Interpretable Learned LASSO

Ye Yuan
ye.yuan@tamu.edu

Xiaohan Chen
chernxh@tamu.edu

Abstract

Solving LASSO (Least Absolute Shrinkage and Selection Operator) is an essential problem in compressive sensing or sparse coding. Numerous algorithms have been proposed to efficiently solve LASSO, such as ISTA (Iterative Shrinkage Thresholding Algorithm), FISTA (Fast ISTA) and AMP (Approximated Message Passing). Above iterative algorithms iteratively perform linear transformation and non-linear mapping (soft thresholding) on inputs, which can be seen as a deep neural network with shared, hand-crafted weights. Therefore, iterative algorithms solving LASSO could be unfolded into deep models whose weights are trained with back-propagation provided with training data. However, once weights in iterative algorithms are changed, good interpretibility of LASSO such as feature selection is lost. In this paper, we propose a variant of learned iterative algorithms which can also be learned with training data while trying our best to preserve the interpretibility of LASSO.

1. Introduction

Our starting point is to solve the least absolute shrinkage and selection operator (LASSO) formulated as below.

$$\min_x \frac{1}{2}||y - Ax||_2^2 + \lambda||x||_1 \tag{1}$$

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

Solving LASSO is an essential problem in compressive sensing and sparse coding. Numerous algorithms have been proposed to solve LASSO. In our project, we are focused with iterative algorithms, ISTA more specifically, which is short for 'iterative shrinkage thresholding algorithm'. Given signal y and dictionary A, ISTA iteratively calculates latent variable x with

$$\begin{aligned} x^{k+1} =& \eta_\tau(x^k + \frac{1}{L}A^T(y - Ax^k)) \quad (2)\\ =& \eta_\tau(By + Wx^k), \quad (3) \end{aligned}$$

and $x^0 = \mathbf{0}, L \geq \lambda_{max}(A)$.

The block diagram of ISTA is shown in Figure 1, in which ISTA can be seen as a deep neural network with shared and hand-crafted weights. From this perspective, [3] proposed a deep model based on ISTA, called Learned ISTA (LISTA), which unfolds ISTA iterations and truncates into certain number of steps, feeds training data and tunes the weights with back-propagation. Emperical results show that an 8-layer LISTA network can produce the same MSE as ISTA after thousands of iterations, which enables us to do some instant application of compressive sensing and sparse coding.

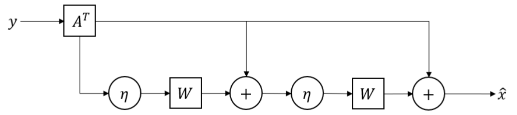

Figure 1. ISTA block diagram. ISTA reads a signal y as inputs, iteratively perfomrs linear transformation and non-linear mapping, which is very similar to a deep neural network.

However, although LISTA is quick, it changes weights in ISTA, losing good theoretical properties of ISTA such as feature selection. Here naturally comes the question: how can we achieve the same performance as deep learning model while preserving good interpretibility of ISTA?

2. Previous Works

In this section, I will briefly introduce ISTA algorithm and AMP algorithm and how LISTA and LAMP work.

2.1. ISTA

ISTA (iterative shrinkage thresholding algorithm) is formulated as:

$$\begin{aligned} x^{k+1} =& \eta_\tau(x^k + \frac{1}{L}A^T(y - Ax^k)) \quad (4)\\ =& \eta_\tau(By + Wx^k), \quad (5) \end{aligned}$$

ISTA is a proximal gradient algorithm, which can be factored into two steps:

$$x_{prox}^{k+1} = x^k + \frac{1}{L}A^T(y - Ax^k), \tag{6}$$

$$x^{k+1} = min_x||x - x_{prox}^{k+1}||_2^2 + \lambda||x||_1. \tag{7}$$

Equation (6) is a gradient descent step with respect to the l_2 norm term in equation (1) with step length $\frac{1}{L}$. And equation (7) is a proximal step which takes into consideration the l_1 norm term in LASSO and introduces the soft thresholding function

$$\eta_\tau(r) = \begin{cases} r - \tau & , r > \tau \\ 0 & , -\tau \leq r \leq \tau \\ r + \tau & , r < \tau \end{cases} \tag{8}$$

2.2. AMP

AMP (approximated messaging passing) algorithm [2] is formulated as:

$$\begin{aligned} z^k &= y - Ax^k + \frac{||x^k||_0}{M} z^{k-1} \\ x^{k+1} &= \eta_{\tau^k}(x^k + A^T z^k) \end{aligned}$$

where $\tau^k = \frac{\alpha ||z^k||_2}{\sqrt{M}}$.

AMP is no more than ISTA but a Onsager correction term $\frac{||x^k||_0}{M} z^{k-1}$ and the threshold τ^k is time dependent. The Onsager correction term is very powerful, largely speeding up the convergence of LASSO, shown in Figure 2.

