CS294-34 Homework 2

The homework is due on October 8. Please submit a PDF on bSpace. If you have questions on the clustering questions, email Sriram; for the dimensionality reduction ones, email Percy.

This assignment will be done entirely in R. You will need the following packages: stats, mclust, MASS, pixmap.

Problem 1: K-means In this problem, you are asked to do some K-means experiments on simulated data and deal with some practical issues, such as how to choose K and whether to do standardization or not before running the algorithm.

(a) Simulate 2-dimensional data from four clusters, each of which is specified by a Gaussian distribution with the same covariance matrix and different means. In function generateFourClusters, fill out the code to plot K (ranging from 2 to 6) against within cluster sum of squares. Choose an appropriate K from the plot and argue why do you choose this particular K. Run function Kmeans with chosen K and plot the clustering result. Write down how many points are in wrong clusters. Don't forget to record the sample covariance matrix of the simulated data at this point.

(b) Standardization means making the data centered and spherical (identity matrix as its sample covariance matrix). In function standardization, fill out the code to standardize the data. Now plot K against within cluster sum of squares on standardized data, choose an appropriate K from the plot. Is the K you choose the same as the one you chose in part(a)? Run function Kmeans with two chosen Ks (if they differ) on standardized data. Compare the clustering results with the one you obtained in part(a).

(c) Explain why standardization in this setting does not work.

Problem 2: Hierarchical Clustering

(a) Find a simple artificial example that single linkage hierarchical clustering produces extended clusters while complete linkage yields round clusters.

(b) Download dataset "election.data" from the course website. This is the dataset of presidential votes by county in California(2004). We'd like to cluster counties such that counties with similar voting behavior should be more likely to be in the same cluster. Run hclust on the data using average linkage. Explain why Los Angeles is so different from other counties and why the dendrogram looks like one using single linkage.

(c) Now we normalize the two columns to avoid population influence. Fill out the code in election Clustering (there are only two lines). Contrast the dendrogram to the one obtained in part (a). Try single linkage and complete linkage as well.

Problem 3: PCA for visualization

(a) Complete `p4a()` in `dimension.R` to do the following. Load the voting dataset and run PCA. (See `house-votes-84.names` for a description.) How quickly or slowly does the variance captured by PCA fall off?

Project the data points onto the first 2 principal components. Scatter plot the data points, with a different marker for the Republicans and the Democrats. Are the two classes well separated? What technique could we use to get better separation?

(b) In (a), we used PCA to plot the *data points* in two dimensions. We can also use PCA to plot the *variables* (features, attributes) in two dimensions by simply running PCA on $X^\top$, which will allow us to visualize which variables are correlated with which others. Optional: look in `house-votes-84.names` to see what the votes are and comment on whether the geometric layout of the variables matches your political intuition. (Note that the eigenvalues are the same in both (a) and (b).)

(c) Choose any dataset from your research. Briefly explain what the data points and variables (features) correspond to. Run PCA and visualize the data as in (a) and (b). Comment on whether PCA provides any insight into the data or not. If you do not have a dataset readily available, just choose one from the UCI machine learning repository (`http://archive.ics.uci.edu/ml/datasets.html`).

Problem 4: PCA for classification In this problem, we will run PCA on a face dataset and apply it to face recognition. First, call the function `loadAllFaces()` to read in both the training and test images. The images will be stored in `trainPos`, `trainNeg`, `testPos`, and `testNeg`, which correspond to the positive and negative examples for both training and test sets. You can draw a face by calling, for example, `plotImg(trainPos[,4])`.

(a) Modify `reconstruct()` to do the following. Run PCA on the positive training examples (`trainPos`). Take one of these images vectors x. Project x onto the first K principal components, and then try to reconstruct the image given the projection. You can call `plotImg()` to see what the reconstruction x' looks like and also compute the reconstruction error $||x - x'||^2$ (squared distance between the vectors). Start with $K = 1$ and incrementally increase K. Observe both how quickly (or slowly) the reconstructed image converges back to the original image, both qualitatively and quantitatively. What is the reconstruction error on `trainPos[,4]` when $K = 5$? $K = 50$? Does the error guaranteed to reach 0?

(b) Now, pick a positive *test* example and do the same as in (a). Does the reconstruction require more principal components to reach the same fixed error threshold? Does the error eventually reach 0?

(c) Now let's used the PCA representation for nearest-neighbors classification of face versus non-face. Modify `pcaClassification()` to do the following. Starting with $K = 1$, plot the test error as K increases. At what value of K does the test error stop decreasing?