Sparsity Methods for Systems and Control

Greedy Algorithms

Masaaki Nagahara[1,2]

[1]The University of Kitakyushu
nagahara@kitakyu-u.ac.jp
[2]IIT Bombay (Visiting Faculty)

Table of Contents

Table of Contents

ℓ^0 Optimization

ℓ^0 optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|x\|_0 \quad \text{subject to} \quad y = \Phi x,$$

Here we directly solve the ℓ^0 optimization problem without any convex relaxations.

Mutual coherence

Mutual coherence

For a matrix $\Phi = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_n] \in \mathbb{R}^{m \times n}$ with $\boldsymbol{\phi}_i \in \mathbb{R}^m$, $i = 1, 2, \ldots, n$, we define the mutual coherence $\mu(\Phi)$ by

$$\mu(\Phi) \triangleq \max_{\substack{i,j=1,\ldots,n\\ i \neq j}} \frac{|\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle|}{\|\boldsymbol{\phi}_i\|_2 \|\boldsymbol{\phi}_j\|_2}.$$

Mutual coherence

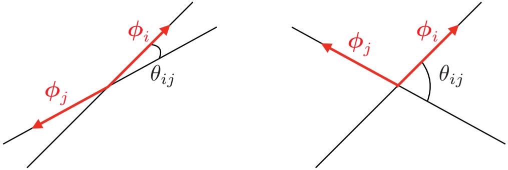
- Mutual coherence

$$\mu(\Phi) \triangleq \max_{\substack{i,j=1,\dots,n\\ i\neq j}} \frac{|\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle|}{\|\boldsymbol{\phi}_i\|_2 \|\boldsymbol{\phi}_j\|_2}.$$

- The cosine of the angle θ_{ij} between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$

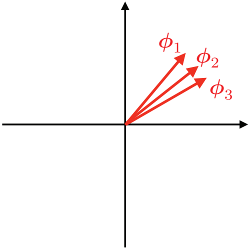
$$\cos\theta_{ij} = \frac{\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle}{\|\boldsymbol{\phi}_i\|_2 \|\boldsymbol{\phi}_j\|_2}.$$

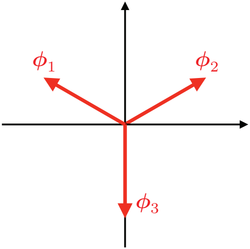
- $\theta_{ij} \approx 0^\circ$ ($|\cos\theta_{ij}| \approx 1$) $\Rightarrow$ $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ are coherent
- $\theta_{ij} \approx 90^\circ$ ($|\cos\theta_{ij}| \approx 0$) $\Rightarrow$ $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ are incoherent



Mutual coherence

- For any Φ, we have $0 \leq \mu(\Phi) \leq 1$.
- Some of $\phi_1, \ldots, \phi_n$ are similar $\Rightarrow$ $\mu(\Phi)$ is large ($\mu(\Phi) \approx 1$)
- $\phi_1, \ldots, \phi_n$ are uniformly spread $\Rightarrow$ $\mu(\Phi)$ is small





Characterization of ℓ^0 solution

ℓ^0 optimization

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad \|\boldsymbol{x}\|_0 \ \text{ subject to } \ \boldsymbol{y} = \Phi\boldsymbol{x},$$

Theorem

If there exists a vector $\boldsymbol{x} \in \mathbb{R}^n$ that satisfies linear equation $\Phi\boldsymbol{x} = \boldsymbol{y}$, and

$$\|\boldsymbol{x}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\Phi)}\right),$$

then $\boldsymbol{x}$ is the sparsest solution of the linear equation (i.e. the solution of the ℓ^0 optimization).

