Onsager Correction, Deep Learning, Sparse Reconstruction and VAMP

Mark Borgerding and Phil Schniter

THE OHIO STATE UNIVERSITY

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

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

IEEE GlobalSIP — Dec 8, 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 $\widehat{\boldsymbol{x}}^0 = \mathbf{0}$

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

$$\begin{aligned} \boldsymbol{v}^t &= \boldsymbol{y} - \boldsymbol{A}\widehat{\boldsymbol{x}}^t && \text{residual error} \\ \widehat{\boldsymbol{x}}^{t+1} &= \boldsymbol{g}(\widehat{\boldsymbol{x}}^t + \boldsymbol{A}^\mathsf{T}\boldsymbol{v}^t) && \text{thresholding} \end{aligned}$$

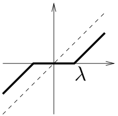
where

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

$\|\boldsymbol{A}\|_2^2 < 1$ 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 $t = 0, 1, 2, \ldots$

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

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

where

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

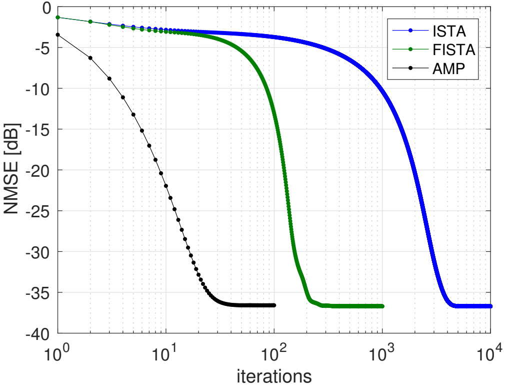
Note:

- “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

The limitations of AMP

- AMP's good performance is guaranteed only for large i.i.d. zero-mean sub-Gaussian $\boldsymbol{A}$.
- Deviations from this condition can cause AMP to diverge.
- Can we extend AMP to a larger class of matrices?

Vector AMP (VAMP)

The recent vector AMP[4] algorithm for linear regression is

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

$\widehat{\boldsymbol{x}}_1^t = \boldsymbol{g}(\boldsymbol{r}_1^t; \gamma_1^t)$	thresholding
$\alpha_1^t = \frac{1}{N} \sum_j \frac{\partial g_r}{\partial r_j}(\boldsymbol{r}_1^t; \gamma_1^t)$	divergence
$\boldsymbol{r}_2^t = \frac{1}{1-\alpha_1^t}\big(\widehat{\boldsymbol{x}}_1^t - \alpha_1^t \boldsymbol{r}_1^t\big)$	Onsager correction
$\gamma_2^t = \gamma_1^t \frac{1-\alpha_1^t}{\alpha_1^t}$	precision of $\boldsymbol{r}_2^t$
$\widehat{\boldsymbol{x}}_2^t = \big(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}/\widehat{\tau}_w + \gamma_2^t \boldsymbol{I}\big)^{-1}\big(\boldsymbol{A}^\mathsf{T}\boldsymbol{y}/\widehat{\tau}_w + \gamma_2^t \boldsymbol{r}_2^t\big)$	LMMSE
$\alpha_2^t = \frac{\gamma_2^t}{N} \operatorname{Tr}\Big[\big(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}/\widehat{\tau}_w + \gamma_2^t \boldsymbol{I}\big)^{-1}\Big]$	divergence
$\boldsymbol{r}_1^{t+1} = \frac{1}{1-\alpha_2^t}\big(\widehat{\boldsymbol{x}}_2^t - \alpha_2^t \boldsymbol{r}_2^t\big)$	Onsager correction
$\gamma_1^{t+1} = \gamma_2^t \frac{1-\alpha_2^t}{\alpha_2^t}$	precision of $\boldsymbol{r}_1^{t+1}$

[4] Rangan,Schniter,Fletcher – arxiv:1610.03082

VAMP without matrix inverses

LMMSE matrix inverse step is easy if the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}$ is known.
Note if $\kappa(\boldsymbol{A}) = 1$ [5], the SVD is trivial $\boldsymbol{U}\boldsymbol{S} \propto \boldsymbol{I}$,$\boldsymbol{V} \propto \boldsymbol{A}^{\mathsf{T}}$.

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

$$\begin{array}{ll}
\widehat{\boldsymbol{x}}^t = \boldsymbol{g}(\boldsymbol{r}_1^t; \gamma_1^t) & \text{thresholding} \\
\alpha_1^t = \frac{1}{N}\sum_j \frac{\partial g}{\partial r_j}(\boldsymbol{r}_1^t; \gamma_1^t) & \text{divergence} \\
\boldsymbol{r}_2^t = \frac{1}{1-\alpha_1^t}\left(\widehat{\boldsymbol{x}}^t - \alpha_1^t \boldsymbol{r}_1^t\right) & \text{Onsager} \\
\gamma_2^t = \gamma_1^t \frac{1-\alpha_1^t}{\alpha_1^t} & \text{precision} \\
\hline
\alpha_2^t = \frac{1}{N}\sum_j \gamma_2^t / \left(s_j^2/\widehat{\tau}_w + \gamma_2^t\right) & \text{divergence} \\
\boldsymbol{r}_1^{t+1} = \boldsymbol{r}_2^t + \frac{1}{1-\alpha_2^t}\boldsymbol{V}\left(\boldsymbol{S}^2 + \widehat{\tau}_w \gamma_2^t \boldsymbol{I}\right)^{-1}\boldsymbol{S}\left(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{y} - \boldsymbol{S}\boldsymbol{V}^{\mathsf{T}}\boldsymbol{r}_2^t\right) & \text{2 mat-vec} \\
\gamma_1^{t+1} = \gamma_2^t \frac{1-\alpha_2^t}{\alpha_2^t} & \text{precision}
\end{array}$$

