CS 754, Ajit Rajwade

Compressive Sensing: Reconstruction Algorithms

Compressive sensing: reconstruction problem

- The basic problem is:

$$(P0): \min \|\boldsymbol{\theta}\|_0 \\ \text{such that } \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 \le \varepsilon \\ \mathbf{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$$

$$\Rightarrow$$

$$(P1): \min \|\boldsymbol{\theta}\|_1 \\ \text{such that } \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 \le \varepsilon$$

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

Compressive Reconstruction Algorithms

- **Category 1:** Problem P0 is NP-hard. So instead, run an *approximation algorithm* which can be efficiently solved.
- **Category 2:** Seek to directly solve problem P1. 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 **θ** of a signal **y** in an over-complete "dictionary" matrix **A** ϵ $\mathcal{R}^{m\text{x}n}$ (i.e. $m << n$)
- 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 **A** which has maximal dot product with a so-called *residual vector* (initialized to **y** in the beginning).

Pseudo-code

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

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

$$\{$$

$$j = \arg\max_{l} |\mathbf{r}^{(i)^T}\mathbf{a}_l / \left\|\mathbf{a}_l\right\|^2|$$

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

$$\}$$

$$\text{OUTPUT}: \{\theta_{\mathrm{j}}\}$$

Select the column of **A** that is maximally correlated (highest dot-product) with the residual

"j" or "l" is an index for columns of **A**

Properties of matching pursuit

- MP decomposes any signal **y** into the form

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

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

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

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

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

Properties of matching pursuit

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

$$\theta_j = \arg\min_{\alpha} \left\| \mathbf{r} - \alpha \mathbf{a_j} \right\|^2 \rightarrow \theta_j = \frac{\mathbf{r}^t \mathbf{a_j}}{\left\| \mathbf{a_j} \right\|^2}$$

- 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 **A**) that are associated with a current "support set".
- The support set is also successively updated.

Pseudo-code

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

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

$$\{$$

$$(1)\ \mathbf{a}_j = \arg\max_j |\mathbf{r}^{(i)^T} \mathbf{a}_j / \left\|\mathbf{a}_j\right\|^2 |$$

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

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

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

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

Support set

Several coefficients are re-computed in each iteration

Sub-matrix containing only those columns which lie in the support set

OMP versus MP

- Unlike MP, in OMP, the residual at an iteration is always orthogonal to all currently selected elements (proof next slide).
- Therefore (unlike MP) OMP never re-selects any element (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

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

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

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

$$\therefore \mathbf{A}_{\mathbf{T}^{(\mathbf{i})}}^T \mathbf{r}^{(i+1)} = \mathbf{A}_{\mathbf{T}^{(\mathbf{i})}}^T (\mathbf{y} - \mathbf{A}_{\mathbf{T}^{(\mathbf{i})}} \boxed{\boldsymbol{\theta}_{\mathbf{T}^{(\mathbf{i})}}})$$

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

$$= \mathbf{0}$$

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 $\delta \in (0,0.36)$, $m >= Cs\log(n/\delta)$. Given m measurements of the S-sparse n-dimensional signal **x** taken with a standard Gaussian matrix of size m x n, we can recover **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 x n
- Data: patches of size 8 x 8 (n = 64) from the Barbara image, with addition of Gaussian noise with mean 0 and sigma = 0.05 x 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)

Ajit Rajwade

Optimization algorithm

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

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

$$\boldsymbol{\theta} = \text{random initialization}$$

$$\text{Repeat till convergence:}$$

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

Denoising: **A** = **U**

Deblurring: **A** = **HU,**

H = circulant/block circulant matrix derived from blur kernel

Iterative Shrinkage and Thresholding Algorithm (ISTA)

- An iterative algorithm to solve:

