AMP-inspired Deep Networks, with Comms Applications

Phil Schniter

THE OHIO STATE UNIVERSITY

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

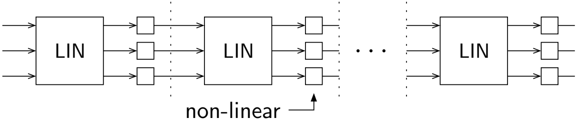
Supported in part by NSF grants IIP-1539960, CCF-1527162, and CCF-1716388.

Intl. Conf. on Signal Processing and Communications — July 2018

Deep Neural Networks

Typical feedforward setup:

- Many layers, consisting of (affine) linear stages and scalar nonlinearities.



- Linear stages often constrained (e.g., small convolution kernels).
- Parameters learned by minimizing training error using backpropagation.

Open questions:

1. How should we interpret the learned parameters?
2. Can we speed up training?
3. Can we design a better network structure?

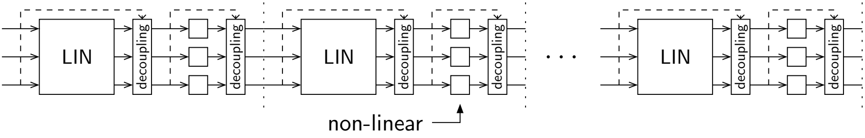
Focus of this talk: Standard Linear Regression

- Consider recovering a vector $\boldsymbol{x}$ from noisy linear observations

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w},$$

where $\boldsymbol{x}$ is drawn from an iid prior (e.g., sparse[1])

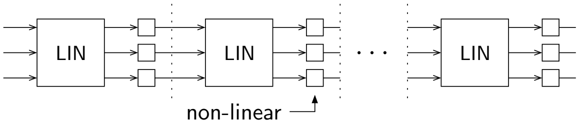
- For this application, we propose a deep network that is
1) asymptotically optimal for a large class of $\boldsymbol{A}$,
2) interpretable, and
3) easy to train.



[1] Gregor/LeCun, Sprechmann/Bronstein/Sapiro, Kamilov/Mansour, Wang/Ling/Huang, Mousavi/Baraniuk, Xin/Wang/Gao/Wipf, Borgerding/Schniter/Rangan, etc.

Understanding Deep Networks via Algorithms

- Many algorithms have been proposed for linear regression.
- Often, such algorithms are iterative, where each iteration consists of a linear operation followed by scalar nonlinearities.
- By "unfolding" such algorithms, we get deep networks.[2]



- Can such algorithms help us design/interpret/train deep nets?

[2] Gregor/LeCun, ICML 10.

Algorithmic Approaches to Standard Linear Regression

- Recall goal: recovering/fitting $\boldsymbol{x}$ from noisy linear observations

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}.$$

- A popular approach is regularized loss minimization:

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

where, e.g., $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ for the lasso.

- Can also be interpreted as MAP estimation of $\boldsymbol{x}$ under priors

$$\boldsymbol{x} \sim \exp(-f(\boldsymbol{x})) \quad \& \quad \boldsymbol{w} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}).$$

- But often the goal is minimizing MSE or inferring marginal posteriors.

High-dimensional MMSE Inference

- High dimensional MMSE inference is difficult in general.
- To simplify things, suppose that 1) $\boldsymbol{x}$ is iid
  2) $\boldsymbol{A}$ is large and random.
- The case of iid Gaussian $\boldsymbol{A}$ is well studied, but very restrictive.
- Instead, consider right-rotationally invariant (RRI) $\boldsymbol{A}$:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\mathsf{T} \text{ with } \boldsymbol{V} \sim \text{Haar and indep of } \boldsymbol{x}.$$

- For this case, the MMSE is[3][4]

$$\mathcal{E}(\gamma) = \text{var}\{x|r\}, \quad r = x + \mathcal{N}(0, 1/\gamma), \quad \gamma = R_{\boldsymbol{A}^\mathsf{T}\boldsymbol{A}/\sigma^2}(-\mathcal{E}(\gamma))$$

[3] Tulino/Caire/Verdu/Shamai, IEEE-TIT'13, [4] Reeves, Allerton'17.

