CS170A — Mathematical Modeling & Methods for Computer Science
HW#1 *SVD, LSQ, PCA*
Due: 3:59pm Monday February 10, 2014

D. Stott Parker, Xingyu Ma, Costas Sideris

Using CourseWeb, please turn in your Matlab code and resulting output (enough to show that the program is working) as a PDF document.

Images are often represented as matrices, and Matlab is a popular tool for analyzing them. In this problem you will develop a few image editing tools with matrix operations. You may want to look up these commands:

```
help imshow
help image
help imagesc
```

The Mandrill image we discussed in class is in the subdirectory `toolbox/matlab/demos/` – load it into Matlab and downsample it using your `downsample` function:

```
load mandrill
% loads in an image and its colormap:
%   X    480x500;  entries are integer color codes
%   map  220x3;    rgb = map(i,:)   gets the RGB triple for code i
imshow(X, map)
Mandrill = ind2rgb( X, map ); %  convert to an RGB image
size(Mandrill)
imshow(Mandrill)
```

This should create and display a $480 \times 500 \times 3$ array `Mandrill`.

1. **Image Resampling (10 points)**
   Write a Matlab function `resample(A,p,q)` that takes a $m \times n$ RGB image `A` as input, as well as two integers p and q, and returns a RGB image of size $\lfloor s\,m \rfloor \times \lfloor s\,n \rfloor$ where $s = p/q$ is a scaling factor.

   For example, when $s = 1/2$ this is downsampling by a factor of 2. This averages every pair of adjacent pixels in the input image on both axes, so altogether 2×2 blocks in the input are averaged to give one pixel in the output. Also, when $s = 2/1$ we get upsampling by a factor of 2. Every pixel in the input is then replicated into a 2×2 block in the output. Almost any scale $s = p/q$ can be implemented by first upsampling by a factor of p, and then downsampling by a factor of q.

   (a) Show the result of `resample(Mandrill, 2, 5)`.

   (b) Prove that this resampling transformation is a bi-linear transformation. That is, it applies linear transformations on both the left and the right of (matrices corresponding to each of the 3 color channels of) the input image.

2. **Anti-Aliasing (10 points)**
   If the sizes p and q for resampling an RGB image A happen to match features in A, the result can look bad — such as with 'jagged-looking edges'. The resampling can lose finer details of A to *aliasing*.

   Often we can fix aliasing problems by smoothing out the result. If S is the $n \times n$ 'anti-aliasing' matrix

$$S = \frac{1}{4}\begin{pmatrix} 3 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 3 \end{pmatrix}$$

   Notice that if we compute the product $B = S\,A$, each entry b_{ij} in the result is a weighted average of adjacent elements in A, something like $b_{ij} = (a_{i(j-1)} + 2a_{ij} + a_{i(j+1)})/4$.

(a) First, write a Matlab function `S(n)` that yields the $n \times n$ anti-aliasing matrix S for any $n > 0$. Notice that S is always a Doubly Stochastic Matrix. For every value of n, what is the largest eigenvalue of S? Show that (up to eigenvector sign) the corresponding eigenvector has all entries the same: $1/\sqrt{n}$.

(b) Next write a Matlab function `anti_alias(A)` that transforms A using two anti-aliasing matrices. Show the result of eight anti-aliasings of `resample(Mandrill, 2, 5)`.

(c) Prove that the bilinear anti-aliasing transform is invertible.

Hints: Color intensities are often `uint8` (8-bit unsigned integer) values, so using the `double()` and `uint8()` conversion functions may be needed to perform arithmetic operations on images. For example, `imshow( uint8(min(255, 1.5 * double( A ))))` multiplies each color intensity in `A` by 1.5. It may help to define define functions to convert between these 3D arrays (like `A`) and 2D matrices (`R`, `G`, and `B`) such as these:

```
function [R,G,B] = image2rgb(A)
        R = double(A(:,:,1));
        G = double(A(:,:,2));
        B = double(A(:,:,3));

function A = rgb2image(R,G,B)
        A(:,:,1) = uint8(R);
        A(:,:,2) = uint8(G);
        A(:,:,3) = uint8(B);
        %  equivalent:  A = cat(3, uint8(R),uint8(G),uint8(B));
```

With `image2rgb` any RGB image is split into 3 matrices, one for each color. These can be transformed mathematically and the converted back with `rgb2image`.

