Reconstruction Methods in Compressed Sensing: From Traditional Iterative Methods to Deep Neural Networks.

Gang-Xuan Lin

Institute of Information Science, Academia Sinica, Taiwan

Jul. 19, 2019

Outline

1. 認識 CS: 求解線性反問題
2. 如何重建原始訊號
3. 求解 LASSO 問題的演算法設計
4. 利用 DNN 來加速重建

認識 CS: 求解線性反問題

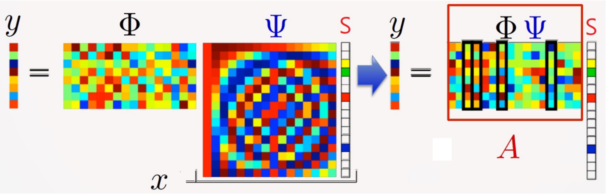
Compressed Sensing (CS) Model : 1D Case (Encoder)

- $x \in \mathbb{R}^n$: origin signal (ground truth)
- $\Phi \in \mathbb{R}^{m \times n}$: sampling matrix
- $\Psi \in \mathbb{R}^{n \times n}$: dictionary (always orthonormal matrix)
- $A = \Phi\Psi \in \mathbb{R}^{m \times n}$
- $y \in \mathbb{R}^m$: measurement vector

The CS model (1D) is defined as:

$$y = \Phi x = \Phi\Psi s = As.$$

- s is k-sparse in Ψ.
- $k < m < n$.



Compressed Sensing (CS) - Introduction

Linear Equations

$$y = Ax_0$$

Compressed Sensing (CS) - Introduction

Linear Equations

$$y = Ax_0$$

- $x_0 \in \mathbb{R}^n$: ground-truth (原始訊號)
- $A \in \mathbb{R}^{m \times n}$: measurement matrix, $m < n$
- $y \in \mathbb{R}^m$: measurement vector

目標: Linear Inverse Problem

給定 y, A, 如何求 x_0?

Compressed Sensing (CS) - Introduction

Linear Equations

$$y = Ax_0$$

- $x_0 \in \mathbb{R}^n$: ground-truth (原始訊號)
- $A \in \mathbb{R}^{m \times n}$: measurement matrix, $m < n$
- $y \in \mathbb{R}^m$: measurement vector

目標: Linear Inverse Problem

給定 y, A, 如何求 x_0?

ANS:

- 用 matrix inverse 不就得了? ...但... A 是不可逆的
- 那就用 Moore-Penrose pseudo-inverse (廣義逆矩陣) 啊!

Compressed Sensing (CS) - 為何它不簡單

Linear Equations / Linear Inverse Problem

$$y = Ax_0$$

給定 y, A, 如何求 x_0?

求解 x_0 的方法: pseudo-inverse

$$y = Ax_0 \implies x_0 = A^+ y = A^T \left(AA^T\right)^{-1} y$$

- 先假設 $\left(AA^T\right)^{-1}$ 存在

問題點

$A^+ y$ 確實是 $y = Ax$ 的其中一個 solution, 可是, 不見得是我們要的 x_0

Compressed Sensing (CS) - 所需的基本設定

Linear Equations / Linear Inverse Problem

$$y = Ax_0$$

給定 y, A, 如何求 x_0?

A^+y 不夠好的理由

- A 是 $m \times n, m < n$ 矩陣
- 如果 $y = Ax$ 有解的話, 那就是無窮多解
- $\dim(\text{null}(A)) \neq 0$
- $x_0 + \text{null}(A)$ 裡的向量通通滿足 $y = Ax_0$
- $A^+y \in x_0 + \text{null}(A)$, 那如何得知 A^+y 就是 x_0?

Compressed Sensing (CS) - 所需的基本設定

Q: 既然早就知道 $y = Ax$ 有無窮多解, 那要怎求出特定的解 x_0?
A: 給 x_0 限制, 使其滿足特定結構, 使其有特色!

讓 x_0 有 sparsity 的特性

- sparse 是指 $|\{i \in [1:n] : (x_0)_i \neq 0\}| = k$, k 很小 (相較於 m, n)
- 稱 x_0 為 k-sparse 向量

Linear Equations / Linear Inverse Problem

假設 x_0 是一個 k-sparse vector, 滿足

$$y = Ax_0$$

給定 y, A, 如何求 x_0

Compressed Sensing (CS) - naive case

Linear Equations / Linear Inverse Problem

假設 x_0 是一個 k-sparse vector, 滿足

$$y = Ax_0$$

給定 y, A, 如何求 x_0?

- $x_0 + \text{null}(A)$ 裡的向量通通滿足 $y = Ax_0$
- 但是, $x_0 + \text{null}(A)$ 裡的 k-sparse vector 就沒有很多了!

- $x = A^+ y \in x_0 + \text{null}(A)$, 是滿足 $y = Ax_0$
- 可是, $x = A^+ y$ 是 k-sparse 嗎?
- 若 $A_{i,j} \sim \mathcal{N}(0,1)$ (normal distribution), 那麼 $A \in \mathbb{R}^{m\times n}$ 有高機率不包含零分量, 是 dense!
- 因為 $y \neq 0$, 因此 $A^+ y = A^T \left(AA^T\right)^{-1} y$ 也是 dense, 解不回 x_0

三個問題

在這裡我們遇到三個問題

- 講了一大堆, 所以 CS 能幹痲?
- 給定 y, A, 如何求 k-sparse 的 x_0?
- 如果 x_0 不稀疏的話勒?