Achieving MMSE in standard linear regression

- Recently a "vector approximate message passing (VAMP)" algorithm has been proposed that iterates linear vector estimation with nonlinear scalar denoising. (Closely related to expectation propagation.[5])

- Under large RRI $\boldsymbol{A}$ and Lipschitz denoisers, VAMP is rigorously characterized by a scalar state-evolution.[6]

- When the state-evolution has a unique fixed point, the VAMP solutions are MMSE!

[5] Opper/Winther, JMLR'05, [6] Rangan/Schniter/Fletcher, arXiv:1610.03082.

VAMP for linear regression

initialize $\boldsymbol{r}_1, \gamma_1$

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

$\widehat{\boldsymbol{x}}_1 \leftarrow \big(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}/\sigma^2 + \gamma_1 \boldsymbol{I}\big)^{-1}\big(\boldsymbol{A}^\mathsf{T}\boldsymbol{y}/\sigma^2 + \gamma_1 \boldsymbol{r}_1\big)$ — LMMSE

$\alpha_1 \leftarrow \frac{\gamma_1}{N}\operatorname{Tr}\Big[\big(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}/\sigma^2 + \gamma_1 \boldsymbol{I}\big)^{-1}\Big]$ — divergence

$\boldsymbol{r}_2 \leftarrow \frac{1}{1-\alpha_1}\big(\widehat{\boldsymbol{x}}_1 - \alpha_1 \boldsymbol{r}_1\big)$ — Onsager correction

$\gamma_2 \leftarrow \gamma_1 \frac{1-\alpha_1}{\alpha_1}$ — precision of $\boldsymbol{r}_2$

$\widehat{\boldsymbol{x}}_2 \leftarrow \boldsymbol{g}(\boldsymbol{r}_2; \gamma_2)$ — (scalar) denoising

$\alpha_2 \leftarrow \frac{1}{N}\operatorname{Tr}\Big[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{r}}(\boldsymbol{r}_2; \gamma_2)\Big]$ — divergence

$\boldsymbol{r}_1 \leftarrow \frac{1}{1-\alpha_2}\big(\widehat{\boldsymbol{x}}_2 - \alpha_2 \boldsymbol{r}_2\big)$ — Onsager correction

$\gamma_1 \leftarrow \gamma_2 \frac{1-\alpha_2}{\alpha_2}$ — precision of $\boldsymbol{r}_1$

end

MMSE-VAMP interpreted

initialize $\boldsymbol{r}_1, \gamma_1$

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

$\widehat{\boldsymbol{x}}_1 \leftarrow$MMSE estimate of $\boldsymbol{x}$ under pseudo-prior $\mathcal{N}(\boldsymbol{r}_1, \boldsymbol{I}/\gamma_1)$ & measurement $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$

$\boldsymbol{r}_2 \leftarrow$linear cancellation of $\boldsymbol{r}_1$ from $\widehat{\boldsymbol{x}}_1$

$\widehat{\boldsymbol{x}}_2 \leftarrow$MMSE estimate of $\boldsymbol{x}$ under prior $p(\boldsymbol{x})$ and pseudo-measurement $\boldsymbol{r}_2 = \boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}/\gamma_2)$

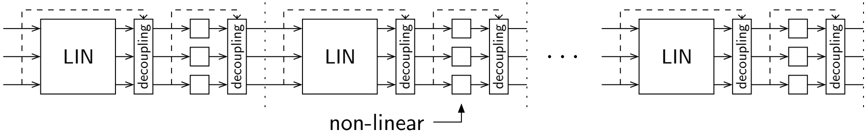
$\boldsymbol{r}_1 \leftarrow$linear cancellation of $\boldsymbol{r}_2$ from $\widehat{\boldsymbol{x}}_2$

end

Linear cancellation "decouples" the iterations, so that global MMSE problem can be tackled by solving simpler local MMSE problems.

Unfolding VAMP

Unfolding the VAMP algorithm gives the network[7]



Notice the two decoupling stages in each layer.

[7] Borgerding/Schniter, IEEE-TSP'17

Learning the network parameters

After unfolding an algorithm, one can use backpropagation (or similar) to "learn" the optimal network parameters.[8]

- Linear stage: $\widehat{\boldsymbol{x}}_1 = \boldsymbol{B}\boldsymbol{r}_1 + \boldsymbol{C}\boldsymbol{y}$
  $\rightarrow$ learn $(\boldsymbol{B}, \boldsymbol{C})$ for each layer.
- Nonlinear stage: $\widehat{x}_{2,j} = g(r_{2,j})\ \forall j$
  $\rightarrow$ learn a scalar function $g(\cdot)$ for each layer.
  e.g., spline, piecewise linear, etc.

