Approximate Message Passing with Built-in Parameter Estimation for Sparse Signal Recovery

Shuai Huang, Trac D. Tran
Department of Electrical and Computer Engineering
Johns Hopkins University, Baltimore, MD, USA
{shuaihuang, trac}@jhu.edu

Abstract
The approximate message passing (AMP) algorithm shows advantage over conventional convex optimization methods in recovering under-sampled sparse signals. AMP is analytically tractable and has a much lower complexity. However, it requires that the true parameters of the input and output channels are known. In this paper, we propose an AMP algorithm with built-in parameter estimation that jointly estimates the sparse signals along with the parameters by treating them as unknown random variables with simple priors. Specifically, the maximum a posterior (MAP) parameter estimation is presented and shown to produce estimations that converge to the true parameter values. Experiments on sparse signal recovery show that the performance of the proposed approach matches that of the oracle AMP algorithm where the true parameter values are known.

1 Introduction
Sparse signal recovery (SSR) is the key topic in Compressive Sensing (CS) [1–4], it lays the foundation for applications such as dictionary learning [5], sparse representation-based classification [6], etc. Specifically, SSR tries to recover the sparse signal \( x \in \mathbb{R}^N \) given a \( M \times N \) sensing matrix \( A \) and a measurement vector \( y = Ax + w \in \mathbb{R}^M \), where \( M < N \) and \( w \in \mathbb{R}^M \) is the unknown noise introduced in this process. Although the problem itself is ill-posed, perfect recovery is still possible provided that \( x \) is sufficiently sparse and \( A \) is incoherent enough [1]. Lasso [7], a.k.a \( l_1 \)-minimization, is one of most popular approaches proposed to solve this problem:

\[
\arg \min_x \| y - Ax \|_2^2 + \gamma \| x \|_1,
\]

where \( \| y - Ax \|_2^2 \) is the data-fidelity term, \( \| x \|_1 \) is the sparsity-promoting term, and \( \gamma \) balances the trade-off between them.

The authors would like to thank NSF-CCF-1117545, NSF-CCF-1422995 and NSF-EECS-1443936 for funding.
From a probabilistic view, Lasso is equivalent to a maximum likelihood (ML) estimation of the signal $x$ under the assumption that the entries of $x$ are i.i.d. distributed following the Laplace distribution $p(x_j) \propto \exp(-\lambda|x_j|)$, and those of $w$ are i.i.d. distributed following the Gaussian distribution $p(w_i) \propto \exp(-w_i^2/2\theta)$. Let $z = Ax$, we have $p(y_i|x) \propto \exp(-(y_i - z_i)^2/2\theta)$. The ML estimation is then arg max$_x p(x, y)$, which is essentially the same as (1). In general, SSR can be described by the Bayesian model from [8], as is shown in Fig. 1.

Under the Bayesian setting it is possible to design efficient iterative algorithms to compute either the maximum a posterior (MAP) or minimum mean square error (MMSE) estimate of the signal $x$. Most notable among them is the approximate message passing (AMP) [9–11]. The AMP algorithm performs probabilistic inferences on the corresponding factor graph using Gaussian and quadratic approximations of loopy belief propagation (loopy BP), a.k.a. message passing [12]. Based on different inference tasks, loopy BP has two variants: sum-product message passing for the MMSE estimate of $x$ and max-sum message passing for the MAP estimate of $x$ [8]. Although AMP is analytically tractable and has low complexity [9–11], it requires the parameters $\{\lambda, \theta\}$ in the input and output channels are known exactly, which is often not satisfied in practice. Various methods have been proposed to include parameter estimation in the AMP algorithm using Expectation-Maximization [13, 14] and adaptive methods [15].