[5] e.g. punctured DFT,DCT,DWT

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[6] under certain large random $\boldsymbol{A}$:

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

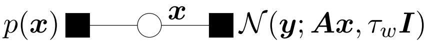
- $\boldsymbol{U}$ *is deterministic*
- $\boldsymbol{S}$ *is deterministic*
- $\boldsymbol{V}$ *is uniformly distributed on the group of orthogonal matrices*

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

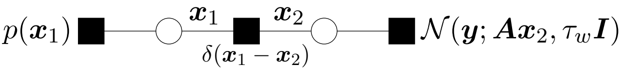
6 Rangan,Fletcher,Schniter–16

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[7] (EP).
  - Also derivable through expectation-consistent approximation[8] (EC).

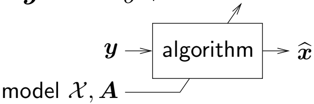
---

[7] Minka–Dissertation'01

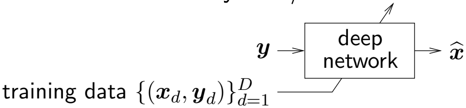
[8] Opper,Winther–NIPS'04, Fletcher,Rangan,Schniter–ISIT'16

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?



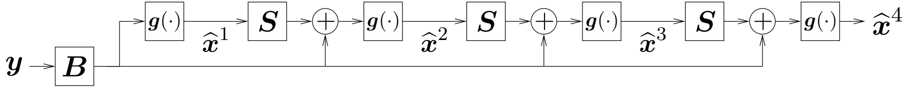
- Are there connections between these approaches?

Unrolling ISTA

First, rewrite ISTA as

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

Then "unroll" into a network:

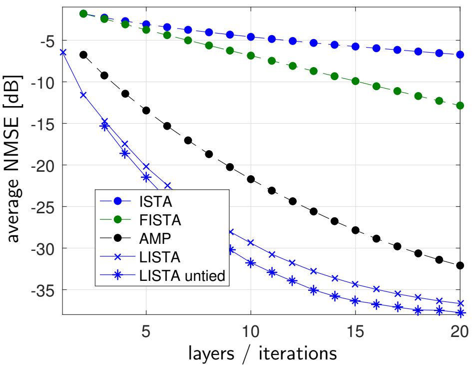


Note cascade of linear "$\boldsymbol{S}$," bias "$\boldsymbol{B}\boldsymbol{y}$," & separable non-linearity "$\boldsymbol{g}(\cdot)$."

ISTA algorithm $\Leftrightarrow$ deep neural network

Learned ISTA (LISTA)

Gregor and LeCun[9] proposed to learn (via backpropagation) the linear transform $\boldsymbol{S}$ and soft thresholds $\{\lambda^t\}_{t=1}^T$ that minimize training MSE



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

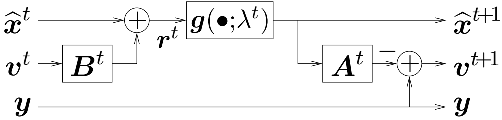
The resulting "LISTA" beats LASSO-AMP in convergence speed *and* asymptotic MSE!

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

[9] Gregor,LeCun–ICML'10

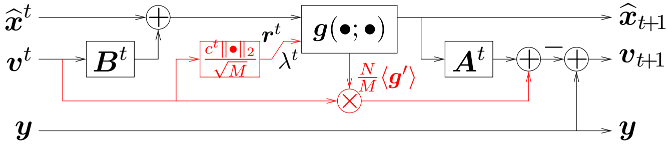
Learned AMP (LAMP)

t^{th} LISTA layer:



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

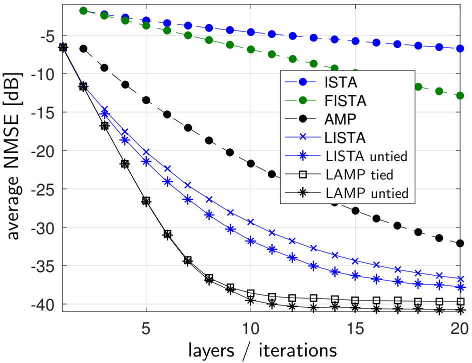
t^{th} LAMP layer:

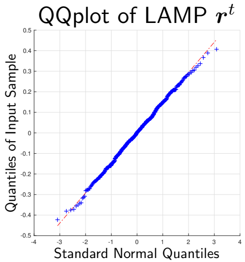


Onsager correction now aims to decouple errors across layers.

