Compressive Sensing: Reconstruction Algorithms

CS 763, Ajit Rajwade

Compressive sensing: reconstruction problem

- The basic problem is:

$$
\begin{aligned}
&(P0): \min \|\boldsymbol{\theta}\|_0 \\
&\text{such that } \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 \le \varepsilon \\
&\mathbf{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}
\end{aligned}
\quad \Longrightarrow \quad
\begin{aligned}
&(P1): \min \|\boldsymbol{\theta}\|_1 \\
&\text{such that } \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 \le \varepsilon
\end{aligned}
$$

- Here **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 **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:** Seek to directly solve problem P1. There are many algorithms in this category, in particular a popular one called Iterative Shrinkage/Thresholding Algorithm (ISTA).

- **Category 2:** Problem P0 is NP-hard. So instead, run an *approximation algorithm* which can be efficiently solved.

Compressive Reconstruction Algorithms

- We will focus on some algorithms in category 2.

- Examples: Matching pursuit, Orthogonal matching pursuit, Iterative Hard Thresholding, Co-SAMP.

Matching Pursuit

- One of the simplest approximation algorithms to obtain the coefficients **s** of a signal **y** in an over-complete basis **A** $\in \mathcal{R}^{mxn}$ (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

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

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

$$\{$$

$$j = \arg\max_{l} |\mathbf{r}^{(i)^T} \mathbf{a}_l / \|\mathbf{a}_l\|_2^2 |$$

$$\mathrm{s}_\mathrm{j} = \mathbf{r}^{(i)^T} \mathbf{a}_j; \mathbf{r}^{(i+1)} = \mathbf{r}^{(i)} - \mathrm{s}_\mathrm{j} \mathbf{a}_j; i = i + 1$$

$$\}$$

$$\text{OUTPUT}: \{\mathrm{s}_\mathrm{j}\}$$

Assumptions: Columns (or atoms) of **A** have unit norm.
"j" or "*l*" is an index for dictionary columns

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

Properties of matching pursuit

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

  $\mathbf{y} = \langle \mathbf{y}, \mathbf{a}_k \rangle \mathbf{a}_k + \mathbf{r}^{-k}$ where $\mathbf{a}_k$ is an atom from **A**, and $\mathbf{r}^{-k}$ is a residual vector.
- Note that $\mathbf{r}^{-k}$ and $\mathbf{a}_k$ are orthogonal.

$$(\mathbf{r}^{-\mathbf{k}})^t \mathbf{a}_\mathbf{k} = (\mathbf{y} - (\mathbf{y}^t \mathbf{a}_\mathbf{k})\mathbf{a}_\mathbf{k})^t \mathbf{a}_\mathbf{k} = \mathbf{y}^t \mathbf{a}_\mathbf{k} - (\mathbf{y}^t \mathbf{a}_\mathbf{k})\mathbf{a}_\mathbf{k}{}^t \mathbf{a}_\mathbf{k}$$

$$= \mathbf{y}^t \mathbf{a}_\mathbf{k} - (\mathbf{y}^t \mathbf{a}_\mathbf{k}) = 0, \text{as } \mathbf{a}_\mathbf{k}{}^t \mathbf{a}_\mathbf{k} = 1$$

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

Properties of matching pursuit

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

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

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

$$\{$$

Support set

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

Several coefficients are re-computed in each iteration

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

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

$$(4)\ \mathbf{r}^{(i+1)} = \mathbf{y} - \mathbf{A}_{\mathbf{T^{(i)}}}\mathbf{s}_{\mathbf{T^{(i)}}}; i = i + 1$$

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

$$\}$$

$$\text{OUTPUT: } \mathbf{s}_{\mathbf{T^{(i-1)}}}, \{\mathbf{a}_j\}$$

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

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

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

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

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

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

$$= \mathbf{0}$$

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