In this paper, we propose an extension to the generalized approximate message passing (GAMP) [8] by treating the parameters $\{\lambda, \theta\}$ as unknown random variables with prior distributions and estimating them jointly with the signal $x$. Using the sum-product GAMP as an example, we give the message passing updates between the factor nodes and the variable nodes, which serves as the basis to write the state evolution equations of the GAMP algorithms. The MAP parameter estimation can then be derived by maximizing the approximated posterior marginals $p(\lambda|y)$ and $p(\theta|y)$. Following the analysis in [15], we can show that the estimated parameter values from the MAP estimation also converge to the true parameter values if they are computed exactly. To compare the proposed GAMP with built-in parameter estimation (PE-GAMP) with oracle-GAMP where the true parameters are known, we run sparse signal recovery experiments with Bernoulli-Gaussian (BG) input channel and additive white Gaussian noise (AWGN) output channel. The experiments show that the performance of PE-GAMP does match that of oracle-GAMP.
2 GAMP with Built-in Parameter Estimation

The generalized factor graph for the proposed PE-GAMP that treats the parameters as random variables is shown in Fig. 2. Here we adopt the same notations used by [8]. Take the messages being passed between the factor node $\Phi_m$ and the variable node $x_n$ for example, $\Delta_{\Phi_m \rightarrow x_n}$ is the message from $\Phi_m$ to $x_n$, and $\Delta_{x_n \rightarrow \Phi_m}$ is the message from $x_n$ to $\Phi_m$. Both $\Delta_{\Phi_m \rightarrow x_n}$ and $\Delta_{x_n \rightarrow \Phi_m}$ can be viewed as functions of $x_n$. In the following section 2.1 and 2.2, we give the messages being passed on the generalized factor graph in log domain for the sum-product message passing algorithm and the max-sum message passing algorithm respectively.

2.1 Sum-product Message Passing

In the following, we first present the sum-product message passing in the $(t+1)$-th iteration.

\[ \Delta^{(t+1)}_{\Phi_m \rightarrow x_n} = \text{const} + \log \int_{\Phi_m(y_m, x, \theta)} \Phi_m(y_m, x, \theta) \cdot \exp \left( \sum_{j \neq n} \Delta^{(t)}_{\Phi_m \leftarrow x_j} + \sum_{v} \Delta^{(t)}_{\Phi_m \leftarrow \theta_v} \right) \]  

(2)

\[ \Delta^{(t+1)}_{x_n \rightarrow \Phi_m} = \text{const} + \Delta^{(t)}_{\Omega_{n \rightarrow x_n}} + \sum_{i \neq m} \Delta^{(t)}_{\Phi_i \rightarrow x_n} \]  

(3)

\[ \Delta^{(t+1)}_{\Omega_n \rightarrow x_n} = \text{const} + \log \int_{\lambda} \Omega_n(x_n, \lambda) \cdot \exp \left( \sum_{u} \Delta^{(t)}_{\Omega_n \leftarrow \lambda_u} \right) \]  

(4)

\[ \Delta^{(t+1)}_{\Omega_n \leftarrow x_n} = \text{const} + \sum_{i} \Delta^{(t)}_{\Phi_i \rightarrow x_n} \]  

(5)

where $\Phi_m(y_m, x, \theta) = p(y_m|x, \theta)$ and $\Omega_n(x_n, \lambda) = p(x_n|\lambda)$. Let $\Gamma(x_n)$ denote the factor nodes in the neighborhood of $x_n$. The posterior marginal of $x_n$ is:

\[ p(x_n|y) \propto \exp \Delta^{(t)}_{\Omega_n \rightarrow x_n} = \exp \left( \Delta^{(t)}_{\Omega_n \rightarrow x_n} + \sum_{\Phi_m \in \Gamma(x_n)} \Delta^{(t)}_{\Phi_m \rightarrow x_n} \right) \]  

(6)

Using $p(x|y)$, the MMSE estimate of $x$ can then be computed:

\[ \hat{x}_n = \mathbb{E}[x_n|y] = \int x_n p(x_n|y) \]  

(7)

