Recent Advances in Approximate Message Passing

Phil Schniter

THE OHIO STATE UNIVERSITY  Duke  iid

Collaborators: Sundeep Rangan (NYU), Alyson Fletcher (UCLA), Mark Borgerding (OSU)

Supported in part by NSF grants IIP-1539960 and CCF-1527162.

SPARS — June 8, 2017
Overview

1. Linear Regression, AMP, and Vector AMP (VAMP)
2. VAMP, ADMM, and Convergence in the Convex Setting
3. VAMP Convergence in the Non-Convex Setting
4. VAMP for Inference
5. EM-VAMP and Adaptive VAMP
6. Plug-and-play VAMP & Whitening
7. VAMP as a Deep Neural Network
8. VAMP for the Generalized Linear Model
Outline

1. Linear Regression, AMP, and Vector AMP (VAMP)
2. VAMP, ADMM, and Convergence in the Convex Setting
3. VAMP Convergence in the Non-Convex Setting
4. VAMP for Inference
5. EM-VAMP and Adaptive VAMP
6. Plug-and-play VAMP & Whitening
7. VAMP as a Deep Neural Network
8. VAMP for the Generalized Linear Model
# The Linear Regression Problem

Consider the following linear regression problem:

<table>
<thead>
<tr>
<th>Recover $x_o$ from $y = Ax_o + w$ with</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_o \in \mathbb{R}^N$ unknown signal</td>
</tr>
<tr>
<td>$A \in \mathbb{R}^{M \times N}$ known linear operator</td>
</tr>
<tr>
<td>$w \in \mathbb{R}^M$ white Gaussian noise.</td>
</tr>
</tbody>
</table>

Typical methodologies:

1. Regularized loss minimization (or MAP estimation):
   $$\hat{x} = \arg \min_x \frac{\theta_2}{2} \|Ax - y\|_2^2 + R(x; \theta_1)$$

2. Approximate MMSE:
   $$\hat{x} \approx \mathbb{E}\{x|y\} \text{ for } x \sim p(x; \theta_1), \; y \sim \mathcal{N}(Ax, I/\theta_2)$$

3. Plug-and-play: iteratively apply a denoising algorithm like BM3D

4. Train a deep network to recover $x_o$ from $y$. 
The AMP Methodology

- All of the aforementioned methodologies can be addressed using the Approximate Message Passing (AMP) framework.¹

- AMP tackles these difficult global optimization/inference problems through a sequence of simpler local optimization/inference problems.

- It does this by appropriate definition of a denoiser \( g_1(\cdot; \gamma, \theta_1) : \mathbb{R}^N \rightarrow \mathbb{R}^N : \)

  - Optimization: \( g_1(r; \gamma, \theta_1) = \arg \min_x R(x; \theta_1) + \frac{\gamma}{2} \|x - r\|_2^2 \triangleq \text{"prox}_{R/\gamma}(r)" \)
  
  - MMSE: \( g_1(r; \gamma, \theta_1) = \mathbb{E} \{ x \mid r = x + \mathcal{N}(0, I/\gamma) \} \)
  
  - Plug-and-play:\(^2\) \( g_1(r; \gamma, \theta_1) = \text{BM3D}(r, 1/\gamma) \)
  
  - Deep network: \( g_1(r; \gamma, \theta_1) \) is learned.

¹Donoho, Maleki, Montanari’09, ²Metzler, Maleki, Baraniuk’14
AMP: the good, the bad, and the ugly

The good:

- **With large i.i.d. sub-Gaussian $\mathbf{A}$**, AMP performs provably$^3$ well, in that it can be rigorously characterized by a scalar state-evolution (SE). When this SE has a unique fixed point, AMP converges to the Bayes optimal solution.

- **Empirically**, AMP behaves well with many other “sufficiently random” $\mathbf{A}$ (e.g., randomly sub-sampled Fourier $\mathbf{A}$ & i.i.d. sparse $\mathbf{x}$).

The bad:

- **With general $\mathbf{A}$**, AMP gives **no guarantees**.

The ugly:

- **With some $\mathbf{A}$**, AMP may **fail to converge!** (e.g., ill-conditioned or non-zero-mean $\mathbf{A}$)

---

$^3$Bayati, Montanari’15, Bayati, Lelarge, Montanari’15
The Vector AMP (VAMP) Algorithm

Take \( SVD \ A = U \text{Diag}(s)V^T \), choose \( \zeta \in (0, 1] \) and Lipschitz \( g_1(\cdot; \gamma_1, \theta_1) : \mathbb{R}^N \to \mathbb{R}^N \).

Initialize \( r_1, \gamma_1 \).

