Recover Sparse Signals from Under-Sampled Observations

Wei Dai, EEE Department, Imperial College London

Aim and outline

Sparse signal processing has become a major component in modern signal processing theory, underpinning compressed sensing, linear regression, machine learning, big data processing, etc. It is based on the observation that most signals, under certain transform or dictionary, only contain a few significant components. Based on this sparsity, efficient ways for data acquisition, processing, and analysis can be developed.

The main aim of this experiment is to familar students with the concept of sparse signals and the easy-to-understand techniques for sparse signal recovery.

In the first section, we highlight the limitations of least squares methods when applied to sparse signal processing, and suggest the paradigm shift from least squares approach to modern sparse recovery techniques.

In the second section, three greedy algorithms are introduced to solve sparse recovery problem. Matlab implementations and numerical comparison of these three algorithms are required.

References

1. Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal Matching Pursuit : recursive function approximation with application to wavelet decomposition", in Asilomar Conf. on Signals, Systems and Comput., 1993.

2. W. Dai and O. Milenkovic, "Subspace Pursuit for Compressive Sensing Signal Reconstruction", IEEE Trans. Inf. Theory, 2009, 55, 2230-2249.

3. T. Blumensath and M. Davies, "Iterative hard thresholding for compressed sensing", Appl. Comput. Harmon. Anal., 2009, 27, 265 - 274.

1 Background

Consider a linear system

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}, \tag{1.1}$$

where $\boldsymbol{y} \in \mathbb{R}^m$ denotes the observation vector, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a given linear transform, and $\boldsymbol{x} \in \mathbb{R}^n$ stands for an unknown data vector. The task is to find $\boldsymbol{x}$ based on the observation vector $\boldsymbol{y}$. It is clear that when $m < n$, i.e., the number of equations is less than the number of unknown variables, the solution of the linear system (1.1) is not unique.

Remark. Henceforth, we assume that each column of $\boldsymbol{A}$ has unit ℓ_2-norm. Let $\boldsymbol{A}_{:,i}$ be the i-th column of $\boldsymbol{A}$. We assume that $\|\boldsymbol{A}_{:,i}\|_2 := \left(\sum_{j=1}^{m} A_{j,i}^2\right)^2 = 1$.

Exercise 1. Let $m = 128$ and $n = 256$. Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ be a random Gaussian matrix. (Generate $\boldsymbol{A}$ using the 'randn' function in Matlab. Pay attention to column normalization.) Generate the observation $\boldsymbol{y}$ by using $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ where $\boldsymbol{x} \in \mathbb{R}^n$ is a Gaussian random vector. In matlab, compute $\hat{\boldsymbol{x}}_1 = \text{pinv}(\boldsymbol{A}) \cdot \boldsymbol{y}$ and $\hat{\boldsymbol{x}}_2 = \boldsymbol{A} \backslash \boldsymbol{y}$. Compare the vectors $\boldsymbol{x}$, $\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$. Typically these three vectors are distinct though all of them are valid solutions.

Let us introduce some definitions.

Definition (Sparse Vectors)**.** A vector $\boldsymbol{x} \in \mathbb{R}^n$ is said to be S-*sparse* if there are S many nonzero entries in $\boldsymbol{x}$. Define the *support set* of $\boldsymbol{x}$ by $\mathcal{S} = \{i : x_i \neq 0\}$. Then the sparsity of $\boldsymbol{x}$ is given by the cardinality (size) of the support set $S = |\mathcal{S}|$. A signal $\boldsymbol{x}$ is called sparse if $S \ll n$.

It has been observed that most natural signals are approximately sparse, that is, they can be approximated by S-sparse signals where S is much less than the signal dimension. For example, an audio signal is typically sparse in the time-frequency domain, and an image is typically sparse under wavelet or discrete cosine transform (DCT) transform, or in other words, most of the wavelet or DCT coefficients of a given image are close to zero.

As an under-deterministic linear system ($m < n$) admits infinite many solutions, we are trying to find the sparsest one among all possible solutions.

2 Greedy Algorithms

Here, we introduce the several greedy algorithms to find a sparse solution of an under-deterministic linear system.

To proceed, we need to introduce several definitions.

