Orthogonal Matching Pursuit Algorithm
A brief introduction

Andersen Ang

Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada

msxang@uwaterloo.ca, where $x = \lfloor \pi \rfloor$

Homepage: angms.science

First draft: August 14, 2017 Last update: April 5, 2022
Signal model and inverse problem

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

$$Ax = b.$$ 

- This is called an inverse problem: given $(A, b)$, find $x$.

- The forward problem: given $(A, x)$, find $b$, is often easier.

- In machine learning, the observed data is usually modelled with noise as

$$b = Ax^* + \epsilon,$$

where $\epsilon \in \mathbb{R}^m$ denotes error, usually the measurement noise.
Signal recovery of sparse signal

▶ We are interested in the case $\mathbf{A}$ has more columns than rows: $\mathbf{A}\mathbf{x} = \mathbf{b}$ 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”.

▶ A want to find $\mathbf{x}$: find $\mathbf{x}$ with only a few non-zero elements$^1$. To find such $\mathbf{x}$ mathematically, we solve the following NP-hard problem

$$(L_0) : \arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\|\mathbf{x}\|_0$ is the $\ell_0$ pseudo norm of $\mathbf{x}$, which is the number of non-zero element in $\mathbf{x}$.

▶ The key message: if $\mathbf{A}$ fulfills some conditions, such NP-hard problem can be solved by the Orthogonal Matching Pursuit (OMP) algorithm, because the sol. of Problem ($L_0$) will be the same as the solution to a $\ell_1$ norm minimization problem, which OMP can solve it.

$^1$Why: for some applications, sparse $\mathbf{x}$ is easier to interpret.
Terminologies and definitions

- **Support**  For a vector $\mathbf{x} \in \mathbb{R}^m$, the set of all indices of non-zero elements in $\mathbf{x}$ is called the support of $\mathbf{x}$, denoted as $\text{supp}(\mathbf{x})$:

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

- **Sparsity**  The sparsity of $\mathbf{x} = \#$ non-zero element in $\mathbf{x} = \text{the cardinality of supp}(\mathbf{x})$. Notation: $|\text{supp}(\mathbf{x})|$ or $\|\mathbf{x}\|_0$.

- **s-sparse**  A vector is $s$-sparse if $\|\mathbf{x}\|_0 \leq s$.

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

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

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

- **Theorem.** Given $A \in \mathbb{R}^{m \times n}$ with $n \gg m$ and $b \in \mathbb{R}^m$. If $Ax = b$, $x \in \mathbb{R}^n$ can be exactly recovered by OMP if $A$ and $x$ satisfy the following inequality:

$$\mu A < \frac{1}{2s_x - 1},$$

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

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

- **Proof:** Theorem 5.14 in *A Mathematical Introduction to Compressive Sensing* by Simon Foucart and Holger Rauhut.

- This document: show the OMP algorithm.
How sparse the recoverable $x$ can be

- Rearranging the inequality $\mu < \frac{1}{2s-1}$ gives $s < \frac{1}{2} \left( \frac{1}{\mu} - 1 \right) = \frac{1}{2\mu} - \frac{1}{2}$.

- $s$ is integer, hence $s \leq \left\lfloor \frac{1}{2\mu} - \frac{1}{2} \right\rfloor$.

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

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

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 problem $(P)$, see page 12 here.
The idea of OMP

- Imagine the solution $x^*$ has only 1 non-zero element, say the 3rd element is non-zero and has the value 0.47 as $x^* = [0, 0, 0.47, 0, \ldots, 0]^\top$.

- The product $Ax^*$ will be the 3rd column of $A$ multiplied by 0.47. Let $a_i$ denotes the $i$th column of $A$ and $x_i$ denotes the $i$th element of $x$. The vector $b = Ax^*$ we observed will be $x_3^*a_3 = 0.47a_3$.

- Now, suppose we ask somebody to recover $x^*$ given only $(A, b)$. To recover $x^*$, a key is to utilize the fact that $x^*$ is sparse $\implies$ we know $b$ is a sparse linear combination of columns of $A$.

- In the example, $b = 0.47a_3$, so $b$ will have the highest correlation towards the 3rd column of $A$.

- We can compute the correlations of $b$ to all the columns of $A$, and see which column gives the “highest correlation”. That column tells which index of $x^*$ is non-zero. This is the “matching” part in OMP.

- The above is the idea behind OMP for 1-sparse $x$.

