Vector Approximate Message Passing and Connections to Deep Learning

Phil Schniter

THE OHIO STATE UNIVERSITY

Duke UNIVERSITY

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

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

IEEE Information Theory Workshop (ITW) — Sep 13, 2016

Sparse Reconstruction

Goal:

Recover $\boldsymbol{x}_o \in \mathbb{R}^N$ from measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o + \boldsymbol{w} \in \mathbb{R}^M$

Assumptions:

- $\boldsymbol{x}_o$ is sparse
- $\boldsymbol{A}$ is known and high dimensional
- often $M \ll N$
- $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \tau_w \boldsymbol{I})$

Regularized loss minimization

Popular approach:

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 + \lambda f(\boldsymbol{x})$$

where

- $f(\boldsymbol{x})$ is a regularizer, e.g., $\|\boldsymbol{x}\|_1$ in LASSO or BPDN
- $\lambda > 0$ is a tuning parameter

The iterative soft thresholding algorithm (ISTA)

ISTA:

initialize $\widehat{\boldsymbol{x}}^0 = \boldsymbol{0}$

for $t = 0, 1, 2, \ldots$

$$\boldsymbol{v}^t = \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}^t \qquad \text{residual error}$$

$$\widehat{\boldsymbol{x}}^{t+1} = \boldsymbol{g}(\widehat{\boldsymbol{x}}^t + \boldsymbol{A}^{\mathsf{T}}\boldsymbol{v}^t) \qquad \text{thresholding}$$

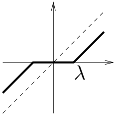
where

$$\boldsymbol{g}(\boldsymbol{r}) = \arg\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{r} - \boldsymbol{x}\|_2^2 + \lambda f(\boldsymbol{x}) \;\triangleq\; \mathsf{prox}_{\lambda f}(\boldsymbol{r})$$

$\|\boldsymbol{A}\|_2^2 < 1$ ensures convergence[1] with convex $f(\cdot)$.

When $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ we get "soft thresholding"

$$[\boldsymbol{g}(\boldsymbol{r})]_j = \mathsf{sgn}(r_j)\max\{0, |r_j| - \lambda\}$$



[1] Daubechies, Defrise, DeMol–CPAM'04

Approximate Message Passing (AMP)

Donoho, Maleki, and Montanari[2] proposed:

initialize $\widehat{\boldsymbol{x}}^0 = \mathbf{0}$, $\boldsymbol{v}^{-1} = \mathbf{0}$

for $t = 0, 1, 2, \ldots$

$$\boldsymbol{v}^t = \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}^t + \tfrac{N}{M}\boldsymbol{v}^{t-1}\langle \boldsymbol{g}^{t-1\prime}(\widehat{\boldsymbol{x}}^{t-1} + \boldsymbol{A}^\mathsf{T}\widehat{\boldsymbol{v}}^{t-1})\rangle \quad \text{corrected residual}$$

$$\widehat{\boldsymbol{x}}^{t+1} = \boldsymbol{g}^t(\widehat{\boldsymbol{x}}^t + \boldsymbol{A}^\mathsf{T}\boldsymbol{v}^t) \quad \text{thresholding}$$

where

$$\langle \boldsymbol{g}'(\boldsymbol{r})\rangle \triangleq \tfrac{1}{N}\textstyle\sum_{j=1}^{N} \frac{\partial g_j(\boldsymbol{r})}{\partial r_j} \qquad \text{“divergence.”}$$

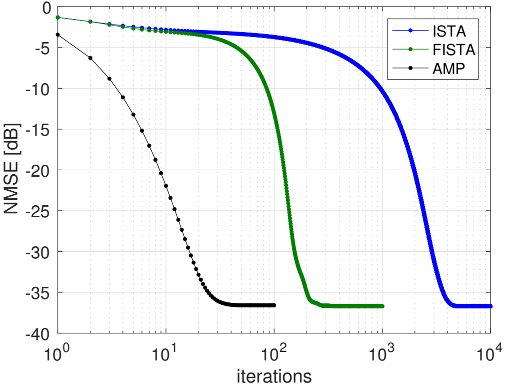
Note:

- “Onsager correction” aims to decouple the errors across iterations.
- The thresholding $\boldsymbol{g}^t(\cdot)$ can vary with iteration t.