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

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

- 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

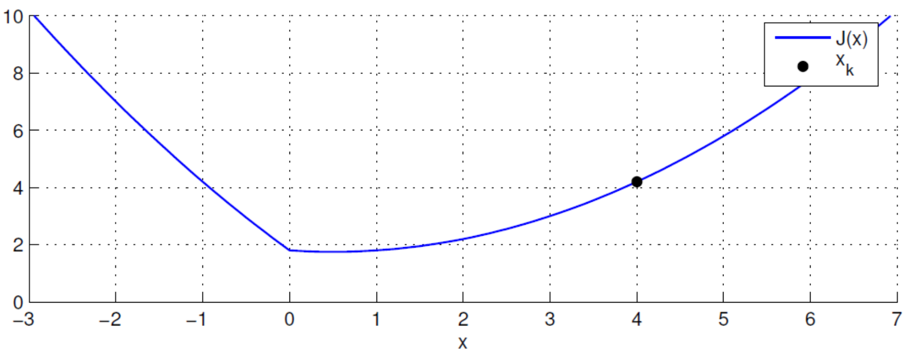
- Directly solving for $\boldsymbol{\theta}$ in closed form is not possible due to the L1-norm term, so we resort to iterative solutions.
- ISTA seeks to solve this difficult minimization problem by a series of smaller minimization problems.
- Given an initial guess $\boldsymbol{\theta}_k$, we seek to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

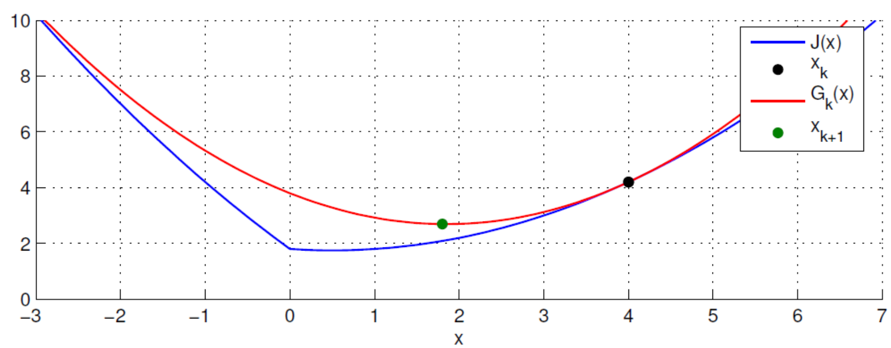
ISTA

- The new vector $\boldsymbol{\theta}_{k+1}$ is chosen to minimize a so-called **majorizer function** $M_k(\boldsymbol{\theta})$ such that $M_k(\boldsymbol{\theta}) >= J(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$, and $M_k(\boldsymbol{\theta}_k) = J(\boldsymbol{\theta}_k)$.

- 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 $\boldsymbol{\theta}_0$.
2. Choose majorizer $M_k(\boldsymbol{\theta})$ such that $M_k(\boldsymbol{\theta}) >= J(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$, and $M_k(\boldsymbol{\theta}_k) = J(\boldsymbol{\theta}_k)$.
3. Select $\boldsymbol{\theta}_{k+1}$ to minimize the majorizer $M_k(\boldsymbol{\theta})$.
4. Set $k = k+1$. Go to step 2.

Note
$J(\boldsymbol{\theta}_{k+1}) <= M_k(\boldsymbol{\theta}_{k+1})$ by definition of M_k
$< M_k(\boldsymbol{\theta}_k)$ as $\boldsymbol{\theta}_{k+1}$ minimizes M_k
$= J(\boldsymbol{\theta}_k)$ by definition of M_k
Hence $J(\boldsymbol{\theta}_{k+1}) < J(\boldsymbol{\theta}_k)$

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:

$$M_k(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + \text{non-negative function of } \boldsymbol{\theta}$$
$$= \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + (\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathbf{k}})^t (\alpha\mathbf{I} - \mathbf{A}^T\mathbf{A})(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathbf{k}}),$$
$$\text{for } \alpha > \text{max eigenvalue of } \mathbf{A}^T\mathbf{A}$$

ISTA

- One possible majorizer is:

