Learned Versions of Sparse Recovery Algorithms

Unnikrishnan N

Indian Institute of Science

unnikrishnann@iisc.ac.in

February 17, 2020

Overview

Compressed Sensing

Model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{z}$$

where $\mathbf{y}, \mathbf{z} \in \mathbb{R}^M$, $\mathbf{x} \in \mathbb{R}^N$ and $\mathbf{A} \in \mathbb{R}^{M \times N}$

Problem Statement

Under lasso setting

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$$

where $\lambda > 0$ is a tunable parameter that controls the tradeoff between sparsity and measurement fidelity in $\widehat{\boldsymbol{x}}$.

Under Bayesian setting

$$\widehat{\mathbf{x}}_{\text{MAP}} = \arg\max_{\mathbf{x}} \, p\left(\mathbf{x}|\mathbf{y};\sigma^2\right)$$

with sparse promoting prior.

Classical Sparse Recovery Algorithms

Greedy algorithms

Such as MP, OMP, IHT etc. They are fast.

Convex Relaxed

Such as BP, ISTA, AMP etc. They are slow and recovery performance is better than greedy algorithms

Non Convex

Such as SBL. Its slow but performance is better than both convex relaxed and greedy techniques.

Both Convex relaxed and Non Convex algorithms are iterative in nature.

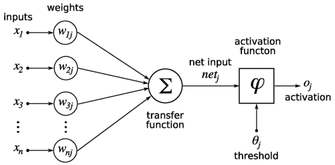
DNN based Sparse Recovery

Deep Neural Networks

Definition

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input

- Learning is done via supervised or unsupervised manner.
- Architecture can be dense, convolutional, etc.
- Number of layers ≥ 3.
- Optimizers can be RMSprop, SGD, Adam etc.
- Activation functions can be ReLU, Linear, sigmoid etc.
- Cost functions can be MSE, binary cross entropy etc..



Single Neuron functionality: $\mathbf{z} = \varphi(\mathbf{w}_i^T \mathbf{x} + \mathbf{b})$

DNN based Sparse Recovery

Motivation

- Universal approximation theorem ensures the existence of mapping $\hat{\mathbf{x}} \approx f(\mathbf{y})$
- Increased performance in terms of computational complexity and NMSE.

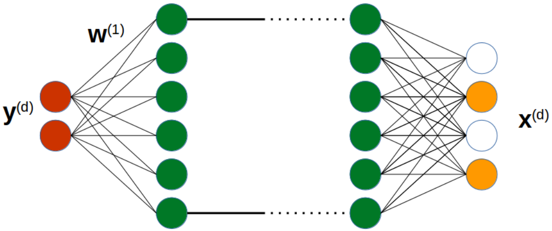
Types of SRA

- Blind sparse vector recovery. Uses existing architectures for sparse recovery.
- Model based Approach. Mimic existing sparse recovery algorithms with deep neural networks. eg: LISTA, LIHT and LSBL.

Supervised Learning of SRA

Training data $\{(\mathbf{y}^{(d)}, \mathbf{x}^{(d)})\}_{d=1}^{d=D}$, with labels $\mathbf{x}^{(d)}$ are continuous, high dimensional and sparse.

SRA using Dense Neural Networks



Inferences

- Training data size is huge (order of 1e6)
- No of trainable parameters increases drastically with input dimension
- Recovery performance
  - Depends on no of trainable parameters used, no of epochs, no of different types of inputs used
  - Better than OMP (wrt NMSE), but inferior to that of l1 recovery algorithms

Learned Iterative Shrinkage Thresholding Algorithm

ISTA derivation

$$
\begin{aligned}
J &= \frac{1}{2}||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2 + \lambda||\mathbf{x}||_1 = J_1(\mathbf{x}) + J_2(\mathbf{x}) \\
\nabla_x(J_1) &= \mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{A}^T\mathbf{y} = -\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}) \\
\frac{\partial J_1}{\partial x_j} &= -(\mathbf{a}_{*j}^T\mathbf{y} - \sum_{i=1}^{N}\mathbf{a}_{*j}^T\mathbf{a}_{*i}x_i) \\
&= -\mathbf{a}_{*j}^T(\mathbf{y} - \sum_{i\neq j}^{N}\mathbf{a}_{*i}x_i) + ||\mathbf{a}_{*j}||_2^2 x_j \\
&= -\rho_j + x_j
\end{aligned}
$$