- 其實真要討論 CS 的話, 還有許多問題, 比如可重建性等等. 此課程聚焦在如何重建原始訊號

先討論第三個問題: 如果 x_0 不稀疏的話勒?

Linear Equations: with dense x_0

$$y = \Phi x_0$$

- $x_0 \in \mathbb{R}^n$: ground-truth (原始訊號)
- $\Phi \in \mathbb{R}^{m \times n}$: sampling matrix
- $y \in \mathbb{R}^m$: measurement vector
- 前面用的符號是 $y = Ax_0$, 為討論清楚, 這裡用的符號是 $y = \Phi x_0$

需要 sparsity 結構

- 是否能找到 $\Psi \in \mathbb{R}^{n \times n}$ 滿足

$$x_0 = \Psi s_0,$$

其中 s_0 是稀疏的

- 亦即 x_0 是否有 sparse representation? $\left(s_0 = \Psi^{-1} x_0\right)$

先討論第三個問題: 如果 x_0 不稀疏的話勒?

需要 sparsity 結構

- 是否能找到 $\Psi \in \mathbb{R}^{n \times n}$ 滿足

$$x_0 = \Psi s_0 \quad \left(s_0 = \Psi^{-1} x_0\right)$$

其中 s_0 是稀疏的

ANS:

- 不會有 Ψ 能使大部份的 s_0 是稀疏的
- 但我們只需要差不多稀疏就行了
- 常見的 Ψ^{-1} 有 wavelet transformation, DCT, FFT, etc.
- Ψ 稱為 dictionary

接下來用 MATLAB 來 implement, 但在那之前, 介紹一些相似度(重建精準度)的測量方式

error

$$\text{error} = \|x^* - x_0\|_2^2$$

- $x_0 \in \mathbb{R}^n$
- 非常受使用限制

MSE (mean square error)

$$\text{MSE} = \frac{1}{n}\|x^* - x_0\|_2^2$$

- 平均每個分量的誤差

接下來用 MATLAB 來 implement, 但在那之前, 介紹一些相似度(重建精準度)的測量方式

RE (relative error)

$$\mathrm{RE} = \frac{\|x^* - x_0\|_2^2}{\|x_0\|_2^2}$$

- 較能適用不同尺度的向量
- 如 $mean(x_0) = 10$ 與 $mean(x_0) = 10^6$ 的差別

SNR (signal-to-noise ratio)

$$\mathrm{SNR} = 10\log_{10}\frac{\|x_0\|_2^2}{\|x^* - x_0\|_2^2}$$

- 與 RE 類似, 便於討論與呈現

(MATLAB implementation)

CS 問題的樣子變成...

$$y = \Phi x_0 = \Phi\Psi s_0 = A s_0$$

- $A = \Phi\Psi \in \mathbb{R}^{n\times n}$, $\Phi \in \mathbb{R}^{m\times n}$ $\Psi \in \mathbb{R}^{n\times n}$

目標: Linear Inverse Problem

給定 y, A, 如何求 k-sparse 的 s_0?

- 求得 s_0 之後, 即可重建原始訊號 $x_0 = \Psi s_0$
- 這回答了前面的第三個問題: 如果 x_0 不稀疏的話勒?

回到第一頁

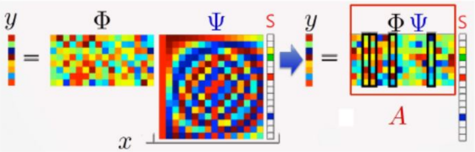
Compressed Sensing (CS) Model : 1D Case (Encoder)

- $x \in \mathbb{R}^n$: origin signal (ground truth)
- $\Phi \in \mathbb{R}^{m \times n}$: sampling matrix
- $\Psi \in \mathbb{R}^{n \times n}$: dictionary (always orthonormal matrix)
- $A = \Phi\Psi \in \mathbb{R}^{m \times n}$
- $y \in \mathbb{R}^m$: measurement vector

The CS model (1D) is defined as:

$$y = \Phi x = \Phi \Psi s = As.$$

- s is k-sparse in Ψ.
- $k < m < n$.



補充: CS Model : 2D Case (Encoder)

- $X \in \mathbb{R}^{N \times N}$: origin image (ground truth)
- $\Phi_1, \Phi_2 \in \mathbb{R}^{M \times N}$: sampling matrices
- $\Psi_2, \Psi_2 \in \mathbb{R}^{N \times N}$: dictionaries (always orthonormal matrix)
- $A_1 = \Phi_1 \Psi_1, A_2 = \Phi_2 \Psi_2 \in \mathbb{R}^{M \times N}$
- $Y \in \mathbb{R}^{M \times M}$: measurement matrix

sparse representation of X:

$$X = \Psi_1 S \Psi_2^T$$

CS model (2D):

$$Y = \Phi_1 X \Phi_2^T = \Phi_1 \Psi_1 S \Psi_2^T \Phi_2^T = A_1 S A_2^T.$$

補充: CS Model : Vectorize into 1D Case (Encoder)

CS model (2D):

$$Y = \Phi_1 X \Phi_2 = \Phi_1 \Psi_1 S \Psi_2^T \Phi_2^T = A_1 S A_2^T.$$

vectorize:

$$y = \Phi x = \Phi \Psi s = As.$$

- $x = vec(X)$, $s = vec(S)$
- $\Phi = \Phi_2 \otimes \Phi_1$
- $\Psi = \Psi_2 \otimes \Psi_1$
- $A = \Phi\Psi$

還有二個問題

在這裡我們遇到三個問題

- 講了一大堆, 所以 CS 能幹麻?
- 給定 y, A, 如何求 k-sparse 的 x_0?
- 如果 x_0 不稀疏的話勒?