Similarly, we can write the message passing updates involving $\lambda, \theta$ as follows:

\[ \Delta^{(t+1)}_{\Omega_{n \rightarrow \lambda_l}} = \text{const} + \log \int_{\Omega_n(x_n, \lambda)} \Omega_n(x_n, \lambda) \cdot \exp \left( \Delta^{(t)}_{\Omega_n \leftarrow x_n} + \sum_{u \neq l} \Delta^{(t)}_{\Omega_n \leftarrow \lambda_u} \right) \]  

(8)

\[ \Delta^{(t+1)}_{\Omega_n \leftarrow \lambda_l} = \text{const} + \sum_{j \neq n} \Delta^{(t)}_{\Omega_j \leftarrow \lambda_l} + \log p(\lambda_l) \]  

(9)

\[ \Delta^{(t+1)}_{\Phi_m \rightarrow \theta_k} = \text{const} + \log \int_{\theta_k \Phi_m(y_m, x, \theta)} \Phi_m(y_m, x, \theta) \cdot \exp \left( \sum_{j} \Delta^{(t)}_{\Phi_m \leftarrow x_j} + \sum_{v \neq k} \Delta^{(t)}_{\Phi_m \leftarrow \theta_v} \right) \]  

(10)

\[ \Delta^{(t+1)}_{\Phi_m \leftarrow \theta_k} = \text{const} + \sum_{i \neq m} \Delta^{(t)}_{\Phi_i \rightarrow \theta_k} + \log p(\theta_k) \]  

(11)

where $p(\lambda_l), p(\theta_k)$ are the pre-specified prior on the parameters. In general, if we don’t have any knowledge about how the parameters are distributed, we can fairly assume a uniform
Figure 2: The factor graph for the proposed PE-GAMP. ■ represents the factor node, and ○ represents the variable node. \( \mathbf{\Lambda} = \{\lambda_1, \cdots, \lambda_L\} \) and \( \mathbf{\Theta} = \{\theta_1, \cdots, \theta_K\} \) are the parameters. \( \mathbf{x} = [x_1, \cdots, x_N]^T \) is the sparse signal.

prior and treat \( p(\lambda_i), p(\theta_k) \) as constants. Additionally, we also have the following posterior marginals of \( \lambda_l, \theta_k \):

\[
p(\lambda_l|y) \propto \exp \Delta^{(t)}_{\lambda_l} = \exp \left( \log p(\lambda_l) + \sum_{\Omega_n \in \Gamma(\lambda_l)} \Delta^{(t)}_{\Omega_n \rightarrow \lambda_l} \right) \tag{12}
\]
\[
p(\theta_k|y) \propto \exp \Delta^{(t)}_{\theta_k} = \exp \left( \log p(\theta_k) + \sum_{\Phi_m \in \Gamma(\theta_k)} \Delta^{(t)}_{\Phi_m \rightarrow \theta_k} \right) \tag{13}
\]

### 2.2 Max-sum Message Passing

For the max-sum message passing, the message updates from the variable nodes to the factor nodes are the same as the aforementioned sum-product message passing, i.e. (3.5, 9.11). We only need to change the message updates from the factor nodes to the variable nodes by replacing \( \int \) with \( \max \):

\[
\Delta^{(t+1)}_{\phi_m \rightarrow x_n} = \max_{x_n, \theta} \log \Phi_m (y_m, x, \theta) + \sum_{j \neq n} \Delta^{(t)}_{\phi_m \leftarrow x_j} + \sum_v \Delta^{(t)}_{\phi_m \leftarrow \theta_v} \tag{14}
\]
\[
\Delta^{(t+1)}_{\Omega_n \rightarrow \lambda_l} = \max_{x_n, \lambda \setminus \lambda_l} \log \Omega_n (x_n, \lambda) + \Delta^{(t)}_{\Omega_n \leftarrow \lambda_l} + \sum_{u \neq l} \Delta^{(t)}_{\Omega_n \leftarrow \lambda_u} \tag{15}
\]
\[
\Delta^{(t+1)}_{\phi_m \rightarrow \theta_k} = \max_{\theta, \theta_k, x} \log \Phi_m (y_m, x, \theta) + \sum_j \Delta^{(t)}_{\phi_m \leftarrow x_j} + \sum_{v \neq k} \Delta^{(t)}_{\phi_m \leftarrow \theta_v} \tag{17}
\]

