Sparse Bayesian Learning Using Approximate Message Passing with Unitary Transformation

Man Luo, Qinghua Guo
School of Electrical, Computer and Telecommunications Engineering, University of Wollongong
qguo@uow.edu.au

Abstract—Sparse Bayesian learning (SBL) can be implemented with low complexity based on the approximate message passing (AMP) algorithm. However, it is vulnerable to ‘difficult’ measurement matrices as AMP can easily diverge. Damped AMP has been used to alleviate the problem at the cost of slowing the convergence speed. In this work, we propose an SBL algorithm based on the AMP with unitary transformation (UTAMP), where the shape parameter of the hyperprior is tuned automatically. It is shown that, compared to the state-of-the-art AMP based SBL algorithm, UTAMP-SBL is much more robust and much faster, leading to remarkably better performance. It is shown that in many cases, UTAMP-SBL can approach the support-oracle bound closely.

Index Terms—Sparse Bayesian learning, approximate message passing

I. INTRODUCTION

In this work, we consider a sparse Bayesian learning (SBL) problem with the following model

\[ y = Ax + w \] (1)

where \( y \) is an observation vector with length \( M \), \( A \) is a known measurement matrix with a size of \( M \times N \), \( x \) is a length-
(\( N \) sparse vector to be recovered, and \( w \) denotes a Gaussian noise vector with mean zero and covariance matrix \( \lambda^{-1} \mathbf{I} \).

It is assumed that the elements of \( x \) are independent and identically distributed, i.e., \( p(x) = \prod_{n=1}^{N} p(x_n) \). The sparsity promoting prior \( p(x_n) = \int \mathcal{N}(x_n|0, \gamma_n^{-1})p(\gamma_n)d\gamma_n \), where \( \mathcal{N}(x_n|0, \gamma_n^{-1}) \) denotes a Gaussian density with mean zero and variance \( \gamma_n^{-1} \), and \( p(\gamma_n) \) is a proper hyperprior.

The approximate message passing (AMP) algorithm was developed for compressive sensing based on the loopy belief propagation [1]. AMP has low complexity and its performance can be rigorously characterized by a scalar state evolution in the case of large i.i.d (sub)Gaussian matrix \( A \) [2]. However, for a generic \( A \), the convergence of AMP cannot be guaranteed, e.g., AMP can easily diverge for non-zero mean, rank-deficient, correlated, or ill-conditioned matrix \( A \) [3]. To address this problem, many AMP variants have been proposed, such as the damped AMP [3], the swept AMP [4], and GAMP with an adaptive damping and mean-removal procedure [5]. More effective variants include the AMP with unitary transform (UTAMP) [6] proposed in 2015, and the vector AMP [7] and the orthogonal AMP [8] proposed in 2016. In particular, UTAMP was derived based on a unitary transform of model [1], and it converges for any matrix \( A \) in the case of Gaussian priors [6].

AMP and its variants have been used for low complexity implementation of SBL [9]. By using AMP to implement the E-step in the expectation maximization (EM) based SBL, a significant reduction in complexity can be achieved. However, AMP-SBL can diverge easily for a difficult matrix \( A \), and exhibits poor performance. In [10], a GAMP based SBL algorithm (GGAMP-SBL) was proposed where the convergence is improved through damping but at the expense of slowing the convergence speed.

In this work, we derive a new SBL algorithm based on UTAMP (called UTAMP-SBL) to achieve low complexity robust SBL by taking advantage of the robustness and low complexity of UTAMP. In UTAMP-SBL, a Gamma distribution is chosen as the hyperprior for precision \( \gamma_n \), and the shape parameter of the Gamma distribution is automatically tuned during the iteration. It is shown that, compared to the state-of-the-art AMP based SBL algorithm GGAMP-SBL [10], UTAMP-SBL can deliver remarkably better performance in terms of robustness, speed and recovery performance. It is observed that, in many cases with difficult measurement matrices, UTAMP-SBL can still approach the support-oracle bound closely.