Definition (Truncation)**.** Let $\mathcal{S} \subset \{1, 2, \cdots, n\}$ be an index set whose cardinality is S. Let $\mathcal{S}^c := \{1, 2, \cdots, n\} \backslash \mathcal{S}$ be its complement. For any given vector $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x}_{\mathcal{S}} \in \mathbb{R}^S$ denotes the subvector of $\boldsymbol{x}$ indexed by $\mathcal{S}$ and $\boldsymbol{x}_{\mathcal{S}^c} \in \mathbb{R}^{n-S}$ is the subvector of $\boldsymbol{x}$ indexed by $\mathcal{S}^c$. Similarly, for any given matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{A}_{\mathcal{S}} \in \mathbb{R}^{m \times S}$ gives the submatrix of $\boldsymbol{A}$ formed by the columns indexed by $\mathcal{S}$ and $\boldsymbol{A}_{\mathcal{S}^c} \in \mathbb{R}^{m \times (n-S)}$ represents the submatrix of $\boldsymbol{A}$ consisting the columns indexed by $\mathcal{S}^c$.

Example. Let $n = 5$, $\mathcal{S} = \{1, 3\}$, and $\boldsymbol{x} = [9, 8, 7, 6, 5]^T$. Then $\boldsymbol{x}_{\mathcal{S}} = [9, 7]^T$ and $\boldsymbol{x}_{\mathcal{S}^c} = [8, 6, 5]^T$.

Definition (Projection and Residue)**.** Let $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ where $d < m$. Suppose that $\boldsymbol{A}^T \boldsymbol{A} \in \mathbb{R}^{d \times d}$ is invertible. The *projection* of $\boldsymbol{y}$ onto the subspace spanned by $\boldsymbol{A}$ is defined as

$$\boldsymbol{y}_p = \mathrm{proj}\,(\boldsymbol{y}, \boldsymbol{A}) := \boldsymbol{A}\boldsymbol{A}^\dagger \boldsymbol{y}$$

where $\boldsymbol{A}^\dagger := \left(\boldsymbol{A}^T \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T$ denotes the pseudo-inverse of the matrix $\boldsymbol{A}$. The projection residue is given by

$$\boldsymbol{y}_r = \mathrm{resid}\,(\boldsymbol{y}, \boldsymbol{A}) := \boldsymbol{y} - \boldsymbol{y}_p.$$

Remark. In order to compute $\boldsymbol{A}^\dagger \boldsymbol{y}$, it is recommended in Matlab to use $\boldsymbol{A} \backslash \boldsymbol{y}$ rather than $\mathrm{pinv}\,(\boldsymbol{A}) \cdot \boldsymbol{y}$.

Definition (Hard Thresholding Function)**.** For given vector $\boldsymbol{x} \in \mathbb{R}^n$ and positive integer $S < n$, the hard thresholding function $H_S\,(\boldsymbol{x})$ set all but the largest S entries (in magnitude) of $\boldsymbol{x}$ to zero.

Example. Let $\boldsymbol{x} = [3, -4, 1]^T$. Then $H_1\,(\boldsymbol{x}) = [0, -4, 0]^T$ and its support set $\mathrm{supp}\,(H_1\,(\boldsymbol{x})) = \{2\}$; and $H_2\,(\boldsymbol{x}) = [3, -4, 0]^T$ and $\mathrm{supp}\,(H_2\,(\boldsymbol{x})) = \{1, 2\}$.

With the above definitions, we describe the orthogonal matching pursuit (OMP), subspace pursuit (SP), and iterative hardthresholding (IHT) algorithms below. There are many possible exit criteria that can be used to terminate the loops in the algorithms. The commonly used criteria include 1) the normalised error $\|\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}\|_2 / \|\boldsymbol{y}\|_2$ is less than the preset tolerance, and/or 2) the algorithm starts to diverge, i.e., the error in the current iteration is larger than the error in the previous iteration.

Algorithm 1 The Orthogonal Matching Pursuit (OMP) Algorithm

```
Input: $S$, $\boldsymbol{A}$, $\boldsymbol{y}$.
Output: $\hat{\boldsymbol{x}}$.
Initialization:
$\boldsymbol{x} = \boldsymbol{0}$, $\mathcal{S} = \phi$, and $\boldsymbol{y}_r = \boldsymbol{y}$.
Iteration: $\ell = 1, 2, \cdots, S$ or the exit criteria are true
    1. $\mathcal{S} = \mathcal{S} \bigcup \mathrm{supp}\left(H_1\left(\boldsymbol{A}^T \boldsymbol{y}_r\right)\right)$.      (Add one index)
    2. $\hat{\boldsymbol{x}}_{\mathcal{S}} = \boldsymbol{A}_{\mathcal{S}}^\dagger \boldsymbol{y}$ and $\hat{\boldsymbol{x}}_{\mathcal{S}^c} = \boldsymbol{0}$.      (Estimate $\ell$-sparse signal)
    3. $\boldsymbol{y}_r = \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}$.      (Compute estimation error)
```

