Homework 5

AMATH 582/482, Winter 2022

Assigned Feb 25, 2022. Due on Mar 18, 2022 at midnight.

Directions, Reminders and Policies

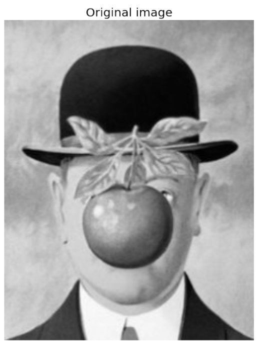
Read these instructions carefully:

- **You are required to upload a PDF report to Canvas along with a zip of your code. Note the PDF and the zip should be uploaded separately.**
- The report should be a maximum of 6 pages long with references included. Minimum font size 10pts and margins of at least 1inch on A4 or standard letter size paper.
- Do not include your code in the report. Simply create a zip file of your main scripts and functions, without figures or data sets included, and upload the zip file to Canvas.
- Your report should be formatted as follows:
  - Title/author/abstract: Title, author/address lines, and short (100 words or less) abstract. This is not meant to be a separate title page.
  - Sec. 1. Introduction and Overview
  - Sec. 2. Theoretical Background
  - Sec. 3. Algorithm Implementation and Development
  - Sec. 4. Computational Results
  - Sec. 5. Summary and Conclusions
  - Acknowledgments (no more than four or five lines, also see the point below on collaborations)
  - References
- I suggest you use LaTeX(Overleaf is a great option) to prepare your reports. A template is provided on Canvas under the Syllabus tab. You are also welcome to use Microsoft Word or any other software that properly typesets mathematical equations.
- I encourage collaborations, however, everything that is handed in (both your report and your code) should be your work. You are welcome to discuss your assignments with your peers and seek their advice but these should be clearly stated in the acknowledgments section of your reports. This also includes any significant help or suggestions from the TAs or any other faculty in the university. You don't need to give all the details of the help you received, just a sentence or two.
- Your homework will be graded based on how completely you solved it as well as neatness and little things like: did you label your graphs and include figure captions. **The homework is worth 20 points. 10 points will be given for the overall layout, correctness and neatness of the report, and 10 additional points will be for specific technical things that the TAs will look for in the report itself.**
- **Late submissions will not be accepted on Canvas, send them to bamdadh@uw.edu directly. Late reports are subject to a 2 points/day penalty up to five days. They are no longer accepted afterwards. For example, if your report is three days late and you managed to get $16/20$, your final grade will be $16 - 6 = 10$.** Basically, you will lose 2% of your overall course grade for each day the report is late. So be careful.

Problem Description: Compressed Image Recovery

This homework makes heavy use of the CVX package for convex optimization so make sure you have the appropriate Python or MATLAB variants installed for your programming language of choice.

The problem we consider is that of recovering an image from limited observations of its pixels. For example, consider a portion of René Magritte's "The Son of Man" along with its corrupted variant. Our goal is to recover the original image from the corrupted version.



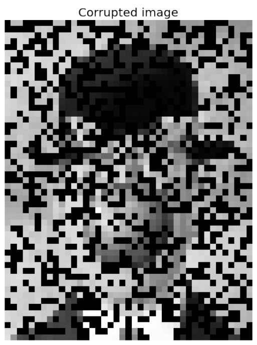


Figure 1 (Left) Original image at high-resolution. (Right) Corrupted low-resolution image with random pixels missing.

Download the data files `SonOfMan.png`, `UnknownImage.py` (or `UnknownImage.mat` if you are using MATLAB) along with the notebook `HW5-Helper.ipynb` to get you started. If you are using MATLAB it is still worthwhile taking a look at the notebook to get you started. Most importantly, the helper notebook rescales the original image to lower resolution to make the computations manageable. Otherwise, the optimization step can take a very long time.

Tasks

Below is a list of tasks to complete in this homework and discuss in your report.