Characterization of ℓ^0 solution

- Suppose $\mu(\Phi) < 1$.
- Then,

$$\frac{1}{2}\left(1 + \frac{1}{\mu(\Phi)}\right) > 1.$$

- From the theorem, if there exists a 1-sparse vector $\boldsymbol{x}$ (i.e. $\|\boldsymbol{x}\|_0 = 1$) satisfying $\Phi \boldsymbol{x} = \boldsymbol{y}$, this is ℓ^0 optimal.

Finding 1-sparse vector

- Assume $\boldsymbol{x}$ is 1-sparse. ($\|\boldsymbol{x}\|_0 = 1$)
- The equation $\boldsymbol{y} = \Phi \boldsymbol{x}$ becomes

$$\boldsymbol{y} = \Phi \boldsymbol{x} = x_1 \boldsymbol{\phi}_1 + x_2 \boldsymbol{\phi}_2 + \cdots + x_n \boldsymbol{\phi}_n.$$

- Define the error

$$e(i) \triangleq \min_{x \in \mathbb{R}} \|x \boldsymbol{\phi}_i - \boldsymbol{y}\|_2^2.$$

- A 1-sparse vector satisfying $\boldsymbol{y} = \Phi \boldsymbol{x}$ is found by searching i that minimizes $e(i)$.
  - Actually, if a 1-sparse solution exists, $e(i) = 0$ for some i.
- It needs $O(n)$ computations.

Characterization of ℓ^0 solution

- Suppose $\mu(\Phi) < 1/3$.
- Then,

$$\frac{1}{2}\left(1 + \frac{1}{\mu(\Phi)}\right) > 2.$$

- From the theorem, if there exists a 2-sparse vector $\boldsymbol{x}$ (i.e. $\|\boldsymbol{x}\|_0 \leq 2$) satisfying $\Phi\boldsymbol{x} = \boldsymbol{y}$, this is ℓ^0 optimal.
- Finding 2-sparse vector needs $O(n^2)$ computations.

Characterization of ℓ^0 solution

- Suppose $\mu(\Phi) < 1/(2k - 1)$.
- Then,

$$\frac{1}{2}\left(1 + \frac{1}{\mu(\Phi)}\right) > k.$$

- From the theorem, if there exists a k-sparse vector $\boldsymbol{x}$ (i.e. $\|\boldsymbol{x}\|_0 \leq k$) satisfying $\Phi\boldsymbol{x} = \boldsymbol{y}$, this is ℓ^0 optimal.
- Finding k-sparse vector $\boldsymbol{x}$ satisfying $\Phi\boldsymbol{x} = \boldsymbol{y}$ is almost impossible when k is large, since it needs $O(n^k)$ computations.
- In this chapter, we learn greedy methods for this problem.

Table of Contents

ℓ^0 optimization

ℓ^0 optimization

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad \|\boldsymbol{x}\|_0 \ \text{ subject to } \ \boldsymbol{y} = \Phi\boldsymbol{x},$$

To solve this, we employ greedy methods.

Matching pursuit

- Finding 1-sparse solution of $\boldsymbol{y} = \Phi\boldsymbol{x}$ is of $O(n)$.
- This is done by searching an index i that minimizes

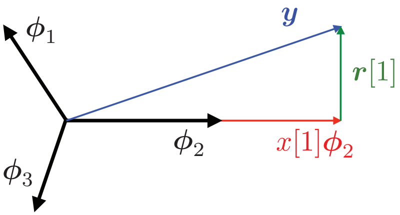
$$e(i) \triangleq \min_{x \in \mathbb{R}} \|x\boldsymbol{\phi}_i - \boldsymbol{y}\|_2^2.$$

- If there is no 1-sparse solution, $e(i)$ never gets 0, but we iterate this minimization.