來討論第一個問題: CS 能幹麻?

CS 的應用

- single-pixel camera
- MRI
- 任何 compressable 的訊號 (image, music, signal, video, etc.)
- IoT (大部份時間, 多數裝置都處於休息或監測狀態, 不會一直傳輸信息給基地台, 具有稀疏性)
- MIMO
- security (壓縮本身具有加密性質)

- 都是為了省 energy, 省 memory

剩最後一個問題

在這裡我們遇到三個問題

- 講了一大堆, 所以 CS 能幹嘛?
- 給定 y, A, 如何求 k-sparse 的 x_0?
- 如果 x_0 不稀疏的話勒?

符號重整

$$y = \Phi x_0 = \Phi \Psi s_0 = A s_0$$

- $A = \Phi \Psi$
- 為後面討論方便, 我們仍用符號 $y = A x_0$, 這裡的 x_0 為 k-sparse

如何重建原始訊號

面對最後的問題: 如何重建原始訊號 (Decoder)

$$y = Ax_0$$

- x_0 是 k-sparse
- 亦即 $\|x_0\|_0 = k$
- $k < m < n$

模型建立: ℓ_0-norm minimization problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|x\|_0 \\ \text{s.t.} \quad & y = Ax \end{aligned}$$

- 求解 ℓ_0-norm minimization problem, 希望其最佳解為 x_0

最佳化的名詞介紹

ℓ_0-norm minimization problem

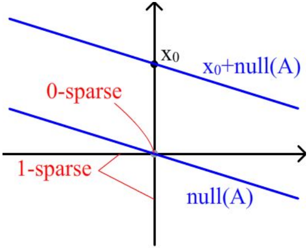
$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \|x\|_0 \\ \text{s.t.} \quad & y = Ax \end{aligned}$$

- $\|x\|_0$: 目標函數 (objective function), 其函數值稱為 objective value
- $y = Ax$: 限制式 (constraint)
- $\{x : y = Ax\}$: 可行解區域 (feasible domain), 裡面的元素稱為可行解 (feasible solution)
- x^*: 最佳解 (optimal solution)
- $\|x^*\|_0$: optimal value

如何重建原始訊號

ℓ_0-norm minimization problem:

$$y = Ax_0 \quad \Longrightarrow \quad \begin{cases} \min\limits_{x \in \mathbb{R}^n} & \|x\|_0 \\ \text{s.t.} & y = Ax \end{cases}$$



GOAL:

在 feasible domain $x_0 + \text{null}(A)$ 求得最佳解 $x^* = x_0$

困難點

ℓ_0-norm minimization problem:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|x\|_0 \\ \text{s.t.} \quad & y = Ax \end{aligned}$$

- $\|x\|_0$ 不是一個連續函數, 很難做基本數學分析
- $\|x\|_0$ 不是一個 norm
- ℓ_0-norm minimization problem 是一個 NP-hard

解決方法

- 一個模型很難求解的話, 常見的招數有:
  ~~reuse/reduce/recycle~~ reduce/relax/reformulate, 或 approximate

BP problem

ℓ_0-norm minimization problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \|x\|_0 \\ \text{s.t.} \quad & y = Ax \end{aligned}$$

利用 ℓ_1 minimization problem 來估計

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \|x\|_1 \\ \text{s.t.} \quad & y = Ax \end{aligned}$$

- 此模型稱為 BP (Basis Pursuit)

其他 ℓ_1 minimization problem

Basis Pursuit (BP) (to recover sparse x from $y = Ax$)

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|x\|_1 \\ \text{s.t.} \quad & y = Ax \end{aligned}$$

Basis Pursuit Denoising (BPDN) (to recover sparse x from $y = Ax + n$, n is noise)

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|x\|_1 \\ \text{s.t.} \quad & \|y - Ax\|_2^2 \leq \epsilon \end{aligned}$$

(LASSO) (to recover sparse x from $y = Ax$ or $y = Ax + n$)

$$\min_{x\in\mathbb{R}^n} \frac{1}{2} \|y - Ax\|_2^2 + \lambda \|x\|_1$$

這裡針對經由求解 LASSO 問題去重建原始訊號

LASSO

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|y - Ax\|_2^2 + \lambda \|x\|_1$$

- $\lambda > 0$: penalty parameter
- BP 與 BPDN 都是有限制式, 一般而言較難求解

原本統計學上的 LASSO 問題

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & \frac{1}{m} \|y - Ax\|_2^2 \\ \text{s.t.} & \|x\|_1 \leq t \end{array}$$

- 這問題的統計上的 Lagrange form 就是上面的 unconstrained minimization problem

早期的重建演算法 (包含 greedy algorithms)

- MP/OMP ((orthogonal) matching pursuit)
- N-BOMP (N-way block OMP)
- SPGL1 (spectral projected gradient for L1 minimization)
- KCS (Kronecker compressive sensing)
- MWCS (multiway compressive sensing)
- GTCS (generalized tensor compressive sensing)
- ADM (alternating direction method)
- ISTA/FISTA ((fast) iterative shrinkage-thresholding algorithm)
- BCS (Iterative Wiener filtering and hard-thresholding)
- BCS-SPL (block-based CS with smoothed projected Landweber)
- FP-qA (fixed point equation with quasi-Armijo rule)
- PGD (projected gradient descent method)
- AMP, DAMP, VAMP, quasi-Newton, etc.

or convert to

- SOCP reformulation (can solved by CPLEX, Gurobi, etc.)
- SDP relaxation (can solved by SDPA, CSDP, SDPLR, SDPT3, SeDuMi, etc.)

