta}_t$ does not perform better than the min NMSE in S iterations, then stop
    end for
  end for

and an M-step which is used to estimate the hyperparameters.

IV. Simulations

The simulations are performed in a server equipped with 30 CPU cores and a GPU. The training and testing are implemented in python using TensorFlow, and we used the Adam optimizer [16]. We consider a measurement matrix $\mathbf{A}$ with dimensions $M = 150$ and $N = 250$ with i.i.d. Gaussian $\mathcal{N}(0, 1/M)$ entries, and the observations are generated at SNR $= 20$ dB. Also, we use a Bernoulli-Gaussian prior to generate the sparse vector $\boldsymbol{x}$, namely, $p(x_i) = p_i\delta(x_i) + (1-p_i)\mathcal{N}(0, 1)$, where $\delta(x_i)$ is the Dirac delta function and $p_i = 0.9$.

We evaluate the performance and robustness of the EM-LAMP-SBL algorithm in two scenarios. In Scenario 1, we evaluate the convergence of the EM-LAMP-SBL algorithm with varying condition number, denoted by κ, of $\mathbf{A}$. We compare the performance of the EM-LAMP-SBL algorithm with the original AMP-SBL and SAVE (space alternating variational estimation)-SBL [17] (which is based on mean-field variational Bayes for the estimation of $\boldsymbol{x}$ and hyperparameters). In Scenario 2, we examine the robustness of the EM-LAMP-SBL algorithm to deviations (errors) in the knowledge of the measurement matrix $\mathbf{A}$.

A. Scenario 1

In this scenario, we consider a matrix $\mathbf{A}$ generated with condition number κ. To generate $\mathbf{A}$ with a specific condition number, we first create a matrix with i.i.d, mean 0, variance $1/M$ Gaussian entries. Then, we perform a singular value decomposition (SVD) of $\mathbf{A}$. We also compute M uniformly spaced points between 1 and $1/\kappa$ in the logarithmic scale, and normalize them to have a unit sum. We replace the singular values of $\mathbf{A}$ with these M values, and create a new $\mathbf{A}$ with condition number κ by left- and right-multiplying by the matrices containing the corresponding singular vectors.

In Fig. 3, we plot the normalized MSE (NMSE) as a function of the iteration number (number of DNN layers for EM-LAMP-SBL) for $\kappa = 1$ and 100. When $\kappa = 1$, EM-LAMP-SBL converges faster than AMP-SBL and SAVE-SBL. For example, to achieve an NMSE of -15 dB, EM-LAMP-SBL requires only 5 iterations, while SAVE-SBL and AMP-SBL require 20 and 30 iterations, respectively. Since a single iteration of the three algorithms is of comparable complexity, this represents a four-fold reduction in complexity compared to the state-of-the-art. When $\kappa = 100$, AMP-SBL diverges. However, the EM-LAMP-SBL algorithm still converges, albeit with a slightly degraded NMSE. We also see that it outperforms SAVE-SBL, which takes around 100 iterations to converge to a slightly worse NMSE than EM-LAMP-SBL. This degradation in the NMSE with the condition number may be because the algorithms converge to sub-optimal local optima.

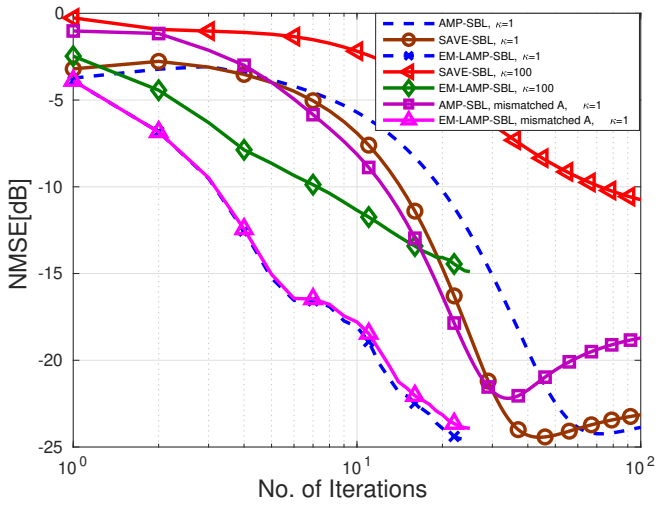

Fig. 3. NMSE versus the number of iterations/layers of the DNN, comprising (a) comparison with state-of-the art approaches; (b) for matrices with different condition numbers; and (c) with mismatched measurement matrices.

B. Scenario 2

We evaluate the case of mismatched measurement matrices, where the matrix $\mathbf{A}$ used to generate the observations is different from the matrix $\hat{\mathbf{A}}$ used by the recovery algorithm. We generate $\mathbf{A}$ with i.i.d. Gaussian $\mathcal{N}(0, 1/M)$ entries and condition number κ, and generate $\hat{\mathbf{A}}$ as $\hat{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A}$, where $\Delta\mathbf{A}$ has from a i.i.d Gaussian $\mathcal{N}(0, \sigma^2/M)$, with σ^2 set equal to the noise variance. We train the DNN by providing observations generated using different sets of $\hat{\mathbf{A}}$.