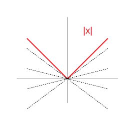
Now define subgradient as

$$
\partial f(x) = \left\{ y | f(z) \geq f(x) + y^T(z - x) \text{ for all } z \in \text{dom } f \right\}
$$

SRA using Learned ISTA

ISTA derivation continues

$$\frac{\partial J_2}{\partial x_j} = \begin{cases} -\lambda, & \text{if } x_j < 0 \\ [-\lambda, \lambda], & \text{if } x_j = 0 \\ \lambda, & \text{if } x_j > 0 \end{cases}$$



$$\frac{\partial J}{\partial x_j} = \begin{cases} -\lambda - \rho_j + x_j, & \text{if } x_j < 0 \\ [-\lambda - \rho_j, \lambda - \rho_j], & \text{if } x_j = 0 \\ \lambda - \rho_j + x_j, & \text{if } x_j > 0 \end{cases}$$

$$x_j^{\text{opt}} = \begin{cases} \lambda + \rho_j, & \text{if } \rho_j < -\lambda \\ 0, & \text{if } \rho_j \in [-\lambda, \lambda] \\ \rho_j - \lambda, & \text{if } \rho_j > \lambda \end{cases}$$

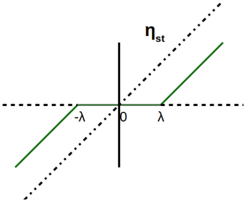
shrinkage function $\eta_{\text{st}}(\rho_j) = sign(\rho_j)(|\rho_j| - \lambda)_+$

SRA using Learned ISTA

ISTA derivation continues

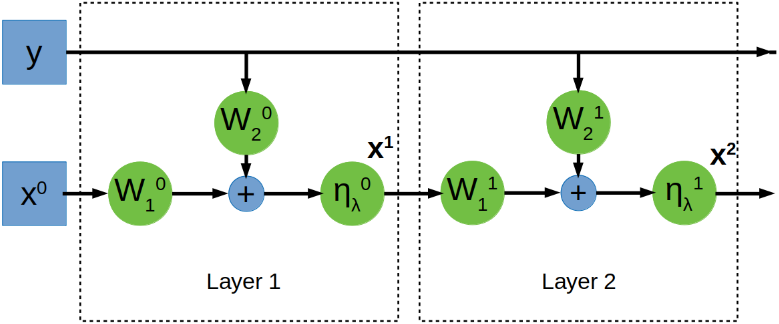
$$\mathbf{x}^{k+1} = \eta_{\text{st}}(\mathbf{x}^k + \beta \mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}^k))$$
$$= \eta_{\text{st}}((\mathbf{I} - \beta \mathbf{A}^T\mathbf{A})\mathbf{x}^k + \beta \mathbf{A}^T\mathbf{y})$$
$$\mathbf{x}^{k+1} = \eta_{\lambda^k}(\mathbf{W}_2^k\mathbf{x}^k + \mathbf{W}_1^k\mathbf{y})$$

where $\beta \in (0, \frac{1}{||\mathbf{A}^T\mathbf{A}||_2}]$ and $\eta_\lambda(\cdot) = \eta_{\text{st}}(\cdot)$



Soft thresholding function

Unfolded Structure of LISTA



Training data	$\{(\mathbf{y}_{(d)}, \mathbf{x}^*_{(d)})\}_{d=1}^{d=D}$
No of Layers	K
Parameters to be learned	$\Theta = \left\{\left(W_1^k, W_2^k, \lambda^k\right)\right\}_{k=0}^{K-1}$
Cost function	$\underset{\Theta}{\text{minimize}} \quad \mathbb{E}_{\mathbf{x}^*,\mathbf{y}} \left\|\mathbf{x}^K\left(\Theta, \mathbf{y}, \mathbf{x}^0\right) - \mathbf{x}^*\right\|_2^2$
Back propagation algorithm	SGD