LAMP performance under soft thresholding

LAMP beats LISTA in both convergence speed and asymptotic MSE.





Simulation/Training Details

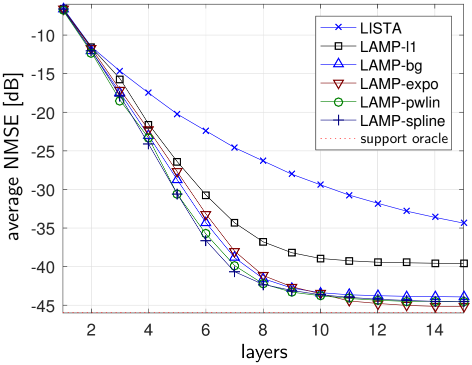
- Bernoulli-Gaussian $\boldsymbol{x} \in \mathbb{R}^{500}$, known Gaussian $\boldsymbol{A} \in \mathbb{R}^{250\times 500}$, 40dB SNR
- TensorFlow implementation on GPU
- Adam[10] SGD in mini-batches of 1000 vectors
- Add layers by greedy extension, then whole-network fine-tuning
- Reported results are for an untrained validation set

[10] Kingma,BA'15

LAMP with more sophisticated denoisers

So far, we used soft-thresholding, so as to compare directly to LISTA.

What happens when we learn other denoisers?



Here we learned the parameters of these denoiser families:

- scaled soft-threshold (LAMP-ℓ_1)
- Bernoulli-Gaussian MMSE
- Exponential kernel[11]
- Piecewise Linear[11]
- Spline[12]

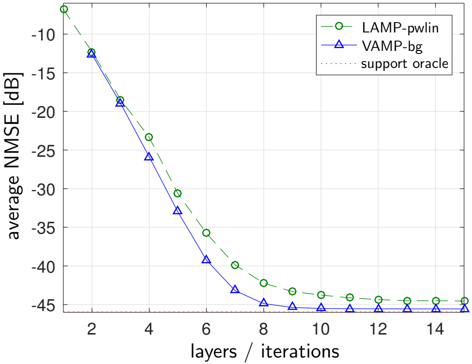
Big improvement!

[11] Guo,Davies–TSP'15

[12] Kamilov,Mansour–SPL'16

LAMP versus VAMP

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

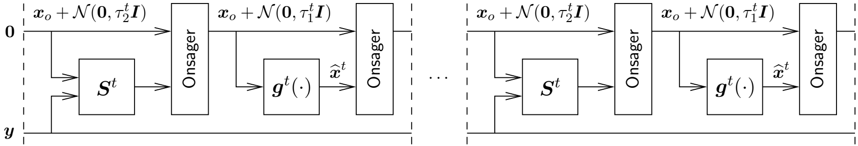


VAMP wins!

So what about "learned VAMP"?

Learned VAMP

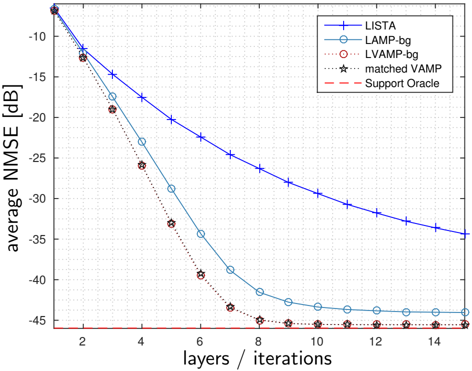
- Suppose we unroll VAMP and learn (via backprop) the parameters $\{\boldsymbol{S}^t, \boldsymbol{g}^t\}_{t=1}^T$ that minimize the training MSE.



Can we improve VAMP with learning?

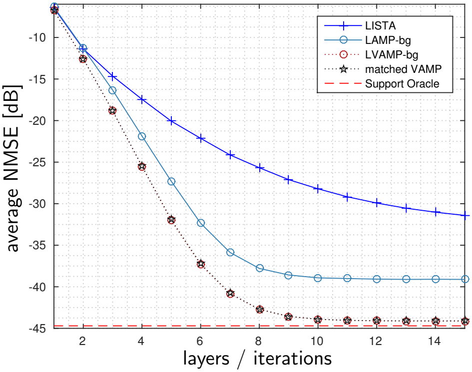
Learning VAMP

- Remarkably, backpropagation does not improve matched VAMP!
- Matched VAMP specifies optimal network parameters in closed form, without training.



Non iid sub-Gaussian A

- What happens if $\boldsymbol{A}$ is not iid sub-Gaussian?
- We replaced the singular values from our previous matrix with a log-spaced sequence so $\kappa(\boldsymbol{A}) = 15$.
- Basic AMP fails to converge!
- VAMP's advantages become even clearer!
- In large-system limit, VAMP can handle arbitrarily large $\kappa(\boldsymbol{A})$



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 $\boldsymbol{A}$.
- Vector AMP has the same nice properties 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.
- MSE-optimal parameters of VAMP can be specified in closed form, without training, when signal/noise statistics are known.

For details, see Borgerding,Schniter arXiv 1612.01183