Algorithm 2 The Subspace Pursuit (SP) Algorithm

```
Input: $S$, $\boldsymbol{A}$, $\boldsymbol{y}$.
Output: $\hat{\boldsymbol{x}}$.
Initialization:
    1. $\mathcal{S} = \operatorname{supp}\left(H_S\left(\boldsymbol{A}^T\boldsymbol{y}\right)\right)$.
    2. $\boldsymbol{y}_r = \operatorname{resid}\left(\boldsymbol{y}, \boldsymbol{A}_{\mathcal{S}}\right)$.
Iteration: $\ell = 1, 2, \cdots$ until the exit criteria are true.
    1. $\tilde{\mathcal{S}} = \mathcal{S} \bigcup \operatorname{supp}\left(H_S\left(\boldsymbol{A}^T\boldsymbol{y}_r\right)\right)$.    (Expand support)
    2. Let $\boldsymbol{b}_{\tilde{\mathcal{S}}} = \boldsymbol{A}_{\tilde{\mathcal{S}}}^{\dagger}\boldsymbol{y}$ and $\boldsymbol{b}_{\tilde{\mathcal{S}}^c} = \boldsymbol{0}$.    (Estimate $2S$-sparse signal)
    3. Set $\mathcal{S} = \operatorname{supp}\left(H_S\left(\boldsymbol{b}\right)\right)$.    (Shrink support )
    4. Let $\hat{\boldsymbol{x}} = \boldsymbol{A}_{\mathcal{S}}^{\dagger}\boldsymbol{y}$ and $\hat{\boldsymbol{x}}_{\mathcal{S}^c} = \boldsymbol{0}$.    (Estimate $S$-sparse signal)
    5. Let $\boldsymbol{y}_r = \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}$.    (Compute estimation error)
```

Algorithm 3 The Iterative Hardthresholding (IHT) Algorithm

```
Input: $S$, $\boldsymbol{A}$, $\boldsymbol{y}$.
Output: $\hat{\boldsymbol{x}}$.
Initialization:
$\hat{\boldsymbol{x}} = \boldsymbol{0}$.
Iteration: $\ell = 1, 2, \cdots$ until exit criterion true.
    $\hat{\boldsymbol{x}} = H_S\left(\hat{\boldsymbol{x}} + \boldsymbol{A}^T\left(\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}\right)\right)$.
```

Exercise 2 (Test greedy algorithms)**.** Let $m = 128$, $n = 256$ and $S = 12$. Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ be a random Gaussian matrix. Generate an S-sparse vector $\boldsymbol{x} \in \mathbb{R}^n$ of which the support set is randomly generated (use Matlab function 'randsample') and the nonzero entries are $\mathcal{N}(0,1)$ distributed. Generate $\boldsymbol{y}$ using $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. Let $\hat{\boldsymbol{x}}_1 = \operatorname{pinv}(\boldsymbol{A}) \cdot \boldsymbol{y}$, $\hat{\boldsymbol{x}}_2 = \boldsymbol{A}\backslash\boldsymbol{y}$, and $\hat{\boldsymbol{x}}_{\mathrm{OMP}}$, $\hat{\boldsymbol{x}}_{\mathrm{SP}}$, $\hat{\boldsymbol{x}}_{\mathrm{IHT}}$ be the outputs of OMP, SP, and IHT respectively. Compare these estimates with the ground truth signal $\boldsymbol{x}$.

Exercise 3 (Success Rate Comparison)**.** In this exercise, we compare the three greedy algorithms using the success rate of recovering the ground truth signal ($\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2 / \|\boldsymbol{x}\| < 10^{-6}$ will be regarded as a success). Let $m = 128$ and $n = 256$. We vary S from 3 to 63 with step size 3. We use the same models in the previous exercise for matrix $\boldsymbol{A}$ and ground truth sparse signal $\boldsymbol{x}$. For each value of S, we run 500 independent tests (both $\boldsymbol{A}$ and $\boldsymbol{x}$ will be generated independently in each test) and compute the corresponding success rate. We compare the performance of the three greedy algorithms in a single figure where the horizontal axis denotes S and the vertical axis denotes the success rate.