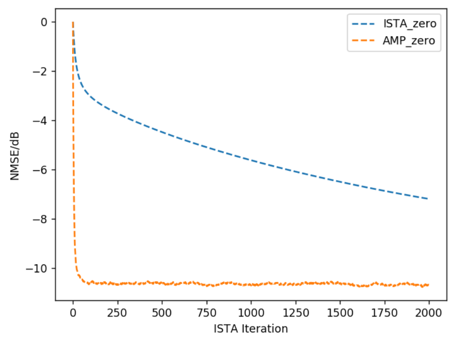

Figure 2. Convergence speed of ISTA and AMP.

2.3. Learned ISTA and Learned AMP

Learned ISTA (LISTA) algorithm was first proposed by LeCun *et al.* in 2010. The main philosophy of LISTA is to unfold iterative algorithm — ISTA, truncate to certain number of steps, say T and train the weights or dictionary in LASSO with training data. The structure of LISTA is shown in Figure 1.

The same philosophy can also be applied to AMP algorithm, which will lead to a slightly more complicated model, named LAMP [1] shown in Figure 3.

Although there is no strict theoretical proof so far, emprical results show that LISTA can largely accelerate the convergence. As you can see in Figure 4.

3. Our Method

In LISTA and LAMP, the estimation of x starts from origin $x^0 = \mathbf{0}$, which is not a good starting point. A start as simple as $x^0 = A^T y$ will be much better than the origin.

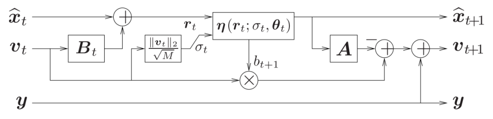
$\widehat{\boldsymbol{x}}_t$
$\boldsymbol{v}_t$
$\boldsymbol{y}$
$\boldsymbol{B}_t$
$\frac{\|\boldsymbol{v}_t\|_2}{\sqrt{M}}$
$\boldsymbol{r}_t$
σ_t
$\boldsymbol{\eta}(\boldsymbol{r}_t; \sigma_t, \boldsymbol{\theta}_t)$
b_{t+1}
$\boldsymbol{A}$
$-$
$\widehat{\boldsymbol{x}}_{t+1}$
$\boldsymbol{v}_{t+1}$
$\boldsymbol{y}$

Figure 3. The architecture of learned AMP model.

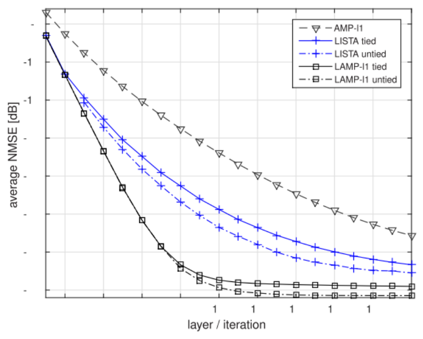

Figure 4. Convergence speed comparison of AMP, LISTA and LAMP.

This simple thought inspires us that what if we fix the weights of ISTA but learn the inputs into ISTA given signal y? Then we come to the idea to use a deep neural network $f : \mathbb{R}^M \to \mathbb{R}^N$ to maps signals y to $x^0 = f(y|\omega)$ as inputs to ISTA. After feeding our model with training data and training it via back-propagation, we expect our model at least can learn a good mapping with respect to data from a specific distribution. We will perform some numerical experiments with synthetic data.

Our model is shown in Figure 5.

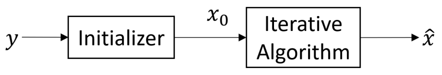

Figure 5. Our model consists of two parts: initializer and a several steps iterative algorithm.

The initializer in Figure 5 is a normal neural network, which reads signal y as input and outputs a initial estimator of sparse code, x^0, formulated as:

$$I(y; \mathbf{W}) : \mathbb{R}^M \to \mathbb{R}^N, y \mapsto x^0(y; \mathbf{W})$$

where $\mathbf{W}$ is the weights of initializer. We train this neural network with training data $(x_i, y_i)_{i=1}^D$ and back-propagation with respect to loss function

$$L_i(x^i, y^i) = ||\hat{x}(x^0(y^i; \mathbf{W})) - x^i||_2^2$$

4. Experiments

We conduct some numerical experiments on our model. We try initializer with different structures and use ISTA and

AMP algorithms as iterative algorithms in Figure 5.

4.1. Experimets Setting

The dimension of signal y is $M = 25$. The dimension of sparse code x is $N = 50$. Entries in sparse code x are non-zero with propability $p = 0.1$ independently. And all non-zero entries obey standard Gaussian distribution. The dictionary A is sampled from i.i.d. Gaussian distribution with zero means and standard deviation $1/\sqrt{M}$. Our problem is noiseless.

