

CS 754, Ajit Rajwade

Compressive Sensing: Reconstruction Algorithms

Compressive sensing: reconstruction problem

- The basic problem is:

$$(P0) : \min \|\boldsymbol{\theta}\|_0$$

$$\text{such that } \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 \leq \varepsilon$$

$$\mathbf{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$$



$$(P1) : \min \|\boldsymbol{\theta}\|_1$$

$$\text{such that } \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 \leq \varepsilon$$

- Here \mathbf{y} is a vector of measurements, as obtained with an instrument with sensing matrix $\boldsymbol{\Phi} \in \mathcal{R}^{m \times n} (m \ll n)$.
- The original signal \mathbf{x} (to be estimated) has a sparse/compressible representation in the orthonormal basis $\boldsymbol{\Psi} \in \mathcal{R}^{n \times n}$, i.e. $\mathbf{x} = \boldsymbol{\Psi}\boldsymbol{\theta}$, i.e. $\boldsymbol{\theta}$ is a sparse/compressible vector.

Compressive Reconstruction Algorithms

- **Category 1:** Problem P_0 is NP-hard. So instead, run an *approximation algorithm* which can be efficiently solved.
- **Category 2:** Seek to directly solve problem P_1 . There are many algorithms in this category, in particular a popular one called Iterative Shrinkage/Thresholding Algorithm (ISTA).

Compressive Reconstruction Algorithms

- We will first focus on some algorithms in category 1.
- Examples:
 - ✓ Matching pursuit (MP)
 - ✓ Orthogonal matching pursuit (OMP)
 - ✓ Iterative Hard Thresholding
 - ✓ Co-SAMP.

Matching Pursuit

- One of the simplest approximation algorithms to obtain the coefficients $\boldsymbol{\theta}$ of a signal \mathbf{y} in an over-complete “dictionary” matrix $\mathbf{A} \in \mathcal{R}^{m \times n}$ (i.e. $m \ll n$)
- Developed by Mallat and Zhang in 1993 (ref: S. G. Mallat and Z. Zhang, [Matching Pursuits with Time-Frequency Dictionaries](#), IEEE Transactions on Signal Processing, December 1993)
- Based on successively choosing the column vector in \mathbf{A} which has maximal dot product with a so-called *residual vector* (initialized to \mathbf{y} in the beginning).

Pseudo-code

$$\mathbf{r}^{(0)} = \mathbf{y}; i = 0$$

$$\text{while } (\|\mathbf{r}^{(i)}\|^2 > \varepsilon)$$

{

$$j = \arg \max_l | \mathbf{r}^{(i)T} \mathbf{a}_l / \|\mathbf{a}_l\|_2 |$$

$$\theta_j = \mathbf{r}^{(i)T} \mathbf{a}_j / \|\mathbf{a}_j\|^2; \mathbf{r}^{(i+1)} = \mathbf{r}^{(i)} - \theta_j \mathbf{a}_j; i = i + 1$$

}

OUTPUT : $\{\theta_j\}$

Select the column of \mathbf{A} that is maximally correlated (highest dot-product) with the residual

"j" or "l" is an index for columns of \mathbf{A}

MP may choose a single column vector of \mathbf{A} (say \mathbf{A}_j) in multiple different iterations, and each time the coefficient θ_j may be different. In such cases, all these different θ_j values will contribute towards the reconstruction of \mathbf{y} .

Properties of matching pursuit

- MP decomposes any signal \mathbf{y} into the form

$$\mathbf{y} = \frac{\mathbf{y}^t \mathbf{a}_k}{\mathbf{a}_k^t \mathbf{a}_k} \mathbf{a}_k + \mathbf{r}^{(-k)}$$

where \mathbf{a}_k is a column from \mathbf{A} , and \mathbf{r}^{-k} is a residual vector.

