Orthogonal Matching Pursuit Algorithm

Andersen Ang

Mathématique et recherche opérationnelle, UMONS, Belgium

manshun.ang@umons.ac.be Homepage: angms.science

First draft: August 14, 2017
Last update: December 31, 2020

Overview

1. Problem Setting: Compressive Sensing
2. The idea behind OMP
3. Orthogonal Matching Pursuit Algorithm

Signal model

- General setting: given $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $n \gg m$ (short-fat matrix, more columns than rows). Find $\mathbf{x} \in \mathbb{R}^m$ such that

$$\mathbf{A}\mathbf{x} = \mathbf{b} + \epsilon.$$

  where $\epsilon \in \mathbb{R}^m$ is modeling error (or interpreted as measurement noise).

- Special case: noiseless ($\epsilon = 0$) and

$$\mathbf{A}\mathbf{x} = \mathbf{b}.$$

  Why noiseless case: easier to understand.

- For noisy case, the analysis is more involved (which will be presented in other document).

Signal recovery of sparse signal

- $\mathbf{A}$ more columns than rows means $\mathbf{Ax} = \mathbf{b}$ is under-determined, which has ∞ many sol.
- Statistician George Box: "all models are wrong, some are useful." Here: "All solutions are wrong, but some are useful".
- For example, one want to find $\mathbf{x}$ with only a few non-zero elements (why: easier to interpret). Mathematically $\mathbf{x}$ can be found by solving the following NP-hard problem

$$(\mathcal{P}) \ : \ \min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ subject to } \mathbf{Ax} = \mathbf{b}.$$

  where $\|\mathbf{x}\|_0 =$ number of non-zero element in vector $\mathbf{x}$.
- The key message: if $\mathbf{A}$ fulfills some conditions, such NP-hard problem can be solved by the *Orthogonal Matching Pursuit* algorithm.

Terminologies and definitions

- **Support** The support of a vector $\mathbf{x} \in \mathbb{R}^m$ is a set, denoted by $\mathrm{supp}(\mathbf{x})$, that contains all the indices of non-zero elements in $\mathbf{x}$:

$$\mathrm{supp}(\mathbf{x}) = \{i \ : \ x_i \neq 0\}.$$

- **Sparsity** The sparsity of $\mathbf{x}$ = the cardinality of the set $\mathrm{supp}(\mathbf{x})$. i.e., sparsity of $\mathbf{x}$ = the number of non-zero element in $\mathbf{x}$. Notation: $|\mathrm{supp}(\mathbf{x})|$ or $\|\mathbf{x}\|_0$. A vector is k-sparse if its sparsity is less than or equal to k. Mathematically, $|\mathrm{supp}(\mathbf{x})| \leq k$.

- **Mutual incoherence** For a set of n vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^m$ for all i, the mutual incoherence M is the largest absolute value of normalized correlation between these vectors.

$$M = \max_{i \neq j} \frac{|\langle \mathbf{x}_i, \mathbf{x}_j \rangle|}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}.$$

A theorem

- **Theorem**. Given $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $n \gg m$, If $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \in \mathbb{R}^n$ can be recovered by OMP if $\mathbf{A}$ and $\mathbf{x}$ satisfy following inequality:
$$\mu < \frac{1}{2k-1},$$
where μ= mutual coherence of column vectors of $\mathbf{A}$ and k= sparsity of $\mathbf{x}$.

- That is, assumes we know $\mathbf{x}$ is k-sparse, then as long as the mutual coherence of $\mathbf{A}$ satisfies the inequality, $\mathbf{x}$ can be recovered from the given $(\mathbf{A}, \mathbf{b})$ by the OMP.

- Proof: in other document.
This document : show the OMP algorithm.