Matching Pursuit (MP)

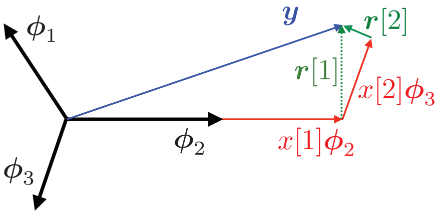
1. Find a 1-sparse vector $\boldsymbol{x}[1]$ that minimizes $\|\Phi\boldsymbol{x} - \boldsymbol{y}\|_2$.
2. For $k = 1, 2, 3, \ldots$ do
   - Compute the residue $\boldsymbol{r}[k] = \boldsymbol{y} - \Phi\boldsymbol{x}[k]$
   - Find a 1-sparse vector $\boldsymbol{x}^*$ that minimizes $\|\Phi\boldsymbol{x} - \boldsymbol{r}[k]\|_2$
   - Set $\boldsymbol{x}[k+1] = \boldsymbol{x}[k] + \boldsymbol{x}^*$

Matching pursuit



- $k = 1$
- $i[1] = 2$ minimizes $e(i) = \min_{x \in \mathbb{R}} \|x\boldsymbol{\phi}_i - \boldsymbol{y}\|_2^2$.
- $\boldsymbol{y} = x[1]\boldsymbol{\phi}_{i[1]} + \boldsymbol{r}[1]$

Matching pursuit



- $k = 2$
- $i[2] = 3$ minimizes $\min_{x \in \mathbb{R}} \|x\boldsymbol{\phi}_i - \boldsymbol{r}[1]\|_2^2$.
- $\boldsymbol{r}[1] = x[2]\boldsymbol{\phi}_{i[2]} + \boldsymbol{r}[2]$
- $\boldsymbol{y} = x[1]\boldsymbol{\phi}_{i[1]} + x[2]\boldsymbol{\phi}_{i[2]} + \boldsymbol{r}[2]$
- We can continue this for $k = 3, 4, 5, \ldots$.

Matching pursuit

- After k step, we have

$$\boldsymbol{y} = x[1]\boldsymbol{\phi}_{i[1]} + x[2]\boldsymbol{\phi}_{i[2]} + \cdots + x[k]\boldsymbol{\phi}_{i[k]} + \boldsymbol{r}[k].$$

- The vector $\boldsymbol{y}$ is approximated by

$$\tilde{\boldsymbol{y}}[k] \triangleq x[1]\boldsymbol{\phi}_{i[1]} + x[2]\boldsymbol{\phi}_{i[2]} + \cdots + x[k]\boldsymbol{\phi}_{i[k]} = \Phi\boldsymbol{x}[k]$$

  with k-sparse vector

$$\boldsymbol{x}[k] \triangleq x[1]\boldsymbol{e}_{i[1]} + x[2]\boldsymbol{e}_{i[2]} + \cdots + x[k]\boldsymbol{e}_{i[k]}$$

- One needs $O(nk)$ computations to obtain k-sparse vector that approximates a solution of $\boldsymbol{y} = \Phi\boldsymbol{x}$.

Finding 1-sparse vector

- For MP, we need to obtain a 1-sparse vector that minimizes $\|\Phi \boldsymbol{x} - \boldsymbol{r}\|_2^2$.
- Since $\Phi = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_n]$ and $\boldsymbol{x} = [x_1, \dots, x_n]^\top$,

$$\Phi \boldsymbol{x} = x_1 \boldsymbol{\phi}_1 + \cdots + x_n \boldsymbol{\phi}_n.$$

- Since $\boldsymbol{x}$ is 1-sparse, we just need to obtain the index i^* and the coefficient x_{i^*} that minimizes $\|x_i \boldsymbol{\phi}_i - \boldsymbol{r}\|_2^2$.
- We have

$$e(i) = \min_x \|x \boldsymbol{\phi}_i - \boldsymbol{r}\|_2^2 = \|\boldsymbol{y}\|_2^2 - \frac{\langle \boldsymbol{\phi}_i, \boldsymbol{r} \rangle^2}{\|\boldsymbol{\phi}_i\|_2^2}$$