- Note that \mathbf{r}^{-k} and \mathbf{a}_k are orthogonal.

$$(\mathbf{r}^{-k})^t \mathbf{a}_k = (\mathbf{y} - \frac{(\mathbf{y}^t \mathbf{a}_k) \mathbf{a}_k}{\mathbf{a}_k^t \mathbf{a}_k})^t \mathbf{a}_k = \mathbf{y}^t \mathbf{a}_k - (\mathbf{y}^t \mathbf{a}_k) = 0$$

- Hence we have: $\|\mathbf{y}\|^2 = (\mathbf{y}^t \mathbf{a}_k)^2 + \|\mathbf{r}^{-k}\|^2$

Properties of matching pursuit

- We want residuals with low magnitudes and hence the choice of the dictionary column with maximal dot product.

$$\theta_j = \arg \min_{\alpha} \|\mathbf{r} - \alpha \mathbf{a}_j\|^2 \rightarrow \theta_j = \frac{\mathbf{r}^t \mathbf{a}_j}{\|\mathbf{a}_j\|^2}$$

- The residual squared magnitude decreases across iterations.

Orthogonal Matching Pursuit (OMP)

- More sophisticated algorithm as compared to matching pursuit (MP).
- MP may pick the same dictionary column multiple times (why?) and hence it is inefficient.
- In OMP, the signal is approximated by successive projection onto those dictionary columns (i.e. columns of \mathbf{A}) that are associated with a current “support set”.
- The support set is also successively updated.

Pseudo-code

$$\mathbf{r}^{(0)} = \mathbf{y}, \boldsymbol{\theta} = \mathbf{0}; T^{(0)} = \phi, i = 0$$

while ($\|\mathbf{r}^{(i)}\|^2 > \varepsilon$)

{

$$(1) j = \arg \max_k | \mathbf{r}^{(i)T} \mathbf{a}_k / \|\mathbf{a}_k\|_2 |$$

$$(2) T^{(i+1)} = T^{(i)} \cup j; i = i + 1$$

$$(3) \boldsymbol{\theta}_{T^{(i)}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{A}_{T^{(i)}} \mathbf{w}\|^2 = \mathbf{A}_{T^{(i)}}^\mp \mathbf{y}$$

$$(4) \mathbf{r}^{(i)} = \mathbf{y} - \mathbf{A}_{T^{(i)}} \boldsymbol{\theta}_{T^{(i)}};$$

}

OUTPUT : $\boldsymbol{\theta}_{T^{(i)}}, \mathbf{A}_{T^{(i)}}$

Support set

Several coefficients
are re-computed in
each iteration

Pseudo-inverse

Sub-matrix of \mathbf{A}
containing only those
columns which lie in the
support set $T^{(i)}$

OMP versus MP

- Unlike MP, in OMP, the residual at an iteration is always orthogonal to all currently selected column vectors of \mathbf{A} (proof next slide).
- Therefore (unlike MP) OMP never re-selects any column vector (why?).
- OMP is costlier per iteration (due to pseudo-inverse).
- OMP always gives the optimal approximation w.r.t. the selected subset of the dictionary (note: this does not mean that the selected subset itself was optimal).
- In order for the pseudo-inverse to be well-defined, the number of OMP iterations should not be more than m .

OMP residuals

$$\boldsymbol{\theta}_{\mathbf{T}^{(i)}} = \arg \min_{\mathbf{w}} \left\| \mathbf{y} - \mathbf{A}_{\mathbf{T}^{(i)}} \mathbf{w} \right\|^2 = \mathbf{A}_{\mathbf{T}^{(i)}}^{\mp} \mathbf{y}$$

$$= (\mathbf{A}_{\mathbf{T}^{(i)}}^T \mathbf{A}_{\mathbf{T}^{(i)}})^{-1} \mathbf{A}_{\mathbf{T}^{(i)}}^T \mathbf{y}$$

$$\mathbf{r}^{(i)} = \mathbf{y} - \mathbf{A}_{\mathbf{T}^{(i)}} \boldsymbol{\theta}_{\mathbf{T}^{(i)}}$$