Modified LISTA

Modification 1 - Partial weight Coupling (CP)

$$\mathbf{x}^{k+1} = \eta_{\text{st}}(\mathbf{x}^k + \beta \mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}^k))$$
$$\mathbf{x}^{k+1} = \eta_{\lambda^k}(\mathbf{x}^k + (\mathbf{W}_1^k)^T(\mathbf{y} - \mathbf{A}\mathbf{x}^k))$$

A is absorbed in non trainable parameters.
Trainable parameters reduced to $\Theta = \left\{\left(W_1^k, \lambda^k\right)\right\}_{k=0}^{K-1}$

Modification 2 - Support Selection (SS)

At each LISTA layer (k^{th} layer) before applying soft thresholding, select a certain percentage ($p^k\%$) of entries with largest magnitudes, and trust them as true support and won't pass them through thresholding.

$$\mathbf{x}^{k+1} = \eta_{\lambda^k}^{p^k}(\mathbf{W}_2^k \mathbf{x}^k + \mathbf{W}_1^k \mathbf{y})$$

Modification 2 Continues

$$(\eta^{p^k}_{\lambda^k}(v))_i = \begin{cases} v_i, & \textit{if} \quad |v_i| > \lambda^k \quad \text{and} \quad i \in S^{p^k}(v) \\ \eta_{\lambda^k}(v_i), & \textit{if} \quad i \notin S^{p^k}(v) \end{cases}$$

where $S^{p^k}(v)$ is the support set of $p\%$ largest magnitude entries in k^{th} layer, $\mathbf{v} = \mathbf{W}_2^k\mathbf{x}^k + \mathbf{W}_1^k\mathbf{y}$

Combined model - LISTA-cpss

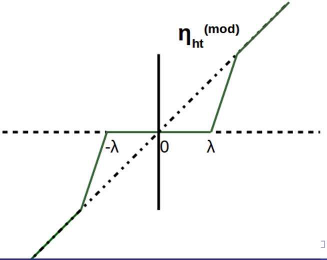
$$\mathbf{x}^{k+1} = \eta^{p^k}_{\lambda^k}(\mathbf{x}^k + (\mathbf{W}_1^k)^T(\mathbf{y} - \mathbf{A}\mathbf{x}^k))$$

Learned Iterative Hard Thresholding (LIHT)

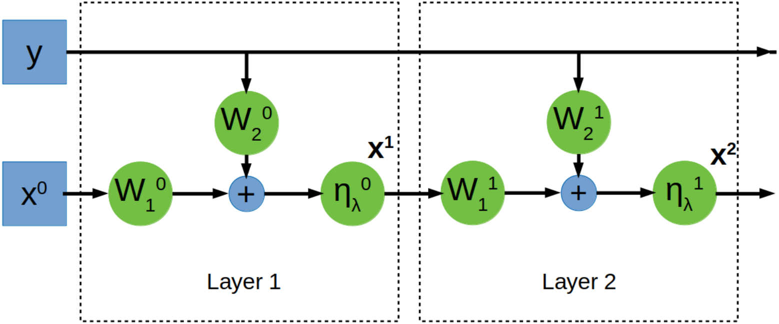
IHT

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{h}_\lambda(\mathbf{x}^k + \beta \mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}^k)) \\ &= \mathbf{h}_\lambda((\mathbf{I} - \beta \mathbf{A}^T\mathbf{A})\mathbf{x}^k + \beta \mathbf{A}^T\mathbf{y}) \\ &= \mathbf{h}_\lambda(\mathbf{W}_2^k\mathbf{x}^k + \mathbf{W}_1^k\mathbf{y})\end{aligned}$$

where $\beta \in (0, \frac{1}{||\mathbf{A}^T\mathbf{A}||_2}]$ and $\mathbf{h}_\lambda$ is modified hard thresholding function.