[2] Donoho,Maleki,Montanari–PNAS'09

AMP vs ISTA (and FISTA)

Example: LASSO problem with i.i.d. Gaussian $\boldsymbol{A}$:



- $M = 250,\ N = 500$
- $\Pr\{x_n \neq 0\} = 0.1$
- SNR$=40$dB
- Convergence to -35dB:
  - ISTA: 2407 iterations
  - FISTA:[3] 174 iterations
  - AMP: 25 iterations

[3] Beck,Teboulle–JIS'09

AMP's state evolution

Define $\mathcal{E}^t := \frac{1}{N}\,\mathrm{E}\left\{\|\widehat{\boldsymbol{x}}^t - \boldsymbol{x}_o\|^2\right\}$ as the iteration-t MSE.

For large i.i.d. sub-Gaussian $\boldsymbol{A}$ and separable Lipschitz $\boldsymbol{g}^t(\cdot)$, AMP has the following scalar state evolution (SE):[4]

$$\boxed{\begin{aligned}
&\text{for } t = 0, 1, 2, \ldots \\
&\qquad \tau_r^t = \tau_w + \tfrac{N}{M}\mathcal{E}^t \\
&\qquad \mathcal{E}^{t+1} = \tfrac{1}{N}\,\mathrm{E}\Big\{\Big\|\boldsymbol{g}^t\big(\underbrace{\boldsymbol{x}_o + \mathcal{N}(\boldsymbol{0}, \tau_r^t \boldsymbol{I})}_{:= \boldsymbol{r}^t}\big) - \boldsymbol{x}_o\Big\|^2\Big\}
\end{aligned}}$$

But for generic $\boldsymbol{A}$, AMP is not well justified and may fail catastrophically.

[4] Bayati,Montanari–TransIT'11

Vector AMP (VAMP)

The vector AMP algorithm for linear regression is

for $t = 0, 1, 2, \ldots$

$$\widehat{\boldsymbol{x}}_1^t = \boldsymbol{g}(\boldsymbol{r}_1^t; \gamma_1^t) \qquad \text{thresholding}$$

$$\alpha_1^t = \frac{1}{N}\sum_j \frac{\partial g_r}{\partial r_j}(\boldsymbol{r}_1^t; \gamma_1^t) \qquad \text{divergence}$$

$$\boldsymbol{r}_2^t = \frac{1}{1-\alpha_1^t}\left(\widehat{\boldsymbol{x}}_1^t - \alpha_1^t \boldsymbol{r}_1^t\right) \qquad \text{Onsager correction}$$

$$\gamma_2^t = \gamma_1^t \frac{1-\alpha_1^t}{\alpha_1^t} \qquad \text{precision of } \boldsymbol{r}_2^t$$

$$\widehat{\boldsymbol{x}}_2^t = \left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}/\widehat{\tau}_w + \gamma_2^t \boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{y}/\widehat{\tau}_w + \gamma_2^t \boldsymbol{r}_2^t\right) \qquad \text{LMMSE}$$

$$\alpha_2^t = \frac{\gamma_2^t}{N}\,\mathrm{Tr}\left[\left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}/\widehat{\tau}_w + \gamma_2^t \boldsymbol{I}\right)^{-1}\right] \qquad \text{divergence}$$

$$\boldsymbol{r}_1^{t+1} = \frac{1}{1-\alpha_2^t}\left(\widehat{\boldsymbol{x}}_2^t - \alpha_2^t \boldsymbol{r}_2^t\right) \qquad \text{Onsager correction}$$

$$\gamma_1^{t+1} = \gamma_2^t \frac{1-\alpha_2^t}{\alpha_2^t} \qquad \text{precision of } \boldsymbol{r}_1^{t+1}$$

Note similarities with standard AMP.

VAMP without matrix inverses

Can avoid matrix inverses by pre-computing an SVD, $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}$:

for $t = 0, 1, 2, \ldots$

$$\widehat{\boldsymbol{x}}^t = \boldsymbol{g}(\boldsymbol{r}_1^t; \gamma_1^t) \qquad \text{thresholding}$$

$$\alpha_1^t = \tfrac{1}{N}\textstyle\sum_j \tfrac{\partial g}{\partial r_j}(\boldsymbol{r}_1^t; \gamma_1^t) \qquad \text{divergence}$$

$$\boldsymbol{r}_2^t = \tfrac{1}{1-\alpha_1^t}\big(\widehat{\boldsymbol{x}}^t - \alpha_1^t \boldsymbol{r}_1^t\big) \qquad \text{Onsager}$$

$$\gamma_2^t = \gamma_1^t \tfrac{1-\alpha_1^t}{\alpha_1^t} \qquad \text{precision}$$