- From this

$$i^* = \arg\min_i e(i) = \arg\max \frac{\langle \boldsymbol{\phi}_i, \boldsymbol{r} \rangle^2}{\|\boldsymbol{\phi}_i\|_2^2}.$$

Matching pursuit: convergence

Theorem

Assume that dictionary $\{\phi_1, \phi_2, \dots, \phi_n\}$ *has* m *linearly independent vectors (i.e.* $\operatorname{rank}\Phi = m$*). Then there exists a constant* $c \in (0,1)$ *such that*

$$\|\boldsymbol{r}[k]\|_2^2 \leq c^k \|\boldsymbol{y}\|_2^2, \quad k = 0, 1, 2, \dots. \tag{1}$$

- The residue $\boldsymbol{r}[k]$ monotonically decreases and

$$\lim_{k \to \infty} \boldsymbol{r}[k] = \boldsymbol{0}$$

- The convergence rate is first order; the residue decreases exponentially, that is, $O(c^k)$.
- Much faster than FISTA with $O(1/k^2)$.

Orthogonal Matching Pursuit (OMP)

- MP cannot always achieve $\boldsymbol{r}[k] = \mathbf{0}$ with finite k.
- This is because MP may choose an index $i[k]$ that was already chosen in previous steps.
- *OMP (Orthogonal Matching Pursuit)* can achieve $\boldsymbol{r}[k] = \mathbf{0}$ in a finite number of iterations.

Orthogonal Matching Pursuit (OMP)

- Choose the index $i[k]$ of a 1-sparse vector that minimizes $\|\Phi \boldsymbol{x} - \boldsymbol{r}[k-1]\|_2$:

$$i[k] = \underset{i \in \{1,\dots,n\}}{\arg\max} \frac{\langle \boldsymbol{\phi}_i, \boldsymbol{r}[k-1] \rangle^2}{\|\boldsymbol{\phi}_i\|_2^2}, \quad \boldsymbol{r}[0] = \boldsymbol{y}, \quad k = 1, 2, \dots$$

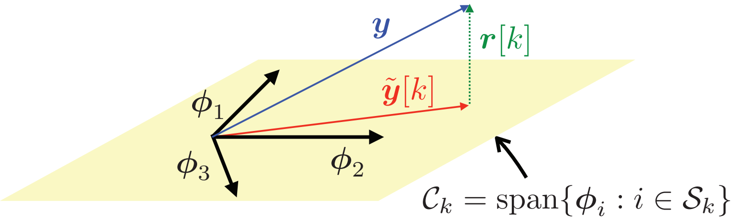
- Store the index in the chosen index set:

$$\mathcal{S}_k = \mathcal{S}_{k-1} \cup \{i[k]\}, \quad \mathcal{S}_0 = \emptyset, \quad k = 1, 2, \dots$$

- Approximate $\boldsymbol{y}$ by a vector $\tilde{\boldsymbol{y}}[k]$ in $C_k = \mathrm{span}\{\boldsymbol{\phi}_i : i \in \mathcal{S}_k\}$, that is,

$$\tilde{\boldsymbol{y}}[k] = \underset{\boldsymbol{v} \in C_k}{\arg\min} \frac{1}{2} \|\boldsymbol{v} - \boldsymbol{y}\|_2^2 = \Pi_{C_k}(\boldsymbol{y}) : \quad \text{projection onto } C_k$$

Orthogonal Matching Pursuit (OMP)



- The coefficient vector $\tilde{\boldsymbol{x}}[k]$ that satisfies

$$\tilde{\boldsymbol{y}}[k] = \sum_{i \in \mathcal{S}_k} \tilde{x}_i[k] \boldsymbol{\phi}_i = \Phi_{\mathcal{S}_k} \tilde{\boldsymbol{x}}[k]$$