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

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

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

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

$$\leftrightarrow \forall\boldsymbol{\theta}, (\boldsymbol{\theta} - \boldsymbol{\theta}_\mathbf{k})^t(\alpha\mathbf{I} - \mathbf{A}^\mathrm{T}\mathbf{A})(\boldsymbol{\theta} - \boldsymbol{\theta}_\mathbf{k}) \geq 0$$

$$\text{Note that } \alpha\mathbf{I} - \mathbf{A}^\mathrm{T}\mathbf{A} = \mathbf{V}\alpha\mathbf{I}\mathbf{V}^T - \mathbf{V}\mathbf{D}\mathbf{V}^T = \mathbf{V}(\alpha\mathbf{I} - \mathbf{D})\mathbf{V}^T$$

has all positive eigenvalues

ISTA

- One possible majorizer is:

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

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

- The minimizer is given as:

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

$$\rightarrow \boldsymbol{\theta} = \boldsymbol{\theta}_k + \frac{1}{\alpha}\mathbf{A}^T(\mathbf{y} - \mathbf{A}\boldsymbol{\theta}_k)$$

$$\rightarrow \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}$$

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:

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

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

ISTA

- The majorizer is now:

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

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

- The minimizer is given as:

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

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

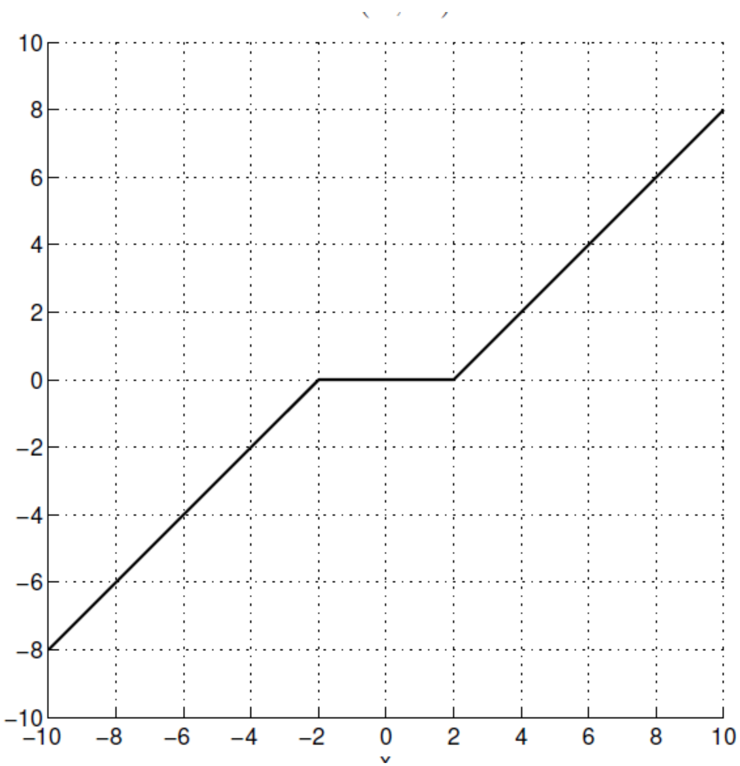
ISTA

- Consider the following equation in *x*

$$y = x + \lambda \text{sign}(x)$$
$$Note: \lambda > 0$$

- Its solution is given as

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



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

ISTA

- The minimizer is given as:

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

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

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

ISTA: Algorithm

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

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

Repeat till convergence :