In this mismatched measurement matrix scenario, EM-LAMP-SBL performs much better than the AMP-SBL and SAVE-SBL. It takes around 25 iterations to converge to an NMSE of -25 dB when $\kappa = 1$ (and an NMSE of -15 dB when $\kappa = 100$). Moreover, the performance of our DNN version is superior to AMP-SBL. Hence, the EM-LAMP-SBL algorithm is robust to mismatch in $\mathbf{A}$.

C. Complexity and Energy Efficiency Comparison

We use the number of floating point operations (flops) per iteration as a proxy for energy efficiency, and compare EM-LAMP-SBL with AMP-SBL. Each iteration of AMP-SBL involves $3MN + 5N$ multiplications, which dominate the computational complexity. EM-LAMP-SBL also involves the same amount of computations as AMP-SBL since the only difference in each layer of the DNN architecture is that the $\mathbf{A}, \mathbf{A}^T$ are learned from the training data. We ignore the complexity of training since it is performed offline. Furthermore, to achieve an NMSE below -20 dB, the total complexity of the AMP-SBL or SAVE-SBL is of the order of $70(3MN + 5N)$ compared to $20(3MN + 5N)$ for the EM-LAMP-SBL, when the condition number is one (see Fig. 3). When the condition number increases to 100, the computational complexity and hence the energy efficiency of EM-LAMP-SBL is almost 5 times lower than the other algorithms.

V. Conclusions

In this paper, we presented EM-LAMP-SBL, a deep learning empowered computationally and energy efficient version of a state-of-the-art AMP-based sparse signal recovery algorithm. Through simulations on synthetic data, we illustrated the superior performance of the EM-LAMP-SBL algorithm in terms of the convergence rate, compared to its iterative model-based parent algorithm. The convergence rate dictates the number of iterations required for achieving a target NMSE, and, therefore, is a proxy for the energy efficiency of the algorithm. We also showed that EM-LAMP-SBL is robust to ill-conditioned matrices and model mismatch. Our algorithm can potentially be used in solving sparse signal recovery problems with structured or parameterized measurement matrices, e.g., in massive MIMO or mmWave channel estimation systems.

VI. Acknowledgments

C. K. Thomas was funded by a French FUI project titled MASS-START. M. Kountouris was partially supported from a Huawei France-funded Chair towards Future Wireless Networks. C. R. Murthy was funded by the Ministry of Electronics and Information Technology, Govt. of India.

References

[1] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, December 2007.

[2] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 129–159, 1998.

[3] D. Wipf and S. Nagarajan, "Iterative reweighted l_1 and l_2 methods for finding sparse solutions," *IEEE J. Sel. Topics Sig. Process.*, vol. 4, no. 2, pp. 317–329, April 2010.

[4] M. E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.

[5] D. P. Wipf and B. D. Rao, "Sparse Bayesian Learning for Basis Selection," *IEEE Trans. on Sig. Proc.*, vol. 52, no. 8, pp. 2153–2164, Aug. 2004.

[6] D. L. Donoho et al., "Message-passing algorithms for compressed sensing," *PNAS*, vol. 106, no. 106, pp. 18914–18919, Nov. 2009.

[7] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int. Symp. Inf. Theory*, Saint Petersburg, Russia, August 2011.

[8] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector Approximate Message Passing," *IEEE Trans. on Info. Theory*, vol. 65, no. 10, pp. 6664–6684, Oct. 2019.

[9] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *in Proc. Int. Conf. Mach. Learn.*, Haifa, Israel, 2010.

[10] U. Kamilov and H. Mansour, "Learning optimal nonlinearities for iterative thresholding algorithms," *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 747–751, 2016.

[11] Z.Wang, Q. Ling, and T. S. Huang, "Learning deep l0 encoders," in *in Proc. AAAI Conf. Artif. Intell.*, Arizona ,USA, 2016, p. 2194–2200.

[12] M. Al-Shoukairi and B. Rao, "Sparse Bayesian learning using approximate message passing," in *48th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, November 2014.

[13] X. Tan and J. Li, "Computationally Efficient Sparse Bayesian Learning via Belief Propagation ," *IEEE Trans. on Sig. Proc.*, vol. 58, no. 4, pp. 2010–2021, Apr. 2013.

[14] M. Bayati and A. Montanari, "The dynamics of message Passing on dense graphs, with Applications to Compressed Sensing ," *IEEE Trans. on Inf. Theory*, vol. 57, no. 2, pp. 764–785, February 2011.

[15] M. Borgerding, P. Schniter, and S. Rangan, "AMP-Inspired Deep Networks for Sparse Linear Inverse Problems," *IEEE Trans. On Sig. Process.*, pp. 4293–4308, Aug. 2017.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *in Proc. Int. Conf. Learn. Representations*, San Diego, USA, 2015.

[17] C. K. Thomas and Dirk Slock, "SAVE - Space alternating variational estimation for sparse Bayesian learning," in *IEEE Data Science Workshop*, Lausanne, Switzerland, June 2018.