For \( k = 1, 2, 3, \ldots \)

\[
\hat{x}_1 \leftarrow g_1(r_1; \gamma_1, \theta_1) \quad \text{denoising of } r_1 = x_o + \mathcal{N}(0, I/\gamma_1)
\]

\[
\eta_1 \leftarrow \gamma_1 N / \text{tr} \left[ \frac{\partial g_1(r_1; \gamma_1, \theta_1)}{\partial r_1} \right]
\]

\[
r_2 \leftarrow (\eta_1 \hat{x}_1 - \gamma_1 r_1) / (\eta_1 - \gamma_1)
\]

\[
\gamma_2 \leftarrow \eta_1 - \gamma_1 \quad \text{Onsager correction}
\]

\[
\hat{x}_2 \leftarrow g_2(r_2; \gamma_2, \theta_2) \quad \text{LMMSE estimate } x \sim \mathcal{N}(r_2, I/\gamma_2)
\]

from \( y = Ax + \mathcal{N}(0, I/\theta_2) \)

\[
\eta_2 \leftarrow \gamma_2 N / \text{tr} \left[ \frac{\partial g_2(r_2; \gamma_2, \theta_2)}{\partial r_2} \right]
\]

\[
r_1 \leftarrow \zeta (\eta_2 \hat{x}_2 - \gamma_2 r_2) / (\eta_2 - \gamma_2) + (1 - \zeta) r_1 \quad \text{Onsager correction}
\]

\[
\gamma_1 \leftarrow \zeta (\eta_2 - \gamma_2) + (1 - \zeta) \gamma_1 \quad \text{damping}
\]

where \( g_2(r_2; \gamma_2, \theta_2) = V \left( \theta_2 \text{Diag}(s)^2 + \gamma_2 I \right)^{-1} \left( \theta_2 \text{Diag}(s)U^T y + \gamma_2 V^T r_2 \right) \)

\[
\eta_2 = \frac{1}{N} \sum_{n=1}^{N} (\theta_2 s_n^2 + \gamma_2)^{-1}
\]

two mat-vec multis per iteration!
Outline

1. Linear Regression, AMP, and Vector AMP (VAMP)
2. VAMP, ADMM, and Convergence in the Convex Setting
3. VAMP Convergence in the Non-Convex Setting
4. VAMP for Inference
5. EM-VAMP and Adaptive VAMP
6. Plug-and-play VAMP & Whitening
7. VAMP as a Deep Neural Network
8. VAMP for the Generalized Linear Model
**PRS-ADMM**

- Consider the optimization problem

  $$\arg \min_x f_1(x) + f_2(x) \text{ with, e.g., } \begin{cases} f_1(x) = -\log p(x; \theta_1) \\ f_2(x) = \frac{\theta_2}{2} \|Ax - y\|^2 \end{cases}$$

  and define the augmented Lagrangian

  $$L_\gamma(x_1, x_2, s) = f_1(x_1) + f_2(x_2) + s^T(x_1 - x_2) + \frac{\gamma}{2} \|x_1 - x_2\|^2.$$ 

- An ADMM variant (via Peaceman-Rachford splitting on the dual) is

  $$\hat{x}_1 \leftarrow \arg \min_{x_1} L_\gamma(x_1, \hat{x}_2, s)$$

  $$s \leftarrow s + \gamma(\hat{x}_1 - \hat{x}_2)$$

  $$\hat{x}_2 \leftarrow \arg \min_{x_2} L_\gamma(\hat{x}_1, x_2, s)$$

  $$s \leftarrow s + \gamma(\hat{x}_1 - \hat{x}_2)$$

- PRS-ADMM has weaker convergence guarantees than standard ADMM, but is supposedly faster.
VAMP Connections to PRS-ADMM

- Now consider VAMP applied to the same optimization problem, but with \( \gamma_1 = \gamma_2 \triangleq \gamma \) enforced at each iteration. Also, define

\[
s_i \triangleq \gamma (\hat{x}_i - r_i) \quad \text{for} \quad i = 1, 2.
\]

- This \( \gamma \)-forced VAMP manifests as

\[
\begin{align*}
\hat{x}_1 & \leftarrow \arg \min_{x_1} L_\gamma(x_1, \hat{x}_2, s_1) \\
s_2 & \leftarrow s_1 + \gamma (\hat{x}_1 - \hat{x}_2) \\
\hat{x}_2 & \leftarrow \arg \min_{x_2} L_\gamma(\hat{x}_1, x_2, s_2) \\
s_1 & \leftarrow s_2 + \gamma (\hat{x}_1 - \hat{x}_2)
\end{align*}
\]

which is identical to Peaceman-Rachford ADMM.

- The full VAMP algorithm adapts \( \gamma_1 \) and \( \gamma_2 \) on-the-fly according to the local curvature of the cost function.
Example of VAMP applied to the LASSO Problem

Solving LASSO to reconstruct 40-sparse \( \mathbf{x} \in \mathbb{R}^{1000} \) from noisy \( \mathbf{y} \in \mathbb{R}^{400} \).

\[
\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \| \mathbf{y} - A \mathbf{x} \|_2^2 + \lambda \| \mathbf{x} \|_1.
\]
VAMP Convergence in the Convex Setting

- Consider arbitrary $A$.

- A double-loop version of VAMP \textit{globally converges} to a unique minimum when the Jacobian of the denoiser $g_1$ is bounded as:

$$
\exists c_1, c_2 > 0 \text{ s.t. } \frac{\gamma}{\gamma + c_1} I \leq \frac{\partial g_1(r, \gamma)}{\partial r} \leq \frac{\gamma}{\gamma + c_2} I,
$$

as occurs in optimization-VAMP under \textit{strictly convex} regularization $R(\cdot; \theta_1)$.