Unfolded Structure of LIHT



Training data	$\{(\mathbf{y}_{(d)}, \mathbf{x}^*_{(d)})\}_{d=1}^{d=D}$
No of Layers	K
Parameters to be learned	$\Theta = \left\{\left(W_1^k, W_2^k, \lambda^k\right)\right\}_{k=0}^{K-1}$
Cost function	$\underset{\Theta}{minimize} \quad \mathbb{E}_{\mathbf{x}^*,\mathbf{y}} \left\Vert \mathbf{x}^K\left(\Theta, \mathbf{y}, \mathbf{x}^0\right) - \mathbf{x}^* \right\Vert_2^2$
Back propagation algorithm	SGD

SRA using Learned Sparse Bayesian Learning

SBL - problem formulation

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{z}$$

where $\mathbf{y}, \mathbf{z} \in \mathbb{R}^M$, $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\mathbf{x} \in \mathbb{R}^N$, $||\mathbf{x}||_0 \leq K < M \ll N$

$\mathbf{z} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ and $\mathbf{x} \sim \mathcal{N}(0, \mathbf{R_x})$, $\mathbf{R_x} = diag\{\frac{1}{\alpha_1}, \ldots, \frac{1}{\alpha_N}\}$

Find sparsest $\mathbf{x}$ from $\mathbf{y}$ under gaussian prior.

Solution

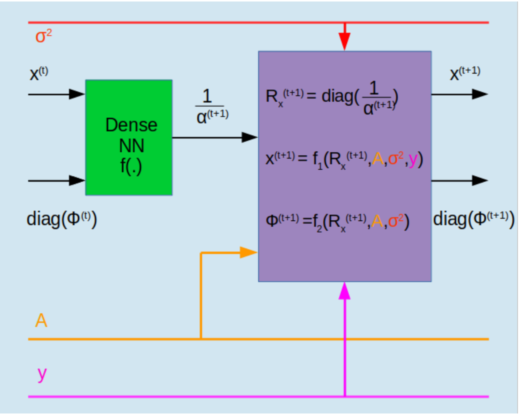
$$\begin{aligned}\hat{\mathbf{x}}_{MAP} &= \arg\max_{\mathbf{x}} \, p(\mathbf{x}|\mathbf{y}, \sigma^2, \mathbf{R_x}) \\ &= \mathbf{R_x}\mathbf{A}^T(\mathbf{A}\mathbf{R_x}\mathbf{A}^T + \sigma^2\mathbf{I})^{-1}\mathbf{y}\end{aligned}$$

Estimation of α_i is by EM algorithm. t^{th} iterate is

$$\frac{1}{\alpha_i^t} = (x_i^{t-1})^2 + (\mathbf{\Phi}^{t-1})_{i,i}$$

Where $\mathbf{\Phi}$ is the error covariance matrix. Once the estimate of $\alpha_i \forall i \in [N]$ is known then using above equation sparse solution can be estimated.

L-SBL Block Diagram

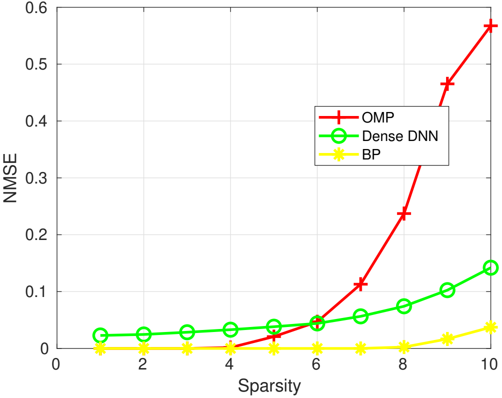


$$\frac{1}{\alpha^{t+1}} = f(\mathbf{x}^t, \mathbf{x}^{t-1}, \ldots, \boldsymbol{\Phi}^t, \boldsymbol{\Phi}^{t-1}, \ldots)$$

$$\mathbf{x}^{t+1} = \mathbf{R_x A}^T(\mathbf{A R_x A}^T + \sigma^2 \mathbf{I})^{-1}\mathbf{y}$$

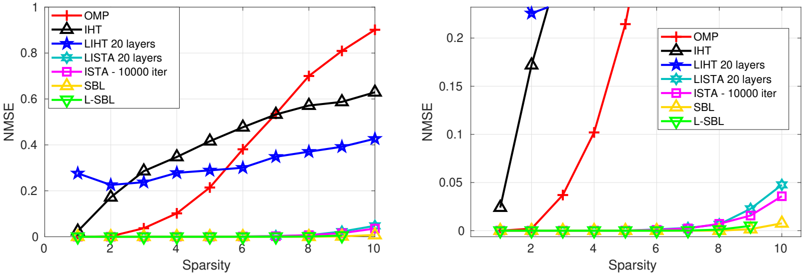
$$\boldsymbol{\Phi}^{t+1} = \mathbf{R_x} - \mathbf{R_x A}^T(\mathbf{A R_x A}^T + \sigma^2 \mathbf{I})^{-1}\mathbf{A R_x}$$