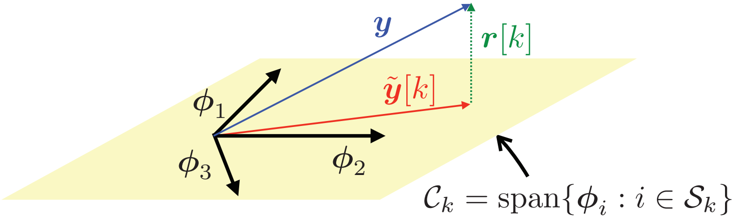
  is given by

$$\tilde{\boldsymbol{x}}[k] = \left(\Phi_{\mathcal{S}_k}^\top \Phi_{\mathcal{S}_k}\right)^{-1} \Phi_{\mathcal{S}_k}^\top \boldsymbol{y}.$$

- Finally, the k-sparse vector is computed by

$$\big(\boldsymbol{x}[k]\big)_{\mathcal{S}_k} = \tilde{\boldsymbol{x}}[k], \quad \big(\boldsymbol{x}[k]\big)_{\mathcal{S}_k^c} = \boldsymbol{0},$$

Orthogonal Matching Pursuit (OMP)



- The residue vector $\boldsymbol{r}[k]$ is orthogonal to the linear subspace $\mathcal{C}_k$.
- From this,

$$\langle \boldsymbol{v}, \boldsymbol{r}[k] \rangle = 0, \quad \forall \boldsymbol{v} \in \mathcal{C}_k.$$

- Any vector $\boldsymbol{\phi}_i$ in $\mathcal{C}_k$ will never be chosen at the next step:

$$i[k+1] = \underset{i \in \{1,2,\dots,n\}}{\arg\max} \frac{\langle \boldsymbol{\phi}_i, \boldsymbol{r}[k] \rangle^2}{\|\boldsymbol{\phi}_i\|_2^2} = \underset{\substack{i \in \{1,2,\dots,n\} \\ \boldsymbol{\phi}_i \notin \mathcal{C}_k}}{\arg\max} \frac{\langle \boldsymbol{\phi}_i, \boldsymbol{r}[k] \rangle^2}{\|\boldsymbol{\phi}_i\|_2^2}$$

- This is why the algorithm is called the orthogonal MP (OMP).

OMP: convergence

Theorem

Assume that $\mathrm{rank}(\Phi) = m$*. Assume also that there exists a vector* $\boldsymbol{x} \in \mathbb{R}^n$ *such that* $\boldsymbol{y} = \Phi \boldsymbol{x}$ *and*

$$\|\boldsymbol{x}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\Phi)}\right).$$

Then, this vector $\boldsymbol{x}$ *is the unique solution of the* ℓ^0 *optimization, and OMP gives it in* $k = \|\boldsymbol{x}\|_0$ *steps.*

- At each step of OMP, we need to compute the matrix inversion of

$$(\Phi_{\mathcal{S}_k}^\top \Phi_{\mathcal{S}_k})^{-1} \Phi_{\mathcal{S}_k}^\top \boldsymbol{y}$$

- If the number $k = \|\boldsymbol{x}\|_0$ is very large, then this inversion may impose a heavy computational burden.

Table of Contents

Optimization problems

In this section, we consider the following two optimization problem:

ℓ^0 regularization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|\Phi \boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{x}\|_0$$

s-sparse approximation

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|\Phi \boldsymbol{x} - \boldsymbol{y}\|_2^2 \text{ subject to } \|\boldsymbol{x}\|_0 \leq s$$

We introduce greedy algorithms called thresholding algorithms for these ℓ^0 optimization problems.

