Theoretical Linear Convergence of Unfolded ISTA and its Practical Weights and Thresholds

By X. CHEN, J. LIU, Z. WANG, W.YIN at NeurIPS 2018

Leo DAVY   Martin GJORGJEVSKI

ENS Lyon
M2 Advanced Mathematics

March 2022

Introduction

- LISTA - A Neural Network designed for sparse recovery
- Coupling of parameters
- Speed of convergence
- Experiments (Validating the results)
- Training the network (strategies and difficulties)

Introduction

A system of the form

$$Ax^* = b + \epsilon$$

with A m by n matrix, $x^* \in \mathbb{R}^n$, $b, \epsilon \in \mathbb{R}^m$ with $m << n$ is known as an inverse problem.

- Generally ill posed but under certain conditions sparse solutions for x^* exist and are unique
- Popular principle for finding them is the LASSO principle : $argmin_x \frac{1}{2}||b - Ax||^2 + \lambda||x||_1$
- Popular algorithms for solving this problem: ISTA and FISTA with error $O(\frac{1}{k})$ and $O(\frac{1}{k^2})$ (in function values) after k iterations

LISTA - definition

The algorithm ISTA is an iterative algorithm with iterative step

$$x_{k+1} = \eta_{\frac{\lambda}{L}}(x_k + \frac{1}{L}A^T(b - Ax_k))$$

The Idea behind LISTA is to introduce weight matrices and thresholding values at each step as follows:

$$x_{k+1} = \eta_{\theta^k}(W_1^k b + W_2^k x^k)$$

The parameters $\{W_1^k, W_2^k, \theta^k\}$ are to be found by training. We assume that we have training data set (x_n^*, ϵ_n) sampled according to a distribution on $\mathcal{X}(B, s, \sigma) = \{(x^*, \epsilon) | |x_i^*| \leq B, ||x^*||_0 \leq s, ||\epsilon||_1 \leq \sigma\}$

Coupling of parameters

Given a sequence $\{W_1^k, W_2^k, \theta^k\}_{k=1}^{\infty}$, b the observed value we denote $x^k(W_k, b)$ the layer wise outcomes of the neural network.

It is desirable to know that after certain number of iterations we can guarantee that the realization of the network approximates the true signal uniformly.

Necessary condition for uniform convergence in the noiseless case

If $x^k(W_k, b) \to x^*$ uniformly on $\mathcal{X}(B, s, 0)$ as $k \to \infty$ and $||W_2^k||$ are bounded then asymptotically:

- $W_2^k - (I - W_1^k A) \to 0$
- $\theta^k \to 0$

as $k \to \infty$

This asymptotic relation motivates the introduction of LISTA-CP where we have $W_2^k = I - W_1^k A)$ for all $k \geq 1$.

Convergence Analysis

Generalized Mutual Coherence

Given an m by n matrix A with normalized columns, the generalized mutual coherence $\bar{\mu}$ is defined as:

$$\bar{\mu}(A) = \inf_{W} \max_{1 \leq i,j \leq n, i \neq j} |W_i^T A_j|$$

Where the infimum is taken over all m by n matrices W such that $W_i^T A_i = 1$ for all $1 \leq i \leq n$.

It can be shown that there exist matrices which achieve this minimum, and they play a key role in speeding up the convergence of LISTA-CP.

Convergence Analysis

For a given matrix A, we denote $\Phi(A)$ the set of m by n matrices W that achieve the minimum for $\bar{\mu}(A)$. A weight matrix $W \in \Phi(A)$ is good if it is a solution to the problem

$$\min_{W \in \Phi(A)} \max_{i,j} |W_{i,j}|$$

We also denote with C_W the common value of $\max_{i,j} |W_{i,j}|$ for any good matrix W^k.

Convergence Analysis

Theorem: Convergence of LISTA-CP

If s is sufficiently small, then by setting

- W^k to be a good weight matrix for A
- $\theta^k = \bar{\mu}(A) \sup_{(x^*,\epsilon)\in\mathcal{X}(B,s,\sigma)} ||x^k(x^*,\epsilon) - x^*||_1 + C_W\sigma$

we have the following convergence guarantee (for all k):

$$||x^k(x^*,\epsilon) - x^*||_2 \leq sB\exp(-ck) + C\sigma$$

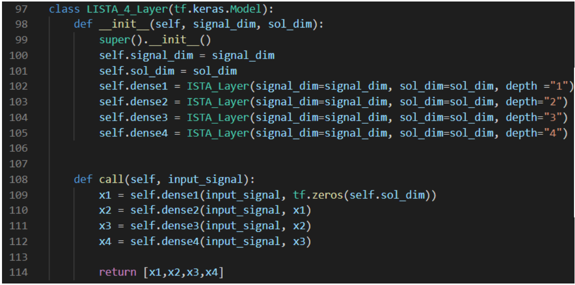
where $c > 0$, $C > 0$ depend only on A and s

In particular in the noiseless case $\sigma = 0$ we see that the proposed coupling is both necessary and sufficient to ensure convergence.