- For convergence, it suffices to choose the \textit{damping parameter} $\zeta \in (0, 1]$ as

$$
\zeta \leq \frac{2 \min\{\gamma_1, \gamma_2\}}{\gamma_1 + \gamma_2}.
$$

Thus

- the damping parameter $\zeta$ can be adapted using $\gamma_1, \gamma_2$, and
- damping is not needed (i.e., $\zeta = 1$ suffices) if $\gamma_1 = \gamma_2$. 
Outline

1. Linear Regression, AMP, and Vector AMP (VAMP)
2. VAMP, ADMM, and Convergence in the Convex Setting
3. VAMP Convergence in the Non-Convex Setting
4. VAMP for Inference
5. EM-VAMP and Adaptive VAMP
6. Plug-and-play VAMP & Whitening
7. VAMP as a Deep Neural Network
8. VAMP for the Generalized Linear Model
VAMP State Evolution

- Suppose the denoiser $g_1(\cdot)$ has identical scalar components $g_1(\cdot)$, where $g_1$ and $g'_1$ are Lipschitz.

- Suppose that $A$ is right-rotationally invariant, in that its SVD
  \[ A = USV^T \]
  has Haar $V$ (i.e., uniformly distributed over the set of orthogonal matrices). Since $U$ and $S$ are arbitrary, this includes iid Gaussian $A$ as a special case.

- In the large-system limit, one can prove\(^4\) that VAMP is rigorously characterized by a scalar state-evolution (using techniques inspired by Bayati-Montanari’10).

  This state-evolution establishes

  1. the convergence of VAMP in the non-convex setting,
  2. the correctness of the denoising model $r_1 = x_o + \mathcal{N}(0, I/\gamma_1)$.

\(^4\)Rangan, Schniter, Fletcher’16
VAMP state evolution

Assuming empirical convergence of \( \{s_j\} \rightarrow S \) and \( \{(r_{1,j}^0, x_{o,j})\} \rightarrow (R^0_1, X_o) \) and Lipschitz continuity of \( g \) and \( g' \), the VAMP state-evolution under \( \hat{\tau}_w = \tau_w \) is as follows:

for \( t = 0, 1, 2, \ldots \)

\[
\begin{align*}
\mathcal{E}_1^t &= E \{ [g(X_o + \mathcal{N}(0, \tau^t_1); \gamma^t_1) - X_o]^2 \} & \text{MSE} \\
\overline{\alpha}_1^t &= E \{ g'(X_o + \mathcal{N}(0, \tau^t_1); \gamma^t_1) \} & \text{divergence} \\
\gamma^t_2 &= \gamma_1^t \frac{1-\overline{\alpha}_1^t}{\overline{\alpha}_1^t}, \quad \tau^t_2 = \frac{1}{(1-\overline{\alpha}_1^t)^2} [\mathcal{E}_1^t - (\overline{\alpha}_1^t)^2 \tau^t_1] \\
\mathcal{E}_2^t &= E \{ [S^2/\tau_w + \gamma^t_2]^{-1} \} & \text{MSE} \\
\overline{\alpha}_2^t &= \gamma_2^t E \{ [S^2/\tau_w + \gamma^t_2]^{-1} \} & \text{divergence} \\
\gamma^{t+1}_1 &= \gamma_2^t \frac{1-\overline{\alpha}_2^t}{\overline{\alpha}_2^t}, \quad \tau^{t+1}_1 = \frac{1}{(1-\overline{\alpha}_2^t)^2} [\mathcal{E}_2^t - (\overline{\alpha}_2^t)^2 \tau^t_2] \\
\end{align*}
\]

More complicated expressions for \( \mathcal{E}_2^t \) and \( \overline{\alpha}_2^t \) exist for the case when \( \hat{\tau}_w \neq \tau_w \).
Outline

1. Linear Regression, AMP, and Vector AMP (VAMP)
2. VAMP, ADMM, and Convergence in the Convex Setting
3. VAMP Convergence in the Non-Convex Setting
4. VAMP for Inference
5. EM-VAMP and Adaptive VAMP
6. Plug-and-play VAMP & Whitening
7. VAMP as a Deep Neural Network
8. VAMP for the Generalized Linear Model
Now consider VAMP applied to the “inference” or “MMSE” problem.

- assume a prior $p(x; \theta_1)$,
- choose the denoiser as $g_1(r_1; \gamma_1, \theta_1) = \mathbb{E}\{x \mid r_1 = x + \mathcal{N}(0, I/\gamma_1)\}$.

What is the corresponding cost function in this case?

What can we say about convergence and performance?

Can we tune the hyperparameters $\theta = [\theta_1, \theta_2]$ if they are unknown?
Variational Inference

- Ideally, we would like to compute the exact posterior density
  \[
p(x|y) = \frac{p(x; \theta_1)\ell(x; \theta_2)}{Z(\theta)} \quad \text{for} \quad Z(\theta) \triangleq \int p(x; \theta_1)\ell(x; \theta_2) \, dx,
\]
  but the high-dimensional integral in \( Z(\theta) \) is difficult to compute.