ℓ^0 regularization and proximal gradient algorithm

ℓ^0 regularization

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{x}\|_0$$

- Consider the optimization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}}\, f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}),$$

  - f_1: differentiable and convex, $\mathrm{dom}(f_1) = \mathbb{R}^n$
  - f_2: proper, closed, and convex
- The proximal gradient algorithm

$$\boldsymbol{x}[k+1] = \mathrm{prox}_{\gamma f_2}\bigl(\boldsymbol{x}[k] - \gamma\nabla f_1(\boldsymbol{x}[k])\bigr).$$

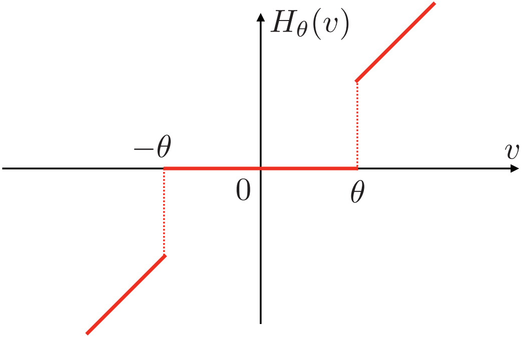
- For the ℓ^0 regularization,

$$f_1(\boldsymbol{x}) \triangleq \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2, \quad f_2(\boldsymbol{x}) \triangleq \lambda\|\boldsymbol{x}\|_0.$$

ℓ^0 regularization and proximal gradient algorithm

- The function $f_2(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_0$ is not convex.
- The proximal operator of $\lambda\|\boldsymbol{x}\|_0$ is given by the hard-thresholding operator $H_\theta(\boldsymbol{v})$ with $\theta = \sqrt{2\gamma\lambda}$, where

$$[H_\theta(\boldsymbol{v})]_i \triangleq \begin{cases} v_i, & |v_i| \geq \theta, \\ 0, & |v_i| < \theta, \end{cases} \quad i = 1, 2, \ldots, n,$$

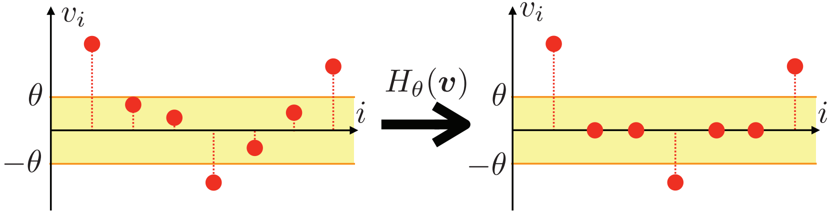


Hard-thresholding operator

Proximal operator of $f_2(\boldsymbol{x}) = \lambda \|\boldsymbol{x}\|_0$

$$\mathrm{prox}_{\gamma\lambda\|\cdot\|_0}(\boldsymbol{v}) = H_{\sqrt{2\gamma\lambda}}(\boldsymbol{v}).$$

- Hard-thresholding operator $H_\theta(\boldsymbol{v})$ rounds small elements ($|v_i| < \theta$) to 0, where $\theta = \sqrt{2\gamma\lambda}$.



Iterative Hard-Thresholding (IHT) Algorithm

Iterative Hard-Thresholding (IHT)

Initialization: Give an initial vector $\boldsymbol{x}[0]$ and positive number $\gamma > 0$.
Iteration: for $k = 0, 1, 2, \ldots$ do

$$\boldsymbol{x}[k+1] = H_{\sqrt{2\gamma\lambda}}\left(\boldsymbol{x}[k] - \gamma\Phi^\top(\Phi\boldsymbol{x}[k] - \boldsymbol{y})\right).$$

Theorem

Assume that

$$\gamma < \frac{1}{\|\Phi\|^2},$$

holds. Then the sequence $\{\boldsymbol{x}[0], \boldsymbol{x}[1], \boldsymbol{x}[2], \ldots\}$ *generated by IHT converges to a local minimizer of the* ℓ^0 *regularization. Moreover, the convergence is first order:*

$$\|\boldsymbol{x}[k+1] - \boldsymbol{x}^*\|_2 \leq c\|\boldsymbol{x}[k] - \boldsymbol{x}^*\|_2, \quad k = 0, 1, 2, \ldots.$$

s-sparse approximation

s-sparse approximation

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|\Phi x - y\|_2^2 \quad \text{subject to} \quad \|x\|_0 \leq s$$

