CS170A — Mathematical Modeling & Methods for Computer Science
HW#2 *SVD, LSQ, PCA*
Due: 11:59pm Sunday November 9, 2014

D. Stott Parker, Luis Angel Larios-Cardenas

Using CCLE, please turn in your work (with enough to demonstrate clearly that programs are working) as a PDF document.

1. **Singular Value Decomposition (20 points)**
One of the most important and useful results in computational linear algebra is that any $n \times p$ matrix A has a *Singular Value Decomposition (SVD)* $A = U\,S\,V'$ where U and V are unitary matrices and, assuming $n > p$, S is a diagonal matrix of nonnegative real *singular values* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$:

$$A = \left(\begin{array}{c|c|c|c|c} & & & & \\ \mathbf{u}_1 & \cdots & \mathbf{u}_k & \cdots & \mathbf{u}_n \\ & & & & \end{array}\right) \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & \ddots & \\ & & & & \sigma_p \end{pmatrix} \left(\begin{array}{c|c|c|c|c} & & & & \\ \mathbf{v}_1 & \cdots & \mathbf{v}_k & \cdots & \mathbf{v}_p \\ & & & & \end{array}\right)' .$$

The singular values are a lot like eigenvalues, but they are always real values, and they are never negative.

In Matlab, the command `[U,S,V] = svd(A)` finds the SVD of `A`.

(a) Implement the SVD as a function

```
[U,S,V] = mysvd(A)   %%  A can be any REAL rectangular matrix
```

Your implementation can build upon the Jacobi eigendecomposition procedure described in the course notes, but however you do this, your implementation of the SVD must be self-contained, without using builtin functions other than trigonometric functions (*sin*, *cos*, *arctan*, ...).

Important: your function must return singular values in descending order.

For each of the following five matrices, find the SVD using your implementation:

```
A = hilb(7)
B = magic(7)
C = pascal(7,1)
D = vander(1:7)
E = rosser()
```

(b) Notice that if $\mathbf{u}_k$ and $\mathbf{v}_k$ are the k-th left and right singular vectors of a matrix $A = U\,S\,V'$, then for any value of θ, replacing both $\mathbf{u}_k$ by $(e^{\mathbf{i}\theta}\,\mathbf{u}_k)$ and $\mathbf{v}_k$ by $(e^{\mathbf{i}\theta}\,\mathbf{v}_k)$ gives valid singular vectors. We say the two SVD results are *unit-equivalent*.

For each of the five matrices above, determine whether the SVD obtained by your implementation is unit-equivalent to the SVD obtained by the builtin function. (If it is, say so; if not, identify the differences.)

(c) The condition number of a matrix A is the ratio of its largest to smallest singular values: $\kappa(A) = \sigma_1(A) \,/\, \sigma_n(A)$. Using the singular values returned by your program, find the condition number of each matrix.

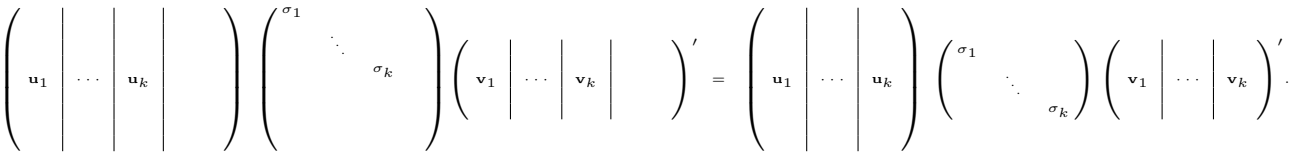
(d) Using the Matlab functions `solve()` and `charpoly()`, find exact values for the singular values of all five matrices, and find the condition number for each matrix. [If possible: `solve()` may not find exact solutions.]

(e) Using the Matlab function `vpa(...,50)` find values for all singular values of the five matrices using 50-digit precision, and find the condition number for each matrix.

(f) The script `imagesvd.m` from `www.mathworks.com/moler/ncmfilelist.html` shows how the SVD can be used to compress images. The *rank-k SVD approximation* to a matrix A keeps only the first k columns in U and V, and keeps only the upper left $k \times k$ portion of S:

$$A^{(k)} = U^{(k)}\,S^{(k)}\left(V^{(k)}\right)'.$$

The schematic of $A^{(k)}$ simply zeroes out some columns:



Modify `imagesvd()` to take a matrix A as input (rather than an image), and display the error matrix

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

when the user selects k with the slider.

To display a matrix like this, you can convert the matrix to an image (basically, convert the magnitudes of values in the matrix to a color), and then use the same approach for rendering used by `imagesvd()`.