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

4-5 lines of MATLAB code

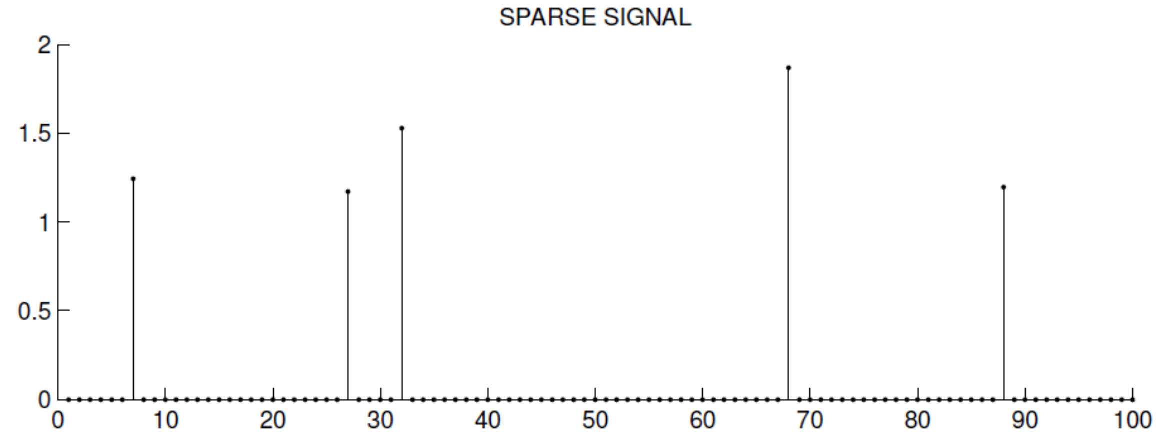
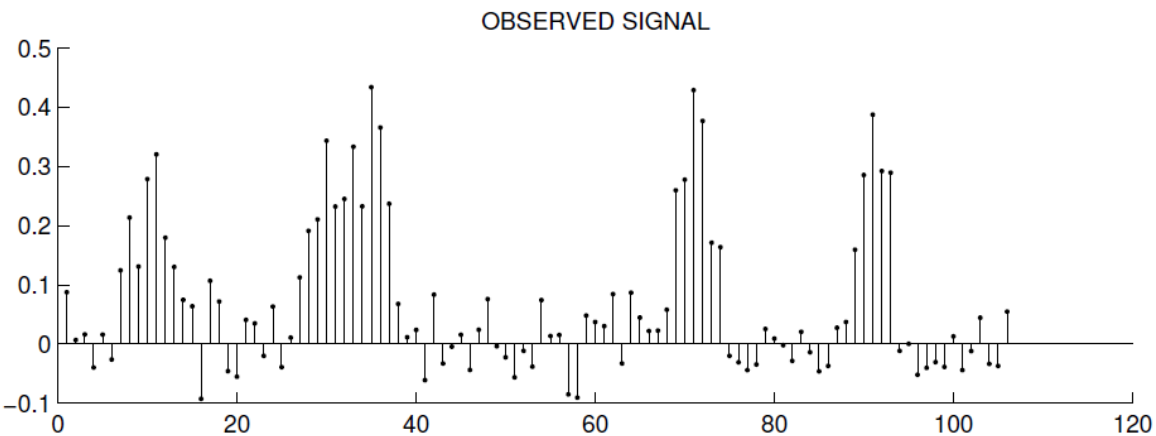
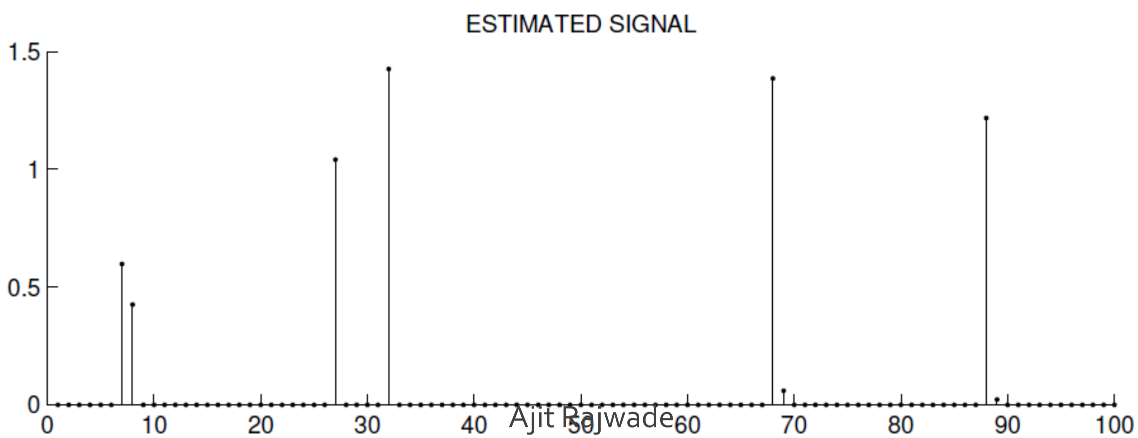