The MAP estimate of \( x \) is then:

\[
\hat{x}_n = \arg \max_{x_n} \Delta^{(t)}_{x_n} = \arg \max_{x_n} \Delta^{(t)}_{\Omega_n \rightarrow x_n} + \sum_{\phi_m \in \Gamma(x_n)} \Delta^{(t)}_{\phi_m \rightarrow x_n} \tag{18}
\]
3 MAP Parameter Estimation for Sum-product Message Passing

In this paper we focus on using the *sum-product* message passing algorithm for sparse signal recovery. Take (2) in *sum-product* message passing for example, it contains integrations with respect to the parameters $\theta$, and they might be difficult to compute. However, the posterior marginals of $\theta$ can be directly estimated in (12). Instead of doing a complete integration w.r.t $\theta$, we can use the following scalar MAP estimate of $\theta$ to simplify the message updates:

$$\hat{\theta}^{(t)}_k = \arg \max_{\theta_k} p(\theta_k | y) = \arg \max_{\theta_k} \Delta^{(t)}_{\theta_k}.$$  

(19)

Similarly, we also have the MAP estimate of $\lambda_l$ as follows:

$$\hat{\lambda}^{(t)}_l = \arg \max_{\lambda_l} p(\lambda_l | y) = \arg \max_{\lambda_l} \Delta^{(t)}_{\lambda_l}.$$  

(20)

The GAMP algorithm with MAP parameter estimation can be summarized in Algorithm 1.

**Algorithm 1** Sum-product GAMP with MAP parameter estimation

```
Require: y, A, p(\lambda), p(x|\lambda), p(\theta), p(w|\theta)
1: Initialize \(\lambda^{(0)}, \theta^{(0)}\)
2: for \(t = \{1, 2, \cdots\}\) do
3:  Perform GAMP [8];
4:  Compute \(\hat{\lambda}^{(t)}_l\) for \(l = 1, \cdots, L\);
5:  Compute \(\hat{\theta}^{(t)}_k\) for \(k = 1, \cdots, K\);
6:  Compute the MMSE estimate \(\hat{x}^{(t)} = \mathbb{E}[x | y]\);
7:  if \(\hat{x}^{(t)}\) reaches convergence then
8:    \(\hat{x} = \hat{x}^{(t)}\);
9:    break;
10: end if
11: end for
12: return Output \(\hat{x}\);
```

**Discussion:** The EM-GAMP [13, 14] and the adaptive-GAMP [15] are both ML estimations. Specifically, EM-GAMP tries to maximize $\mathbb{E}[\log p(x, w; \lambda, \theta) | y, \lambda^{(t)}, \theta^{(t)}]$ iteratively using the Expectation-Maximization (EM) algorithm [16]; adaptive-GAMP tries to maximize the log-likelihood of two new random variables $r, p$ introduced in the original GAMP: $\log p(r | \lambda), \log p(y, p | \theta)$. For the proposed MAP parameter estimation, we have the following using Bayes’ rule:

$$p(\lambda | y) \propto p(y | \lambda)p(\lambda)$$  

(21a)

$$p(\theta | y) \propto p(y | \theta)p(\theta).$$  

(21b)

If the priors $p(\lambda), p(\theta)$ are chosen to be uniform distributions, the MAP estimation and ML estimation would have the same solutions since they were both maximizing $p(y | \lambda)$ and $p(y | \theta)$. 


For the state evolution analysis, we make the same assumptions as [8, 11, 15] about the PE-GAMP algorithm. In general, the estimation problem is indexed by the dimensionality \( N \) of \( x \), and the output dimensionality \( M \) satisfies \( \lim_{N \to \infty} N/M = \beta \) for some \( \beta > 0 \).