- The set of s-sparse vectors in $\mathbb{R}^n$:

$$\Sigma_s \triangleq \{x \in \mathbb{R}^n : \|x\|_0 \leq s\}.$$

- This is non-convex.
- The indicator function:

$$I_{\Sigma_s}(x) = \begin{cases} 0, & \|x\|_0 \leq s, \\ \infty, & \|x\|_0 > s. \end{cases}$$

- The problem is equivalently described by

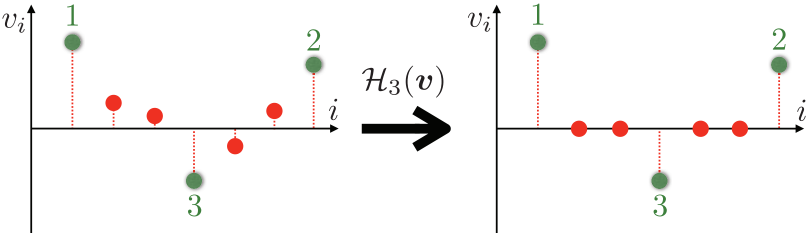
$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|\Phi x - y\|_2^2 + I_{\Sigma_s}(x).$$

s-sparse operator

- We apply the proximal gradient algorithm for

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|\Phi\boldsymbol{x} - \boldsymbol{y}\|_2^2 + I_{\Sigma_s}(\boldsymbol{x}).$$

- The proximal operator of $I_{\Sigma_s}(\boldsymbol{x})$ is the projection onto Σ_s.
- The projection is given by s-sparse operator $\mathcal{H}_s(\boldsymbol{v})$ that sets all but the s largest (in magnitude) elements of $\boldsymbol{v}$ to 0.
- Note that this is not unique.



Iterative s-sparse algorithm

Iterative s-sparse algorithm

Initialization: Give an initial vector $\boldsymbol{x}[0]$ and a positive number $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$\boldsymbol{x}[k+1] = \mathcal{H}_s\left(\boldsymbol{x}[k] - \gamma\Phi^\top(\Phi\boldsymbol{x}[k] - \boldsymbol{y})\right).$$

Theorem

Assume that $\operatorname{rank}(\Phi) = m$*, and the column vectors* $\boldsymbol{\phi}_i$*,* $i = 1, 2, \ldots, n$*, are non-zero. Assume also that the constant* $\gamma > 0$ *satisfies*

$$\gamma < \frac{1}{\|\Phi\|^2}.$$

Then the sequence $\{\boldsymbol{x}[0], \boldsymbol{x}[1], \boldsymbol{x}[2], \ldots\}$ *generated by the s-sparse algorithm converges to a local minimizer of the s-sparse approximation. Moreover, the convergence is first order.*

Compressed Sampling Matching Pursuit (CoSaMP)

s-sparse approximation

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2 \quad \text{subject to} \quad \|\boldsymbol{x}\|_0 \leq s$$

- The OMP can be extended to solve the s-sparse approximation.
- This is called the Compressed Sampling Matching Pursuit (CoSaMP)

CoSaMP algorithm for s-sparse approximation

CoSaMP algorithm for s-sparse approximation

Initialization: $\boldsymbol{x}[0] = \boldsymbol{0}$, $\boldsymbol{r}[0] = \boldsymbol{y}$, $\mathcal{S}_0 = \emptyset$