Notations: Boldface lowercase letters and uppercase symbols represent column vectors and matrices, respectively. Scalars are represented by non-boldface letters. \( \mathbf{1} \) and \( \mathbf{0} \) represent an all-one column vector and an all-zero column vector with proper sizes. The \( n \)-th element of vector \( \mathbf{c} \) is denoted by \( c_n \). \( \text{Diag}(\mathbf{a}) \) returns a diagonal matrix with the elements of \( \mathbf{a} \) on its diagonal. \( (\mathbf{A})_D \) returns a diagonal matrix by forcing the off-diagonal elements of \( \mathbf{A} \) to zero. \( \mathcal{N}(\mathbf{x}|\mu, \Sigma) \) denotes a Gaussian density of \( \mathbf{x} \) with mean \( \mu \) and covariance \( \Sigma \), and \( \text{Ga}(\gamma|\epsilon, \eta) \) denotes a Gamma distribution with shape parameter \( \epsilon \) and rate parameter \( \eta \). Let \((\cdot)^H\), \(\|\cdot\|\), \(|\cdot|\) and \(\delta(\cdot)\) denote the (conjugate) transpose, the \(l_2\) norm, the element-wise magnitude squared operations and the Dirac delta function, respectively. The notation \( \langle f(x) \rangle_{q(x)} \) denotes the expectation of the function \( f(x) \) with respect to probability density \( q(x) \).

II. AMP WITH UNITARY TRANSFORMATION

The UTAMP algorithm, inspired by [11], was derived based on the vector stepsize AMP algorithm shown in Algorithm [1] and a unitary transform of model [1] [6]. In AMP and UTAMP,
The function $g_x(q, \tau_q)$ returns a column vector whose $n$-th element, denoted as $[g_x(q, \tau_q)]_n$, is given by

$$[g_x(q, \tau_q)]_n = \frac{\int x_n p(x_n | N(x_n | q_n, \tau_{q_n})) dx_n}{\int p(x_n | N(x_n | q_n, \tau_{q_n})) dx_n}. \tag{2}$$

Equation (2) can be interpreted as the minimum mean square error (MMSE) estimation of $x_n$ based on the following model

$$q_n = x_n + \omega \tag{3}$$

where $\omega$ is a Gaussian noise with mean zero and variance $\tau_q$. The function $g_x'(q, \tau_q)$ returns a column vector and the $n$-th element is denoted by $[g_x(q, \tau_q)]_n$, where the derivative is with respect to $q_n$. It is not hard to show that $\tau_{q_n} [g_x(q, \tau_q)]_n$ is the posterior variance of $x_n$ with model (3). Note that $g_x(q, \tau_q)$ can also be changed for MAP (maximum a posteriori) estimation of $x$.

The derivation of UTAMP is briefly introduced in the following. As any matrix $A$ can have its singular value decomposition (SVD) $A = U \Lambda V$, a unitary transformation with $UH$ to (1) can be performed, yielding

$$r = AVx + \omega \tag{4}$$

where $r = UH y$, and $\omega = UH w$ is still a zero-mean Gaussian noise vector with the same covariance matrix $\lambda^{-1} I$.

It is not hard to verify that

$$|C|^2 d = (C \text{Diag}(d) C^H) d 1. \tag{5}$$

Now suppose we have a variance vector $\tau_p^1$. According to Line 1 in the vector stepsize AMP and using (5), we have

$$\tau_p = (AV \text{Diag}(\tau_p^1) V^H \Lambda^H) d 1. \tag{6}$$

We can find that if $\text{Diag}(\tau_p^1)$ is a scaled identity matrix, the computation of (6) can be significantly simplified. This motivates the replacement of $\tau_p^1$ with $\tau_p^1 1$ where $\tau_p^1$ is the average of the elements of $\tau_p^1$. So (6) is reduced to

$$\tau_p = \tau_p^1 \Lambda \Lambda^H 1 \tag{7}$$

which is Line 1 of the UTAMP algorithm. Lines 2, 3 and 4 of UTAMP can be obtained according to Lines 2, 3 and 4 of the vector stepsize AMP by simply replacing $A$ with $AV$. According to (5) again, Line 5 of the vector stepsize AMP with matrix $AV$ can be represented as

$$1/\tau_q = (V^H A^H \text{Diag}(\tau_s) A V) d 1. \tag{8}$$

We then replace the diagonal matrix $\Lambda^H \text{Diag}(\tau_s) A$ with a scaled identity matrix $\beta I$ where $\beta$ is the average of the diagonal elements of $\Lambda^H \text{Diag}(\tau_s) A$, i.e.,