[8] Gregor/LeCun, ICML'10.

Result of learning

Suppose that the training data $\{\boldsymbol{y}^{(d)}, \boldsymbol{x}^{(d)}\}_{d=1}^{D}$ were constructed using
1) iid $x_j^{(d)} \sim p$
2) $\boldsymbol{y}^{(d)} = \boldsymbol{A}\boldsymbol{x}^{(d)} + \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$
3) right-rotationally invariant $\boldsymbol{A}$.

Backpropagation recovers the parameter settings $(\boldsymbol{B}, \boldsymbol{C}, g)$ originally prescribed by the VAMP algorithm!

Result of learning

Suppose that the training data $\{\boldsymbol{y}^{(d)}, \boldsymbol{x}^{(d)}\}_{d=1}^{D}$ were constructed using
1) iid $x_j^{(d)} \sim p$
2) $\boldsymbol{y}^{(d)} = \boldsymbol{A}\boldsymbol{x}^{(d)} + \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$
3) right-rotationally invariant $\boldsymbol{A}$.

Backpropagation recovers the parameter settings $(\boldsymbol{B}, \boldsymbol{C}, g)$ originally prescribed by the VAMP algorithm!

- The learned linear stages are MMSE under pseudo-prior $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{r}_1, \boldsymbol{I}/\gamma_1)$ & measurement $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$
- The learned scalar nonlinearities are MMSE under prior $x_j \sim p$ and pseudo-measurements $r_{2,j} = x_j + \mathcal{N}(0, 1/\gamma_2)$

$\rightsquigarrow$ *This deep network is interpretable!*

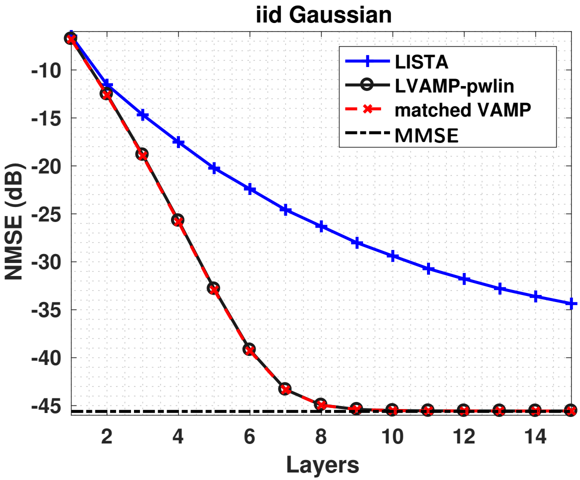
Implications for training

Due to the decoupling stages...

- each linear stage is locally MSE optimal, so
  $\rightsquigarrow$ *the linear params* $(\boldsymbol{B}, \boldsymbol{C})$ *can be learned locally in each layer!*
- each non-linear stage is locally MSE optimal, so
  $\rightsquigarrow$ *the nonlinear function* $g(\cdot)$ *can be learned locally in each layer!*

This deep network is easy to train!