3. **Color Vision Repair in Video Games (20 points)**
Color blindness, as mentioned in the course notes, is a result of changes in the wavelength sensitivities of red and green cones in the eyes. Shifts in wavelength peak sensitivities of red and green, or different densities of red and green (or blue) cones in the retina, produce changes in the range of perceived colors. Approximately 5% of all people are estimated to experience some such difference, with an order of magnitude greater prevalence among men than among women.

The site daltonize.org has a good summary of different forms of color blindness. Four of the most common are: Protanomaly (red altered: 1% of males); Deuteranomaly (green altered: 5% of males); Protanopia (red-blind: 1% of males); Deuteranopia (green-blind: 1% of males). The Wikipedia pages on Color blindness and LMS color space have good background reading for this problem.

Color blindness has been looked at historically as a deficiency, but it actually might have been the opposite. Adjusted wavelengths can work well in environments where differences between red and green are less important — such as in night vision. It also can come with enhanced light sensitivity. Differences that involve genetic variations like these might reflect some historical competitive advantage.

New 'sunglasses' for color blindness have appeared recently; that is the basis for this problem. These modify light wavelengths in ways that permit better discrimination between red and green. These are not science fiction; they are available on Amazon.

Apparently most game vendors also do some kind of color correction for color blindness. Computer algorithms are used to shift colors so that important features can be seen by everyone. A popular algorithm is called **Daltonization** specified at daltonize.org.

Your job is to develop a function `daltonize(A,type)` that takes an RGB image A and converts it into an image Daltonized for either `type = 'protanopia'` or `type = 'deuteranopia'` using the materials at http://www.daltonize.org/search/label/Daltonize

(a) Show the result for `daltonize(Mandrill, 'protanopia')`.

(b) Show the result for `daltonize(Ishihara45, 'deuteranopia')` using `Ishihara45.jpg`.

(c) Show the result for `daltonize(Ishihara74, 'protanopia')` using `Ishihara74.jpg`.

4. **imagesvd (20 points)**
Get the script `imagesvd.m` from `http://www.mathworks.com/moler/ncmfilelist.html`.

Modify the script in two ways:

(a) At the bottom of the figure, or in another figure, include a *scree plot*, showing the sequence of singular values for the image. Whenever an image is selected, modify the program to display the scree plot for the image SVD immediately.

(b) The *rank-k SVD approximation* to a matrix A is

$$A^{(k)} \;=\; U^{(k)}\, S^{(k)}\, (V^{(k)})'$$

keeping only the first k columns in U and V, and keeping only the upper left $k \times k$ portion of S.

The common wisdom is that one should pick a value of k that is at the 'elbow' of the scree plot. Formalize this mathematically as a function of the sequence of singular values, and state it clearly.

(c) Show how to modify `imagesvd` to use your formalization of an elbow to pick an initial value of k. (Instead of k it uses 'r' as the rank, with a default initial value of 1, and makes the slider value match the value of r.)

5. **Least Squares (20 points)**
In the file `auto.m` is a set of data involving MPG in autos, as described in the course notes.

(a) Use least squares to find coefficients α and δ that yield the best model

$$1/\text{MPG} \;=\; \alpha \cdot \text{horsepower} \;+\; \delta$$

— that is, find the instance of this formula with minimal squared error.

(b) Find coefficients α, β, γ, δ such that the model

$$1/\text{MPG} \;=\; \alpha \cdot \text{horsepower} \;+\; \beta \cdot \text{displacement} \;+\; \gamma \cdot \text{weight} \;+\; \delta$$

best fits the data.

(c) What are the R^2 coefficients for these two models?

6. **PCA (20 points)**
Do PCA for the auto data, but:

- use 1/MPG instead of MPG — because attributes of the engine (volume, displacement, horsepower) are more nearly proportional to gallons than they are to miles.
- omit the Cylinder, Year, and Origin attributes from the table; they are not continuous numeric values and so they do not add information to the analysis.
- use the correlation matrix, not the covariance matrix.

(a) Generate a scree plot for the PCA. Where is the elbow?

(b) Identify the dominant 'loadings' (weights, eigenvector entries) in the first three principal components. Interpret their significance.

(c) Generate a 3D plot, showing the data projected onto the first three principal components. Autos with Origin = 3 are from Japan. Highlight these in the 3D plot, and comment on their position.