Math 285 HW2, Fall 2015.

Instructions:

- Due date: **Tuesday, October 20, in class**.
- Please type your homework in Word or Latex.
- All the Matlab scripts and data needed by this homework can be found on the course website: `http://www.math.sjsu.edu/~gchen/math285.html`.
- For each programming question, submit both the Matlab script and the output.

Answer the following questions.

(1) Implement MDS in Matlab (although Matlab already has coded MDS in a function called *cmdscale.m*, I still would like you to do it by yourself in order to fully understand the steps of MDS). It's preferable to write a function for MDS with input being distances and dimension of target Euclidean space and output being the low dimensional coordinates and the Kruskal stress score:

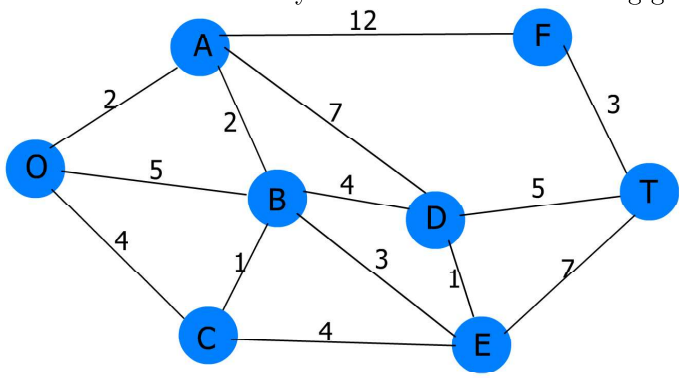
$$\text{function } [\mathbf{Y}, \text{stress}] = \text{mds}(\mathbf{X}, k).$$

Afterwards, do the following:

- Apply your function to a data set (called *ChineseCityData.mat*) that contains the mutual distances of 12 Chinese cities to produce a two-dimensional map (for clarity let's place the City of Urumqi in the top left corner). How good is your map?
- Suppose you had airline distances for 50 cities around the world. Could you use these distances to construct a world map?

(2) Download the ISOmap code from the course website (note that the code provided on the ISOmap website has an error; it has been fixed in the version I provide). Also, find a real data set from the Internet (anything but those already listed on the ISOmap website) to which it makes sense to apply ISOmap (be sure to describe your data clearly but briefly). Perform ISOmap on your data set and interpret the low dimensional representation you got.

(3) Use Dijkstra's algorithm to calculate, by hand, the shortest distance from the node O to every other node in the following graph:



(4) This question concerns Kernel PCA with the Gaussian kernel, also called the Radial Basis Function (RBF) kernel:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}}$$

and aims to help you understand what the combined algorithm does:

- The value $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ represents dot product between the images of $\mathbf{x}_i, \mathbf{x}_j$ in some infinite-dimensional feature space $\mathcal{F}$;
- Each data point $\mathbf{x}_i$ is mapped to a unit vector in $\mathcal{F}$ (as $\kappa(\mathbf{x}_i, \mathbf{x}_i) = 1$);
- If two points $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$ are spatially "close" under the Euclidean distance, then their feature vectors $\phi_i, \phi_j \in \mathcal{F}$ will have a small angle;
- If two points $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$ are spatially "far" from each other, then their feature vectors $\phi_i, \phi_j \in \mathcal{F}$ will have an angle close to 90 degrees;
- The scale parameter $\sigma > 0$ defines how far is "far" and how close is "close". It is normally chosen to be the average distance between each point and its kth nearest neighbor in the data set (say $k = 8$).

Overall, Kernel PCA maps nearby points to unit feature vectors with small angles and faraway points into orthogonal directions and applies PCA in the feature space $\mathcal{F}$, thus preserving all (and only) local geometry. It does not depend on the shape of the data set and thus is a general-purpose kernel. For more reference, you should read the first two papers listed under Kernel PCA on the course website.

Now, implement Kernel PCA in Matlab as a function and apply it to the data in *kernelpca_data.mat*. Display the two dimensional representation of the data obtained by Kernel PCA. What do you find?

(5) The Iris data set in the University of California, Irvine (UCI) Machine Learning Repository (`http://archive.ics.uci.edu/ml/datasets/Iris`) contains 3 classes of 50 instances each, where each class refers to a type of iris plant. First, download this data set to your computer and use the file *script_read_irisdata.m* to read it into Matlab. Afterwards, perform the following tasks:

- Apply kmeans with 10 restarts to the iris data set to divide it into three groups. What is the error percentage of your clustering? Is it good and why?
- Now, suppose we do not know how many classes there are and would like to estimate it by kmeans with $k = 1, \ldots, 6$, each with 10 restarts. Plot the scatter versus k. How many clusters does the plot indicate?