  For $s$-sparse $x$ with $s > 1$, 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. This is the “orthogonal” part in OMP.
Orthogonal Matching Pursuit Algorithm

- OMP is
  - a **greedy algorithm** : at each stage, the problem is solved optimally based on current info.

- 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:

$$0 \leq |\langle a_i, a_j \rangle| \leq 1.$$
OMP algorithm ... initialization phase

- (Optional step) Normalize the columns of $\mathbf{A}$ to unit $\ell_2$-norm.

- (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 $\Lambda_0 = \emptyset$
  $\Lambda_k$ 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.

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

“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 \( \arg\max_{j \notin \Lambda_{k-1}} |\langle a_j, r_{k-1} \rangle| \) produces multiple solutions (if \( A \) has duplicated columns). So it is useful to remove duplicated columns in the initialization stage.

▶ 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 at $k = n$ step ($n$ is the dimension of $x$), $\Lambda_n$ will hold all the column indices in $\mathbf{A}$. That means we should stop OMP at this point and $x$ is fully-dense (there is no zero element).

- As we assume $x$ is $s$-sparse, so we should stop at iteration $k = s$. 


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 $\dagger$ is 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 $x_i$ is set to zero.
OMP algorithm ... main loop steps 4 and 5

► Step-4. Compute $\hat{b}_k = A x_k$.
$\hat{b}_k$ 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 $A x_k$.

► If we use the notation $A_{\Lambda_k}$ to form $\hat{b}$, 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. Theoretically $\hat{b}_k = A x_k$ and $\hat{b}_k = A_{\Lambda_k} x_k (i \in \Lambda_k)$ are the same, but for implementation, the later one is more efficient (since we are now working on a vector with fewer entries).

► 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.

► Steps 4 & 5 can be combine into one single step: $r_k = b - A x_k$ or $b - A_{\Lambda_k} x_k (i \in \Lambda_k)$. 

13 / 16
The OMP algorithm

**Algorithm 1: OMP**(A, b)

**Input:** A, b

**Result:** x_k

1 **Initialization** r_0 = b, Λ_0 = ∅;
2 Normalize all columns of A to unit L_2 norm;
3 Remove duplicated columns in A ;
4 for k = 1, 2, ... do
  5     Step-1. \( \lambda_k = \arg\max_{j \notin \Lambda_{k-1}} |\langle a_j, r_{k-1} \rangle|; \)
  6     Step-2. \( \Lambda_k = \Lambda_{k-1} \cup \{ \lambda_k \}; \)
  7     Step-3. \( x_k (i \in \Lambda_k) = \arg\min_x \| A_{\Lambda_k} x - b \|_2, \quad x_k (i \notin \Lambda_k) = 0; \)
  8     Step-4. \( \hat{b}_k = A x_k; \)
  9     Step-5. \( r_k \leftarrow b - \hat{b}_k; \)
50 end
Compact OMP algorithm

Algorithm 2: OMP(A, b)

Input: A, b
Result: x_k

1 Initialization \( r_0 = b, \Lambda_0 = \emptyset \);
2 - Normalize all columns of A to unit \( L_2 \) norm;
3 - Remove duplicated columns in A (make A full rank);
4 for \( k = 1, 2, \ldots \) do
   5 Step-1-2. \( \Lambda_k = \Lambda_{k-1} \cup \{ \arg \max_{j \notin \Lambda_{k-1}} |\langle a_j, r_{k-1} \rangle| \} \);
   6 Step-3. \( x_k (i \in \Lambda_k) = \arg \min_x \| A_{\Lambda_k} x - b \|_2 \), \( x_k (i \notin \Lambda_k) = 0 \);
   7 Step-4-5. \( r_k \leftarrow b - Ax_k \);
Another form of compact OMP algorithm using p.10

Algorithm 3: OMP(A, b)

**Input:** A, b
**Result:** x_k

1 **Initialization** r_0 = b, Λ_0 = ∅;
2 - Normalize all columns of A to unit L_2 norm;
3 - Remove duplicated columns in A (make A full rank);
4 **for** k = 1, 2, ... **do**
5   - Step-1-2. Λ_k = Λ_{k-1} ∪ \left\{ \arg\max_{j \notin Λ_{k-1}} |A^T r_{k-1}| \right\};
6   - Step-3. x_k (i ∈ Λ_k) = \arg\min_x \|A_{Λ_k} x - b\|_2, \ x_k (i \notin Λ_k) = 0;
7   - Step-4-5. r_k ← b - A x_k;
8 **end**