Example with iid Gaussian $\boldsymbol{A}$



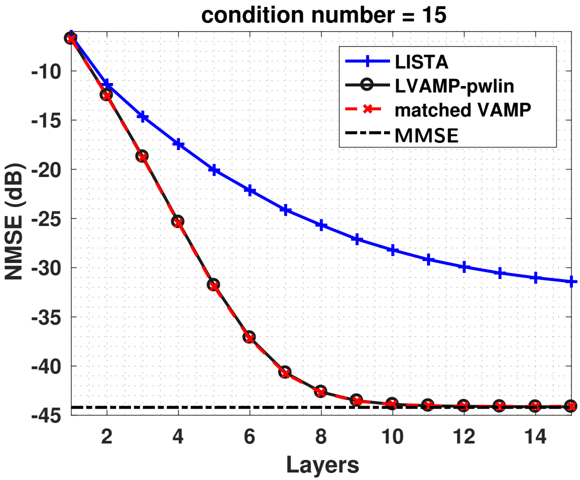
$n = 1024$
$m/n = 0.5$

$\boldsymbol{A} \sim$ iid $\mathcal{N}(0, 1)$

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

SNR $= 40$ dB

Example with non-iid Gaussian $\boldsymbol{A}$



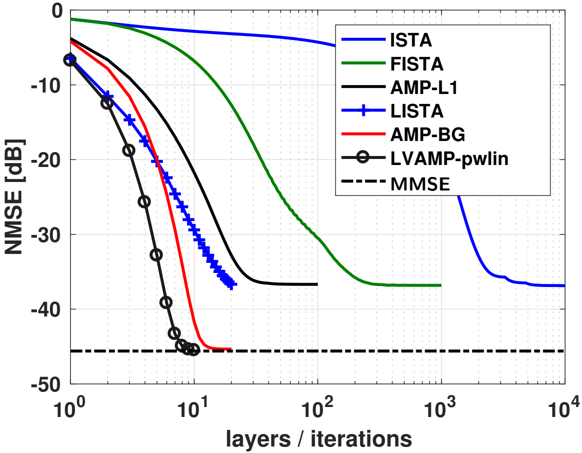
$n = 1024$
$m/n = 0.5$

$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\mathsf{T}$
$\boldsymbol{U}, \boldsymbol{V} \sim$ Haar
$s_n / s_{n-1} = \phi \ \forall n$

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

SNR $= 40$ dB

Deep nets vs algorithms



$n = 1024$
$m/n = 0.5$

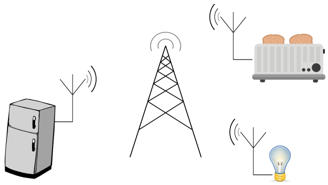
$\boldsymbol{A} \sim$ iid $\mathcal{N}(0, 1)$

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

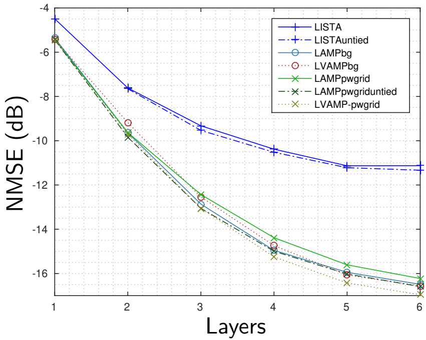
SNR $= 40$ dB

Application: Compressive random access

- Devices periodically wake up and broadcast short data bursts.
- The BS must simultaneous detect which devices are active and estimate their multipath gains.
- We use deep networks for this.



“Internet of Things”

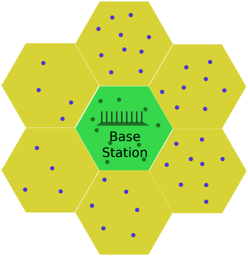


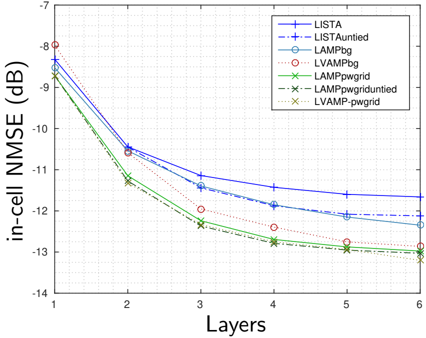
Experiment:

- 512 users, 1% active
- single antenna BS
- Pilots: i.i.d. QPSK, length 64
- flat Ricean fading
- SNR: 10dB

Application: Massive-MIMO channel estimation

- Massive-array BS communicates with single-antenna mobiles.
- Pilot reuse contaminates channel estimates. Random pilots can alleviate this problem.
- We use deep networks to simultaneously estimate channels of in-cell and out-of-cell users.





Experiment:

- BS: 64-element ULA
- 64 in-cell users, 448 total users
- Pilots: i.i.d. QPSK, length 64
- flat Ricean fading
- SNR: 20dB

Conclusions

- Our goal is to understand the design and interpretation of deep nets.
- For this talk, we focused on the task of sparse linear regression.
- We proposed a deep net that is
  1) asymptotically MSE-optimal (for iid $\boldsymbol{x}$ and RRI $\boldsymbol{A}$)
  2) interpretable: ...LMMSE/decoupling/NL-MMSE/decoupling ...
  3) locally trainable.
- The proposed network is obtained by "unfolding" the VAMP algorithm and learning its parameters.
- We demonstrated the approach in wireless comms applications.

Thanks!

TensorFlow code at

https://github.com/mborgerding/onsager_deep_learning