$$\alpha_2^t = \tfrac{1}{N}\textstyle\sum_j \gamma_2^t / \big(s_j^2/\widehat{\tau}_w + \gamma_2^t\big) \qquad \text{divergence}$$

$$\boldsymbol{r}_1^{t+1} = \boldsymbol{r}_2^t + \tfrac{1}{1-\alpha_2^t}\boldsymbol{V}\big(\boldsymbol{S}^2 + \widehat{\tau}_w\gamma_2^t\boldsymbol{I}\big)^{-1}\boldsymbol{S}\big(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{y} - \boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{r}_2^t\big) \qquad \text{2 mat-vec}$$

$$\gamma_1^{t+1} = \gamma_2^t \tfrac{1-\alpha_2^t}{\alpha_2^t} \qquad \text{precision}$$

And can tune noise precision $\widehat{\tau}_w$ using EM.

Why call this "Vector AMP"?

1) Can be derived using an approximation of message passing on a factor graph, now with vector-valued variable nodes.

2) Performance characterized by a rigorous state-evolution[5] under certain large random $\boldsymbol{A}$:

SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\mathsf{T}$

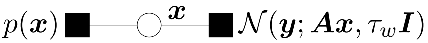
- $\boldsymbol{U}$ *is deterministic*
- $\boldsymbol{S}$ *is deterministic*
- $\boldsymbol{V}$ *is uniformly distributed on the group of orthogonal matrices*

"$\boldsymbol{A}$ *is right rotationally invariant*"

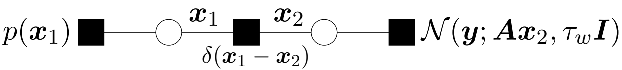
[5] Rangan,Fletcher,Schniter–16

Message-passing derivation of VAMP

- Write joint density as $p(\boldsymbol{x},\boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x}) = p(\boldsymbol{x})\mathcal{N}(\boldsymbol{y};\boldsymbol{A}\boldsymbol{x},\tau_w\boldsymbol{I})$



- Variable splitting: $p(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{y}) = p(\boldsymbol{x}_1)\delta(\boldsymbol{x}_1-\boldsymbol{x}_2)\mathcal{N}(\boldsymbol{y};\boldsymbol{A}\boldsymbol{x}_2,\tau_w\boldsymbol{I})$



- Perform message-passing with messages approximated as $\mathcal{N}(\boldsymbol{\mu},\sigma^2\boldsymbol{I})$.
  - An instance of expectation-propagation[6] (EP).
  - Also derivable through expectation-consistent approximation[7] (EC).

[6] Minka–Dissertation'01

[7] Opper,Winther–NIPS'04, Fletcher,Rangan,Schniter–ISIT'16

VAMP state evolution

Assuming empirical convergence of $\{s_j\} \to S$ and $\{(r_{1,j}^0, x_{o,j})\} \to (R_1^0, X_o)$ and Lipschitz continuity of g and g', the SE under $\widehat{\tau}_w = \tau_w$ is as follows:

for $t = 0, 1, 2, \ldots$

$$\mathcal{E}_1^t = \mathrm{E}\left\{\left[g\big(X_o + \mathcal{N}(0, \tau_1^t); \overline{\gamma}_1^t\big) - X_o\right]^2\right\} \qquad \text{MSE}$$

$$\overline{\alpha}_1^t = \mathrm{E}\left\{g'(X_o + \mathcal{N}(0, \tau_1^t); \overline{\gamma}_1^t)\right\} \qquad \text{divergence}$$

$$\overline{\gamma}_2^t = \overline{\gamma}_1^t \tfrac{1-\overline{\alpha}_1^t}{\overline{\alpha}_1^t}, \quad \tau_2^t = \tfrac{1}{(1-\overline{\alpha}_1^t)^2}\left[\mathcal{E}_1^t - \left(\overline{\alpha}_1^t\right)^2 \tau_1^t\right] \qquad \text{precision}$$

$$\mathcal{E}_2^t = \mathrm{E}\left\{\left[S^2/\tau_w + \overline{\gamma}_2^t\right]^{-1}\right\} \qquad \text{MSE}$$