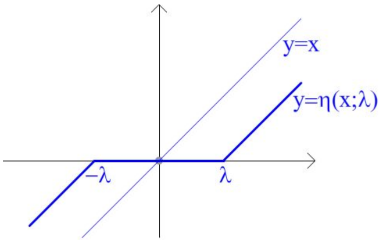
求解 LASSO 問題的演算法

求解 LASSO 問題, 最早期最簡單的迭代法: ISTA (Iterative Shrinkage-Thresholding Algorithm)

ISTA

$$\begin{cases} r_t = x_t - \beta A^T (Ax_t - y) & \text{gradient descent step} \\ x_{t+1} = \eta(r_t; \lambda) & \text{soft-threshold} \end{cases}$$

- η: soft-thresholding operator
- $\eta(x;\lambda) = \text{sgn}(x)(|x| - \lambda)_+$



ISTA

ISTA 常見的二種寫法

$$\begin{cases} r_t = x_t - \beta A^T (Ax_t - y) & \text{gradient descent step} \\ x_{t+1} = \eta (r_t; \lambda) & \text{soft-threshold} \end{cases}$$

$$\begin{cases} r_t = y - Ax_t & \text{residue measurement error} \\ x_{t+1} = \eta \left(x_t + \beta A^T r_t; \lambda \right) & \text{gradient descent and soft-threshold} \end{cases}$$

ISTA

LASSO

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|y - Ax\|_2^2 + \lambda \|x\|_1$$

- $-\nabla \left(\frac{1}{2} \|Ax - b\|_2^2 \right) = -A^T (Ax - b)$: descent direction
- β: step size
- η: soft-thresholding operator, it is a proximal mapping of $\lambda \|x\|_1$

ISTA

$$\begin{cases} r_t = x_t - \beta A^T (Ax_t - y) & \text{gradient descent step} \\ x_{t+1} = \eta (r_t; \lambda) & \text{soft-threshold} \end{cases}$$

二個問題

- $\lambda > 0$ 怎麼選
- $\beta > 0$ 怎麼選

LASSO

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|y - Ax\|_2^2 + \lambda \|x\|_1$$

ISTA

$$\begin{cases} r_t = x_t - \beta A^T (Ax_t - y) & \text{gradient descent step} \\ x_{t+1} = \eta(r_t; \lambda) & \text{soft-threshold} \end{cases}$$

第二個問題: $\beta > 0$ 怎麼選

ISTA

$$\begin{cases} r_t = x_t - \beta A^T (Ax_t - y) & \text{gradient descent step} \\ x_{t+1} = \eta(r_t; \lambda) & \text{soft-threshold} \end{cases}$$

Guaranteed to converge under $\beta \in \left(0, \frac{1}{\|A\|_2^2}\right)$, with convergence rate $\frac{1}{k}$.

第一個問題: $\lambda > 0$ 怎麼選

LASSO

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|y - Ax\|_2^2 + \lambda \|x\|_1$$

- $\frac{1}{2} \|y - Ax\|_2^2$: residue in measurement
- $\|x\|_1$: sparsity
- $\lambda > 0$: penalty parameter, depending on magnitude of x, tradeoff between sparsity and residue in measurement

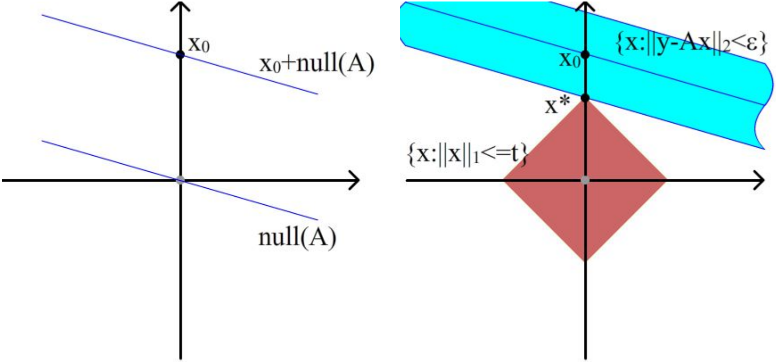
- if λ large, then $\|x\|_1$ should small (w.r.t. residue)
  the optimal solution will sparse, with relative large error
- if λ small, then residue should small
  the optimal solution may not sparse

LASSO 的模型内涵

LASSO

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|y - Ax\|_2^2 + \lambda \|x\|_1$$

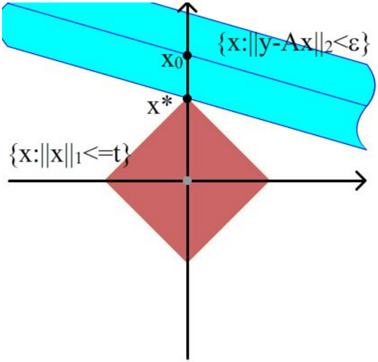
- $\lambda > 0$: tradeoff between sparsity and residue in measurement