$$\beta = (1/N)^2 \Lambda^H \Lambda. \tag{9}$$

Hence (8) is reduced to Line 5 of the UTAMP algorithm. Line 6 can be obtained from Line 6 of the vector stepsize AMP by replacing $A = UAV$ with $AV$. Compared with Line 7 in the vector stepsize AMP, an additional average operation is performed in Line 7 in UTAMP to meet the requirement of a scalar $\tau_p^1$ in Line 1. We note that the average operation is not necessarily in Line 7 as we can also put the additional average operation in Line 8 in UTAMP is the same as Line 8 of the vector stepsize AMP except that $\tau_q$ is a scalar. The UTAMP algorithm is summarized in Algorithm 2.

**Remarks:** It is worth pointing out that UTAMP is not equivalent to the vector step size AMP due to the approximations made in the derivation. Interestingly, it is these approximations that make UTAMP much more robust than AMP.

### III. Sparse Bayesian Learning with UTAMP

In this Section, the SBL algorithm UTAMP-SBL is proposed based on the UTAMP algorithm, where we assume that the noise precision $\lambda$ is unknown. The UTAMP-SBL algorithm is derived using the factor graph representation of model (4). The factor graph is shown in Fig. [H] where $f_{x_m}(r_m | h_m, \lambda) = \mathcal{N}(r_m | h_m, \lambda^{-1})$, $f_{\delta_n}(h_m | x_n) = \delta(h_m - (AV)_m x_n)$ with $(AV)_m$ being the $m$-th row of matrix $AV$, $f_\lambda(\lambda)$ denotes the prior of the noise precision $\lambda$, $f_{x_n}(x_n | \gamma_n) = \mathcal{N}(x_n | 0, \gamma_n^{-1})$ denotes the Gaussian prior of $x_n$, and $f_{\tau_q}(\gamma_n) = \text{Ga}(\lambda, \eta)$ is the hyperprior. UTAMP-SBL is derived by incorporating the UTAMP algorithm to the message passing in the factor graph shown in Fig. [I] where the whole graph is divided into 3 subgraphs. The subgraph in the middle is mainly handled by UTAMP, and the message computations in Subgraphs 1 and 2 are detailed in the following.
Fig. 1: Factor graph for deriving UTAMP-SBL.

A. Message Computations in Subgraph 1

1) Backward Message Passing: According to the derivation of AMP using loopy belief propagation, UTAMP provides the message $m_{r_m \rightarrow h_m}(h_m) = \mathcal{N}(\hat{h}_m, \tau_m)$ where the mean $\hat{h}_m$ and the variance $\tau_m$ are given in Lines 1 and 2 of the UTAMP algorithm (Algorithm 2), which are also Lines 1 and 2 in the UTAMP-SBL algorithm (Algorithm 3). The mean field (MF) rule is used at the function nodes $\{f_{r_m}\}$. Hence, we need to compute the belief $b(h_m)$, i.e., $b(h_m) \propto m_{h_m \rightarrow f_{r_m}(h_m)m_{r_m \rightarrow h_m}(h_m)}$. Later we will see that $m_{f_{r_m} \rightarrow h_m}(h_m) \propto \mathcal{N}(\hat{h}_m, \hat{\lambda}^{-1})$ where $\hat{\lambda}^{-1}$ is an estimate of $\lambda$ (in the last iteration), and its computation is delayed to Line 12. Hence $b(h_m)$ is Gaussian i.e., $b(h_m) = \mathcal{N}(\hat{h}_m, \tau_m)$, where

$$v_{h_m} = (1/\tau_{p_m} + \hat{\lambda}^{-1})^{-1}, \quad \hat{h}_m = v_{h_m}(r_m \hat{\lambda} + p_m/\tau_{p_m}) \quad (10)$$

leading to Lines 3 and 4 of the UTAMP-SBL algorithm. The message from $f_{r_m}$ to $\lambda$ is calculated by the MF rule, which reads

$$m_{f_{r_m} \rightarrow \lambda}(\lambda) = \exp\left\{\langle \log f_{r_m}(r_m, h_m, \lambda^{-1}) \rangle_{b(h_m)}\right\} \propto \sqrt{\lambda} \exp\left\{-\frac{\lambda}{2} \left((r_m - \hat{h}_m)^2 + v_{h_m}\right)\right\} \quad (11)$$