Show the result of your program with $k = 50$ and $k = 100$ on the 400×400 matrix `TestMatrix`, which is in the file `TestMatrix.csv` and also in `TestMatrix.mat`. To read this in, try: `load 'TestMatrix'`

2. **Exponential Growth (15 points)**

In the files `guinea.csv`, `liberia.csv`, `sierra_leone.csv` are cumulative counts of Ebola virus cases and deaths in the three hardest-hit countries in West Africa, compiled by Gregory Piatetsky-Shapiro from reports from WHO (`www.who.int/csr/disease/ebola/situation-reports/en/`) as well as earlier reports issued by CDC (`www.cdc.gov/vhf/ebola/outbreaks/2014-west-africa/index.html`), and updated to Oct 24.

We want to determine whether the growth of Ebola cases over time is exponential or polynomial. (A function $f(t)$ is called **exponential** (or **geometric**) if $f(t) = O(c^t)$ for some nonzero constant c, and **polynomial** if $f(t) = O(t^n)$ for some positive integer n.) In order to resolve this question, modify `ebola.m` so as to accomplish the following:

(a) Read in each file `F.csv` file (e.g., using `csvread('F.csv',1,0)` in Matlab).

(b) Compute actual time values in days (using the month and day values in the file), and subtract the starting time, to obtain a vector of time values t for the country that starts at 0.

(c) For each of the three countries (where t is its time vector):

- Where x is first *Cases*, and then x is *Deaths*, obtain a degree-4 polynomial fit $x \sim \sum_{i=0}^{4} c_i t^i$.
- Where x is first *Cases*, and then x is *Deaths*, obtain an exponential fit $x \sim e^{c_0 + c_1 t}$ by fitting a linear model $\log(x) \sim c_0 + c_1 t$.
- Plot the *Cases* and *Deaths* data and the best polynomial and exponential fits together in one plot, and include the coefficients and squared error in the plot (e.g., with `title(sprintf(...))`).
- Determine whether a polynomial or exponential fit has lower squared error for *Cases*, and for *Deaths*.

3. **Fuel Economy (25 points)**

The homework zip file includes a table `mpg.csv` with 12 variables as follows:

Col #	*Name*	*Description*
1	id	numeric vehicle id
2	year	year manufactured
3	origin	1 = America, 2 = Europe, 3 = Asia
4	automatic-transmission	0 = No, 1 = Yes
5	cylinders	one of: 2, 3, 4, 5, 6, 8, 10, 12
6	displacement	engine displacement in liters; a value between 1.0 and 8.3 liters)
7	drive	1 = front-wheel drive, 2 = rear-wheel drive, 4 = 4 or all-wheel drive
8	cityMPG	a value between 10 and 72 mpg
9	highwayMPG	a value between 15 and 85 mpg
10	class-size	0 = compact, 1=small, 2=mid, 3=full, 4=large, 5=Large, 6=LARGE
11	guzzler	subject to gas guzzler tax: 0 = no, 1 = yes
12	smog-score	EPA smog rating, from 1 [great] to 9 [terrible]

We extracted this from the data at `www.fueleconomy.gov/feg/download.shtml`, which has information about fuel economy of vehicles since 1978.

(a) Use least squares to find coefficients c_0, c_1 that yield the best model

$$\log(\text{city MPG}) \;\sim\; c_0 \;+\; c_1 \;\log(\text{displacement})$$

— that is, find the coefficients of the model with minimal squared error.

(b) Find least squares coefficients $c_0, \ldots, c_4$ for the model

$$1/(\text{city MPG}) \;\sim\; c_0 \;+\; c_1\,\text{displacement} \;+\; c_2\,\text{cylinders} \;+\; c_3\,\text{class-size} \;+\; c_4\,\text{guzzler}$$

Give the R^2 for this model.

(c) Find the linear model for 1/(city MPG) whose R^2 value is highest, using at least one of these six variables: *origin, automatic-transmission, cylinders, displacement, drive, class-size*. This requires a search over all $2^6 - 1$ nonempty subsets of these variables, which can be implemented easily with 6 nested for-loops.

(d) Perform PCA for the vehicle data using the correlation matrix for the variables *cylinders, displacement, drive, class-size, year* and 1/(city MPG).

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

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

(g) Generate a 3D plot, showing the data projected onto the first three principal components. Color the points by their origin (1 = America, 2 = Europe, 3 = Asia), and comment on their position.

We use 1/(city MPG) instead of (city MPG) because attributes of the engine (displacement, cylinders) are more proportional to gallons than to miles.