4.2. Experment Results

We implement one-layer initializer with sigmoid, leaky ReLU, PReLU, and soft thresholding activation and two-layer initializer with leaky ReLU and PReLU activation, and concatenate our initializer with one-layer, two-layer, three-layer, four-layer ISTA and two-layer, three-layer AMP. We compare the performance of our model with standard ISTA, standard AMP with zero input, transposition input $A^T y$ and pseudo-inverse input $A^T(AA^T)^{-1}y$ as base line.

Figure 6 shows performance with ISTA following a two-layer initializer with leaky ReLU activation compared with standard ISTA algorithm with zero, transposition and pseudo-inverse inputs. As you can see, although transposition input is not good at first but it converges much faster than zero input. And as expected, pseudo-inverse is a much better initialization than zero and transposition inputs. However, our model achieves better performance than all of the above initialization.

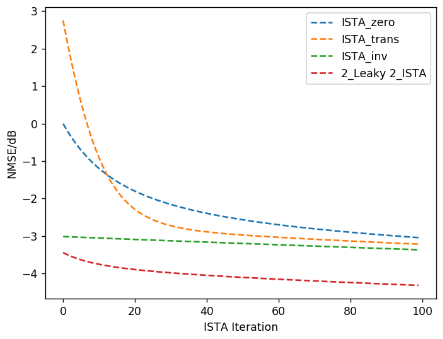

Figure 6. The plot of NMSE against iteration with iteration of two-layer initializer with leaky ReLU activation followed by two-step ISTA compared with standard ISTA algorithm with zero, transposition and pseudo-inverse inputs.

We also compare between the performance of initializers with different structures, shown in Figure 7 and between the same initializer followed by different number of layers iterative algorithm, shown in Figure 8

Figure 9 shows comparison of performance between two and three steps AMP following two-layer initializer with

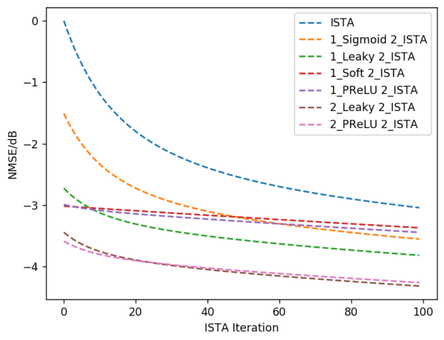

Figure 7. Compare performance of different initializers.

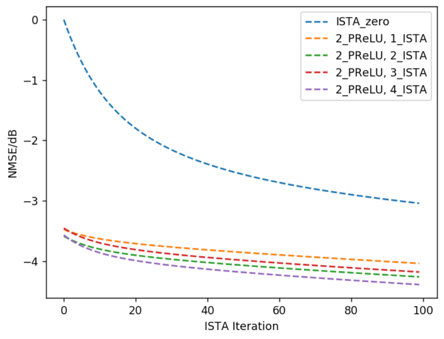

Figure 8. Compare performance of the same initializer followed by ISTA with different numbers of layers.

leaky ReLU activation and standard AMP algorithm with zero, transposition and pseudo-inverse inputs. There are a lot of interesting phenomenon in this experiment:

- Transposition and pseudo-inverse inputs are not always good for AMP algorithm. Actually pseudo-inverse input is not solvable for AMP.
- Deeper initializer doesn't mean better performance.

5. Conclusions and Futur Work

We propose a variant of learned ISTA and learned AMP models in which we keep the dictionaries in LASSO unchanged but learn a better initialization of standard iterative algorithms such as ISTA and AMP to contain the interpretability of LASSO, say feature selection.

Emperical results shows that our model can speed up the convergence of iterative algorithms to some extent.

For future work, we will need to work more on how the interpretibility of LASSO is preserved, for which we might look into the support selection status of intermediate steps of ISTA and AMP.

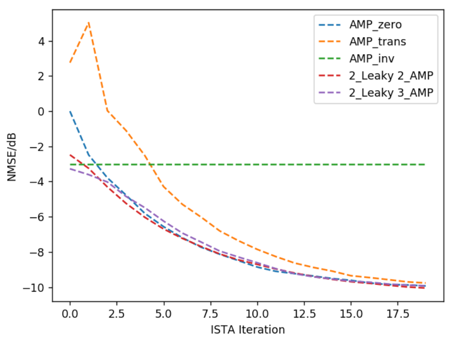

Figure 9. The plot of NMSE against iteration with iteration of two-layer initializer with leaky ReLU activation followed by two-step and three-step AMP compared with standard AMP algorithm with zero, transposition and pseudo-inverse inputs.

References

[1] M. Borgerding, P. Schniter, and S. Rangan. Amp-inspired deep networks for sparse linear inverse problems. *IEEE Transactions on Signal Processing*, 2017.

[2] D. L. Donoho, A. Maleki, and A. Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

[3] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 399–406, 2010.