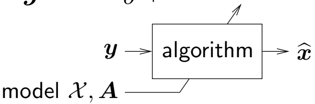
$$\overline{\alpha}_2^t = \overline{\gamma}_2^t\, \mathrm{E}\left\{\left[S^2/\tau_w + \overline{\gamma}_2^t\right]^{-1}\right\} \qquad \text{divergence}$$

$$\overline{\gamma}_1^{t+1} = \overline{\gamma}_2^t \tfrac{1-\overline{\alpha}_2^t}{\overline{\alpha}_2^t}, \quad \tau_1^{t+1} = \tfrac{1}{(1-\overline{\alpha}_2^t)^2}\left[\mathcal{E}_2^t - \left(\overline{\alpha}_2^t\right)^2 \tau_2^t\right] \qquad \text{precision}$$

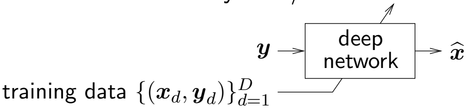
More complicated expressions for $\mathcal{E}_2^t$ and $\overline{\alpha}_2^t$ apply when $\widehat{\tau}_w \neq \tau_w$.

Deep learning for sparse reconstruction

- Until now we've focused on designing algorithms to recover $\boldsymbol{x}_o \in \mathcal{X}$ from measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o + \boldsymbol{w}$.



- What about training deep networks to predict $\boldsymbol{x}_o$ from $\boldsymbol{y}$? Can we increase accuracy and/or decrease computation?



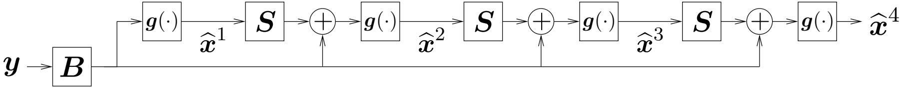
- Are there connections between these approaches?

Unrolling ISTA

First, rewrite ISTA as

$$\boxed{\begin{aligned} \boldsymbol{v}^t &= \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}^t \\ \widehat{\boldsymbol{x}}^{t+1} &= \boldsymbol{g}(\widehat{\boldsymbol{x}}^t + \boldsymbol{A}^\mathsf{T}\boldsymbol{v}^t) \end{aligned}} \quad \Leftrightarrow \quad \boxed{\widehat{\boldsymbol{x}}^{t+1} = \boldsymbol{g}(\boldsymbol{S}\widehat{\boldsymbol{x}}^t + \boldsymbol{B}\boldsymbol{y}) \text{ with } \begin{aligned} \boldsymbol{S} &\triangleq \boldsymbol{I} - \boldsymbol{A}^\mathsf{T}\boldsymbol{A} \\ \boldsymbol{B} &\triangleq \boldsymbol{A}^\mathsf{T} \end{aligned}}$$

Then "unroll" into a network:

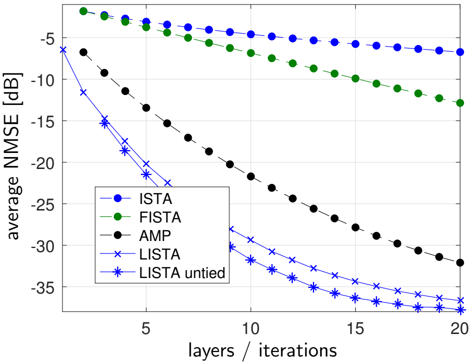


Note cascade of linear "$\boldsymbol{S}$," bias "$\boldsymbol{B}\boldsymbol{y}$," & separable non-linearity "$\boldsymbol{g}(\cdot)$."

ISTA algorithm $\Leftrightarrow$ deep neural network

Learned ISTA (LISTA)

Gregor and LeCun[8] proposed to learn (via backpropagation) the linear transform $\boldsymbol{S}$ and soft thresholds $\{\lambda^t\}_{t=1}^T$ that minimize training MSE



$$\arg\min_{\boldsymbol{\Theta}} \sum_{d=1}^{D} \left\| \widehat{\boldsymbol{x}}(\boldsymbol{y}_d; \boldsymbol{\Theta}) - \boldsymbol{x}_d \right\|^2.$$

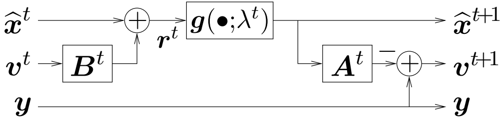
The resulting "LISTA" beats LASSO-AMP in convergence speed *and* asymptotic MSE!

Further improvement when $\boldsymbol{S}$ is "untied" to $\{\boldsymbol{S}^t\}_{t=1}^T$.