- We can avoid computing \( Z(\theta) \) through variational optimization:
  \[
p(x|y) = \arg \min_b D(b(x)\|p(x|y)) \quad \text{where} \quad D(\cdot\|\cdot) \text{ is KL divergence}
  \]
  \[
  = \arg \min_b D(b(x)\|p(x; \theta_1)) + D(b(x)\|\ell(x; \theta_2)) + H(b(x))
  \]
  Gibbs free energy
  \[
  = \arg \min_{b_1, b_2, q} D(b_1(x)\|p(x; \theta_1)) + D(b_2(x)\|\ell(x; \theta_2)) + H(q(x))
  \]
  \[
  \text{s.t.} \quad b_1 = b_2 = q, \quad \triangleq J_{\text{Gibbs}}(b_1, b_2, q; \theta)
  \]
  but the density constraint keeps the problem difficult.
Expectation Consistent Approximation

- In expectation-consistent approximation (EC)\(^5\), the density constraint is relaxed to moment-matching constraints:

\[
p(x|y) \approx \arg \min_{b_1, b_2, q} J_{\text{Gibbs}}(b_1, b_2, q; \theta)
\]

\[
\text{s.t. } \begin{cases} E\{x|b_1\} = E\{x|b_2\} = E\{x|q\} \\ \text{tr}(\text{Cov}\{x|b_1\}) = \text{tr}(\text{Cov}\{x|b_2\}) = \text{tr}(\text{Cov}\{x|q\}). \end{cases}
\]

- The stationary points of EC are the densities

\[
b_1(x) \propto p(x; \theta_1)N(x; r_1, I/\gamma_1)
\]

\[
b_2(x) \propto \ell(x; \theta_2)N(x; r_2, I/\gamma_2)
\]

\[
q(x) = N(x; \hat{x}, I/\eta)
\]

s.t.

\[
\begin{cases} E\{x|b_1\} = E\{x|b_2\} = \hat{x} \\ \text{tr}(\text{Cov}\{x|b_1\}) = \text{tr}(\text{Cov}\{x|b_2\}) = N\eta, \end{cases}
\]

where VAMP iteratively solves for the quantities \(r_1, \gamma_1, r_2, \gamma_2, \hat{x}, \eta\).

- For large right-rotationally invariant \(A\), the these stationary points are “good” in that \(\text{MSE}(\hat{x})\) matches the MMSE predicted by the replica method.\(^6\)\(^7\)

---

\(^5\)Opper, Winther'04, \(^6\)Kabashima, Vehkaperä'14, \(^7\)Fletcher, Sahraee, Rangan, Schniter'16
The VAMP Algorithm for Inference

When applied to inference, the VAMP algorithm manifests as

Initialize \( r_1, \gamma_1 \).

For \( k = 1, 2, 3, \ldots \)

\[
\hat{x}_1 \leftarrow g_1(r_1; \gamma_1, \theta_1) \quad \text{MMSE estimate of } x \sim p(x; \theta_1) \text{ from } r_1 = x + N(0, I/\gamma_1)
\]

\[
\eta_1 \leftarrow \gamma_1 N / \text{tr} \left[ \frac{\partial g_1(r_1; \gamma_1, \theta_1)}{\partial r_1} \right] \quad \text{posterior precision}
\]

\[
r_2 \leftarrow (\eta_1 \hat{x}_1 - \gamma_1 r_1) / (\eta_1 - \gamma_1)
\]

\[
\gamma_2 \leftarrow \eta_1 - \gamma_1
\]

\[
\hat{x}_2 \leftarrow g_2(r_2; \gamma_2, \theta_2) \quad \text{LMMSE estimate of } x \sim N(r_2, I/\gamma_2) \text{ from } y = Ax + N(0, I/\theta_2)
\]

\[
\eta_2 \leftarrow \gamma_2 N / \text{tr} \left[ \frac{\partial g_2(r_2; \gamma_2, \theta_2)}{\partial r_2} \right] 
\]

\[
r_1 \leftarrow \zeta(\eta_2 \hat{x}_2 - \gamma_2 r_2) / (\eta_2 - \gamma_2) + (1 - \zeta) r_1
\]

\[
\gamma_1 \leftarrow \zeta(\eta_2 - \gamma_2) + (1 - \zeta) \gamma_1
\]

and yields \( \hat{x}_1 = \hat{x}_2 = \hat{x} \) and \( \eta_1 = \eta_2 = \eta \) at a fixed point.
Experiment with Matched Priors

Comparison of several algorithms\(^8\) with priors matched to data.

\[
\begin{align*}
N &= 1024 \\
M/N &= 0.5 \\
A &= U \text{ Diag}(s)V^T \\
U, V &\sim \text{Haar} \\
s_n/s_{n-1} &= \phi \ \forall n \\
\phi &\text{ determines } \kappa(A) \\
X_o &\sim \text{Bernoulli-Gaussian} \\
\operatorname{Pr}\{X_0 \neq 0\} &= 0.1 \\
\text{SNR} &= 40\text{dB} \\
\end{align*}
\]

VAMP follows replica prediction\(^9\) over a wide range of condition numbers.

---

\(^8\)S-AMP: Cakmak, Fleury, Winther’14, AD-GAMP: Vila, Schniter, Rangan, Krzakala, Zdeborová’15

\(^9\)Tulino, Caire, Verdú, Shamai’13
Experiment with Matched Priors

Comparison of several algorithms with priors matched to data.