2) Forward Message Passing: According to the MF rule, the message $m_{f_{r_m} \rightarrow h_m}(h_m)$ reads

$$m_{f_{r_m} \rightarrow h_m}(h_m) = \exp\left\{\langle \log f_{r_m}(r_m, h_m, \lambda^{-1}) \rangle_{b(h_m)}\right\} \propto \mathcal{N}(r_m, \hat{\lambda}^{-1}) \quad (12)$$

where $\hat{\lambda} = \langle \lambda \rangle_{b(\lambda)}$. With the prior density $f_{\lambda}(\lambda) \propto \frac{1}{\lambda}$,

$$b(\lambda) \propto f_{\lambda}(\lambda) \prod_m m_{f_{r_m} \rightarrow \lambda}(\lambda) \propto \lambda^{N/2} \exp\left\{-\frac{\lambda}{2} \sum_m (||r_m - \hat{h}_m||^2 + v_{h_m})\right\} \quad (13)$$

Hence

$$\hat{\lambda} = M/\sum_m (||r_m - \hat{h}_m||^2 + v_{h_m}), \quad (14)$$

which is Line 5 of the UTAMP-SBL algorithm.

Algorithm 3 UTAMP-SBL

Unitary transform: $x = U^H y = \Lambda V x + \omega$, where $\Lambda = UAV$.

Define vector $\lambda_p = \Lambda \lambda A^T$.

Initialization: $\tau_x^{(0)} = 1$, $\tilde{x}^{(0)} = 0$, $\tilde{\epsilon} = 0.001$, $\tilde{\gamma}^{(0)} = 1$, $\lambda_r^{(-1)} = 1$, $s^{(-1)} = 0$, and $t = 0$.

Do

1: $\tau_p = \tau_x^t \lambda_p$
2: $p = \Lambda V \tilde{x}^t - \tau_p \cdot s^{(-1)}$
3: $v_h = 1/(1/\tau_p + \lambda_r^{(-1)})$
4: $\hat{h} = v_h \cdot (\lambda_r - r + p/\tau_p)$
5: $\hat{\lambda} = M/(||r - \hat{h}||^2 + 1) V \tilde{v}_h$;
6: $\tau_s = 1/(\tau_p + (\lambda_r^{(-1)} - 1))$
7: $s^t = s \cdot (r - p)$
8: $1/\tau_q = (1/N) \lambda^H \tau_s$
9: $q = \tilde{x}^t + \tau_q (V H^H s')$
10: $\tau_x^{t+1} = (\tau_q/N) \lambda^H (1/(1 + \tau_q \tilde{\gamma}^{(t)}))$
11: $\tilde{x}^{t+1} = q / (1 + \tau_q \tilde{\gamma}^{(t)})$
12: $\tilde{\gamma}^{t+1} = (2 \epsilon + 1)/(||\tilde{x}^{t+1}||^2 + \tilde{\gamma}^{t+1})$, $n = 1, ... , N$.
(Note: $\tilde{\gamma}^{t+1} = (\epsilon + 1)/(||\tilde{x}^{t+1}||^2 + \tilde{\gamma}^{t+1})$ for complex case)
13: $\tilde{\epsilon} = \frac{1}{2} \sqrt{\sum_n \tilde{\gamma}_n^{t+1} - \frac{1}{N} \sum_n \log \tilde{\gamma}_n^{t+1}}$
14: $t = t + 1$

while $(\|\tilde{x}^{t+1} - \tilde{x}^t\|^2 / \|\tilde{x}^{t+1}\|^2 > \delta_x$ and $t < t_{max})$