[8] Gregor,LeCun–ICML'10

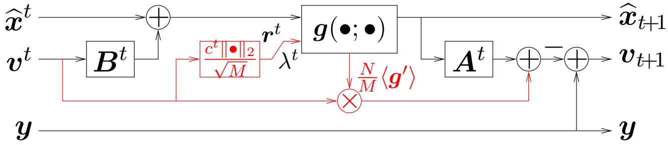
Learned AMP (LAMP)

t^{th} LISTA layer:



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

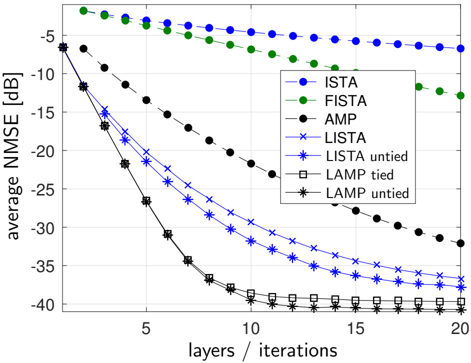
t^{th} LAMP layer:

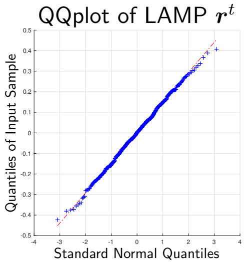


Onsager correction now aims to decouple errors across layers.

LAMP performance under soft thresholding

LAMP beats LISTA in both convergence speed and asymptotic MSE.

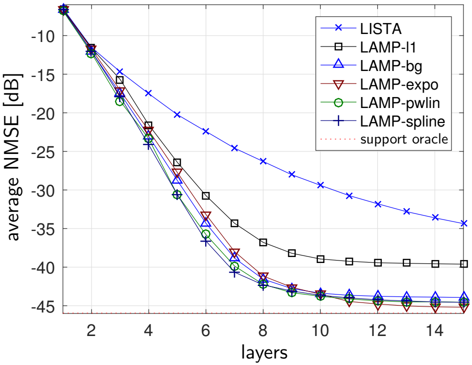




LAMP with more sophisticated denoisers

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

What happens with more sophisticated (learned) denoisers?



Here we learned the parameters of these denoiser families:

- scaled soft-thresholding
- Bernoulli-Gaussian MMSE
- Exponential kernel[9]
- Piecewise Linear[9]
- Spline[10]

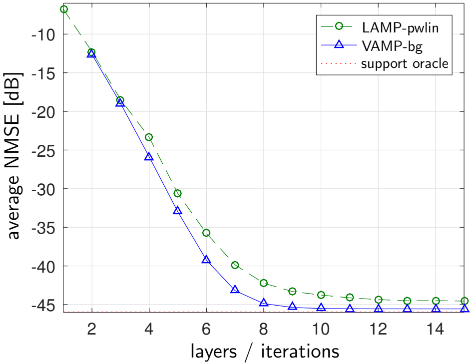
Big improvement!

[9] Guo,Davies–TSP'15

[10] Kamilov,Mansour–SPL'16

LAMP  versus VAMP 

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

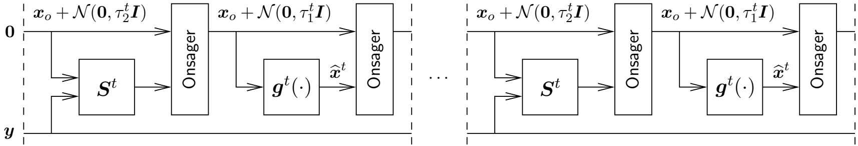


VAMP wins!

So what about "learned VAMP"?

Local optimality of VAMP

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



- Remarkably, backpropagation does not improve matched VAMP!

  VAMP is locally optimal

- Essentially, Onsager correction decouples the design of $\{\boldsymbol{S}^t, \boldsymbol{g}^t(\cdot)\}_{t=1}^T$:

  Layer-wise optimal $\boldsymbol{S}^t, \boldsymbol{g}^t(\cdot) \quad \Rightarrow \quad$ Network optimal $\{\boldsymbol{S}^t, \boldsymbol{g}^t(\cdot)\}_{t=1}^T$

Conclusions

- For sparse reconstruction, AMP has some nice properties:
  - low cost-per-iteration
  - fast convergence,
  - rigorous state evolution,

  but only under large i.i.d. Gaussian $\boldsymbol{A}$.

- We proposed a Vector AMP, where the same nice properties hold under large rotationally invariant $\boldsymbol{A}$.

- "Learned ISTA" results from unrolling ISTA and fitting its parameters to training data. We proposed learned AMP & learned VAMP.

- Remarkably, the original VAMP is locally optimal.