Specifically, (19, 20) can be viewed as different adaptation functions defined in [15]. The empirical convergence of \( x, \lambda, \theta \) can then be obtained directly from the conclusion of Theorem 2 in [15]. Following the analysis in [15], we also assume the MAP adaptation functions \( \arg \max_\lambda p(\lambda|y) \), \( \arg \max_\theta p(\theta|y) \) satisfy the weak pseudo-Lipschitz continuity property [11] over \( y \). Using Theorem 3 in [15], we can see that the MAP estimations converge to the true parameters as \( N \to \infty \) when they are computed exactly.

### 3.1 MAP Estimation of \( \lambda, \theta \)

We next show how to compute the MAP estimates in (19, 20). For clarification purposes, we will remove the superscript \( ^{(t)} \) from the notations. Starting with some initial solutions \( \lambda_l(0), \theta_k(0) \), we will maximize the quadratic approximations of \( \Delta \lambda_l, \Delta \theta_k \) iteratively. Take \( \Delta \lambda_l \) for example, it is approximated as follows in the \((h+1)\)-th iteration:

\[
\Delta \lambda_l \approx \Delta \lambda_l(h) + \Delta' \lambda_l(h) (\lambda_l - \lambda_l(h)) + \frac{1}{2} \Delta'' \lambda_l(h) (\lambda_l - \lambda_l(h))^2 ,
\]

(22)

where \( \lambda_l(h) \) is the solution in the \( h \)-th iteration. If \( \Delta'' \lambda_l(h) < 0 \), we can get the following \( \lambda_l(h+1) \) using Newton’s method:

\[
\lambda_l(h+1) = \lambda_l(h) - \frac{\Delta' \lambda_l(h)}{\Delta'' \lambda_l(h)}.
\]

(23)

If \( \Delta'' \lambda_l(h) > 0 \), Newton’s method will produce local minimum solution. In this case we will use the line search method. This iterative approach can be summarized in Algorithm 2. The MAP estimation \( \theta_k \) can also be computed in a similar way.

**Algorithm 2** MAP parameter estimation

**Require:** \( \Delta \lambda_l \)

1: Initialize \( \lambda_l(0) \)

2: for \( h = \{0, 1, \ldots\} \) do

3: Compute \( \Delta' \lambda_l(h), \Delta'' \lambda_l(h) \);

4: if \( \Delta'' \lambda_l(h) < 0 \) then \( \lambda_l(h+1) = \lambda_l(h) - \frac{\Delta' \lambda_l(h)}{\Delta'' \lambda_l(h)} \) end if;

5: if \( \Delta'' \lambda_l(h) \geq 0 \) then Perform line search for \( \lambda_l(h+1) \) end if;

6: if \( \lambda_l(h+1) \) reaches convergence then

7: \( \lambda_l = \lambda_l(h+1) \);

8: break;

9: end if

10: end for

11: return Output \( \lambda_l \);
3.2 MAP estimation for Special Input and Output Channels

We then give the following three examples on how to perform MAP estimation of the parameters in the input and output channels.

3.2.1 Bernoulli-Gaussian Input Channel

In this case, the sparse signal $x$ can be modeled as a mixture of Bernoulli and Gaussian distribution:

$$p(x_j|\lambda) = (1 - \lambda_1)\delta(x_j) + \lambda_1\mathcal{N}(x_j; \lambda_2, \lambda_3),$$  \hfill (24)

where $\delta(\cdot)$ is Dirac delta function, $\lambda_1 \in [0, 1]$ is the sparsity rate, $\lambda_2 \in \mathbb{R}$ is the nonzero coefficient mean and $\lambda_3 \in \mathbb{R}^+$ is the nonzero coefficient variance.