$$\therefore \mathbf{A}_{\mathbf{T}^{(i)}}^T \mathbf{r}^{(i)} = \mathbf{A}_{\mathbf{T}^{(i)}}^T (\mathbf{y} - \mathbf{A}_{\mathbf{T}^{(i)}} \boldsymbol{\theta}_{\mathbf{T}^{(i)}})$$

$$= \mathbf{A}_{\mathbf{T}^{(i)}}^T \mathbf{y} - \mathbf{A}_{\mathbf{T}^{(i)}}^T \mathbf{A}_{\mathbf{T}^{(i)}} (\mathbf{A}_{\mathbf{T}^{(i)}}^T \mathbf{A}_{\mathbf{T}^{(i)}})^{-1} \mathbf{A}_{\mathbf{T}^{(i)}}^T \mathbf{y}$$

$$= \mathbf{0}$$

OMP for noisy signals

- For non-noisy signals, OMP is run with $\varepsilon = 0$, i.e. until the magnitude of the residual is zero.
- If there is noise, then ε should be set based on the noise variance.

OMP: error bounds

- Various error bounds on the performance of OMP have been analyzed.
- Example the following theorem due to Tropp and Gilbert (2007): Let $\delta \in (0, 0.36)$, $m \geq Cs \log(n/\delta)$. Given m measurements of the s -sparse n -dimensional signal \mathbf{x} taken with a standard Gaussian matrix of size $m \times n$, we can recover \mathbf{x} exactly with probability more than $1 - 2\delta$. The constant C turns out to be ≤ 20 .

Experiment on OMP

- Simulation of compressive sensing with a Gaussian random measurement matrix of size $m \times n$
- Data: patches of size 8×8 ($n = 64$) from the Barbara image, with addition of Gaussian noise with mean 0 and sigma = $0.05 \times$ mean intensity of coded patch
- m varied from $0.1n$ to $0.7n$.



Original barbara image

Reconstruction results with $m = fn$ where $f = 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1$ in **column-wise** order



Iterative Shrinkage and Thresholding Algorithm (ISTA)

Optimization algorithm

- There are many algorithms for solving the optimization problem below.

$$J(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1$$

$\boldsymbol{\theta}$ = random initialization

Repeat till convergence :

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

Denoising: $\mathbf{A} = \mathbf{U}$

Deblurring: $\mathbf{A} = \mathbf{H}\mathbf{U}$,

\mathbf{H} = circulant/block circulant matrix derived from blur kernel

Iterative Shrinkage and Thresholding Algorithm (ISTA)

- An iterative algorithm to solve:

$$\text{Problem PU : } \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1$$

$$\mathbf{y} \in \mathbf{R}^m, \boldsymbol{\theta} \in \mathbf{R}^n, \mathbf{A} \in \mathbf{R}^{m \times n}, m \ll n, n \text{ is large}$$

- Useful in compressive reconstructions, but also used in various image restoration problems such as deblurring.
- Can be implemented in 4 lines of MATLAB code.

