ECE 269: Linear Algebra and Applications
Winter 2024

Mini Project # 1
Due: Sunday, March 3 11:59pm,
via Gradescope

Collaboration Policy: This Mini project **DOES NOT allow collaboration**. You are expected to work individually on the project. If you have any questions about the project, you are only allowed to consult TAs and Prof. Pal. You are not allowed to discuss anything with your friends. Any violation of this policy will be considered a breach of academic integrity and will be duly reported to the Office of Academic Integrity. *

Instructions for Programming Assignment: You can use MATLAB or Python for the programming assignment. You must submit the source code and a report with results and discussions. The source code must be presented in a way that it can be directly executed by the TAs. This can be done either by separately submitting the source code files, or embedding them in a Jupyter notebook. You cannot simply copy paste the source code into a .pdf or word file as part of the report.

Programming Assignment: Finding Sparse Solutions via Orthogonal Matching Pursuit (OMP)

In this mini project, we will implement and study the performance of the Orthogonal Matching Pursuit (OMP) algorithm for recovering sparse signals and images.

Required Reading: Read the paper "Greed is good: algorithmic results for sparse approximation" by Joel Tropp to learn about OMP (available on Canvas under Files>Homework), and relevant lectures and discussion sessions. For this assignment, you will mainly need to understand the algorithm (and not the proofs concerning the theoretical performance guarantees of OMP), although you are certainly encouraged to go over them and discuss with Prof. Pal if you are interested.

Consider the measurement model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}$$

where $\mathbf{y} \in \mathbb{R}^M$ is the (compressed, $M < N$) measurement, $\mathbf{A} \in \mathbb{R}^{M \times N}$ is the measurement matrix, and $\mathbf{n} \in \mathbb{R}^M$ is the additive noise. Here, $\mathbf{x} \in \mathbb{R}^N$ is the unknown signal (to be estimated) with $s \ll N$ non-zero elements. The indices of the non-zero entries of $\mathbf{x}$ (also known as the support of $\mathbf{x}$) is denoted by $\mathcal{S} = \{i | x_i \neq 0\}$, with $|\mathcal{S}| = s$.

1. **Performance Metrics:** Let $\hat{\mathbf{x}}$ be the estimate of $\mathbf{x}$ obtained from OMP. To measure the performance of OMP, we consider the Normalized Error defined as

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2}$$

*For more information on Academic Integrity Policies at UCSD, please visit `http://academicintegrity.ucsd.edu/excel-integrity/define-cheating/index.html`

The average Normalized Error is obtained by averaging the Normalized Error over 2000 Monte Carlo runs.

2. **Experimental setup:**

   (a) Generate $\mathbf{A}$ as a random matrix with independent and identically distributed entries drawn from the standard normal distribution. Normalize the columns of $\mathbf{A}$.

   (b) Generate the sparse vector $\mathbf{x}$ with random support of cardinality s (i.e. s indices are generated uniformly at random from integers 1 to N), and non-zero entries drawn as uniform random variables in the range $[-10, -1] \cup [1, 10]$.

   (c) The entries of noise $\mathbf{n}$ are drawn independently from the normal distribution with standard deviation σ and zero mean.

   (d) For each cardinality $s \in [1, s_{max}]$, the average Normalized Error should be computed by repeating step (a) to step (c) 2000 times and averaging the results over these 2000 Monte Carlo runs.

3. **Noiseless case: ($\mathbf{n = 0}$)**
   Implement OMP (you may stop the OMP iterations once $\|\mathbf{y} - \mathbf{A}\mathbf{x}^{(k)}\|_2$ is close to 0, here $\mathbf{x}^{(k)}$ is the estimation of $\mathbf{x}$ after kth iteration) and evaluate its performance. Calculate the probability of Exact Support Recovery (i.e. the fraction of runs when $\hat{\mathcal{S}} = \mathcal{S}$) by averaging over 2000 random realizations of $\mathbf{A}$, as a function of M and s_{max} (for different fixed values of N). For each N, the probability of exact support recovery is a two dimensional plot (function of M and s_{max}) and you can display it as an image. The resulting plot is called the "noiseless phase transition" plot, and it shows how many measurements (M) are needed for OMP to successfully recover the sparse signal, as a function of s_{max}. Do you observe a sharp transition region where the probability quickly transitions from a large value (close to 1) to a very small value (close to 0)? Generate different phase transition plots for the following values of N: 20, 50 and 100. Regenerate phase transition plots for average Normalized Error (instead of probability of successful support recovery). Comment on both kinds of plots.

