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

  $$Ax = b + \epsilon.$$ 

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

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

  $$Ax = b.$$ 

  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

▶ A more columns than rows means \( Ax = b \) is under-determined, which has \( \infty \) 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 \( x \) with only a few non-zero elements (why: easier to interpret). Mathematically \( x \) can be found by solving the following NP-hard problem

\[
(\mathcal{P}) : \min_x \| x \|_0 \text{ subject to } Ax = b.
\]

where \( \| x \|_0 = \text{number of non-zero element in vector } x \).

▶ The key message: if \( A \) 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 \( x \in \mathbb{R}^m \) is a set, denoted by \( \text{supp}(x) \), that contains all the indices of non-zero elements in \( x \):

  \[
  \text{supp}(x) = \{ i : x_i \neq 0 \}.
  \]

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

- **Mutual incoherence**  For a set of \( n \) vectors \( \{x_1, x_2, \ldots, x_n\} \), where \( x_i \in \mathbb{R}^m \) for all \( i \), the mutual incoherence \( M \) is the largest absolute value of normalized correlation between these vectors.

  \[
  M = \max_{i \neq j} \frac{|\langle x_i, x_j \rangle|}{\|x_i\|_2 \|x_j\|_2}.
  \]
Theorem. Given $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ with $n \gg m$, if $Ax = b$, $x \in \mathbb{R}^n$ can be recovered by OMP if $A$ and $x$ satisfy following inequality:

$$\mu < \frac{1}{2k - 1},$$

where $\mu =$ mutual coherence of column vectors of $A$ and $k =$ sparsity of $x$.

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

Proof: in other document.
This document : show the OMP algorithm.
How sparse the recoverable $x$ can be

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

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

- 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

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

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

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

- The inequality $k \leq \left\lfloor \frac{1}{2\mu} \right\rfloor$ makes sense: Imagine if $x$ 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.

$$x = [0, 0, 0.47, 0, \ldots, 0]^\top.$$ 

- The product $Ax$ will be the 3rd column of $A$ multiplied by 0.47. Let $a_i$ denotes the $i$th columns of $A$ and $x_i$ denotes the $i$th element of $x$. So for $Ax = b$, the vector $b$ we get will be $x_3a_3 = 0.47a_3$.

- Now, suppose we ask somebody to find $x$ given only $(A, b)$. How to recover $x$?
The idea behind OMP ... (1/2)

▶ The key to find \( x \) is to **utilize the fact that \( x \) is sparse** \( \Rightarrow \) we know \( b \) will be a **sparse linear combination of columns of \( A \)**.

▶ In the example, \( b = \) the 3rd column of \( A \) scaled by 0.47, so \( b \) will have the highest correlation towards the 3rd column of \( A \).

▶ Thus we can compute the correlations of \( b \) to all columns of \( A \), and see which columns gives the “highest correlation”. The column with the highest correlation with \( b \) tells which index of \( x \) is non-zero.

▶ The above is the idea behind OMP for 1-sparse \( x \).

▶ In general, \( x \) is \( k \)-sparse with \( k > 1 \), but the same idea applies with one more step: each time when a column in \( A \) is extracted, the effect of the extracted column on vector \( b \) 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 $x$ element-by-element in a step-by-step iterative manner.
  - **a greedy algorithm**: at each stage, the problem is solved optimally.

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

  $$ a_i \leftarrow \frac{a_i}{\|a_i\|_2}. $$

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

  $$ 0 \leq |\langle a_i, a_j \rangle| \leq 1. $$

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

10 / 17
OMP algorithm ... initialization phase

- (Optional step) Normalize the columns of $A$.

- (Optional step) Remove duplicated columns in $A$.

- Set residue $r_0 \leftarrow b$
  $r_k$ is the key in extracting the “important columns” of $A$. It is the “remaining portion” of $b$ that has not been “explained” by $Ax_k$.

