ECE 269: Linear Algebra and Applications
Fall 2020

Project
Due: Tuesday, December 13, 11:59 pm,
via Gradescope

Collaboration Policy: Each student should work individually on the project. Choose any one of the following three Mini Projects. You can use MATLAB or Python for writing the source code. You are expected to submit a project report (along with figures) and suitably commented MATLAB/Python source codes. Wherever applicable, you should write your own code. For example, if the project asks you to compute the principal components, you should not use in-built library functions to compute them. If you use any source code from the internet without properly citing the source, it will be considered as an act of violation of academic integrity and will be duly reported. Wherever applicable, you should provide critical discussions and interpretation of your results.

---

1. **Mini Project 1: Orthogonal Matching Pursuit (OMP) and Sparse Signal Recovery:**
   In this mini project, we will implement and study the performance of the Orthogonal Matching Pursuit (OMP) algorithm for recovering sparse signals. Please read the paper "Greed is good: algorithmic results for sparse approximation" by Joel Tropp to learn about OMP(available on Canvas). Consider the measurement model

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

   where $\mathbf{y} \in \mathbb{R}^M$ is the measurement, $\mathbf{A} \in \mathbb{R}^{M \times N}$ is the measurement matrix, and $\mathbf{n} \in \mathbb{R}^N$ 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 $|\mathcal{S}| = s$.

   (a) **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

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

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

   (b) **Experimental setup:**

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

   ii. 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 $[-10, -1] \cup [1, 10]$.

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

iv. For each cardinality $s \in [1, s_{max}]$, the average Normalized Error should be computed by repeating step (1) to step (3) 2000 times and averaging the results over these 2000 Monte Carlo runs.

(c) **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) 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 recovery). Comment on both kinds of plots.

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

i. Assume that sparsity s is known. Implement OMP (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.

ii. 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.

iii. Design a numerical experiment to test OMP on real images. Describe your approach in detail about how you generate the measurement model, and comment on the quality of reconstructed images as you vary the number of measurements. What is the maximum compression (i.e the ratio of M/N) that still leads to (visually) satisfactory reconstruction? Show the reconstructed image for different values of M to justify your answer.

2. **Mini Project 2: Face Recognition Using Principal Component Analysis:**
Please read the following paper which shows how to use Principal Component Analysis (PCA) for Face Recognition (uploaded on Canvas).

M. A Turk and A. P. Pentland, "Face Recognition Using Eigenfaces", Proceedings of IEEECVPR 1991.

The paper puts forward a simple yet effective idea of using eigenfaces (obtained via PCA) to perform unsupervised face recognition. Read and understand the basic principle, and then conduct the following numerical experiments to implement and test the eigenface-based face recognition algorithm.

Data: The following data sets consist of faces of 200 people and each person has two frontal images (one with a neutral expression and the other with a smiling facial expression),there are 400 full frontal face images manually registered and cropped.
`http://fei.edu.br/~cet/frontalimages_spatiallynormalized_cropped_equalized_part1.zip`
`http://fei.edu.br/~cet/frontalimages_spatiallynormalized_cropped_equalized_part2.zip`

Implementation and Experiments:

(a) Compute the principal components (PCs) using first 190 individuals' neutral expression image. Plot the singular values of the data matrix and justify your choice of principal components.

(b) Reconstruct one of 190 individuals' neutral expression image using different number of PCs. As you vary the number of PCs, plot the mean squared error (MSE) of reconstruction versus the number of principal components to show the accuracy of reconstruction. Comment on your result.

(c) Reconstruct one of 190 individuals' smiling expression image using different number of PCs. Again, plot the MSE of reconstruction versus the number of principal components and comment on your result.

(d) Reconstruct one of the other 10 individuals' neutral expression image using different number of PCs. Again, plot the MSE of reconstruction versus the number of principal components and comment on your result.

(e) Use any other non-human image (e.g., car image, resize and crop to the same size), and try to reconstruct it using all the PCs. Comment on your results.

(f) Rotate one of 190 individuals' neutral expression image with different degrees and try to reconstruct it using all PCs. Comment on your results.

3. **Mini Project 3: Robust Linear Regression**
   Reference: G. Papageorgiou, P. Bouboulis, S. Theodoridis, K. Themelis, 2015. Robust linear regression analysis - a greedy approach. IEEE Transactions on Signal Processing 63: 3872-3887.

   This paper studies a robust linear regression problem based on OMP. Please read the whole paper carefully (including the appendix) and implement the original GARD algorithm (see Algorithm 1 on page 4). The authors further use Cholesky decomposition to speed up their algorithm. Try to use QR decomposition as well as Matrix Inversion Lemma instead of Cholesky decomposition to speed up the algorithm. Compare GARD (one of your fast implementations) with least squares and M-estimators (see MATLAB robustfit function, and choose any two weight functions) by conducting the experiments described in section V.A, V.C, V.D of the paper (no need to compare with other methods listed in the paper). In experiment section V.A, compare the run-time of all versions of GARD (original, QR, Matrix Inversion Lemma) with least squares and M-estimators.

   Finally, test these methods on any two real data sets (x01.txt-x28.txt) from
   `http://people.sc.fsu.edu/~jburkardt/datasets/regression/regression.html`

   E.g., The selling price of houses is to be represented as a function of 11 variables:
   `http://people.sc.fsu.edu/~jburkardt/datasets/regression/x26.txt`