1. **(Image compression):** Your goal here is to investigate the compressibility of the discrete cosine transform (DCT) of `SonOfMan.png`. Given a discrete signal $\mathbf{f} \in \mathbb{R}^K$ we define its DCT as $\mathrm{DCT}(\mathbf{f}) \in \mathbb{R}^K$ where

$$\mathrm{DCT}(\mathbf{f})_k = \sqrt{\frac{1}{K}} \left[f_0 \cos\left(\frac{\pi k}{2K}\right) + \sqrt{2} \sum_{j=1}^{K-1} f_j \cos\left(\frac{\pi k(2j+1)}{2K}\right) \right].$$

This transform is analogous to taking the real part of the FFT of $\mathbf{f}$, i.e., writing the signal as a sum of cosines (other definitions of DCT exist in the literature; see `scipy.fftpack.dct` manual page). The inverse DCT (iDCT) transform reconstructs the signal $\mathbf{f}$ from $\mathrm{DCT}(\mathbf{f})$. The 2D DCT (resp. iDCT) is defined analogously to 2D FFT (resp. iFFT) by successively applying the 1D DCT (resp. iDCT) to the rows and columns of a 2D image.

*Perform the following using the rescaled 53×41 image and not the original.

   (a) Let N_y, N_x denote the number of rows and columns of the image with $N = N_y \times N_x$ denoting the number of pixels in the image. Let $F \in \mathbb{R}^{N_y \times N_x}$ denote the image and write $\mathrm{vec}(F) \in \mathbb{R}^N$ for its vectorization, i.e., pixel values stacked into a long vector akin to Numpy's `flatten` function. Use the functions `construct_DCT_Mat` and `construct_iDCT_Mat` to construct the forward and inverse DCT matrices for the image.

Note 1: If you are using MATLAB you may directly use the `dct2` function. Python users do not have access to these.

Note 2: The above functions construct matrices $D, D^{-1} \in \mathbb{R}^N$ such that $D\text{vec}(F) = \text{vec}(\text{DCT}(F))$ and $D^{-1}\text{vec}(\text{DCT}(F)) = \text{vec}(F)$.

(b) Plot DCT(F) and investigate its compressibility. Do you see a lot of large coefficients?

(c) Reconstruct and plot the image after thresholding its DCT to keep the top $5, 10, 20$, and 40 percent of DCT coefficients.

2. **(Compressed Image Recovery)** Your goal in this step is to recover the image F from limited random observations of its pixels using the prior knowledge that the DCT(F) is nearly sparse.

*Perform the following using the rescaled 53×41 image and not the original.

(a) Let $M < N$ be an integer and construct a measurement matrix $B \in \mathbb{R}^{M \times N}$ by randomly selecting M rows of the identity matrix $I \in \mathbb{R}^{N \times N}$ (Hint: look up `numpy.random.permutation`).

(b) Generate a vector of measurements $\mathbf{y} \in \mathbb{R}^M$ by applying B to vec(F).

(c) Define the matrix $A = BD^{-1} \in \mathbb{R}^{M \times N}$ and use CVX to Solve the optimization problem

$$\begin{aligned} &\text{minimize}_{\mathbf{x}\in\mathbb{R}^N} && \|\mathbf{x}\|_1, \\ &\text{Subject to} && A\mathbf{x} = \mathbf{y}, \end{aligned} \tag{1}$$

and denote the minimizer as $\mathbf{x}^*$. Clearly, this vector $\mathbf{x}^*$ is the DCT vector of an image F^* that, hopefully, resembles the original image F.

Note: This problem can take several minutes to converge. Make sure you run `CVX`'s solve command with the options: "`verbose=True, solver = 'CVXOPT', max_iter= 1000, reltol=1e-2, featol = 1e-2`" to speed up the computations and monitor convergence of the solver.

(d) Take $M = r \times N$ for $r = 0.2, 0.4, 0.6$. For each choice of M perform the steps (2a-2c) above to obtain an approximation F^* to F and repeat these calculations three times by re-drawing the random matrix B. The entire calculation should lead to a total of 9 images (3 per each value of M) that approximate the original image F. Present a plot of these images and discuss your results.

Note 1: This step is very expensive and might take a long time. My advice to you is to prototype your code with an even lower resolution image than 53×41 so `CVX` converges quickly. Once you are happy with your code run it to generate plots for your report.

Note 2: The reason for repeating the experiments with new B's is to make sure that you are not getting too lucky/unlucky with your random matrix B.

3. **(A Mysterious Image)** Hopefully, by this point you have a working code that is capable of recovering an image from limited and indirect measurements by solving problem (1). The file `UnknownImage.py` (resp. `UnknownImage.mat`) contains the measurement vector $\mathbf{y}$ and the measurement matrix B for an unknown image of size 50×50 pixels. Read the file using `numpy.load` and use your code for Step 2c to reconstruct and visualize the unknown image.

(0 mark question) Do you recognize it?