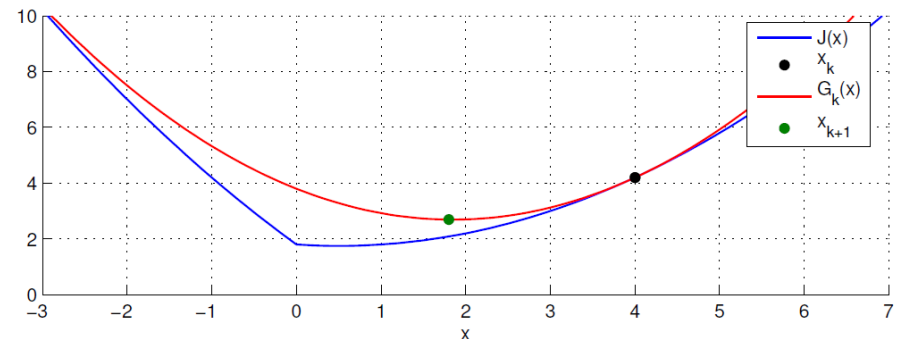
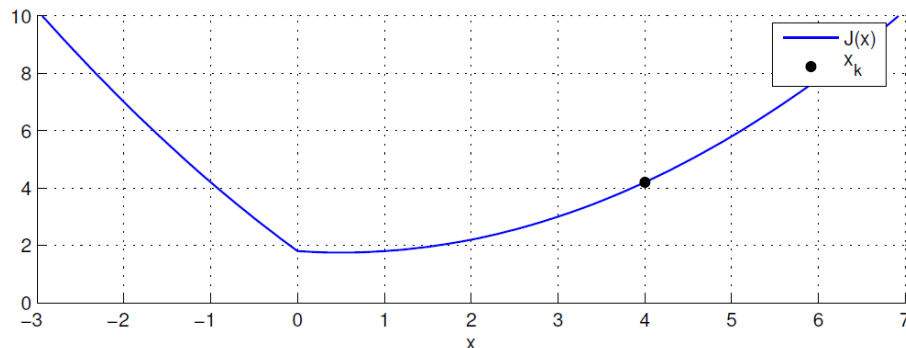
ISTA

- Directly solving for θ in closed form is not possible due to the L1-norm term, so we resort to iterative solutions.
- ISTA seeks to solve this difficult minimization problem by a series of smaller minimization problems.
- Given an initial guess θ_k , we seek to find a new vector θ_{k+1} such that $J(\theta_{k+1}) < J(\theta_k)$.

ISTA

- The new vector θ_{k+1} is chosen to minimize a so-called **majorizer function** $M_k(\theta)$ such that $M_k(\theta) \geq J(\theta)$ for all θ , and $M_k(\theta_k) = J(\theta_k)$.
- The majorizer function will be different at each iteration (hence the subscript k).

Image source: tutorial by Ivan Selesnick



ISTA

- Overall algorithm is as follows:
 1. For $k = 0$, initialize θ_0 .
 2. Choose majorizer $M_k(\theta)$ such that $M_k(\theta) \geq J(\theta)$ for all θ , and $M_k(\theta_k) = J(\theta_k)$.
 3. Select θ_{k+1} to minimize the majorizer $M_k(\theta)$.
 4. Set $k = k+1$. Go to step 2.

Note

$J(\theta_{k+1}) \leq M_k(\theta_{k+1})$ by definition of M_k
 $< M_k(\theta_k)$ as θ_{k+1} minimizes M_k
 $= J(\theta_k)$ by definition of M_k
Hence $J(\theta_{k+1}) < J(\theta_k)$

ISTA

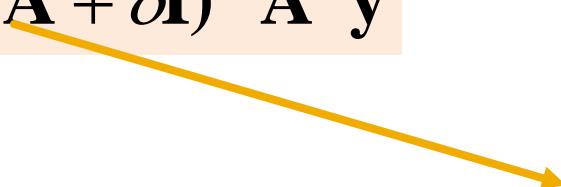
- Suppose we want to solve:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2$$

- The intuitive solution is:

$$\mathbf{A}^T \mathbf{y} = \mathbf{A}^T \mathbf{A} \boldsymbol{\theta}$$

$$\boldsymbol{\theta} = (\mathbf{A}^T \mathbf{A} + \delta \mathbf{I})^{-1} \mathbf{A}^T \mathbf{y}$$



Humongous matrix – very hard to store in memory – forget about inverting it! ☹️

ISTA

- Suppose we want to solve:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2$$