B. Message Computations in Subgraph 2

1) Forward Message Passing: According to the derivation of AMP and UTAMP, UTAMP produces the message $m_{x_n \rightarrow f_{r_n}(x_n)} \propto \mathcal{N}(q_n, \tau_\eta)$ with mean $q_n$ and variance $\tau_\eta$, which correspond to Lines 5 and 6 of the UTAMP algorithm and Line 8 and Line 9 of the UTAMP-SBL algorithm. The MF rule is also used at nodes $\{f_{x_n}\}$. Hence, we need to compute the belief $b(x_n)$, i.e., $b(x_n) \propto m_{x_n \rightarrow f_{r_n}(x_n)m_{f_{r_n} \rightarrow x_n}(x_n)}$. The message $m_{f_{r_n} \rightarrow x_n}(x_n) \propto \mathcal{N}(x_n|\tilde{x}_n, \tau_x)$ will be defined in (22), where $\tilde{\gamma}_n = \langle \gamma \rangle_{b(\gamma)}$. So $b(x_n) = \mathcal{N}(x_n|\tilde{x}_n, \tau_x)$ with

$$\tau_x = (1/\tau_\eta + \tilde{\gamma}_n)^{-1}, \quad \tilde{x}_n = q_n/(1 + \tau_\eta \tilde{\gamma}_n) \quad (15)$$

leading to Lines 10 and 11 of the UTAMP-SBL algorithm (noting that $\tau_x$ is the average of $\{\tau_x\}$).

By the MF rule, the message $m_{f_{r_n} \rightarrow \gamma_n}(\gamma_n)$ from $f_{x_n}$ to $\gamma_n$ is calculated by

$$m_{f_{r_n} \rightarrow \gamma_n}(\gamma_n) = \exp\left\{\langle \log f_{x_n}(x_n, 0, \gamma_n^{-1}) \rangle_{b(\gamma_n)}\right\} \propto \sqrt{\gamma_n} \exp\left\{-0.5 \gamma_n (||\tilde{x}_n||^2 + \tau_x)\right\} \quad (16)$$

As defined later in (19) $m_{f_{r_n} \rightarrow \gamma_n}(\gamma_n) \propto \gamma_n^{\tilde{\epsilon} - 1} \exp\left\{-\frac{\tau_\eta}{2} (||\tilde{x}_n||^2 + \tau_x)\right\}$ where $\tilde{\epsilon} = \langle \epsilon \rangle_{b(\epsilon)}$, the belief $b(\gamma_n)$ is scaled to a Gamma distribution i.e.,

$$b(\gamma_n) \propto m_{f_{r_n} \rightarrow \gamma_n}(\gamma_n)m_{f_{r_n} \rightarrow x_n}(x_n) \propto \gamma_n^{\tilde{\epsilon} - 1} \exp\left\{-\frac{\tau_\eta}{2} (||\tilde{x}_n||^2 + \tau_x + 2\eta)\right\} \quad (17)$$

Hence $\tilde{\gamma}_n = 1/2^{n+1} \tilde{x}_n^2$. Here we set $\eta = 0$, and $\tilde{\gamma}_n$ is reduced to $1/\tilde{x}_n^2 + \tau_x$, which leads to Line 12 of the UTAMP-SBL algorithm.
According to the MF rule, \( m_{f_n \rightarrow \epsilon} = \frac{\epsilon}{N} \exp \left\{ (\epsilon - 1) \left( \log \gamma_n \right) \right\} \). Thus,

\[
b(\epsilon) \propto \prod_n m_{f_n \rightarrow \epsilon} = \prod_n \frac{\eta N e^{\eta \gamma_n}}{(\Gamma(\epsilon))^{\eta}} \exp \left\{ (\epsilon - 1) \sum_n \left( \log \gamma_n b(\gamma_n) - \eta \gamma_n \right) \right\}
\]

(18)

2) Backward Message Passing: According to the MF rule, the message \( m_{f_n \rightarrow \gamma_n} \) is calculated as

\[
m_{f_n \rightarrow \gamma_n}(\gamma_n) = \exp \left\{ (\log f_x(\gamma_n, \eta)) \right\} \propto \mathcal{N}(\gamma_n | \hat{\epsilon}, \eta)
\]

where \( \hat{\epsilon} = (\epsilon)_{b(\epsilon)} \). However, it is difficult to calculate \( \hat{\epsilon} \). To circumvent this problem, we may use \( \hat{\epsilon}' = \arg \max_{\epsilon} b(\epsilon) \) to replace \( \hat{\epsilon} \), and \( \hat{\epsilon}' \) can be obtained iteratively with the following equation [12]

\[
\frac{1}{\epsilon_{\text{new}}} = \frac{1}{\epsilon} + \frac{g(\epsilon)}{\epsilon^2 (1/\epsilon - \Psi'(\epsilon))}
\]