4. **Los Angeles Power and Los Angeles Temperature (20 points)**

UCLA's `California Center for Sustainable Communities (CCSC)` has obtained data about power utilization in Los Angeles. We extracted a small subset from this data called `LApower.csv` that contains 2011 data for about 2500 'parcels' (geographic regions in Los Angeles). The dataset has 38 variables describing each parcel, including the longitude (x-coordinate) and latitude (y-coordinate) of its center:

Col #	*Name*	*Description*
1–12	logPowerJan11, ..., logPowerDec11	log of power utilization in the parcel in each month of 2011
13–24	TempJan11, ..., TempDec11	average temperature in the parcel in each month of 2011
25	Longitude	Longitude of the center of the Parcel (x-coordinate)
26	Latitude	Latitude of the center of the Parcel (y-coordinate)
27	logSqMeters	log of the size of the parcel (in square meters)
28	logAvgIncome	log of the average income of customers in the parcel
29	AvgYearBuilt	average year built (age) of buildings in the parcel
30	ResidentialCustomers	percentage of customers who are residential (integer between 0 and 100)
31	CommercialCustomers	percentage of customers who are commercial (integer between 0 and 100)
		Customer distribution (integers between 0 and 100):
32	PercentMXR	percentage of customers who are merchandisers
33	PercentMFR	percentage of customers who are manufacturers
34	PercentCOM	percentage of customers who are companies
35	PercentIND	percentage of customers who are industrial
36	PercentNA	percentage of customers who are (not available)
37	PercentOTH	percentage of customers who are (other)
38	PercentSFR	percentage of customers who are single-family residences

Notice that power utilization by a parcel depends on the size of the parcel (square meters). Taking logs of these quantities results in better models using least squares.

(a) Use least squares to fit the model

$$(\text{logPowerJul11} \;-\; \text{logSqMeters}) \;\sim\; f(\,\text{AvgTempJul11}\,)$$

where f is a cubic polynomial.

(b) Plot AvgTempJan11 as x, and (logPowerJan11 $-$ logSqMeters) as y. Then plot your fit superimposed on the data. Is there a relationship between average temperature and average power utilization (per sq. meter)?

(c) Use least squares to fit this linear model to the data:

$$\text{logPowerJul11} \sim c_0 + c_1\,\text{AvgTempJul11} + c_2\,\text{AvgYearBuilt} + c_3\,\text{logAvgIncome} + c_4\,\text{logSqMeters}.$$

Also determine the R^2 value.

(d) For parcels that are more than 50% residential, construct a matrix `Data` with the following variables: (logPowerJul11 − logSqMeters), AvgTempJul11, Longitude, Latitude, logAvgIncome, AvgYearBuilt, PercentSFR. Use `plotmatrix(Data)` to get scatter plots of these variables against each other.

(e) Display the correlation matrix for `Data`. Perform PCA on the correlation matrix for derive the first 3 PCs (principal components), and interpret their loadings (eigenvector entries).

(f) **Extra Credit**: Find something interesting in the data, such as extreme outliers (like parcels with very high power utilization). `CCSC` may be interested in pursuing it.

5. **Eigenfaces (20 points)** The course notes include a chapter on Eigenfaces. For this assignment you can use the files in the directory `eigenfaces/` in the homework zip file, which include Matlab scripts and a database of 177 face images, each a `.bmp` bitmap file of size 64×64 pixels. The face images have been pre-processed so that the background and hair are removed and the faces have similar lighting conditions.

Figure 1: Sample 64×64 face image

If we reshape each face image as a 1×64^2 row vector, and collect them into a matrix, then the principal components of the matrix define the main dimensions of variance in the faces. These principal components are called *eigenfaces*. The course notes, and specifically the script `eigenfaces/eigenfaces.m` in this directory, show how to do this.

Using Matlab, perform the following steps:

(a) Modify the script `eigenfaces/eigenfaces.m` so as to create a function that takes as input a string array of filenames of face images, an integer k, and an integer sample size s — and yields the average face and the first k singular values and eigenfaces as output values for a sample of size s.

(b) Use your function and the information in file `eigenfaces/face_descriptions.m` (and omitting the images it suggests), to compute the first $k = 30$ eigenfaces for: (1) Females with Blue eyes, (2) Males with Blue eyes.
Print the top k singular values for each.
Render the mean Female face $\bar{\mathbf{f}}_0$ and the mean Male face $\bar{\mathbf{f}}_1$.
Also render (the absolute value of) $\bar{\mathbf{f}}_0 - \bar{\mathbf{f}}_1$, the difference of the mean Female and mean Male face having Blue eyes.

(c) For faces having Blue eyes, find the *most extreme male face*, and the *most extreme female face*. That is, if $\mathbf{c}$ is the vector of 30 coefficients for a face, find the face whose value of $\|\mathbf{c}\|$ is largest.

(d) Develop a Male/Female face classifier using the eigenfaces you've computed: write a function that, when given as input the filename of a face image, yields the output value 0 if the face is Male and 1 if the face is Female.
To determine whether a face $\mathbf{f}$ is Female, for example, a possible classification scheme would be to determine whether it is closer to the average Female face $\bar{\mathbf{f}}_0$ than it is to the average Male face $\bar{\mathbf{f}}_1$.
Implement your own classifier using a method of your own design. Then, using a Matlab program, determine for each of the Blue-eyed images whether your classifier agrees with its Male/Female description in `eigenfaces/face_descriptions.m`, and determine its accuracy (percentage of correct classifications).