4. **Noisy case: ($\mathbf{n \neq 0}$)**

   (a) Assume that sparsity s is known. Implement OMP, and terminate the algorithm after first s columns of $\mathbf{A}$ are selected. Generate "noisy phase transition" plots (for fixed N and σ) where success is defined as the event that the Normalized Error is less than 10^{-3}. Repeat the experiment for two values of σ (one small and one large) and choose N as 20, 50, and 100. Comment on the results.

   (b) Assume the sparsity s is NOT known, but $\|\mathbf{n}\|_2$ is known. Implement OMP where you may stop the OMP iterations once $\|\mathbf{y} - \mathbf{A}\mathbf{x}^{(k)}\| \leq \|\mathbf{n}\|_2$). Generate phase transition plots using the same criterion for success as the previous part. Comment on the results.

5. **Decode a Compressed Message:** In this part of the assignment, you will uncover a hidden message from their compressed sketches (generated using random measurement matrices) using the OMP algorithm that seeks the sparsest solution. An unknown and sparse image $\mathbf{X}$,

containing a message, has been compressed using three different random matrices (of different sizes) $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3$ to produce three corrupted images as follows

$$\mathbf{Y}_i = \mathbf{A}_i \mathbf{X}$$

The corrupted images and the measurement matrices can be accessed from this link

(a) Can you guess the message by simply displaying the compressed images?

(b) Using OMP, recover $\mathbf{X}$ from $\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3$. Figure out the appropriate stopping criteria. Reshape the recovered $\mathbf{X}$ into a 2D image of size 90×160 and decode the message. Show your results for each case. Compare these results with the Least Squares Solution[†].

(c) Which corrupted image gave you the best result for OMP? Can you explain why?

(d) (*For Fun*) Can you make an (educated) guess about the meaning of this message?

6. **Decode a Compressed Audio Signal**: In this part, you will uncover a hidden message from its compressed sketches of an audio signal using the OMP algorithm. An unknown audio signal $\mathbf{x}$ with frame per second (fps) of 7350 containing a message, has a sparse representation over a special basis $\mathbf{D}$:

$$\mathbf{x} = \mathbf{D}\mathbf{s}$$

where $\mathbf{s}$ is 100-sparse vector of size 15980. The audio signal $\mathbf{x}$ has been compressed using a random matrix $\mathbf{A}$ to produce compressed signal $\mathbf{y}$, as follows

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{D}\mathbf{s}$$

The compressed signal, the measurement matrix $\mathbf{A}$ and the special basis $\mathbf{D}$ can be accessed from this link.

The basis matrix $\mathbf{D}$ is normalized, converted to a 8bit format and saved as a *.tiff* file to reduce the storage size. To recover $\mathbf{D}$ use the provided code: *loading-data*. This function takes the address of the folder containing the data as input and returns $\mathbf{y}, \mathbf{A}, \mathbf{D}$

(a) Can you guess the message by simply playing the compressed signal? use the following Matlab command:
sound(y,fps)

(b) For all values of $k \in K = \{10, 50, 100, 200, 300, 1000, 2000, 3000\}$, select first k elements of $\mathbf{y}$, ($\mathbf{y}_k = [\mathbf{y}]_{1:k}$) and first k rows of $\mathbf{A}$, ($\mathbf{A}_k = [\mathbf{A}]_{1:k,:}$). Using OMP, recover $\mathbf{s}$ from $\mathbf{y}_k$ for all $k \in K$ and play $\mathbf{x} = \mathbf{D}\mathbf{s}$ using `sound(x,fps)`. What is the message? Compare these results with the Least Squares Solution.

(c) What is minimum number of measurements (call it $K_{\min}$) that is enough to understand the message from the reconstructed audio signal (based on your perception)? Can you explain what parameters of problem this number depends on? What is the relation of this number to size and sparsity of $\mathbf{s}$? Can you comment on what advantages more measurements (over $K_{\min}$) offer?

[†]You can use "lsqr()" command in Matlab or equivalent build-in function in python

(d) (Bonus) Can you think of the method that has been employed to construct the basis $\mathbf{D}$ that results in a sparse representation of our audio signal $\mathbf{x}$? Can you propose a method to construct such basis, $\mathbf{D}$?