EECS 16A Designing Information Devices and Systems I
Summer 2017 D. Aranki, F. Maksimovic, V. Swamy Homework 8

This homework is due on Wednesday, August 9, 2017, at 23:59.
Self-grades are due on Thursday, August 10, 2017, at 23:59.

Submission Format
Your homework submission should consist of **two** files.

- `hw8.pdf`: A single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

  If you do not attach a PDF of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible.

- `hw8.ipynb`: A single IPython notebook with all of your code in it.

  In order to receive credit for your IPython notebook, you must submit both a "printout" and the code itself.

Submit each file to its respective assignment on Gradescope.

1. Mechanical Gram-Schmidt

(a) Use Gram-Schmidt to find a matrix $\mathbf{U}$ whose columns form an orthonormal basis for the column space of $\mathbf{V}$.

$$\mathbf{V} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Show that you get the same resulting vector when you project $\vec{w} = \begin{bmatrix} 1 & -1 & 0 & -1 & 0 \end{bmatrix}^T$ onto $\mathbf{V}$ and onto $\mathbf{U}$, i.e. show that

$$\mathbf{V}(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\vec{w} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\vec{w}.$$

(b) Use Gram-Schmidt to find a matrix $\mathbf{U}$ whose columns form an orthonormal basis for the column space of $\mathbf{V}$.

$$\mathbf{V} = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & -1 & 1 \end{bmatrix}$$

Show that you get the same resulting vector when you project $\vec{w} = \begin{bmatrix} 1 & -1 & 0 & -1 & 0 \end{bmatrix}^T$ onto $\mathbf{V}$ and onto $\mathbf{U}$.

(c) Compute the QR decomposition of the above matrices, that is, find an orthogonal matrix $\mathbf{Q}$ and an upper triangular matrix $\mathbf{R}$, such that $\mathbf{V} = \mathbf{QR}$.

2. Deconstructing Trolls

(a) Consider two vectors that are orthogonal to each other, $\vec{a}$ and $\vec{b}$. Suppose we scale both vectors by some scalar c. Are they still orthogonal? Now suppose we scale them by two different scalars, c_1 and c_2. Are they still orthogonal?

(b) Now, let's construct two new vectors $\vec{x}_1$ and $\vec{x}_2$ that are both linear combinations of $\vec{a}$ and $\vec{b}$. In general, are $\vec{x}_1$ and $\vec{x}_2$ orthogonal? If yes, provide some justification, if not, provide a counter example.

(c) Suppose now we have a matrix $\mathbf{M}$ whose columns are $\vec{x}_1$ and $\vec{x}_2$. We find the QR factorization of $\mathbf{M}$. What is one possible $\mathbf{Q}$ matrix? Recall that the $\mathbf{Q}$ matrix is a set of orthogonal vectors that span the column space of the matrix $\mathbf{M}$.

(d) Now let's revisit the troll problem we saw in homework 1. Recall that there were two speakers and two microphones. The microphones recorded sound that was a linear combination of the two speakers. More specifically, let the sound of the two speakers be represented by the length n vectors $\vec{a}$ and $\vec{b}$ and the sound recorded by the microphones be represented by the length n vectors $\vec{m}_1$ and $\vec{m}_2$, where $\vec{m}_1$ contains mostly the troll. We know that $\vec{a}$ and $\vec{b}$ are orthogonal and that $\vec{m}_1$ and $\vec{m}_2$ are linear combinations of $\vec{a}$ and $\vec{b}$. Given just $\vec{m}_1$ and $\vec{m}_2$, devise a way to find $\vec{a}$ and $\vec{b}$. Justify your answer.

(e) Implement your method in the provided IPython notebook and comment on the results.

3. Sparse Imaging

Recall the imaging lab where we projected masks on an object to scan it to our computer using a single pixel measurement device, that is, a photoresistor. In that lab, we were scanning a 30×40 image having 1200 pixels. In order to recover the image, we took exactly 1200 measurements because we wanted our 'measurement matrix' to be invertible.