Building an L-layer LISTA Network

$$x^{k+1} = \eta_{\theta^k}(W_1^k b + W_2^k x^k), \quad \text{for } k = 1, \cdots, L$$

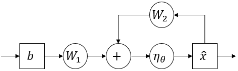
LISTA consists in applying L times a "one-layer LISTA", each of those layers having its own parameters (W_1^k, W_2^k, θ^k).

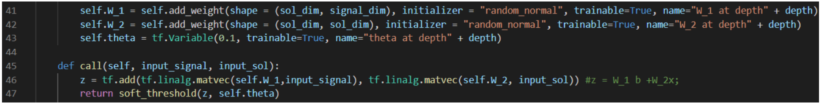


```python
class LISTA_4_Layer(tf.keras.Model):
    def __init__(self, signal_dim, sol_dim):
        super().__init__()
        self.signal_dim = signal_dim
        self.sol_dim = sol_dim
        self.dense1 = ISTA_Layer(signal_dim=signal_dim, sol_dim=sol_dim, depth ="1")
        self.dense2 = ISTA_Layer(signal_dim=signal_dim, sol_dim=sol_dim, depth="2")
        self.dense3 = ISTA_Layer(signal_dim=signal_dim, sol_dim=sol_dim, depth="3")
        self.dense4 = ISTA_Layer(signal_dim=signal_dim, sol_dim=sol_dim, depth="4")


    def call(self, input_signal):
        x1 = self.dense1(input_signal, tf.zeros(self.sol_dim))
        x2 = self.dense2(input_signal, x1)
        x3 = self.dense3(input_signal, x2)
        x4 = self.dense4(input_signal, x3)

        return [x1,x2,x3,x4]
```





```python
        self.W_1 = self.add_weight(shape = (sol_dim, signal_dim), initializer = "random_normal", trainable=True, name="W_1 at depth" + depth)
        self.W_2 = self.add_weight(shape = (sol_dim, sol_dim), initializer = "random_normal", trainable=True, name="W_2 at depth" + depth)
        self.theta = tf.Variable(0.1, trainable=True, name="theta at depth" + depth)

    def call(self, input_signal, input_sol):
        z = tf.add(tf.linalg.matvec(self.W_1,input_signal), tf.linalg.matvec(self.W_2, input_sol)) #z = W_1 b +W_2x;
        return soft_threshold(z, self.theta)
```

Code available at `https://github.com/DavyL/LISTA_DNN`

Learning a LISTA Network

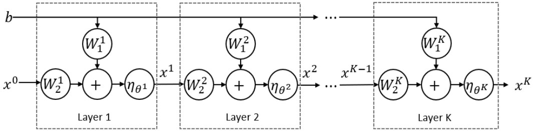


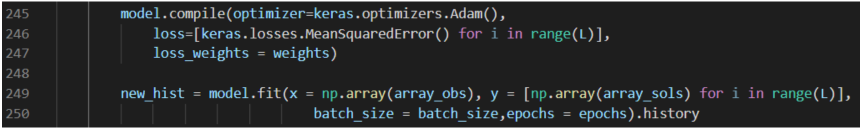
Figure: Unfolded LISTA Network

The goal is to minimize the last layer output MSE:

$$\mathbb{E}_{x^*,b}||x^L(\Theta,b)-x^*||_2^2$$

over all layers parameters:

$$\Theta = (W_1^k, W_2^k, \theta^k)_{k=1}^L.$$



```
model.compile(optimizer=keras.optimizers.Adam(),
    loss=[keras.losses.MeanSquaredError() for i in range(L)],
    loss_weights = weights)

new_hist = model.fit(x = np.array(array_obs), y = [np.array(array_sols) for i in range(L)],
                    batch_size = batch_size,epochs = epochs).history
```

Examples of recoveries through layers

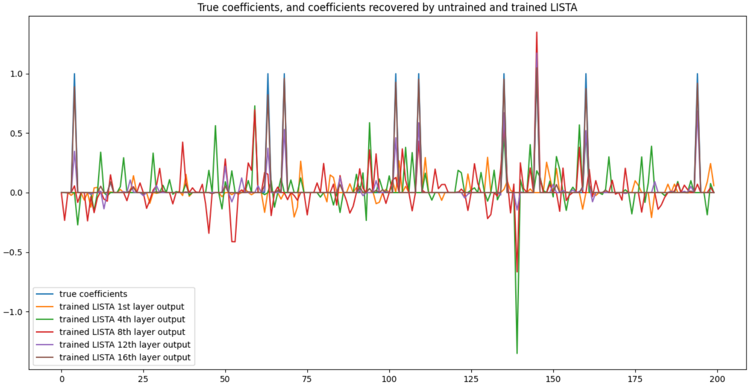


Figure: LISTA (CP) recovery through layers with 200 columns, 100 rows and 10000 samples

Learning layer by layer, or all at once ?