VAMP is fast even when $A$ is ill-conditioned.

\[
N = 1024 \\
M/N = 0.5 \\
A = U \text{Diag}(s)V^T \\
U, V \sim \text{Haar} \\
s_n/s_{n-1} = \phi \ \forall n \\
\phi \text{ determines } \kappa(A) \\
X_o \sim \text{Bernoulli-Gaussian} \\
\Pr\{X_0 \neq 0\} = 0.1 \\
\text{SNR} = 40\text{dB}
\]
Outline

1. Linear Regression, AMP, and Vector AMP (VAMP)
2. VAMP, ADMM, and Convergence in the Convex Setting
3. VAMP Convergence in the Non-Convex Setting
4. VAMP for Inference
5. **EM-VAMP and Adaptive VAMP**
6. Plug-and-play VAMP & Whitening
7. VAMP as a Deep Neural Network
8. VAMP for the Generalized Linear Model
Expectation Maximization

- What if the hyperparameters $\theta$ of the prior & likelihood are unknown?

- The EM algorithm\textsuperscript{10} is majorization-minimization approach to ML estimation that iteratively minimizes a tight upper bound on $-\ln p(y|\theta)$:

$$\hat{\theta}^{k+1} = \arg\min_\theta \left\{ -\ln p(y|\theta) + D(b^k(x)||p(x|y; \hat{\theta}^k)) \right\}$$

with $b^k(x) = p(x|y; \hat{\theta}^k) \geq 0$

- We can also write EM in terms of the Gibbs free energy:\textsuperscript{11}

$$\hat{\theta}^{k+1} = \arg\min_\theta D(b^k(x)||p(x; \theta_1)) + D(b^k(x)||\ell(x; \theta_2)) + H(b^k(x))$$

$$J_{\text{Gibbs}}(b^k, b^k, b^k; \theta)$$

- Thus, we can interleave EM and VAMP to solve

$$\min_{\theta} \min_{b_1, b_2, q} J_{\text{Gibbs}}(b_1, b_2, q; \theta) \text{ s.t. } \left\{ \begin{array}{l} E\{x|b_1\} = E\{x|b_2\} = E\{x|q\} \\ \text{tr[Cov}\{x|b_1\}\} = \text{tr[Cov}\{x|b_2\}\} = \text{tr[Cov}\{x|q\}\}. \end{array} \right.$$\textsuperscript{10}\textsuperscript{11}Dempster,Laird,Rubin’77, Neal,Hinton’98
The EM-VAMP Algorithm

Input conditional-mean $g_1(\cdot)$ and $g_2(\cdot)$, and initialize $r_1, \gamma_1, \hat{\theta}_1, \hat{\theta}_2$.

For $k = 1, 2, 3, \ldots$

\[ \hat{x}_1 \leftarrow g_1(r_1; \gamma_1, \hat{\theta}_1) \quad \text{MMSE estimation} \]

\[ \eta_1 \leftarrow \gamma_1 N / \text{tr} \left[ \frac{\partial g_1(r_1; \gamma_1, \hat{\theta}_1)}{\partial r_1} \right] \]

\[ r_2 \leftarrow \frac{(\eta_1 \hat{x}_1 - \gamma_1 r_1)}{(\eta_1 - \gamma_1)} \]

\[ \gamma_2 \leftarrow \eta_1 - \gamma_1 \]

\[ \hat{\theta}_2 \leftarrow \arg \max_{\theta_2} \mathbb{E}\{ \ln \ell(x; \theta_2) | r_2; \gamma_2, \hat{\theta}_2 \} \quad \text{EM update} \]

\[ \hat{x}_2 \leftarrow g_2(r_2; \gamma_2, \hat{\theta}_2) \quad \text{LMMSE estimation} \]

\[ \eta_2 \leftarrow \gamma_2 N / \text{tr} \left[ \frac{\partial g_2(r_2; \gamma_2, \hat{\theta}_2)}{\partial r_2} \right] \]

\[ r_1 \leftarrow \zeta (\eta_2 \hat{x}_2 - \gamma_2 r_2) / (\eta_2 - \gamma_2) + (1 - \zeta) r_1 \]

\[ \gamma_1 \leftarrow \zeta (\eta_2 - \gamma_2) + (1 - \zeta) \gamma_1 \]

\[ \hat{\theta}_1 \leftarrow \arg \max_{\theta_1} \mathbb{E}\{ \ln p(x; \theta_1) | r_1; \gamma_1, \hat{\theta}_1 \} \quad \text{EM update} \]

Experiments suggest it helps to update $\hat{\theta}_2$ several times per VAMP iteration.
State Evolution and Consistency

- EM-VAMP has a rigorous state-evolution when the prior is i.i.d. and $A$ is large and right-rotationally invariant.\textsuperscript{12}

- Furthermore, a variant known as “adaptive VAMP” can be shown to yield consistent parameter estimates with an i.i.d. prior in the exponential-family or with finite-cardinality $\theta_1$.\textsuperscript{12}

- Essentially, adaptive VAMP replaces the EM update

\[
\hat{\theta}_1 \leftarrow \arg\max_{\theta_1} \mathbb{E}\{\ln p(x; \theta_1) \mid r_1, \gamma_1, \hat{\theta}_1\}
\]

with

\[
(\hat{\theta}_1, \hat{\gamma}_1) \leftarrow \arg\max_{(\theta_1, \gamma_1)} \mathbb{E}\{\ln p(x; \theta_1) \mid r_1, \gamma_1, \hat{\theta}_1\},
\]

which also re-estimates the precision $\gamma_1$. (And similar for $\theta_2, \gamma_2$.)