How sparse the recoverable $\mathbf{x}$ can be

- Rearranging the inequality $\mu < \frac{1}{2k-1}$ gives

$$k < \frac{1}{2}\Big(\frac{1}{\mu} - 1\Big) = \frac{1}{2\mu} - \frac{1}{2}.$$

- As k is integer, a "better" expression is

$$k \leq \left\lfloor \frac{1}{2\mu} - \frac{1}{2} \right\rfloor$$

- Algebra of floor function $\lfloor a + b \rfloor \leq \lfloor a \rfloor + \lfloor b \rfloor + 1$ gives

$$k \leq \left\lfloor \frac{1}{2\mu} - \frac{1}{2} \right\rfloor \leq \left\lfloor \frac{1}{2\mu} \right\rfloor + \underbrace{\left\lfloor -\frac{1}{2} \right\rfloor}_{-1} + 1 = \left\lfloor \frac{1}{2\mu} \right\rfloor,$$

i.e., recoverable $\mathbf{x}$ can be at most $\left\lfloor \frac{1}{2\mu} \right\rfloor$-sparse.

- This $\frac{1}{2\mu}$-sparse condition on $\mathbf{x}$ links to the uniqueness of solving the problem $(\mathcal{P})$, see page12 here for details.

The idea behind OMP ... (1/2)

- The inequality $k \le \left\lfloor \frac{1}{2\mu} \right\rfloor$ makes sense:
  Imagine if $\mathbf{x}$ has only 1 non-zero element and all the rest are zeros, say the 3rd element is non-zero and has the value 0.47 and all rest are zeros.
  $$\mathbf{x} = [0,\ 0,\ 0.47,\ 0,\ \ldots,\ 0]^\top.$$
- The product $\mathbf{Ax}$ will be the 3rd column of $\mathbf{A}$ multiplied by 0.47. Let $\mathbf{a}_i$ denotes the ith columns of $\mathbf{A}$ and x_i denotes the ith element of $\mathbf{x}$, So for $\mathbf{Ax} = \mathbf{b}$, the vector $\mathbf{b}$ we get will be $x_3\mathbf{a}_3 = 0.47\mathbf{a}_3$.
- Now, suppose we ask somebody to find $\mathbf{x}$ given only $(\mathbf{A}, \mathbf{b})$. How to recover $\mathbf{x}$?

The idea behind OMP ... (1/2)

- The key to find $\mathbf{x}$ is to **utilize the fact that $\mathbf{x}$ is sparse** $\Longrightarrow$ we know $\mathbf{b}$ will be **a sparse linear combination of columns of $\mathbf{A}$**.
- In the example, $\mathbf{b} =$ the 3rd column of $\mathbf{A}$ scaled by 0.47, so $\mathbf{b}$ will have the highest correlation towards the 3rd column of $\mathbf{A}$.
- Thus we can compute the correlations of $\mathbf{b}$ to all columns of $\mathbf{A}$, and see which columns gives the "highest correlation". The column with the highest correlation with $\mathbf{b}$ tells which index of $\mathbf{x}$ is non-zero.
- The above is the idea behind OMP for 1-sparse $\mathbf{x}$.
- In general, $\mathbf{x}$ is k-sparse with $k > 1$, but the same idea applies with one more step: each time when a column in $\mathbf{A}$ is extracted, the effect of the extracted column on vector $\mathbf{b}$ has to be "removed" so that next time the same column will not be extracted again.

Orthogonal Matching Pursuit Algorithm

- OMP is
  - **an iterative algorithm** : it finds $\mathbf{x}$ element-by-element in a step-by-step iterative manner.
  - **a greedy algorithm**: at each stage, the problem is solved optimally.

- Given $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, an optional step is to normalize all the column vectors of $\mathbf{A}$ to unit norm:

$$\mathbf{a}_i \leftarrow \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|_2}.$$

  This normalization make sure the dot product (correlation) between any two columns of $\mathbf{A}$ is within the range $[-1 \ +1]$ and hence the absolute value of it is bounded by 1. i.e.

$$0 \leq \Big|\langle \mathbf{a}_i, \mathbf{a}_j \rangle\Big| \leq 1.$$

  Note : here $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y}$.

