Orthogonal Matching Pursuit Algorithm

A brief introduction

Andersen Ang

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

ms$\mathbf{x}$ang@uwaterloo.ca, where $\mathbf{x} = \lfloor \pi \rfloor$
Homepage: angms.science

First draft: August 14, 2017 Last update: April 5, 2022

Signal model and inverse problem

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

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

- This is called an *inverse problem*: given $(\mathbf{A}, \mathbf{b})$, find $\mathbf{x}$.

- The *forward problem*: given $(\mathbf{A}, \mathbf{x})$, find $\mathbf{b}$, is often easier.

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

$$\mathbf{b} = \mathbf{A}\mathbf{x}_* + \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^m$ 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 ∞ 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

$$(\mathcal{L}_0) \;:\; \operatorname*{argmin}_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b},$$

  where $\|\mathbf{x}\|_0$ is the ℓ_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 $(\mathcal{L}_0)$ will be the same as the solution to a ℓ_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 $\operatorname{supp}(\mathbf{x})$:

$$\operatorname{supp}(\mathbf{x}) = \left\{ i \quad : \ x_i \neq 0 \right\}.$$

- **Sparsity** The sparsity of $\mathbf{x}$ = # non-zero element in $\mathbf{x}$ = the cardinality of $\operatorname{supp}(\mathbf{x})$. Notation: $|\operatorname{supp}(\mathbf{x})|$ 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 $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $n \gg m$ and $\mathbf{b} \in \mathbb{R}^m$. If $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \in \mathbb{R}^n$ can be exactly recovered by OMP if $\mathbf{A}$ and $\mathbf{x}$ satisfy following inequality:

$$\mu_{\mathbf{A}} < \frac{1}{2s_{\mathbf{x}} - 1},$$

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

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

- Rearranging the inequality $\mu < \frac{1}{2s-1}$ gives $s < \dfrac{1}{2}\left(\dfrac{1}{\mu} - 1\right) = \dfrac{1}{2\mu} - \dfrac{1}{2}$.
- s is integer, hence $s \leq \left\lfloor \dfrac{1}{2\mu} - \dfrac{1}{2} \right\rfloor$.
- Algebra of floor function $\lfloor a + b \rfloor \leq \lfloor a \rfloor + \lfloor b \rfloor + 1$ gives

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

  i.e., recoverable $\mathbf{x}$ can be at most $\left\lfloor \dfrac{1}{2\mu} \right\rfloor$-sparse.
- This $\frac{1}{2\mu}$-sparse condition on $\mathbf{x}$ links to the uniqueness of solving problem $(\mathcal{P})$, see page 12 here.

The idea of OMP

- Imagine the solution $\mathbf{x}^*$ has only 1 non-zero element, say the 3rd element is non-zero and has the value 0.47 as $\mathbf{x}^* = [0,\ 0,\ 0.47,\ 0,\ \dots,\ 0]^\top$.
- The product $\mathbf{A}\mathbf{x}^*$ will be the 3rd column of $\mathbf{A}$ multiplied by 0.47. Let $\mathbf{a}_i$ denotes the ith column of $\mathbf{A}$ and x_i denotes the ith element of $\mathbf{x}$. The vector $\mathbf{b} = \mathbf{A}\mathbf{x}^*$ we observed will be $x_3^*\mathbf{a}_3 = 0.47\mathbf{a}_3$.
- Now, suppose we ask somebody to recover $\mathbf{x}^*$ given only $(\mathbf{A}, \mathbf{b})$. To recover $\mathbf{x}^*$, a key is to **utilize the fact that $\mathbf{x}^*$ is sparse** $\implies$ we know $\mathbf{b}$ is **a sparse linear combination of columns of $\mathbf{A}$**.
- In the example, $\mathbf{b} = 0.47\mathbf{a}_3$, so $\mathbf{b}$ will have the highest correlation towards the 3rd column of $\mathbf{A}$.
- We can compute the correlations of $\mathbf{b}$ to all the columns of $\mathbf{A}$, and see which column gives the "highest correlation". That column tells which index of $\mathbf{x}^*$ is non-zero. **This is the "matching" part in OMP**.
- The above is the idea behind OMP for 1-sparse $\mathbf{x}$.
  For s-sparse $\mathbf{x}$ with $s > 1$, 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. **This is the "orthogonal" part in OMP**.

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 based on current info.

- 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:

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

OMP algorithm ... initialization phase

- (Optional step) Normalize the columns of $\mathbf{A}$ to unit ℓ_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 $\Lambda_0 = \varnothing$
  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 $\underset{j \notin \Lambda_{k-1}}{\operatorname{argmax}} \left| \langle \mathbf{a}_j, \mathbf{r}_{k-1} \rangle \right|$ produces multiple solutions (if $\mathbf{A}$ has duplicated columns). So it is useful to remove duplicated columns in the initialization stage.

- Implementation: this step can be done as

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

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

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

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

  where $\dagger$ is 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 is set to 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 $\hat{\mathbf{b}}$, 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.
  Theoretically $\hat{\mathbf{b}}_k = \mathbf{A}\mathbf{x}_k$ and $\hat{\mathbf{b}}$, then $\hat{\mathbf{b}} = \mathbf{A}_{\Lambda_k}\mathbf{x}_k(i \in \Lambda_k)$ 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 $\mathbf{r}_{k+1} \leftarrow \mathbf{b} - \hat{\mathbf{b}}_k$.
  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.

- Steps 4 & 5 can be combine into one single step: $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ or $\mathbf{b} - \mathbf{A}_{\Lambda_k}\mathbf{x}_k(i \in \Lambda_k)$.

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 $\mathbf{A}$;

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

Another form of compact OMP algorithm using p.10

Algorithm 3: 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. $\Lambda_k = \Lambda_{k-1} \cup \left\{ \underset{j \notin \Lambda_{k-1}}{\operatorname{argmax}} \left| \mathbf{A}^\top \mathbf{r}_{k-1} \right| \right\}$;

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