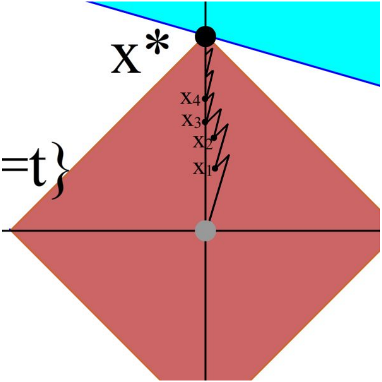


ISTA 的過程

ISTA

$$\begin{cases} r_t = x_t - \beta A^T (Ax_t - y) & \text{gradient descent step} \\ x_{t+1} = \eta(r_t; \lambda) & \text{soft-threshold} \end{cases}$$





(MATLAB implementation)

FISTA (fast ISTA, 2009)[1]

ISTA

$$\begin{cases} r_t = x_t - \beta A^T (Ax_t - y) & \text{gradient descent step} \\ x_{t+1} = \eta(r_t; \lambda) & \text{soft-threshold} \end{cases}$$

FISTA

$$\begin{cases} r_t = x_t - \beta A^T (Ax_t - y) + \frac{t-2}{t+1}(x_t - x_{t-1}) \\ x_{t+1} = \eta(r_t; \lambda) \end{cases}$$

- descent direction: $-\beta A^T (Ax_t - y) + \frac{t-2}{t+1}(x_t - x_{t-1})$
- 根據上一次的迭代方向(momentum)來調整此次的下降方向(descent direction).

FISTA convergence rate $\frac{1}{k^2}$.

[1] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imag. Sci., vol. 2, no. 1, 2009

AMP (approximate message passing)[2] [3] [4]

ISTA

$$\begin{cases} r_t = y - Ax_t & \text{residue measurement error} \\ x_{t+1} = \eta\left(x_t + \beta A^T r_t; \lambda\right) & \text{gradient descent and soft-threshold} \end{cases}$$

AMP (順序應該是錯的)

$$\begin{cases} x_{t+1} = \eta\left(x_t + A^T r_t; \lambda + \gamma_t\right) \\ r_t = y - Ax_t + \frac{1}{\delta} r_{t-1} \left\langle \eta'\left(x_{t-1} + A^T r_{t-1}; \lambda + \gamma_t\right)\right\rangle \end{cases}$$

- $\delta = \frac{m}{n}$: measurement rate, $\langle \cdot \rangle$ average of a vector, η': derivative
- $\gamma_{t+1} = \frac{\lambda + \gamma_t}{\delta} \left\langle \eta'\left(x_t + A^T r_t; \lambda + \gamma_t\right)\right\rangle$

[2] D. L. Donoho, A. Maleki, and A. Montanari, *Message Passing Algorithms for Compressed Sensing*, Proc. Nat. Acad. Sci., vol. 106, Nov. 2009 (arXiv: Jul. 2009)

[3] —, *Message passing algorithms for compressed sensing: I. motivation and construction*, Proc. IEEE Info. Theory Workshop (ITW), Jan. 2010

[4] —, *Message passing algorithms for compressed sensing: II. analysis and validation*, Proc. IEEE Info. Theory Workshop (ITW), Jan. 2010

AMP (較廣為使用的版本)[5]

AMP (Donoho ver.)

$$\begin{cases} x_{t+1} = \eta\left(x_t + A^T r_t; \lambda + \gamma_t\right) \\ r_t = y - Ax_t + \frac{1}{\delta} r_{t-1} \left\langle \eta'\left(x_{t-1} + A^T r_{t-1}; \lambda + \gamma_t\right)\right\rangle \end{cases}$$

- $\gamma_{t+1} = \frac{\lambda+\gamma_t}{\delta} \left\langle \eta'\left(x_t + A^T r_t; \lambda + \gamma_t\right)\right\rangle$
- $\frac{1}{\delta} r_{t-1} \left\langle \eta'\left(x_{t-1} + A^T r_{t-1}\right)\right\rangle$ is called Onsager reaction term

AMP (Montanari ver.)[5]

$$\begin{cases} x_{t+1} = \eta\left(x_t + A^T r_t; \lambda_t\right) \\ r_t = y - Ax_t + b_t r_{t-1} \end{cases}$$

- $b_t = \dfrac{1}{m} \|x_t\|_0 , \quad \lambda_t = \dfrac{\alpha}{\sqrt{m}} \|r_t\|_2$
- $b_t r_{t-1}$ is called Onsager term.

[5] A. Montanari, *Graphical models concepts in compressed sensing*, in Compressed Sensing: Theory and Applications (edited by Y. C. Eldar and G. Kutyniok, eds.), Cambridge Univ. Press, 2012

About AMP

AMP

$$\begin{cases} r_t = y - Ax_t + b_t r_{t-1} \\ x_{t+1} = \eta\left(x_t + A^T r_t; \lambda_t\right) \end{cases}$$

- $b_t = \frac{1}{m}\|x_t\|_0, \quad \lambda_t = \frac{\alpha}{\sqrt{m}}\|r_t\|_2.$

- $A_{ij} \sim \mathcal{N}\left(0, \frac{1}{m}\right)$ (each column has 2-norm ≈ 1)
- Soft-thresholding operator $\eta(\bullet; \bullet)$ is denoiser.
- The input of denoiser is

$$x_t + A^T r_t \sim x_t + \mathcal{N}\left(0, \frac{1}{m}\|r_t\|_2^2 I_n\right),$$

is corrupted version of groundtruth with additive white Gaussian noise of variance $\frac{1}{m}\|r_t\|_2^2$.