(20)

where

\[
g(\epsilon) = \frac{1}{N} \sum_n \log \gamma_n - \Psi(\epsilon) + \log \left( \frac{\epsilon N}{\sum_n \gamma_n} \right) + \Psi(\hat{\epsilon} + 0.5) - \log(\hat{\epsilon} + 0.5),
\]

\( \Psi \) is the digamma function and \( \Psi' \) is the derivative of the digamma function. We find the following simple but more effective equation to update \( \hat{\epsilon}' \), where iteration is not required and \( \hat{\epsilon}' \) can be calculated straightforwardly

\[
\hat{\epsilon}' = \frac{1}{2} \sqrt{\log \left( \frac{1}{N} \sum \gamma_n \right) - \frac{1}{N} \sum \log(\gamma_n)}.
\]

(21)

which is Line 13 of the UTAMP-SBL algorithm. By the MF rule, the message \( m_{f_n \rightarrow x_n} \) is updated by

\[
m_{f_n \rightarrow x_n}(x_n) = \exp \left\{ (\log f_x(x_n, 0, \gamma_n^{-1})) \right\} \propto \mathcal{N}(x_n | 0, \gamma_n^{-1}).
\]

(22)

Remarks: In the above derivation, we assume that all variables are real-valued. The results can be easily extended to the case of proper complex variables. In this case, the UTAMP-SBL algorithm is still the same except that the coefficient 2 in Line 12 is removed and the superscript “H” represents the conjugate transpose operation.

IV. NUMERICAL RESULTS

We compare UTAMP-SBL with the state-of-the-art AMP based SBL algorithm GGAMP-SBL algorithm [10] with estimated noise variance and 3 times of the true noise variance (as suggested in [10]). As a performance benchmark, the support-oracle MMSE bound [10] is also included. We set \( N = 1000 \) and \( M = 800 \). The vector \( x \) is drawn from a Bernoulli-Gaussian distribution with a non-zero probability \( \rho \). The signal to noise power ratio SNR \( \triangleq 10 \log 10(E \| Ax \|^2 / E \| w \|^2) \) (dB). We use the normalized mean squared error NMSE \( \triangleq 10 \log 10(\frac{1}{T} \sum_t \| \hat{x}_t - x^t \|^2 / ||x^t||^2) \) (dB) to evaluate the recovery performance, where \( \hat{x}^t \) is an estimate of \( x^t \), and \( T \) is the number of Monte Carlo simulations.

For UTAMP-SBL, we set the maximum iteration number \( t_{\text{max}} = 300 \) (noting that there is no inner iteration in UTAMP-SBL). For GGAMP-SBL, the maximum numbers of E-step and outer iteration are set to be 50 and 1000 respectively. The damping factor is 0.2. The SNR in all simulations is 60dB. All results are obtained by an ordinary PC with an Intel Core i7 3.50 GHz CPU and 64.0 GB RAM. We examine
O N E T W O T H R E E

the performance of the algorithms with the following different types of measurement matrices.

1) Ill-Conditioned Matrix: Matrix $A$ is constructed based on the SVD $A = U \Lambda V^T$ where $A$ is a singular value matrix with $\Lambda_{i,i}/\Lambda_{1,1,i+1} = \kappa^{1/(M-1)}$ (i.e., the condition number of the matrix is $\kappa$). The NMSE performance of all algorithms is shown in Fig. 2.

2) Correlated Matrix: The correlated matrix $A$ is constructed using $A = C_L^{1/2} G C_R^{1/2}$, where $G$ is an i. i. d. Gaussian matrix, and $C_L$ is an $M \times M$ matrix with the $(m,n)$th element given by $c(m-n)$ where $c \in [0, 1]$. Matrix $C_R$ is generated in the same way but with a size of $N \times N$. The parameter $c$ controls the correlation of matrix $A$. The NMSE performance of the algorithms with the parameter $c$ is shown in Fig. 3.