\textsuperscript{12}Fletcher, Rangan, Schniter’17
Experiment with Unknown Hyperparameters $\theta$

Learning both noise precision $\theta_2$ and BG mean/variance/sparsity $\theta_1$:

$$ N = 1024 $$
$$ M/N = 0.5 $$

$$ A = U \text{ Diag}(s)V^T $$
$$ U, V \sim \text{Haar} $$
$$ s_n/s_{n-1} = \phi \ \forall n $$
$$ \phi \text{ determines } \kappa(A) $$

$$ X_0 \sim \text{Bernoulli-Gaussian} $$
$$ \Pr\{X_0 \neq 0\} = 0.1 $$

$$ \text{SNR} = 40 \text{dB} $$

EM-VAMP achieves oracle performance at all condition numbers.$^{13}$

$^{13}$EM-AMP proposed in Vila,Schniter’11 and Krzakala,Mézard,Sausset,Sun,Zdeborová’12
Experiment with Unknown Hyperparameters $\theta$

Learning both noise precision $\theta_2$ and BG mean/variance/sparsity $\theta_1$:

\[ N = 1024 \]
\[ M/N = 0.5 \]

\[ A = U \text{ Diag}(s)V^T \]
\[ U, V \sim \text{Haar} \]
\[ s_n/s_{n-1} = \phi \forall n \]
\[ \phi \text{ determines } \kappa(A) \]

\[ X_o \sim \text{Bernoulli-Gaussian} \]
\[ \text{Pr}\{X_0 \neq 0\} = 0.1 \]

\[ \text{SNR} = 40\text{dB} \]

EM-VAMP nearly as fast as VAMP and much faster than damped EM-GAMP.
Outline

1. Linear Regression, AMP, and Vector AMP (VAMP)
2. VAMP, ADMM, and Convergence in the Convex Setting
3. VAMP Convergence in the Non-Convex Setting
4. VAMP for Inference
5. EM-VAMP and Adaptive VAMP
6. Plug-and-play VAMP & Whitening
7. VAMP as a Deep Neural Network
8. VAMP for the Generalized Linear Model
Plug-and-play VAMP

- Recall that the nonlinear estimation step in VAMP (or AMP)
  \[ \hat{x}_1 \leftarrow g_1(r_1; \gamma_1) \text{ where } r_1 = x_o + \mathcal{N}(0, I/\gamma_1) \]
  can be interpreted as “denoising” the pseudo-measurement \( r_1 \).

- For certain signal classes, very sophisticated non-scalar denoising procedures have been developed (e.g., BM3D for images).

- Such denoising procedures can be “plugged into” signal recovery algorithms like ADMM\(^{14}\), AMP\(^{15}\), or VAMP\(^{16}\).

- For AMP and VAMP, the divergence can be approximated using Monte-Carlo:
  \[ \frac{1}{N} \text{tr} \left[ \frac{\partial g_1}{\partial r_1} \right] \approx \frac{1}{K} \sum_{k=1}^{K} \frac{p_k^T [g_1(r + \epsilon p_k, \gamma_1) - g_1(r, \gamma_1)]}{N \epsilon} \]
  with random vectors \( p_k \in \{\pm 1\}^N \) and small \( \epsilon > 0 \). Often, \( K = 1 \) suffices.

\(^{14}\)Bouman et al’13, \(^{15}\)Metzler,Maleki,Baraniuk’14, \(^{16}\)Schniter,Rangan,Fletcher’16
Experiment: Image Recovery with Random Matrices

Plug-and-play versions of VAMP and AMP work similarly when $\mathbf{A}$ is i.i.d., but VAMP can handle a larger class of random matrices $\mathbf{A}$.

Results above are averaged over $128 \times 128$ versions of

*lena, barbara, boat, fingerprint, house, peppers*

and 10 random realizations of $\mathbf{A}, \mathbf{w}$. 
Plug-and-play with Non-Random Matrices

- Many imaging applications (e.g., MRI) use low-frequency Fourier measurements, in which case $A = USV^T = I [I \ 0] F$.

- This causes problems for VAMP because the signal correlation structure interacts with $V^T$ in a way that VAMP is not designed to handle.

- Why? Say $x$ is a natural image, and consider $q = V^T x$.
  - If $V$ is large and Haar, then $q$ will be iid Gaussian.
  - If $V^T = F$, the low-freq entries of $q$ will be much stronger than the others.

*PnP VAMP treats $V^T x$ as iid Gaussian and thus diverges when $V^T = F$!*
Whitened VAMP for Image REcovery (VAMPire)

To apply VAMP with non-random Fourier measurements, we propose to operate on the whitened signal:

\[ y = \begin{bmatrix} I & 0 \end{bmatrix} FR_x^{1/2} s + w \quad \text{for} \quad \begin{cases} R_x = \mathbb{E}\{xx^T\} \\ s = \text{whitened signal coefficients} \end{cases} \]

and perform plug-and-play denoising from the whitened-coefficient space:

\[ \hat{s}_1 = g_1(r_1, \gamma_1) = R_x^{-1/2} \text{denoise}(R_x^{1/2} r_1; \gamma_1 N/ \text{tr}(R_x)). \]

In practice, we approximate \( R_x \approx W^T \text{Diag}(\tau)^2 W \), where \( W \) is a wavelet transform and \( \tau_i^2 \) specifies the energy of the \( i \)th wavelet coefficient (which is easy to predict for natural images).
Whitened VAMP for Image REcovery (VAMPFire)

- The resulting matrix $A = [I \; 0]FW \; \text{Diag}(\tau)$ does not yield a right singular vector matrix $V$ with a fast multiplication.