- The behavior of AMP is well understood when $A_{ij} \sim \mathcal{N}\left(0, \frac{1}{m}\right)$, but even small deviations from this model can lead AMP to diverge or at least behave in ways that are not well understood.

ISTA vs. AMP

$$\text{ISTA}: \begin{cases} r_t = y - Ax_t \\ x_{t+1} = \eta\left(x_t + \beta A^T r_t; \lambda\right) \end{cases} \quad \text{AMP}: \begin{cases} r_t = y - Ax_t + b_t r_{t-1} \\ x_{t+1} = \eta\left(x_t + A^T r_t; \lambda_t\right) \end{cases}$$

- The denoiser input error of AMP follows Gaussian due to Onsager correction.
- Without Onsager correction, denoiser input error of ISTA does not follow Gaussian.

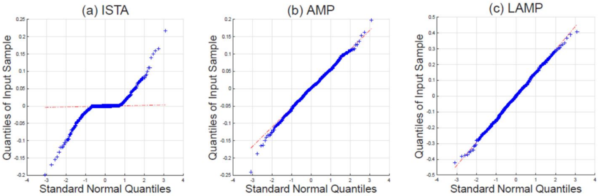


Fig. 5. QQplots of the denoiser input error evaluated at the first iteration t for which $\text{NMSE}(\widehat{\boldsymbol{x}}_t) < -15$ dB. Note ISTA's error is heavy tailed while AMP's and LAMP's errors are Gaussian due to Onsager correction.

ISTA vs. FISTA vs. AMP

- $n = 500,\ m = 250,\ k \sim 50$
- $x_{\text{nonzero}} \sim \mathcal{N}(0,1)$
- average in 1000 realizations

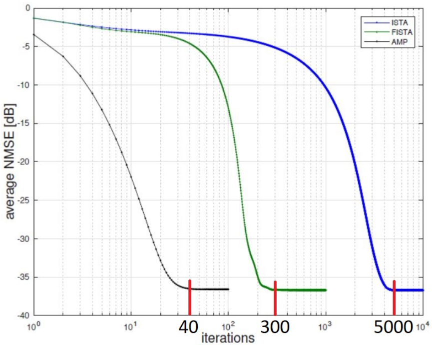


Fig. 1. Average NMSE versus iteration number for AMP, FISTA, ISTA (from left to right).

$$NMSE = 10\log_{10}\frac{\|x_t - x\|_2^2}{\|x\|_2^2}$$

利用 DNN 來加速重建

From ISTA to LISTA

ISTA

$$x_{t+1} = \eta\left(x_t - \beta A^T (Ax_t - y); \lambda\right)$$

The input of denoiser is

$$x_t - \beta A^T (Ax_t - y) \;=\; \left(I_n - \beta A^T A\right) x_t + \beta A^T y \;=\; S x_t + B y$$

where $B = \beta A^T$ and $S = I_n - BA$.

$$\text{(ISTA)}: \; x_{t+1} = \eta\left(S x_t + B y; \lambda\right)$$

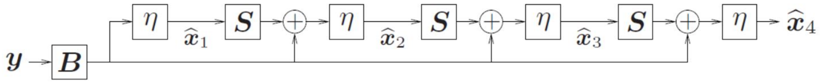


Fig. 2. The feed-forward neural network constructed by unfolding $T = 4$ iterations of ISTA.

LISTA (Learned ISTA)[6]

ISTA

$$x_{t+1} = \eta\left(x_t - \beta A^T (Ax_t - y); \lambda\right) = \eta(Sx_t + By; \lambda)$$

where $B = \beta A^T$ and $S = I_n - BA$.

LISTA[6]

$$x_{t+1} = \eta(\color{red}{S}x_t + \color{red}{B}y; \color{red}{\lambda_t})$$

Learning parameters:

- $B \in \mathbb{R}^{n\times m}$, $S \in \mathbb{R}^{n\times n}$, layer-dependent thresholds $\lambda = (\lambda_1, \cdots, \lambda_T)$

With quadratic loss function:

$$\mathcal{L}_T(B, S, \lambda) = \frac{1}{D}\sum_{d=1}^{D}\left\|x_T\left(y^{(d)}; B, S, \lambda\right) - x^{(d)}\right\|_2^2.$$

- $x_T\left(y^{(d)}; B, S, \lambda\right)$: output of the T-layer network.

[6] K. Gregor and Y. LeCun, *Learning Fast Approximations of Sparse Coding*, ICML, 2010

Some Idea to Improve LISTA

LISTA

$$x_{t+1} = \eta\left(Sx_t + By; \lambda_t\right)$$

- $B = \beta A^T$ and $S = I_n - BA$ in ISTA
- $B \in \mathbb{R}^{n\times m}$, $S \in \mathbb{R}^{n\times n}$, $\lambda_t \in \mathbb{R}$

- 若把 LISTA 的 S 拆回原本的樣子:

$$x_{t+1} = \eta\left((I_n - BA)x_t + By; \lambda_t\right)$$

- $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times m}$, $\lambda_t \in \mathbb{R}$

- S 自由參數(free parameters)變成 $2mn$ 個, 在 $m < \frac{n}{2}$ 時, 記憶體與訓練時間將得到好處.