- One possible majorizer is:

$$M_k(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + \text{non - negative function of } \boldsymbol{\theta}$$

$$= \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^t (\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A})(\boldsymbol{\theta} - \boldsymbol{\theta}_k),$$

for $\alpha > \max \text{ eigenvalue of } \mathbf{A}^T \mathbf{A}$

ISTA

- One possible majorizer is:

$$\begin{aligned} M_k(\boldsymbol{\theta}) &= \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + \text{non-negative function of } \boldsymbol{\theta} \\ &= \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^t (\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A})(\boldsymbol{\theta} - \boldsymbol{\theta}_k), \end{aligned}$$

for $\alpha > \text{max eigenvalue of } \mathbf{A}^T \mathbf{A}$

So that $\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A}$ is a positive semi-definite matrix

$$\Leftrightarrow \forall \boldsymbol{\theta}, (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^t (\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A})(\boldsymbol{\theta} - \boldsymbol{\theta}_k) \geq 0$$

Note that $\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A} = \mathbf{V} \alpha \mathbf{I} \mathbf{V}^T - \mathbf{V} \mathbf{D} \mathbf{V}^T = \mathbf{V}(\alpha \mathbf{I} - \mathbf{D}) \mathbf{V}^T$
has all positive eigenvalue s

ISTA

- One possible majorizer is:

$$M_k(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^t (\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A})(\boldsymbol{\theta} - \boldsymbol{\theta}_k),$$

for $\alpha > \max$ eigenvalue of $\mathbf{A}^T \mathbf{A}$

- The minimizer is given as:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}} (M_k(\boldsymbol{\theta})) &= 0 \\ \rightarrow \boldsymbol{\theta} &= \boldsymbol{\theta}_k + \frac{1}{\alpha} \mathbf{A}^T (\mathbf{y} - \mathbf{A}\boldsymbol{\theta}_k) \\ \rightarrow \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta} \end{aligned}$$

Notice: this yields us an iterative solver that does not require inversion of $\mathbf{A}^T \mathbf{A}$ which is a very large matrix

ISTA

- Now consider:

$$J(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1$$

- The majorizer is now:

$$M_k(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^t (\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A})(\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \lambda \|\boldsymbol{\theta}\|_1$$

for $\alpha > \max$ eigenvalue of $\mathbf{A}^T \mathbf{A}$

ISTA

- The majorizer is now:

$$M_k(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^t (\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A})(\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \lambda \|\boldsymbol{\theta}\|_1$$

for $\alpha > \max$ eigenvalue of $\mathbf{A}^T \mathbf{A}$

- The minimizer is given as:

$$\frac{\partial}{\partial \boldsymbol{\theta}} (M_k(\boldsymbol{\theta})) = 0$$

$$\rightarrow \boldsymbol{\theta}_k + \frac{1}{\alpha} \mathbf{A}^T (\mathbf{y} - \mathbf{A}\boldsymbol{\theta}_k) = \boldsymbol{\theta}_{k+1} + \frac{\lambda}{2\alpha} \text{sign}(\boldsymbol{\theta}_{k+1})$$