- But since $A$ has a fast implementation, the LMMSE stage can be computed via (preconditioned) LSQR:

$$g_2(r_2; \gamma_2) = (\gamma_w A^T A + \gamma_2 I)^{-1}(\gamma_w A^T y + \gamma_2 r_2) = \left[ \sqrt{\gamma_w} A \right]^+ \left[ \begin{array}{c} \sqrt{\gamma_w} y \\ \sqrt{\gamma_2} I \end{array} \right]$$

- The divergence $\langle g'_2(r_2; \gamma_2) \rangle$ can be approximated using Monte-Carlo:

$$\langle g'_2 \rangle = \frac{\gamma_2}{N} \text{tr} \left[ (\gamma_w A^H A + \gamma_2 I)^{-1} \right] \approx \frac{1}{NK} \sum_{k=1}^{K} p_k \left[ \sqrt{\gamma_w} A \right]^+ \left[ \begin{array}{c} 0 \\ \sqrt{\gamma_2} I \end{array} \right]$$

where $\mathbb{E}\{p_k p_k^H\} = I$. Here again, (preconditioned) LSQR can be used. In practice, $K = 1$ suffices.
Image Recovery Experiments

- Fourier measurements sampled at $M$ lowest frequencies
- SNR = 40dB
- $128 \times 128$ images \{lena, barbara, boat, fingerprint, house, peppers\}
- db1 wavelet decomposition, $D = 2$ levels
Outline

1. Linear Regression, AMP, and Vector AMP (VAMP)
2. VAMP, ADMM, and Convergence in the Convex Setting
3. VAMP Convergence in the Non-Convex Setting
4. VAMP for Inference
5. EM-VAMP and Adaptive VAMP
6. Plug-and-play VAMP & Whitening
7. VAMP as a Deep Neural Network
8. VAMP for the Generalized Linear Model
Deep learning for sparse reconstruction

- Until now we’ve focused on designing algorithms to recover $x_o \sim p(x)$ from measurements $y = Ax_o + w$.

- What about training deep networks to predict $x_o$ from $y$? Can we increase accuracy and/or decrease computation?

- Are there connections between these approaches?
Consider, e.g., the classical sparse-reconstruction algorithm, ISTA.\(^{17}\)

\[
\begin{align*}
\mathbf{v}^t &= \mathbf{y} - A\hat{\mathbf{x}}^t \\
\hat{\mathbf{x}}^{t+1} &= g(\hat{\mathbf{x}}^t + A^\top \mathbf{v}^t)
\end{align*}
\]

\[\iff\]

\[
\begin{align*}
\hat{\mathbf{x}}^{t+1} &= g(S\hat{\mathbf{x}}^t + B\mathbf{y}) \quad \text{with} \quad S \doteq I - A^\top A \\
B \doteq A^\top
\end{align*}
\]

Gregor & LeCun\(^{18}\) proposed to “unfold” it into a deep net and “learn” improved parameters using training data, yielding “learned ISTA” (LISTA):

\[
\begin{array}{cccccc}
\mathbf{y} & \rightarrow & B & \rightarrow & \mathbf{g}() & \rightarrow & \hat{\mathbf{x}}^1 \\
& & & & \mathbf{S} & \rightarrow & + \\
& & & & \mathbf{g}() & \rightarrow & \hat{\mathbf{x}}^2 \\
& & & & \mathbf{S} & \rightarrow & + \\
& & & & \mathbf{g}() & \rightarrow & \hat{\mathbf{x}}^3 \\
& & & & \mathbf{S} & \rightarrow & + \\
& & & & \mathbf{g}() & \rightarrow & \hat{\mathbf{x}}^4
\end{array}
\]

The same “unfolding & learning” idea can be used to improve AMP, yielding “learned AMP” (LAMP).\(^{19}\)

\(^{17}\)Daubechies, Defrise, DeMol’04. \(^{18}\)Gregor, LeCun’10. \(^{19}\)Borgerding, Schniter’16.
Onsager-Corrected Deep Networks

$t^{th}$ LISTA layer:

\[
\hat{x}^t \xrightarrow{+} r^t \xrightarrow{g(\bullet;\lambda^t)} \hat{x}^{t+1}
\]

\[
v^t \xrightarrow{B^t} \xrightarrow{r^t} \xrightarrow{A^t} \xrightarrow{-} v^{t+1}
\]

\[
y \xrightarrow{+} \xrightarrow{-} y
\]

to exploit low-rank $B^t A^t$ in linear stage $S^t = I - B^t A^t$.