Modest improvement LISTA [7] [8]

$$x_{t+1} = \eta\left((I_n - BA)x_t + By; \lambda_t\right)$$

- $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times m}$, $\lambda_t \in \mathbb{R}$

ISTA: $x_{t+1} = \eta\left(x_t - \beta A^T\left(Ax_t - y\right); \lambda\right)$

$$\begin{cases} r_t = y - Ax_t \\ x_{t+1} = \eta\left(x_t + \beta A^T r_t; \lambda\right) \end{cases} \Longrightarrow \begin{cases} r_t = y - Ax_t \\ x_{t+1} = \eta\left(x_t + B r_t; \lambda\right) \end{cases}$$

- $B = \beta A^T$

Modest improvement LISTA

Here allows both A and B to vary with the layer t:

$$\begin{cases} r_t = y - A_t x_t \\ x_{t+1} = \eta\left(x_t + B_t r_t; \lambda_t\right) \end{cases}$$

[7] M. Borgerding and P. Schniter, *Onsager-corrected deep learning for sparse linear inverse problems*, IEEE GlobalSIP, Dec. 2016 (arXiv: Jul. 2016)

[8] M. Borgerding, P. Schniter, and S. Rangan, *AMP-Inspired Deep Networks for Sparse Linear Inverse Problems*, IEEE TSP, Aug. 2017 (arXiv: Dec. 2016)

Modest improvement LISTA

$$\begin{cases} r_t = y - A_t x_t \\ x_{t+1} = \eta\left(x_t + B_t r_t; \lambda_t\right) \end{cases}$$

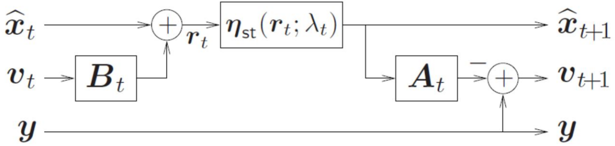


Fig. 3. The tth layer of the LISTA network, with learnable parameters $\boldsymbol{A}_t$, $\boldsymbol{B}_t$, and λ_t.

- The Modest improvement LISTA's performance does not degrade.

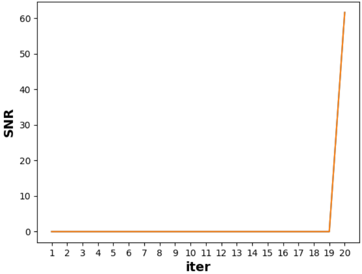
- $n = 500,\ \ m = 250,\ \ k \sim 50$
- $x_{\text{nonzero}} \sim \mathcal{N}\left(0, 1\right),\ A \sim \mathcal{N}\left(0, \frac{1}{m}\right),$
- average in 1000 realizations
- To reach NMSE of -35 dB,

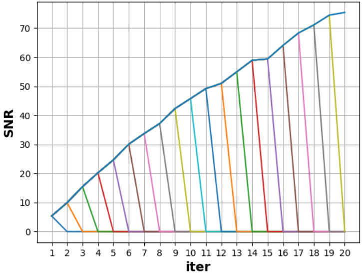
ISTA:	4402 (iterations)
FISTA:	216
AMP:	25
LISTA:	16 (layers)

一堆 DNN 的 LISTA 算法, 結論呢?

- 即使是同一個 DNN 模型, 不同的訓練方式將有不同的結果
- 比如, 不同的 learning rate 設定, 不同的 initial 等等, 皆將導致不同訓練結果

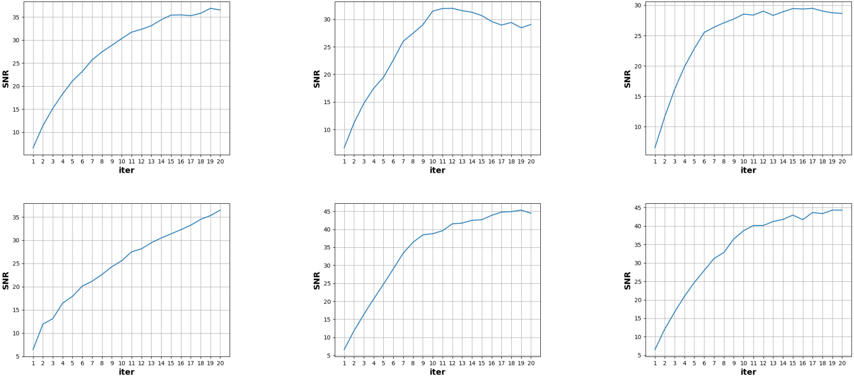
一堆 DNN 的 LISTA 算法, 結論呢? [9]





[9] 左: 一口氣訓練 20 個 layers, 右: train layer-by-layer

一堆 DNN 的 LISTA 算法, 結論呢? [10] [11]



[10] 上排左: $\{S, B, \lambda_i\}$, 上排中: $\{S_i, B_i, \lambda_i\}$, 上排右: $\{S_i, B_i, \lambda_i\}$ 利用前一次的結果當 initial

[11] 下排左: $\{A, B(= A^T), \lambda_i\}$, 下排中: $\{A_i, B_i, \lambda_i\}$, 下排右: $\{A_i, B_i, \lambda_i\}$ 利用前一次的結果當 initial

From AMP to LAMP

LAMP (Learned AMP)[12] [13]

Review: AMP (approximate message passing)

$$\begin{cases} r_t = y - A x_t + b_t r_{t-1} \\ x_{t+1} = \eta\left(x_t + A^T r_t; \lambda_t\right) \end{cases}$$

where $b_t = \frac{1}{m}\|x_t\|_0, \quad \lambda_t = \frac{\alpha}{\sqrt{m}}\|r_t\|_2.$

LAMP

$$\begin{cases} r_t = y - A_t x_t + b_t r_{t-1} \\ x_{t+1} = \eta\left(x_t + B_t r_t; \lambda_t\right) \end{cases}$$

where $b_t = \frac{1}{m}\|x_t\|_0, \quad \lambda_t = \frac{\alpha_t}{\sqrt{m}}\|r_t\|_2.$

Learning parameters: $A_t \in \mathbb{R}^{m\times n}$, $B_t \in \mathbb{R}^{n\times m}$, $\alpha_t \in \mathbb{R}$.