ISTA

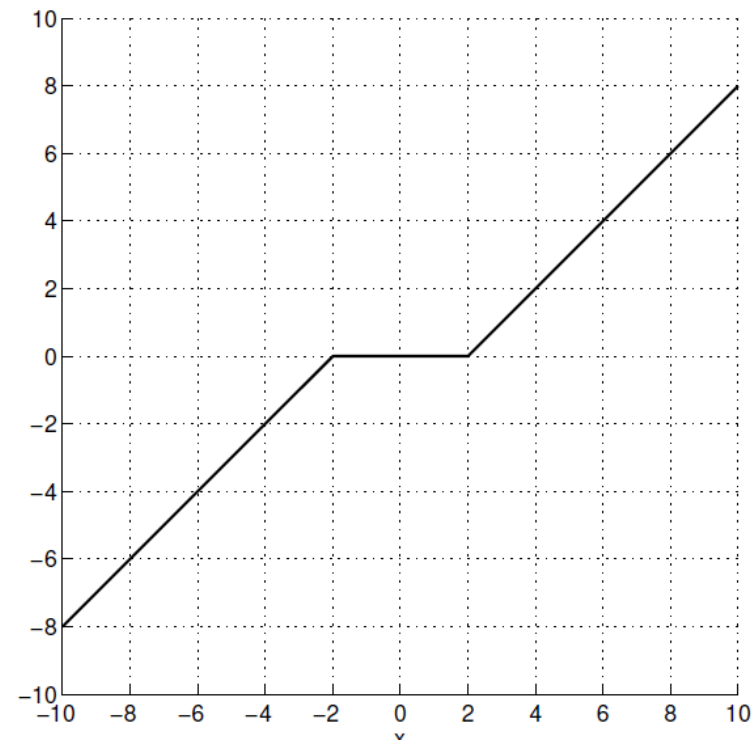
- Consider the following equation in x

$$y = x + \lambda \text{sign}(x)$$

Note: $\lambda > 0$

- Its solution is given as

$$\begin{aligned} x &= \text{soft}(y; \lambda) = y - \lambda, y \geq \lambda \\ &= y + \lambda, y \leq -\lambda \\ &= 0, |y| < \lambda \end{aligned}$$



ISTA: explaining soft thresholding

- When x is positive, $y = x + \lambda$. When $y > \lambda$, $x = y - \lambda$.
- When x is negative, $y = x - \lambda$. When $y < -\lambda$, $x = y + \lambda$.
- When x is zero, one has to refer to the **sub-differential** of the L_1 norm at $x = 0$, which is $[-1, 1]$.
- As per optimality conditions, we must have $0 \in x - y + \lambda[-1, 1]$, i.e. $0 \in -y + \lambda[-1, 1]$, i.e. $|y| \leq \lambda$.

ISTA: explaining soft thresholding – explaining subdifferential

- The sub-derivative of a convex function f at point x_0 in open interval I is a real number c such that
$$\forall x \in I, f(x) - f(x_0) \geq c(x - x_0)$$
- The set of sub-derivatives in the closed interval $[a, b]$ is called the sub-differential where we have:

$$a = \lim_{x \rightarrow x_0^-} \frac{f(x) - f(x_0)}{x - x_0}, b = \lim_{x \rightarrow x_0^+} \frac{f(x) - f(x_0)}{x - x_0}$$

ISTA

- The minimizer is given as:

$$\frac{\partial}{\partial \boldsymbol{\theta}} (\mathbf{M}_k(\boldsymbol{\theta})) = 0$$

$$\rightarrow \boldsymbol{\theta}_k + \frac{1}{\alpha} \mathbf{A}^T (\mathbf{y} - \mathbf{A} \boldsymbol{\theta}_k) = \boldsymbol{\theta}_{k+1} + \frac{\lambda}{2\alpha} \text{sign}(\boldsymbol{\theta}_{k+1})$$

$$\boldsymbol{\theta}_{k+1} = \text{soft} \left(\boldsymbol{\theta}_k + \frac{1}{\alpha} \mathbf{A}^T (\mathbf{y} - \mathbf{A} \boldsymbol{\theta}_k), \frac{\lambda}{2\alpha} \right)$$