$t^{th}$ LAMP layer:

\[
\hat{x}^t \xrightarrow{+} r^t \xrightarrow{g(\bullet;\bullet)} \hat{x}^{t+1}
\]

\[
v^t \xrightarrow{B^t} \xrightarrow{\frac{c^t \|\|_2}{\sqrt{M}} \lambda^t} \xrightarrow{\frac{N}{\sqrt{M}} \langle g' \rangle} v^{t+1}
\]

\[
y \xrightarrow{+} \xrightarrow{-} y
\]

Onsager correction now aims to decouple errors across layers.
LAMP performance with soft-threshold denoising

LISTA beats AMP, FISTA, ISTA
LAMP beats LISTA

in convergence speed and asymptotic MSE.

Graph showing the comparison of different methods in terms of average NMSE [dB] vs. layers / iterations. The graph includes lines for ISTA, FISTA, AMP, LISTA tied, LISTA untied, LAMP tied, and LAMP untied. The QQplot of LAMP $r^t$ also shows quantiles of input sample against standard normal quantiles.
LAMP beyond soft-thresholding

So far, we used soft-thresholding to isolate the effects of Onsager correction.

What happens with more sophisticated (learned) denoisers?

![Graph showing average NMSE vs. layers for different denoisers](image)

Here we learned the parameters of these denoiser families:

- scaled soft-thresholding
- conditional mean under BG
- Exponential kernel\(^{20}\)
- Piecewise Linear\(^{20}\)
- Spline\(^{21}\)

Big improvement!

\(^{20}\text{Guo,Davies’15.}\) \(^{21}\text{Kamilov,Mansour’16.}\)
LAMP versus VAMP

How does our best Learned AMP compare to (unlearned) VAMP?

VAMP wins!

So what about “learned VAMP”? 

LAMP-pwlin
VAMP-bg
support oracle
Learned VAMP

- Suppose we unfold VAMP and learn (via backprop) the parameters \( \{S^t, g^t\}_{t=1}^T \) that minimize the training MSE.

Remarkably, backpropagation does not improve matched VAMP! **VAMP is locally optimal**

- Onsager correction decouples the design of \( \{S^t, g^t(\cdot)\}_{t=1}^T \):
  - Layer-wise optimal \( S^t, g^t(\cdot) \)  \( \Rightarrow \)  Network optimal \( \{S^t, g^t(\cdot)\}_{t=1}^T \)
Outline

1. Linear Regression, AMP, and Vector AMP (VAMP)
2. VAMP, ADMM, and Convergence in the Convex Setting
3. VAMP Convergence in the Non-Convex Setting
4. VAMP for Inference
5. EM-VAMP and Adaptive VAMP
6. Plug-and-play VAMP & Whitening
7. VAMP as a Deep Neural Network
8. VAMP for the Generalized Linear Model
Generalized linear models

- Until now we have considered linear regression: $y = Ax_0 + w$.

- VAMP can also be applied to the generalized linear model (GLM)\(^{23}\)

$$y \sim p(y|z) \text{ with hidden } z = Ax_0$$

which supports, e.g.,

- $y_i = z_i + w_i$: additive, possibly non-Gaussian noise
- $y_i = \text{sgn}(z_i + w_i)$: binary classification / one-bit quantization
- $y_i = |z_i + w_i|$: phase retrieval in noise
- Poisson $y_i$: photon-limited imaging

- How? A simple trick turns the GLM into a linear regression problem:

$$z = Ax \iff \begin{bmatrix} 0 \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} A & -I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}$$

\(^{23}\)Schniter, Rangan, Fletcher’16
One-bit compressed sensing / Probit regression

Learning both $\theta_2$ and $\theta_1$:

VAMP and EM-VAMP robust to ill-conditioned $A$. 

\[ N = 512 \]
\[ M/N = 4 \]

\[ A = U \text{Diag}(s)V^T \]
\[ U, V \text{ drawn uniform} \]
\[ s_n/s_{n-1} = \phi \forall n \]
\[ \phi \text{ determines } \kappa(A) \]

\[ X_0 \sim \text{Bernoulli-Gaussian} \]
\[ \Pr\{X_0 \neq 0\} = 1/32 \]

\[ \text{SNR} = 40\text{dB} \]
One-bit compressed sensing / Probit regression

Learning both $\theta_2$ and $\theta_1$:

EM-VAMP mildly slower than VAMP but much faster than damped AMP.

$N = 512$
$M/N = 4$

$A = U \text{Diag}(s)V^T$
$U, V$ drawn uniform
$s_n/s_{n-1} = \phi \ \forall n$
$\phi$ determines $\kappa(A)$

$X_0 \sim \text{Bernoulli-Gaussian}$
$\Pr\{X_0 \neq 0\} = 1/32$

$\text{SNR} = 40\text{dB}$
Conclusions

- VAMP is an efficient algorithm for linear and generalized-linear regression.

- For convex optimization problems, VAMP is provably convergent and related to Peaceman-Rachford ADMM.

- For inference under right rotationally-invariant $A$, VAMP has a rigorous state evolution and fixed-points that agree with the replica MMSE prediction.

- VAMP can be combined with EM to handle priors/likelihood with unknown parameters, again with a rigorous state evolution.

- Can unfold VAMP into an interpretable deep network.

- In non-convex settings (e.g., plug-and-play) with deterministic matrices, more work is needed to understand the performance and convergence of VAMP.

- Still lots to do! (multilayer generative models, bilinear problems . . . )