Iteration: for $k = 1, 2, \dots$ do

$$
\begin{aligned}
\mathcal{I}[k] &:= \operatorname{supp}\left\{\mathcal{H}_{2s}\left(\left\langle \frac{\boldsymbol{\phi}_i}{\|\boldsymbol{\phi}_i\|_2}, \boldsymbol{r}[k-1]\right\rangle^2\right)\right\},\\
\mathcal{S}_k &:= \mathcal{S}_{k-1} \cup \mathcal{I}[k],\\
\tilde{\boldsymbol{x}}[k] &:= \left(\Phi_{\mathcal{S}_k}^\top \Phi_{\mathcal{S}_k}\right)^{-1} \Phi_{\mathcal{S}_k}^\top \boldsymbol{y},\\
(\boldsymbol{z}[k])_{\mathcal{S}_k} &:= \tilde{\boldsymbol{x}}[k],\\
(\boldsymbol{z}[k])_{\mathcal{S}_k^c} &:= \boldsymbol{0},\\
\boldsymbol{x}[k] &:= \mathcal{H}_s\left(\boldsymbol{z}[k]\right),\\
\mathcal{S}_k &:= \operatorname{supp}\{\boldsymbol{x}[k]\},\\
\boldsymbol{r}[k] &:= \boldsymbol{y} - \Phi_{\mathcal{S}_k}\tilde{\boldsymbol{x}}[k].
\end{aligned}
$$

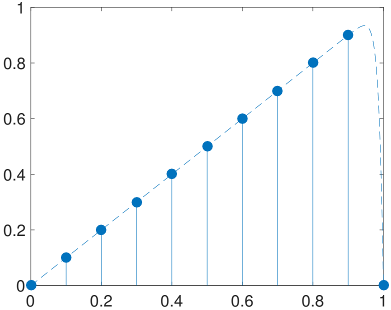
Table of Contents

Sparse polynomial curve fitting

- 80th-order sparse polynomial: $y = -t^{80} + t$.
- Generate data from this polynomial

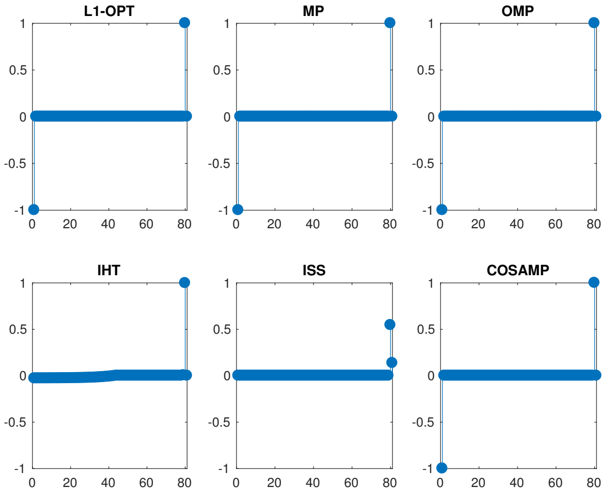
$$\mathcal{D} = \{(t_1, y_1), (t_2, y_2), \ldots, (t_{11}, y_{11})\}, \quad y_i = -t_i^{80} + t_i.$$

on $t_1 = 0, t_2 = 0.1, t_3 = 0.2, \ldots, t_{11} = 1$.



Results

$$c = (-1, \underbrace{0, \ldots, 0}_{78}, 1, 0)$$



Results

Methods	ℓ^1 OPT	MP	OMP	IHT	ISS	CoSaMP
Error	2.7×10^{-10}	9.1×10^{-6}	4.1×10^{-16}	0.0017	0.83	4.1×10^{-11}
Iterations	10	18	2	10^5	10^5	3

- The error $\boldsymbol{r} = \boldsymbol{y} - \Phi \boldsymbol{x}^*$ is smallest with OMP.
- The number of iteration is smallest with OMP.
- IHT and ISS converged to local minimizers with large residues.
- OMP is best for this example, but this is not always true.
- The performance depends on the problem and data, and we should adopt trial and error to seek the best algorithm.

Conclusion

- Greedy algorithms are available to directly solve ℓ^0 optimization.
- The greedy algorithms introduced in this chapter show the linear convergence, which are much faster than the proximal splitting algorithms.
- A local optimal solution is obtained by a greedy algorithm, which is not necessarily a global optimizer.