OMP algorithm ... initialization phase

- (Optional step) Normalize the columns of $\mathbf{A}$.

- (Optional step) Remove duplicated columns in $\mathbf{A}$.

- Set residue $\mathbf{r}_0 \leftarrow \mathbf{b}$
  $\mathbf{r}_k$ is the key in extracting the "important columns" of $\mathbf{A}$. It is the "remaining portion" of $\mathbf{b}$ that has not been "explained" by $\mathbf{A}\mathbf{x}_k$.

- Set the index set $\Lambda_0 = \varnothing$
  Λ_k stores all the indices of the "important columns" of $\mathbf{A}$.

- Set iteration counter $k \leftarrow 1$
  k keeps track of the number of times the "column extraction" has occurred.

OMP algorithm ... main loop step 1

- Step-1. Important column extraction.

$$\lambda_k = \underset{j \notin \Lambda_{k-1}}{\operatorname{argmax}} \left| \langle \mathbf{a}_j, \mathbf{r}_{k-1} \rangle \right|.$$

  "Important column"= the column in $\mathbf{A}$ that has the largest absolute value of correlation with the residue vector $\mathbf{r}_{k-1}$.

- The constraint $j \notin \Lambda_{k-1}$ is to avoid repeatedly extracting the same column index that has been extracted previously.
- It is possible that $\operatorname{argmax}_j \left| \langle \mathbf{a}_j, \mathbf{r}_{k-1} \rangle \right|$ produces multiple solutions (if $\mathbf{A}$ has duplicated columns). The step in the initialization to remove duplicated columns is thus necessary.
- For implementation: this step can be done as

$$\begin{aligned} \mathbf{h}_k &= \mathbf{A}^\top \mathbf{r}_{k-1}. \\ \lambda_k &= \underset{j \notin \Lambda_k}{\operatorname{argmax}} \left| \mathbf{h}_k \right|. \end{aligned}$$

OMP algorithm ... main loop steps 2

- Step-2. Augment the index set: $\Lambda_k = \Lambda_{k-1} \cup \{\lambda_k\}$ (put the index into the index set).

- At $k = 0$, $\Lambda_k = \varnothing$

- At $k = 1$, Λ_k holds 1 index

- At $k = 2$, Λ_k holds 2 indices

- As Λ_k holds k indices, so in the nth step, Λ_n will hold all the column indices in $\mathbf{A}$. That means we should stop OMP at this point and the $\mathbf{x}$ is just fully-dense (there is no zero element).

- As λ_k is selected based on $j \notin \Lambda_{k-1}$, hence $\Lambda_{k-1} \cup \{\lambda_k\}$ will not have duplicated indices.

OMP algorithm ... main loop step 3

- Step-3. Obtain signal estimate $\mathbf{x}_k$. This can be done by solving a regression

$$\mathbf{x}_k(i \in \Lambda_k) = \underset{\mathbf{x}}{\operatorname{argmin}} \, \|\mathbf{A}_{\Lambda_k}\mathbf{x} - \mathbf{b}\|_2, \quad \mathbf{x}_k(i \notin \Lambda_k) = 0,$$

  where $\mathbf{A}_{\Lambda_k}$ is a sub-matrix of $\mathbf{A}$ with columns indicated by Λ_k. The analytical solution of this problem is

$$\mathbf{x}_k(\Lambda_k) = \mathbf{A}_{\Lambda_k}^{\dagger}\mathbf{b},$$

  where $\dagger$ denotes pseudo-inverse.

- What this means: use the $\underbrace{\text{columns in } \mathbf{A}_{\Lambda_k}}_{\text{selected columns in } \mathbf{A}}$ to regress the vector $\mathbf{b}$.

  As we only use some columns of $\mathbf{A}$ to regress $\mathbf{b}$, for those unused columns in $\mathbf{A}$, they contribute nothing in such regression, and hence those corresponding x_i should be zero.