3) Non-Zero Mean Matrix: The elements of matrix $A$ are drawn from a non-zero mean Gaussian distribution, i.e., $a_{m,n} \sim N(\mu_m, 1/N)$. The mean $\mu$ measures the derivation from the i. i. d. zero-mean Gaussian matrix. The NMSE performance of all the algorithms is presented in Fig. 4.

4) Low Rank Matrix: The measurement matrix $A = BC$, where the size of $B$ and $C$ are $M \times R$ and $R \times N$, respectively, and $R < M$. Both $B$ and $C$ are i.i.d. Gaussian matrices with mean zero and unit variance. The rank ratio $R/N$ is used to measure the deviation of matrix $A$ from the i.i.d. Gaussian matrix. The NMSE performance is shown in Fig. 5.

It can be seen from Fig. 2 to Fig. 5 that, when the deviation of the measurement matrices from the i.i.d. zero-mean Gaussian matrix is small, GGAMP-SBL (with $3 \times$ true noise variance) and UTAMP-SBL deliver similar performance, and both of them can approach the bound closely. However, when the deviation is relatively large, UTAMP-SBL can significantly outperform GGAMP-SBL, and in many cases, the performance of UTAMP-SBL is still close to the bound, which demonstrates that UTAMP-SBL is much more robust.

Due to limited space, we only show the average running time for different algorithms for the case of ill-conditioned measurement matrices in Fig. 6. It can be seen that UTAMP-SBL is much faster than GGAMP-SBL.

V. C O N C L U S I O N

In this work, we have proposed a UTAMP based SBL algorithm UTAMP-SBL, which inherits the low complexity and robustness of UTAMP to difficult measurement matrix $A$. It has been demonstrated that UTAMP-SBL can significantly outperform the state-of-the-art AMP based SBL algorithm in terms of robustness, speed and recovery accuracy for difficult measurement matrices.

R E F E R E N C E S

[1] D. L. Donoho, A. Maleki, and A. Montanari, “Message-passing algorithms for compressed sensing,” Proceedings of the National Academy of Sciences, vol. 106, no. 45, pp. 18914–18919, 2009.
[2] A. Javanmard and A. Montanari, “State evolution for general approximate message passing algorithms, with applications to spatial coupling,” Information and Inference: A Journal of the IMA, vol. 2, no. 2, pp. 115–144, 2013.
[3] S. Rangan, P. Schniter, A. K. Fletcher, and S. Sarkar, “On the convergence of approximate message passing with arbitrary matrices,” IEEE Transactions on Information Theory, 2019.
[4] A. Manoel, F. Krzakala, E. W. Tramel, and L. Zdeborová, “Sparse estimation with the swept approximated message-passing algorithm,” arXiv preprint arXiv:1406.4311, 2014.
[5] J. Vila, P. Schniter, S. Rangan, F. Krzakala, and L. Zdeborová, “Adaptive damping and mean removal for the generalized approximate message passing algorithm,” in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015, pp. 2021–2025.
[6] Q. Guo and J. Xi, “Approximate message passing with unitary transformation,” arXiv preprint arXiv:1504.07479, 2015.
[7] S. Rangan, P. Schniter, and A. K. Fletcher, “Vector approximate message passing,” in 2017 IEEE International Symposium on Information Theory (ISIT). IEEE, 2017, pp. 1588–1592.
[8] J. Ma and L. Ping, “Orthogonal amp.” IEEE Access, vol. 5, pp. 2020–2033, 2017.
[9] M. Al-Shoukairi and B. Rao, “Sparse bayesian learning using approximate message passing,” in 2014 48th Asilomar Conference on Signals, Systems and Computers. IEEE, 2014, pp. 1957–1961.
[10] M. Al-Shoukairi, P. Schniter, and B. D. Rao, “A gamp-based low complexity sparse bayesian learning algorithm,” IEEE Transactions on Signal Processing, vol. 66, no. 2, pp. 294–308, 2018.
[11] Q. Guo, D. D. Huang, S. Nordholm, J. Xi, and Y. Yu, “Iterative frequency domain equalization with generalized approximate message passing,” IEEE Signal Processing Letters, vol. 20, no. 6, pp. 559–562, 2013.
[12] D. Fink, “A compendium of conjugate priors.” See http://www.people. cornell.edu/pages/df56/CONJINTRev%20TEX. pdf, vol. 46, 1997.