ISTA: Algorithm

Initialize $\boldsymbol{\theta}_0$ randomly, $k = 0$,

$\alpha = \max$ eigenvalue of $\mathbf{A}^T \mathbf{A}$

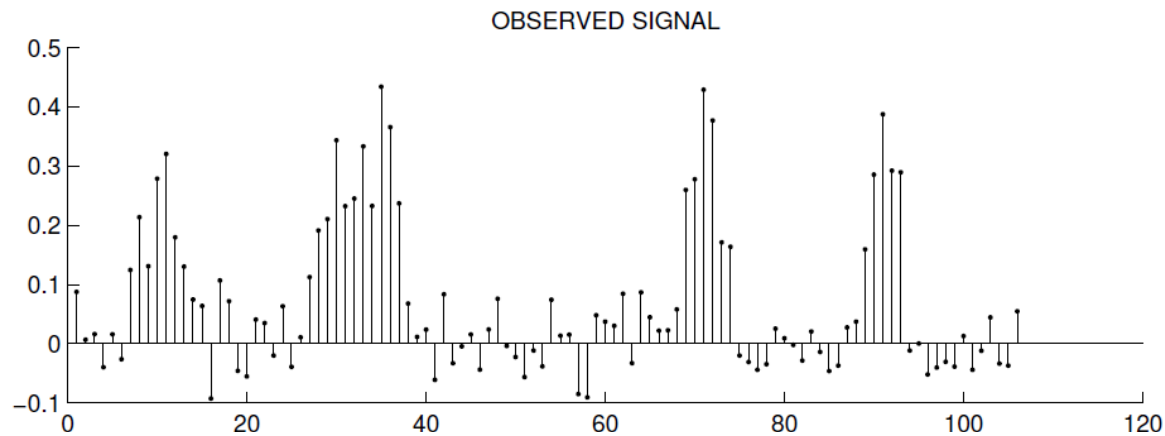
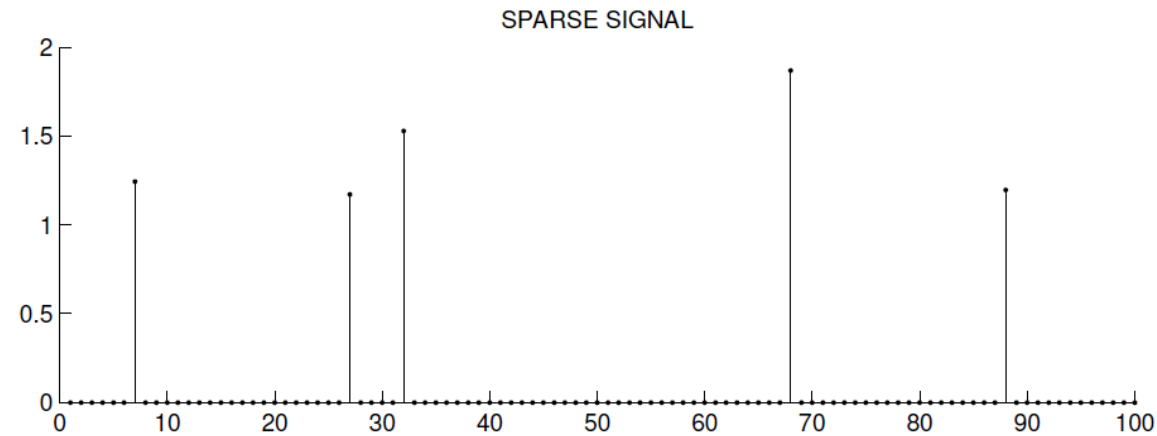
Repeat till convergence :

$$\boldsymbol{\theta}_{k+1} = \text{soft} \left(\boldsymbol{\theta}_k + \frac{1}{\alpha} \mathbf{A}^T (\mathbf{y} - \mathbf{A} \boldsymbol{\theta}_k), \frac{\lambda}{2\alpha} \right)$$