However, we saw that an iterative algorithm that does "matching and peeling" can enable reconstruction of a sparse vector while reducing the number of samples that need to be taken from it. In the case of imaging, the idea of sparsity corresponds to most parts of the image being black with only a small number of light pixels. In these cases, we can reduce the overal number of samples necessary. This would reduce the time required for scanning the image. (This is a real-world concern for things like MRI where people have to stay still while being imaged.)

In this problem, we have a 2D image I of size 91×120 pixels for a total of 10920 pixels. The image is made up of mostly black pixels except for 476 of them that are white.

Although the imaging illumination masks we used in the lab consisted of zeros and ones, in this question, we are going to have masks with real values — i.e. the light intensity is going to vary in a controlled way. Say that we have an imaging mask M_0 of size 91×120. The measurements using the solar cell using this imaging mask can be represented as follows.

First, let us vectorize our image to $\vec{i}$ which is a length 10920 column vector. Likewise, let us vectorize the mask M_0 to $\vec{m}_0$ which is a length 10920 column vector. Then the measurement using M_0 can be represented as

$$b_0 = \vec{m}_0^T \vec{i}.$$

Say we have a total of M measurements, each taken with a different illumination mask. Then, these measurements can collectively be represented as

$$\vec{b} = \mathbf{A}\vec{i},$$

where $\mathbf{A}$ is an $M \times 10920$ size matrix whose rows contain the vectorized forms of the illumination masks, that is

$$\mathbf{A} = \begin{bmatrix} \vec{m}_1^T \\ \vec{m}_2^T \\ \vdots \\ \vec{m}_M^T \end{bmatrix}.$$

To show that we can reduce the number of samples necessary to recover the sparse image I, we are going to only generate 6500 masks. The columns of $\mathbf{A}$ are going to be approximately uncorrelated with each other. The following question refers to the part of IPython notebook file accompanying this homework related to this question.

(a) In the IPython notebook, we call a function `simulate` that generates masks and the measurements. You can see the masks and the measurements in the IPython notebook file. Complete the function `OMP` that does the OMP algorithm described in lecture.

Remark: Note that this remark is not important for solving this problem; it is about how such measurements could be implemented in our lab setting. When you look at the vector `measurements` you will see that it has zero average value. Likewise, the columns of the matrix containing the masks $\mathbf{A}$ also have zero average value. To satisfy these conditions, they need to have negative values. However, in an imaging system, we cannot project negative light. One way to get around this problem is to find the smallest value of the matrix $\mathbf{A}$ and subtract it from all entries of $\mathbf{A}$ to get the actual illumination masks. This will yield masks with positive values, hence we can project them using our real-world projector. After obtaining the readings using these masks, we can remove their average value from the readings to get measurements as if we had multiplied the image using the matrix $\mathbf{A}$.

(b) Run the code `rec = OMP((height, width), sparsity, measurements, A)` and see the image being correctly reconstructed from a number of samples smaller than the number of pixels of your figure. What is the image?

(c) **PRACTICE:** We have supplied code that reads a PNG file containing a sparse image, takes measurements, and performs OMP to recover it. An example input file is also supplied together with the code. Generate an image of size 91×120 pixels of sparsity less than 400 and recover it using OMP with 6500 measurements.

You can answer the following parts of this question in very general terms. Try reducing the number of measurements. Does the algorithm start to fail in recovering your sparse image? Why do you think it fails? Make an image having fewer white pixels. By how much can you reduce the number of measurements that needs to be taken?

4. Speeding Up OMP

Consider the Sparse Imaging problem above.

(a) Modify the code to run faster by using a Gram-Schmidt orthonormalization to speed it up. (Edit the code given to you in `prob8.ipynb`.)

(b) **PRACTICE:** Do any other modifications you want to further speed up the code.

Hint: When possible, how would you safely extract multiple peaks corresponding to multiple pixels in one go and add them to the recovered list? Would this speed things up?

5. Homework Process

(a) Who else did you work with on this homework? List names and student ID's. (In case of homework party, you can also just describe the group.) How did you work on this homework?

Working in groups of 3-5 will earn you credit for your participation grade.

(b) Copy the following statement into your homework submission.

> I, [insert name here], affirm that all of my solutions are entirely my work and that I have properly credited and acknowledged any sources or work that I have consulted.