[12] M. Borgerding and P. Schniter, *Onsager-corrected deep learning for sparse linear inverse problems*, IEEE GlobalSIP, Dec. 2016 (arXiv: Jul. 2016)

[13] M. Borgerding, P. Schniter, and S. Rangan, *AMP-Inspired Deep Networks for Sparse Linear Inverse Problems*, IEEE TSP, Aug. 2017 (arXiv: Dec. 2016)

LAMP (Learned AMP)

$$\begin{cases} r_t = y - \color{red}{A_t} x_t + b_t r_{t-1} \\ x_{t+1} = \eta\left(x_t + \color{red}{B_t} r_t; \lambda_t\right) \end{cases}$$

where $b_t = \dfrac{1}{m}\|x_t\|_0\,, \quad \lambda_t = \dfrac{\color{red}{\alpha_t}}{\sqrt{m}}\|r_t\|_2.$

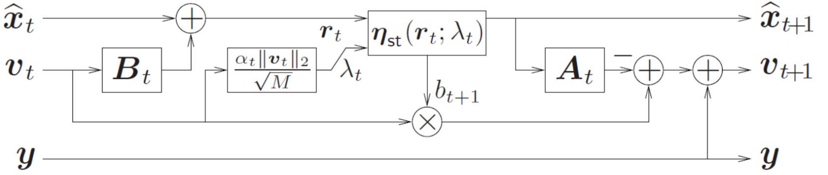
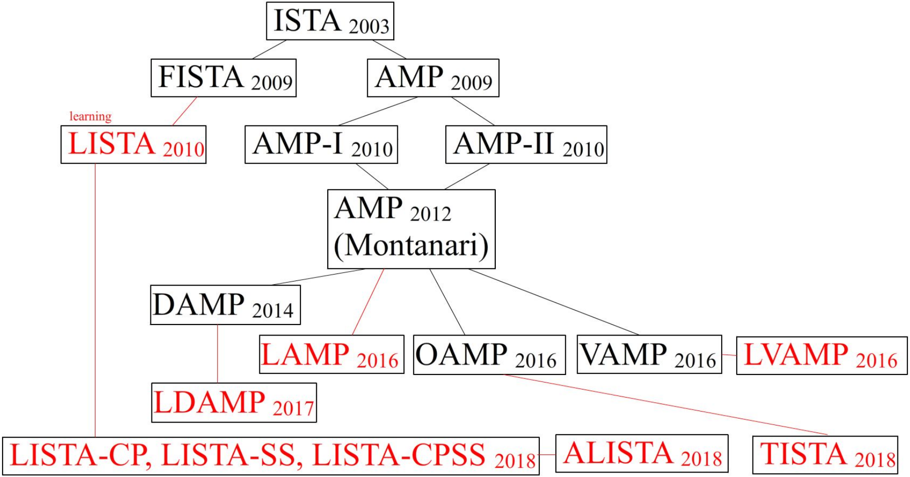


Fig. 4. The tth layer of the LAMP-ℓ_1 network, with learnable parameters $\boldsymbol{A}_t$, $\boldsymbol{B}_t$, and α_t.

一堆 ISTA / AMP 家族的演算法

[14] [15]



[14] 年份代表首次網路亮相, 非刊登時間
[15] 黑色: 傳統迭代法, 紅色: 利用 DNN

	Publications		arXiv	Authors	I/L[16]	param in 1 iter
ISTA	2003?			Daubechies?	I	
FISTA	2009.03	SIAM[17]		Beck	I	
AMP	2009.11	PNAS[18]	2009.07	Donoho	I	
AMP-I	2010.01	ITW[19]	2009.11	Donoho	I	
AMP-II	2010.01	ITW	2009.11	Donoho	I	
AMP	2012.11	book[20]	2010.11	Montanari	I	
LISTA	2010.06	ICML		LeCun	L	$n^2 + mn + 1$
DAMP	2016.04	TIT	2014.06	Baraniuk	I	
OAMP	2017.01	Access	2016.02	Ping	I	
LAMP	2016.12	GlobalSIP	2016.07	Schniter	L	$mn + 2$
VAMP	2017.06	ISIT	2016.10	Rangan	I	
LVAMP	2017.05	TSP	2016.12	Schniter	L	$2mn + m + 1$
TISTA	2018.05	ICC workshop	2018.01	Ito	L	1

[16] iterative method vs. learning method

[17] SIAM imaging science

[18] proceedings of the national academy of sciences of the USA

[19] information theory workshop

[20] in Compressed Sensing: Theory and Applications (edited by Y. C. Eldar and G. Kutyniok, eds.), Cambridge Univ. Press, 2012

還有一堆

- Generalized AMP (GAMP)
- Bayesian AMP (BAMP)
- Multiple Measurement Vector BAMP (MMV-BAMP)
- Distributed AMP
- Centralized AMP
- Hybrid generalized AMP (HyGAMP)
- S-AMP (S comes from the uses of S-transform)
- ISTA-NET

Thank you for listening!
Any question?