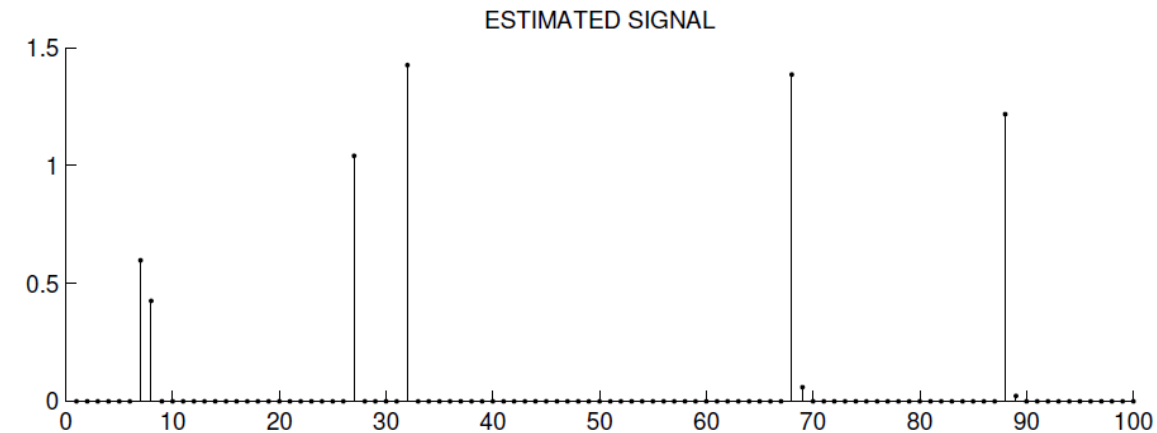
4-5 lines of MATLAB code

Image source: tutorial
by Ivan Selesnick

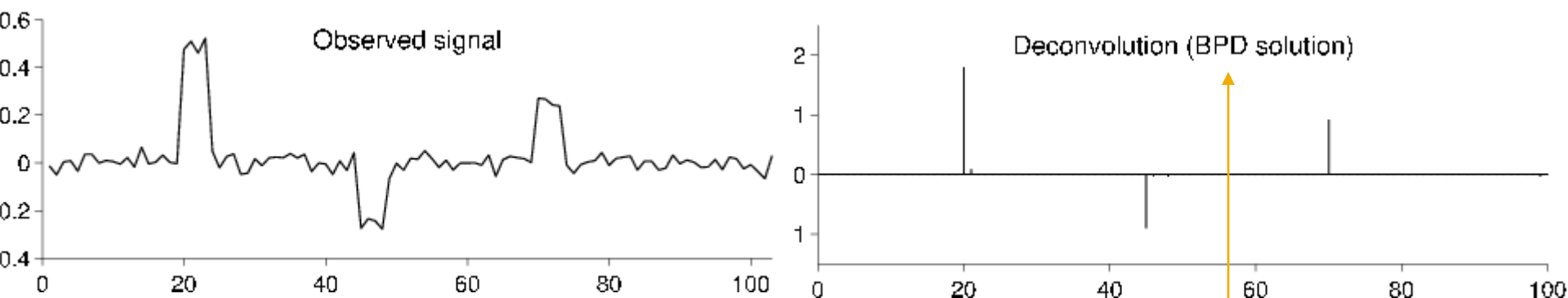
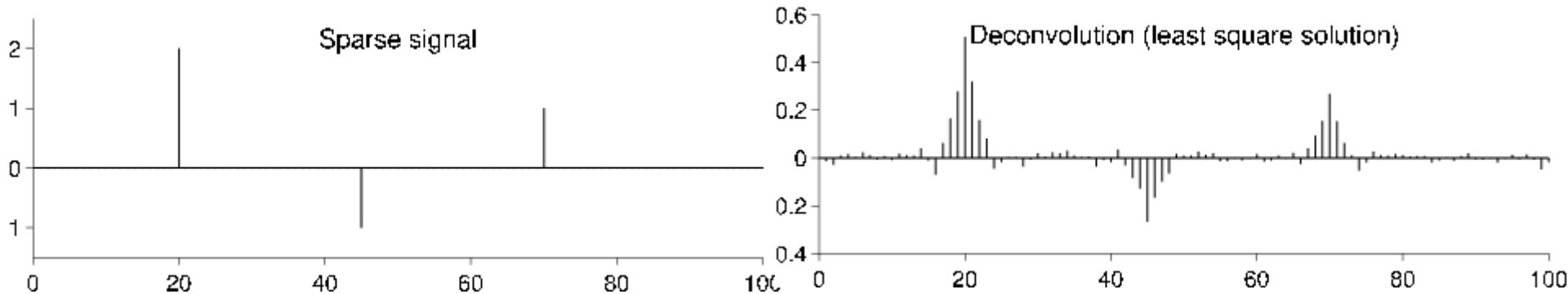
Signal x



Signal $y = h * x + \text{noise}$



Estimate
of signal x



i.e. solution with
Laplacian prior

Image source: tutorial
by Ivan Selesnick