Performance Comparison



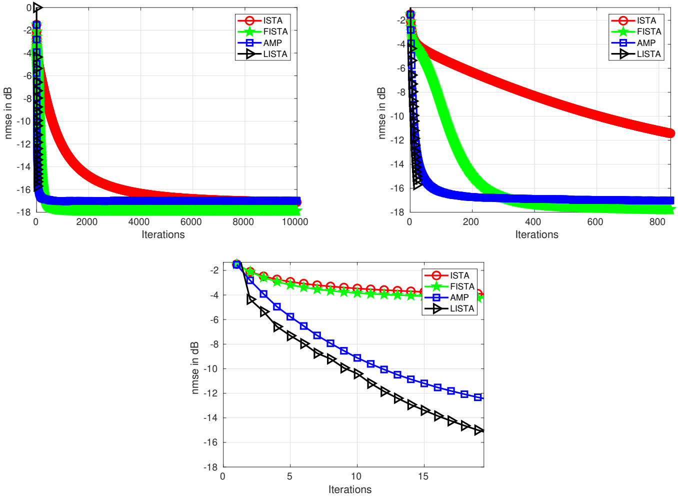
NMSE plot for OMP, Dense and BP against sparsity under noiseless setup

Performance Comparison Continues



NMSE plot for OMP, IHT, ISTA, LISTA, SBL and LSBL sparsity under noiseless setup

Convergence Comparison



Convergence of ISTA, FISTA, AMP and LISTA for $\mathbf{A} \in \mathbb{R}^{M\times N}$ $M = 30, N = 50$, sparsity around 10

Conclusion And Future Scope

Conclusion

1. Tested LISTA-cpss, LIHT and LSBL.
2. LISTA-cpss
   - Each layer of LISTA is equivalent to one iteration of ISTA.
   - Faster convergence in LISTA compared to ISTA.
   - For any measurement matrix (under certain conditions) LISTA network is trainable.
   - Faster training (with less number of inputs(order of 1e3) and less number of trainable parameters).
   - Less computational complexity.
3. Various sparse recovery algorithms such as OMP, IHT, BP, AMP, ISTA, FISTA, SBL are tested and results are compared
4. NMSE performance of LISTA is similar to ISTA
5. LSBL outperforms LISTA and LIHT

Future Scope

1. Try support selection in L-SBL for better performances. Incorporate sparsity information to deep neural net model.
2. Train these networks with structured sparse inputs.

References

- Chen, Xiaohan, Liu, Jialin, Wang, Zhangyang, Yin, Wotao,"Theoretical Linear Convergence of Unfolded ISTA and its Practical Weights and Thresholds", Advances in Neural Information Processing Systems 2018 Conference paper.
- Karol Gregor and Yann LeCun, "Learning fast approximations of sparse coding," In Proceedings of the 27th International Conference on International Conference on Machine Learning, pages 399406. Omnipress, 2010.
- Ali Mousavi and Richard G. Baraniuk, "Learning to invert: Signal recovery via Deep Convolutional Networks,"in International Conference on Acoustics, Speech and Signal Processing, ICASSP, New Orleans, LA, USA, March 5-9, 2017.
- M. Borgerding and P. Schniter, Onsager-corrected deep learning for sparse linear inverse problems, in 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Dec 2016, pp. 227231.
- C. R. Murthy and R. J. Peter, Learned-sbl: A deep learning architecture for sparse signal recovery, cs.IT, vol. abs/1909.08185, 2019. [Online]. Available: https://arxiv.org/abs/1909.08185.

I. Daubechies, M. Defrise, and C. De Mol,"An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," Communications on Pure and Applied Mathematics 57 (11): 1413–1457 (2004).

Zhangyang Wang, Qing Ling, and Thomas Huang, "Learning deep l0 encoders,". In AAAI Conference on Artificial Intelligence, pages 21942200, 2016.

Thank You