**Step 1 - Loading USPS digit data**

1.1. Get the file digits.mat from the folder, and load this file into your workspace. List the variables in your workspace.

1.2. The variable X contains 1000 images of handwritten digits from 0-9 collected by the United States Postal Service. Each digit is represented as a 64x64 pixel black and white image. However, each image has been vectorized, meaning each row of the images has been concatenated together to make a flat vector representation of the image. Get the 5th image in X as a vector and store that image vector in a variable called v. v should be of length 256x1 Do not display the output.

1.3. Use the reshape command to make v into a 16x16 matrix, and plot this matrix as an image in figure(1) in black and white.

**Step 2 - Dimensionality reduction**

Currently each image can be thought of as a single point in a 256 dimensional space, where each dimension corresponds to the intensity of the image at that single pixel location. If we want to plot all of these 1000 points using only 2 dimensions, so that images that are similar are placed next to each other, we need to reduce the dimensionality of the data from 1000 to 2 without losing a significant amount of information.

2.1. Let's start by computing the similarity between all of the 1000 images in X. The vectorized representation of X makes it simple to do this. We can use the matrix inner product between two image vectors as a measure of similarity. What is the inner product?

Generalization of dot product. Results in a scalar, based on multiplication of vectors. (What does the inner product actually mean?)

2.2. Use matrix multiplcation to compute the inner product of X's transpose and X, so that you get a matrix A which is of size 1000x1000. A can be thought of as an affinity matrix. Each value of A corresponds to computing the inner product between two vectors in X, and also represents the similarity between those two vectors; big values in A correspond to a pair of images that are very similar. You should be able to compute A with one single matrix multiply in one line.

2.3. Plot A as an image, use the jet colormap

This plot doesn't look right to me...Its solid red?

2.4. Principal Component Analysis (PCA) allows us to take this high-dimensional data: 1000 points in 256 dimensions, and reduce the dimensionality while preserving as much information as possible. Let's start by computing the eigenvalues and eigenvectors of A, call the eigenvalues D, and the eigenvectors V.

2.5. Set d equal to the diagonal of D, then show a bar plot of the last 10 elements of d. These are the 10 biggest eigenvalues of A.

2.6. Each column of V is an Eigenvector of the matrix A, meaning that if we want to plot the data using only 2 dimensions, the best 2 dimensions to choose correspond to these last 2 eigenvectors (the ones with the biggest eigenvalues). Store the last column of V in the variable x, and the second to last column of V in the variable Y

2.6. Now we have 2 numbers to describe every image instead of 256, so we can plot the data in a 2D plane with one point per digit image by using the variables x and y. Display a scatter plot of x,y. Set the scatter plot so each dot is filled in, and not empty.

**Step 3 - Clustering**

3.1. We know that the data has 10 clusters, each cluster corresponding to a digit between 0-9. The goal of clustering is to give each point a label such that all the points with a certain label are more similar then points with different labels. We can use matlab's built-in clustering tool called K-means. Apply K-means to X transpose, to get a list called C, where C labels each point in X with a value from 1 to numClusters representing which cluster that point is in. numClusters should equal 10.

3.2. Plot a new scatter plot, where each point has coordinates in x and y computed from PCA above, has size 50, and has the color determined from C. Use the colormap jet. You should be able to see how different clusters of points are grouped in similar areas on the map. Mark each color by the corresponded digit.