The AMP algorithm \cite{8,9} uses quadratic approximation of the loopy BP. In the $(t+1)$-th iteration, we have the following from \cite{8}:

$$\Delta_{\Omega_n \leftarrow x_n} \approx -\frac{1}{2\tau_{\Omega_n \leftarrow x_n}}(x_n - x_n^{(t)})^2 + \text{const},$$  \hfill (25)

where $\tau_{\Omega_n \leftarrow x_n}$, $x_n^{(t)}$ “correspond” to the variance and mean of $x_n$ in the $t$-th iteration respectively. Take $\lambda_1$ for example, $\Delta_{\Omega_n \rightarrow \lambda_1}^{(t+1)}$ can be computed as follows:

$$c(x_n) = \exp \left(-\frac{|x_n^{(t)}|^2}{2\tau_{\Omega_n \leftarrow x_n}}\right)$$  \hfill (26)

$$d(x_n) = \sqrt{\frac{\tau_{\Omega_n \leftarrow x_n}}{\lambda_3^{(t)} + \tau_{\Omega_n \leftarrow x_n}}} \exp \left(-\frac{(\lambda_2^{(t)} - x_n^{(t)})^2}{\lambda_3^{(t)} + \tau_{\Omega_n \leftarrow x_n}}\right)$$  \hfill (27)

$$\Delta_{\Omega_n \rightarrow \lambda_1}^{(t+1)} = \log ((1 - \lambda_1)c(x_n) + \lambda_1d(x_n)) + \text{const},$$  \hfill (28)

where $\lambda_2^{(t)}$, $\lambda_3^{(t)}$ are the MAP estimations in the $t$-th iteration. $\Delta_{\Omega_n \rightarrow \lambda_1}^{(t+1)}$ can then be computed accordingly for the MAP estimation of $\lambda_1^{(t+1)}$ in the $(t+1)$-th iteration. $\lambda_2^{(t+1)}$, $\lambda_3^{(t+1)}$ can be updated similarly.

3.2.2 Laplace Input Channel

Laplace input channel assumes the signal $x$ follows the Laplace distribution:

$$p(x_j|\lambda) = \frac{\lambda_1}{2} \exp (-\lambda_1|x_j|)$$  \hfill (29)

where $\lambda_1 \in (0, \infty)$. We can compute $\Delta_{\Omega_n \rightarrow \lambda_1}^{(t+1)}$ similarly using the quadratic approximation of $\Delta_{\Omega_n \leftarrow x_n}$ from \cite{8}:

$$c(x_n) = \frac{\lambda_1}{2} \exp \left(\frac{\tau_{\Omega_n \leftarrow x_n}\lambda_2^{(t)} - 2x_n^{(t)}\lambda_1}{2}\right) \sqrt{\frac{\tau_{\Omega_n \leftarrow x_n}\pi}{2}} \left(1 - \text{erf} \left(\frac{-x_n^{(t)} - \lambda_1\tau_{\Omega_n \leftarrow x_n}}{\sqrt{2\tau_{\Omega_n \leftarrow x_n}}}\right)\right)$$  \hfill (30)
\[
d(x_n) = \frac{\lambda_1}{2} \exp \left( \frac{\tau_{\Omega_n \leftarrow x_n} \lambda_1^2 + 2x_n(t) \lambda_1}{2} \right) \sqrt{\frac{\tau_{\Omega_n \leftarrow x_n} \pi}{2}} \left( 1 + \text{erf} \left( \frac{-x_n(t) + \lambda_1 \tau_{\Omega_n \leftarrow x_n}}{\sqrt{2t \tau_{\Omega_n \leftarrow x_n}}} \right) \right)
\]

\[
\Delta^{(t+1)}_{\Omega_n \rightarrow \lambda_1} = \log(c(x_n) + d(x_n)) + \text{const},
\]

where \(\text{erf}(\cdot)\) is the error function. \(\Delta^{(t+1)}_{\lambda_1}\) can then be computed accordingly for the MAP estimation of \(\lambda_1^{(t+1)}\) in the \((t+1)\)-th iteration.