Image source: tutorial by Ivan Selesnick

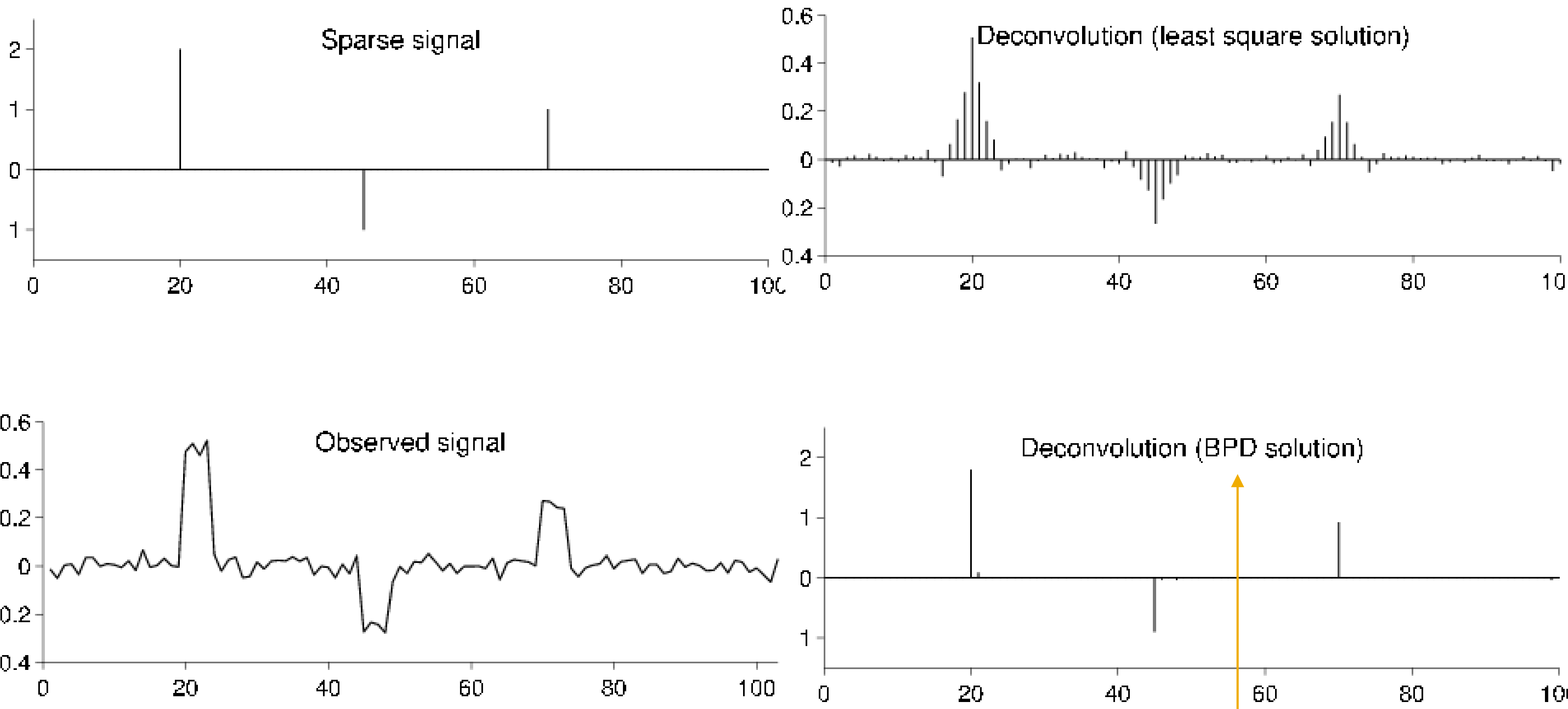
Signal x



Signal y = h*x + noise



Estimate of signal x



i.e. solution with
Laplacian prior

Image source: tutorial by Ivan Selesnick