- Set the index set $\Lambda_0 = \emptyset$
  $\Lambda_k$ stores all the indices of the “important columns” of $A$.

- 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.**

  \[
  \lambda_k = \arg \max_{j \notin \Lambda_k} |\langle a_j, r_{k-1} \rangle|.
  \]

  “Important column” = the column in A that has the largest absolute value of correlation with the residue vector \(r_{k-1}\).

- 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 \(\arg \max_{j} |\langle a_j, r_{k-1} \rangle|\) produces multiple solutions (if \(A\) has duplicated columns). The step in the initialization to remove duplicated columns is thus necessary.

- For implementation: this step can be done as

  \[
  h_k = A^\top r_{k-1},
  \lambda_k = \arg \max_{j \notin \Lambda_k} |h_k|.
  \]
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 $k = 0$, $\Lambda_k = \emptyset$

- At $k = 1$, $\Lambda_k$ holds 1 index

- At $k = 2$, $\Lambda_k$ holds 2 indices

- As $\Lambda_k$ holds $k$ indices, so in the $n$th step, $\Lambda_n$ will hold all the column indices in $A$. That means we should stop OMP at this point and the $x$ is just fully-dense (there is no zero element).

- As $\lambda_k$ 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 $x_k$. This can be done by solving a regression

$$x_k(i \in \Lambda_k) = \arg \min_x \|A_{\Lambda_k} x - b\|_2, \quad x_k(i \notin \Lambda_k) = 0,$$

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

$$x_k(\Lambda_k) = A_{\Lambda_k}^\dagger b,$$

where $\dagger$ denotes pseudo-inverse.

- What this means: use the columns in $A_{\Lambda_k}$ to regress the vector $b$.

As we only use some columns of $A$ to regress $b$, for those unused columns in $A$, 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 $\hat{b}_k = Ax_k$.
  $\hat{b}_k$ is the approximation of $b$ using the column $A$ with the coefficients $x_k$ at iteration $k$. In other words, $\hat{b}_k$ is the portion of $b$ being "explained" by $Ax_k$.

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

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

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

  \[ r_k = b - Ax_k \]
The OMP algorithm

Algorithm 1: OMP(A, b)

Input: A, b
Result: x_k

Initialization r_0 = b, \Lambda_0 = \emptyset;
- Normalize all columns of A to unit L_2 norm;
- Remove duplicated columns in A (make A full rank);

for k = 1, 2, ... do
  Step-1. \lambda_k = \arg\max_{j \notin \Lambda_{k-1}} |\langle a_j, r_{k-1} \rangle|;
  Step-2. \Lambda_k = \Lambda_{k-1} \cup \{\lambda_k\};
  Step-3. x_k(i \in \Lambda_k) = \arg\min_x \|A_{\Lambda_k} x - b\|_2, \ x_k(i \notin \Lambda_k) = 0;
  Step-4. \hat{b}_k = Ax_k;
  Step-5. r_k \leftarrow b - \hat{b}_k;
end
Compact OMP algorithm

Algorithm 2: OMP(A, b)

Input: A, b
Result: x_k

Initialization r_0 = b, \( \Lambda_0 = \emptyset \);
- Normalize all columns of A to unit \( L_2 \) norm;
- Remove duplicated columns in A (make A full rank);

for \( k = 1, 2, \ldots \) do
  \begin{align*}
    \text{Step-1-2. } & \Lambda_k = \Lambda_{k-1} \cup \left\{ \arg\max_{j \notin \Lambda_{k-1}} |\langle a_j, r_{k-1} \rangle| \right\}; \\
    \text{Step-3. } & x_k(i \in \Lambda_k) = \arg\min_x \| A_{\Lambda_k} x - b \|_2, \quad x_k(i \notin \Lambda_k) = 0; \\
    \text{Step-4-5. } & r_k \leftarrow b - A x_k;
  \end{align*}
end

End of document