### 3.2.3 Additive White Gaussian Noise Output Channel

AWGN output channel assumes the noise \(w\) is white Gaussian noise:

\[
p(w_i|\theta) = \mathcal{N}(w_i; 0, \theta_1),
\]

where \(\theta_1 \in \mathbb{R}^+\) is the noise variance. Using the quadratic approximation from \([8]\), we can get:

\[
\Delta^{(t+1)}_{\Phi_m \rightarrow \theta_1} = -\frac{1}{2} \log \left( \theta_1 + \tau_{\Phi_m}^p \right) - \frac{1}{2} \frac{(y_m - p_{\Phi_m}^{(t)})^2}{\left( \theta_1 + \tau_{\Phi_m}^p \right)} + \text{const},
\]

where \(\tau_{\Phi_m}^p, p_{\Phi_m}^{(t)}\) are from \([8]\) and “correspond” to the variance and mean of \(z_m\) in the \(t\)-th iteration respectively. \(\Delta^{(t+1)}_{\theta_1}\) can then be computed accordingly for the MAP estimation of \(\theta_1^{(t+1)}\) in the \((t+1)\)-th iteration.

### 4 Numerical Experiments

In this section, we use the proposed GAMP algorithm with built-in parameter estimation (PE-GAMP) to recover sparse signal \(x\) from the under-sampled measurement \(y = Ax + w\). Specifically, we will use BG input channel and AWGN output channel to generate \(x, y\). Since we don’t have any knowledge about the priors of \(\lambda, \theta\), we will fairly choose the uniform prior for each parameter. The recovered signal \(\hat{x}\) is then computed using the MMSE estimator \(E[x|y]\) from sum-product message passing, and we will compare the recovery performance with the oracle-GAMP that knows the true parameters.

#### 4.1 Noiseless Sparse Signal Recovery

We first perform noiseless sparse signal recovery experiments and draw the empirical phase transition curves (PTC) of PE-GAMP, oracle-GAMP. We fix \(N = 1000\) and vary the over-sampling ratio \(\sigma = \frac{M}{N} \in [0.05, 0.1, 0.15, \ldots, 0.95]\) and the under-sampling ratio \(\rho = \frac{S}{M} \in [0.05, 0.1, 0.15, \ldots, 0.95]\), where \(S\) is the sparsity of the signal, i.e. the number of nonzero coefficients. For each combination of \(\sigma\) and \(\rho\), we randomly generate 100 pairs of \(\{x, A\}\). The entries of \(A\) and the nonzero entries of \(x\) are i.i.d. Gaussian \(\mathcal{N}(0,1)\). Given the measurement vector \(y = Ax\) and the sensing matrix \(A\), we try to recover the sparse signal \(x\). If \(\epsilon = \|x - \hat{x}\|_2/\|x\|_2 < 10^{-3}\), the recovery is considered to be a success. Based on the 100 trials, we compute the success rate for each combination of \(\sigma\) and \(\rho\).
The absolute difference of the success rates between PE-GAMP and oracle-GAMP are shown in Fig. 3(a). The PTC curve is the contour that correspond to the 0.5 success rate in the domain \((\sigma, \rho) \in (0, 1)^2\), it divides the domain into a “success” phase (lower right) and a “failure” phase (upper left). The PTC curves of the two GAMP methods along with the theoretic Lasso [9] are shown in Fig. 3(b). We can see from Fig. 3 that the performance of PE-GAMP generally matches that of the oracle-GAMP. PE-GAMP is able to estimate the parameters fairly well while recovering the sparse signals.

4.2 Noisy Sparse Signal Recovery

We next try to recover the sparse signal \(x\) from a noisy measurement vector \(y\). In this case, we would like to see how the two algorithms behave when an increasing amount of noise is added to the measurement. Specifically, \(S = 100, M = 500, N = 1000\) are fixed, and \(y\) is generated as follows:

\[
y = Ax + \nu w ,
\]

where we choose \(\nu \in \{0.1, 0.2, \cdots, 1\}\). For each \(\nu\), we randomly generate 100 triples of \(\{x, A, w\}\). The entries of \(A, w\) and the nonzero entries of \(x\) are i.i.d Gaussian \(\mathcal{N}(0, 1)\).