Two different strategies for learning:

1. As a traditional DNN, using only the last layer output
2. Enforcing progressive convergence of the *k*-th layer output towards the true solution, using layer by layer training

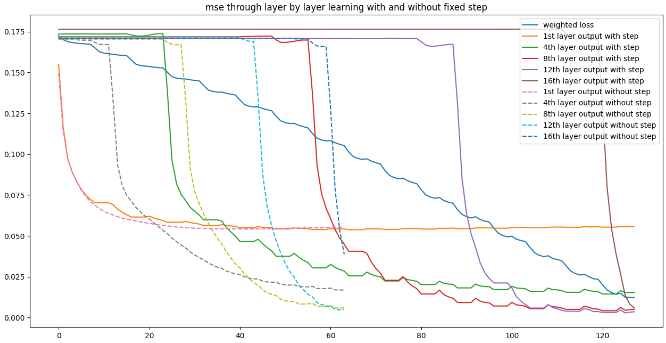


Figure: Two different strategies of layer by layer training

Layer by layer or all at once ?

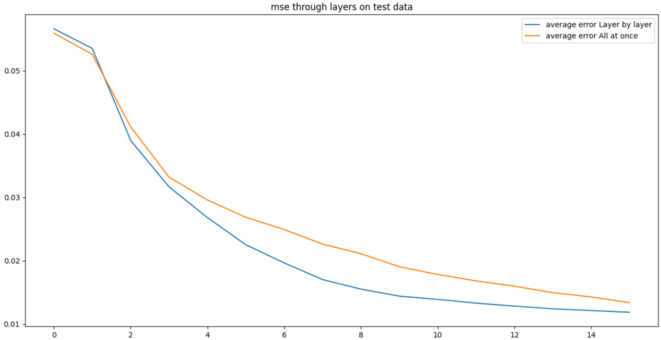


Figure: Decay through layers for all at once training and layer by layer

Comparing ISTA, FISTA, LISTA & LISTA-CP

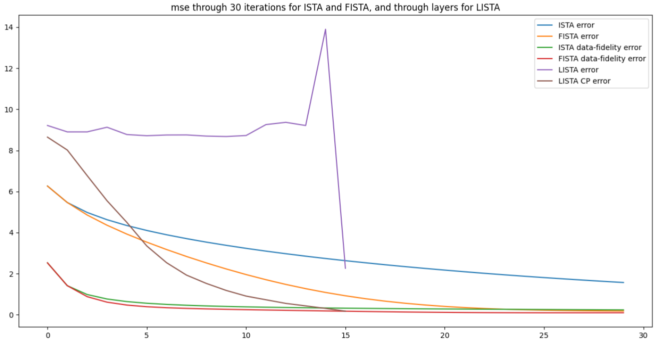


Figure: Comparison of ISTA, FISTA, LISTA & LISTA-CP, with 200 columns, 100 rows and 10000 samples

Convergence of layers parameters

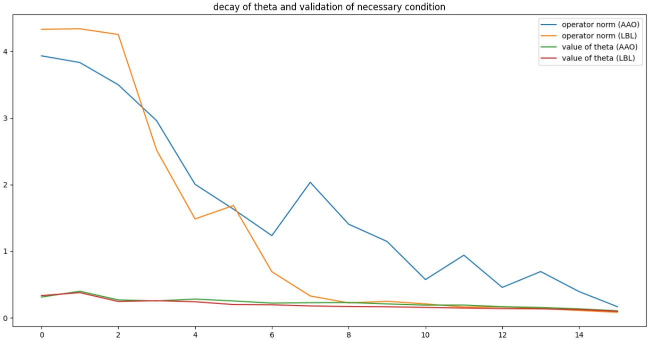


Figure: Coupling of layer parameters and decay of θ through layers

Some words on the experiments

- A well-trained LISTA clearly outperforms (F)ISTA
- Training well a LISTA is a more complicated task than it might seem (sample sizes,over-fitting, instability,...)
- Several training schemes available, it doesn't seem clear which one should be used (LBL is longer than AAO to train, but can succeed on "small" datasets where AAO fails)
- Using the partial coupling (LISTA-CP) makes the learning easier, other "tricks" can help to improve the situation
- Very few comments in the literature on the learning part of LISTA

$\rightarrow$ Do we need to learn the weights (from the data) ?

ALISTA, do we need to learn the weights (from the data) ?

Theorem (Analytic weights are as good as learned weights)

For any $x^* \in \mathcal{X}(B, s)$, W maximally incoherent with D and any sequence γ_k in some bounded interval, then for a LISTA-CP parametrised by

$$W_k = \gamma_k W, \quad \theta_k = \gamma_k \tilde{\mu}(D) \sup_{x^* \in \mathcal{X}(B,S)} ||x^k(x^*) - x^*||_1$$

then it holds that $x^k(x^*)$ has a support contained in the support of x^*, and

$$||x^k(x^*) - x^*||_2 \leq sB \exp\left(-\sum_{i=0}^{k-1} c_i\right)$$

for some positive c_i.