OMP algorithm ... main loop steps 4 and 5

- Step-4. Compute $\hat{\mathbf{b}}_k = \mathbf{A}\mathbf{x}_k$.
  $\hat{\mathbf{b}}_k$ is the approximation of $\mathbf{b}$ using the column $\mathbf{A}$ with the coefficients $\mathbf{x}_k$ at iteration k. In other words, $\hat{\mathbf{b}}_k$ is the portion of $\mathbf{b}$ being "explained" by $\mathbf{A}\mathbf{x}_k$.

- If we use the notation $\mathbf{A}_{\Lambda_k}$ to form $\hat{\mathbf{b}}$, the expression is then $\hat{\mathbf{b}} = \mathbf{A}_{\Lambda_k}\mathbf{x}_k(i \in \Lambda_k)$. Note that it is important to limit the vector $\mathbf{x}_k$ for those $i \in \Lambda_k$, otherwise the dimensions of the matrix and vector do not match.

- Step-5. Update residue $\mathbf{r}_{k+1} \leftarrow \mathbf{b} - \hat{\mathbf{b}}_k$.
  It means removing the "explained portion of $\mathbf{b}$ at iteration k" from $\mathbf{b}$, and take this "unexplained portion" of $\mathbf{b}$ as the residue.

- Step-4 and Step 5 can be combine into one single step

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$$

The OMP algorithm

Algorithm 1: OMP$(\mathbf{A}, \mathbf{b})$

Input: $\mathbf{A}, \mathbf{b}$

Result: $\mathbf{x}_k$

Initialization $\mathbf{r}_0 = \mathbf{b}$, $\Lambda_0 = \varnothing$;

- Normalize all columns of $\mathbf{A}$ to unit L_2 norm;

- Remove duplicated columns in $\mathbf{A}$ (make $\mathbf{A}$ full rank);

for $k = 1, 2, \ldots$ **do**

Step-1. $\lambda_k = \underset{j \notin \Lambda_{k-1}}{\operatorname{argmax}} \left| \langle \mathbf{a}_j, \mathbf{r}_{k-1} \rangle \right|$;

Step-2. $\Lambda_k = \Lambda_{k-1} \cup \{\lambda_k\}$;

Step-3. $\mathbf{x}_k(i \in \Lambda_k) = \underset{\mathbf{x}}{\operatorname{argmin}} \, \|\mathbf{A}_{\Lambda_k} \mathbf{x} - \mathbf{b}\|_2, \quad \mathbf{x}_k(i \notin \Lambda_k) = 0$;

Step-4. $\hat{\mathbf{b}}_k = \mathbf{A}\mathbf{x}_k$;

Step-5. $\mathbf{r}_k \leftarrow \mathbf{b} - \hat{\mathbf{b}}_k$;

end

Compact OMP algorithm

Algorithm 2: OMP($\mathbf{A}, \mathbf{b}$)

Input: $\mathbf{A}, \mathbf{b}$

Result: $\mathbf{x}_k$

Initialization $\mathbf{r}_0 = \mathbf{b}$, $\Lambda_0 = \varnothing$;

- Normalize all columns of $\mathbf{A}$ to unit L_2 norm;
- Remove duplicated columns in $\mathbf{A}$ (make $\mathbf{A}$ full rank);

for $k = 1, 2, \ldots$ **do**

Step-1-2. $\Lambda_k = \Lambda_{k-1} \cup \left\{ \underset{j \notin \Lambda_{k-1}}{\operatorname{argmax}} \left| \langle \mathbf{a}_j, \mathbf{r}_{k-1} \rangle \right| \right\}$;

Step-3. $\mathbf{x}_k(i \in \Lambda_k) = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{A}_{\Lambda_k}\mathbf{x} - \mathbf{b}\|_2, \quad \mathbf{x}_k(i \notin \Lambda_k) = 0$;

Step-4-5. $\mathbf{r}_k \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_k$;

end

End of document