\(\epsilon = \|x - \hat{x}\|_2/\|x\|_2\) is used to evaluate the performances of the algorithms. The mean \pm standard deviation of the \(\epsilon\)s from the 100 trials are shown in Fig. 4. We can see that the proposed PE-GAMP is able to perform as well as the oracle-GAMP in recovering noisy sparse signals.
Figure 4: Comparison on the recovery of noisy sparse signals at different noise levels \( \nu \), the error bar represents the mean \( \pm \) standard deviation of the \( \epsilon \).

5 Conclusion and Future Work

In this paper we proposed the GAMP algorithm with built-in parameter estimation to recover under-sampled sparse signals. The parameters are treated as random variables with pre-specified priors, their posterior marginals can then be directly approximated by loopy belief propagation. This allows us to perform MAP estimation of the parameters and update to the recovered signals jointly. Following the same assumptions made by the original GAMP, the proposed PE-GAMP converges empirically to the true parameter values, as is evident from a series of noiseless and noisy sparse signal recovery experiments. We have mainly focused on the GAMP algorithm based on sum-product message passing here. In the future we would like to include the max-sum message passing and conduct analysis on its convergence and asymptotic consistency behavior.

References

[1] E. Candès and T. Tao, “Decoding by linear programming,” IEEE Trans. on Information Theory, vol. 51(12), pp. 4203–4215, 2005.

[2] E.J. Candes, J. Romberg, and T. Tao, “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,” IEEE Trans. on Information Theory, vol. 52(2), pp. 489–509, 2006.

[3] D.L. Donoho, “Compressed sensing,” IEEE Trans. on Information Theory, vol. 52, no. 4, pp. 1289–1306, 2006.

[4] E.J. Candès and T. Tao, “Near-optimal signal recovery from random projections: Universal encoding strategies?,” IEEE Trans. on Information Theory, vol. 52, no. 12, pp. 5406–5425, 2006.
[5] M. Aharon, M. Elad, and A. Bruckstein, “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[6] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[7] Robert Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society*, vol. 58, pp. 267–288, 1994.

[8] S. Rangan, “Generalized approximate message passing for estimation with random linear mixing,” in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, July 2011, pp. 2168–2172.

[9] D.L. Donoho, A. Maleki, and A. Montanari, “Message-passing algorithms for compressed sensing,” *Proc. Nat. Acad. Sci.*, vol. 106, no. 45, pp. 18914–18919, 2009.

[10] D.L. Donoho, A. Maleki, and A. Montanari, “Message-passing algorithms for compressed sensing: I. motivation and construction,” *Proc. Inform. Theory Workshop*, Jan 2010.

[11] M. Bayati and A. Montanari, “The dynamics of message passing on dense graphs, with applications to compressed sensing,” *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 764–785, Feb 2011.

[12] M. J. Wainwright and M. I. Jordan, “Graphical models, exponential families, and variational inference,” *Found. Trends Mach. Learn.*, vol. 1, no. 1-2, Jan. 2008.

[13] J. Vila and P. Schniter, “Expectation-maximization bernoulli-gaussian approximate message passing,” in *Conf. Rec. 45th Asilomar Conf. Signals, Syst. Comput*, Nov 2011, pp. 799–803.

[14] J. P. Vila and P. Schniter, “Expectation-maximization gaussian-mixture approximate message passing,” *IEEE Transactions on Signal Processing*, vol. 61, no. 19, pp. 4658–4672, Oct 2013.

[15] U. S. Kamilov, S. Rangan, A. K. Fletcher, and M. Unser, “Approximate message passing with consistent parameter estimation and applications to sparse learning,” *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2969–2985, May 2014.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.