EECS 16A Designing Information Devices and Systems I
Spring 2015 Homework 3a

This homework is due February 26, 2015 at 5PM.

1. **Image Deblurring**

   In this problem, we will blur and distort images through matrix multiplication and try to recover the original image. Suppose we want to blur the image A using the distortion matrix D. Then the matrix multiplication

   $$DA = B$$

   would result in the blurred image B. Note that in this problem, the image A is represented as a matrix, not a vector.

   (a) We want B to be an image where each pixel is the average of the pixel above and below it in image A. An example below:

   $$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, B = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ \frac{a_{11}+a_{31}}{2} & \frac{a_{12}+a_{32}}{2} & \frac{a_{13}+a_{33}}{2} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

   Note that the edge pixels (top and bottom border) of image B are the same as those of image A. Design a matrix D that would achieve this "blurring" for a 5×5 image A.

   (b) Download the iPython notebook `prob3.ipynb` and the image `campanile.npy`. Extend your distortion matrix from part (a) to blur the campanile image.

   (c) Given a distortion matrix D and the blurred image B, can you *always* recover the original image A? Why or why not?

   (d) Download `distortion.npy` (D) and `blurred.npy` (B). Recover the original image A.

2. **Counting the paths of a Random Surfer**

   In class, we discussed the behavior of a random web-surfer who jumps from webpage to webpage. We would like to know how many possible paths there are for a random surfer to get from a page to another page. To do this, we represent the webpages as a graph. If page 1 has a link to page 2, we have a directed edge from page 1 to page 2. This graph can further be represented by what is known as an "incidence matrix", A, with elements a_{ij}. $a_{ji} = 1$ if there is link from page i to page j. Matrix operations on the incidence matrix make it very easy to compute the number of paths to get from one webpage to webpage.

   This path counting actually is an implicit part of the how the "importance scores" for each webpage are described. Recall that the "importance score" of a website is the steady-state frequency of the fraction of people on that website.

   Consider the following graphs.

   (a) Write out the incidence matrix for graph A.

   (b) For graph A: How many one-hop paths are there from webpage-1 to webpage-2? How many two-hop paths are there from webpage-1 to webpage-2? How about 3-hop?

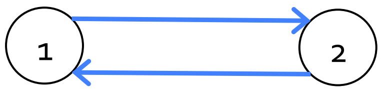

Figure 1: Graph A

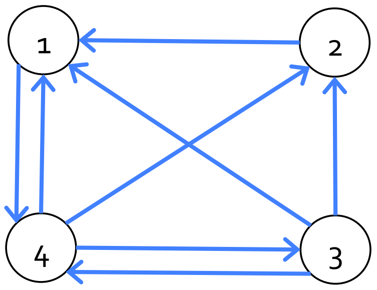

Figure 2: Graph B

(c) For graph A: What are the importance scores of the two webpages?

(d) Write out the incidence matrix for graph B.

(e) For graph B: How many two-hop paths are there from webpage-1 to webpage-3? How many three-hop paths are there from webpage-1 to webpage-2?

(f) For graph B: What are the importance scores of the webpages?

(g) Write out the incidence matrix for graph C.

(h) For graph C: How many paths are there from webpage-1 to webpage-3?

(i) For graph C: What are the importance scores of the webpages? How is graph (c) different from graph (b), and how does this relate the importance scores and eigenvalues and eigenvectors you found?

3. Rotation matrix

Find the eigenvalues and eigenvectors for the 2D θ-rotation matrix. Remember, an eigenvector is a direction that remains invariant when it is transformed by the matrix multiplication. Comment on what your solution means.

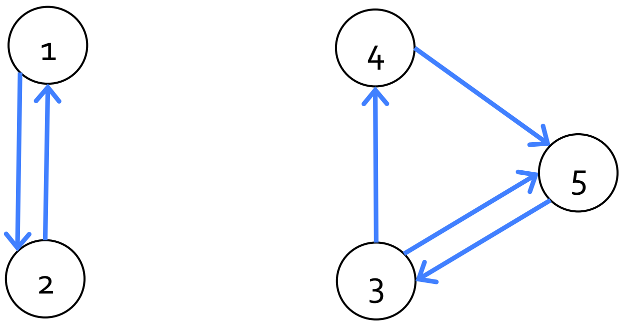

Figure 3: Graph C

4. Image Compression

In this question, we explore how eigenvalues and eigenvectors can be used for image compression. We have seen that a grayscale image can be represented as a data grid. Say a symmetric, square image is represented with a 2D data grid, A. We've been transforming the images to vectors in the past to make it easier to process them as data, but here we will understand them as 2D data. Let $\lambda_1 \cdots \lambda_n$ be the eigenvalues of A with corresponding eigenvectors $v_1 \cdots v_n$. Then, the matrix can be represented as

$$A = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \cdots + \lambda_n v_n v_n^T$$

However, the matrix A can also be *approximated* with the k largest eigenvalues and corresponding eigenvectors. That is,

$$A \approx \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \cdots + \lambda_k v_k v_k^T$$

(a) Can you construct appropriate matrices U, V (using v_i's as rows and columns) and a matrix Λ with the eigenvalues λ_i as components such that

$$A = U \Lambda V$$

.

(b) Download the iPython notebook `prob3.ipynb` and the image `pattern.npy`. Use the `numpy.linalg` command `eig` to find the U and Λ matrices for the image. Mathematically, how many eigenvectors are required to fully capture the information within the image?

(c) Find an approximation for the image using the 100 largest eigenvalues and eigenvectors. That is, use these eigenvalues and eigenvectors to form the U, V and Λ matrices from part (a) and multiply them to form an approximation of image.

(d) Repeat part (c) with $k = 50$. By further experimenting with the code, what seems to be the lowest value of k that retains most of the salient features of the given image?

5. Applications

Are there any interesting applications of the Linear Algebra we have learned in class so far that we have not discussed? List your favorite application. Feel free to make up an application, or use a real life example that we have not discussed in class.