Good job!

100/100

Duttaabhinivesh Devathi
MATH 444
Assignment 2
February 24, 2017

Problem 1

Summary:

The problem gives a data set $X \in \mathbb{R}^{4x150}$ which describes a set of 150 data points, each of which has four characteristics of the *Iris* genus of flowers, which are arbitrary. Also, another given is that the data forms 3 clusters, around 3 different species of the *Iris* genus. And the data set also gives us the cluster (each species of flower) to which each data point (column) in X belongs to.

However, we are asked to write a k-means and k-medoids algorithm to cluster the data and check whether the determined clusters match the given clusters of the data. All data for this problem is found in `IrisDataAnnotated.mat,` which is a matlab file containing X, the data and $I \in \mathbb{R}^{1x150}$, the index of the cluster to which each data point belongs to. In the problem, it is given that the index of the clusters are as follows:

$$\begin{aligned} 1 &= \textit{Iris setosa}, \\ 2 &= \textit{Iris versacolor}, \\ 3 &= \textit{Iris virginica}. \end{aligned}$$

a) Each k-means and k-medoids algorithms were written as functions in matlab, and located in `kmeans.m` and `kmedoids.m.` Each of these functions give two outputs, the first being the final mean or medoid data point for each cluster given. Further investigation can be done by examining the m-files.

b) Both k-means and k-medoids algorithms were run for the data set, for k = 3 because we know that there are three clusters. The three clusters as defined looks like this. **Figure 1** shows a comparison between the actual clustering and the clustering defined by the different k-means and k-medoids algorithm.

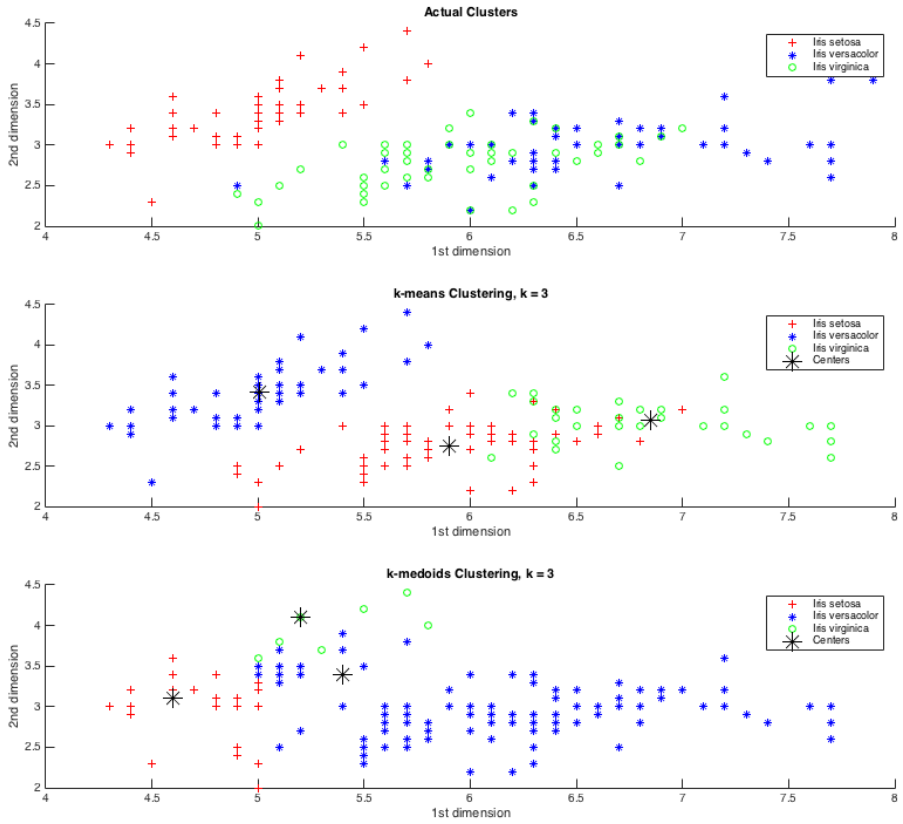

Figure 1: A clustering of Iris Data compared with the actual clustering. Centroids were initialized at random. The black stars denote the final centroids in each algorithm. The legend means nothing for k-means and k-medoids because the assignment of each species depends on initalization of candidate vectors, which was chosen at random.

In the k-means clustering, identifying the correct species itself happens at random. Upon re-running the code to make these graphs, one can see that the identification of the clusters change, and depend on the initialization of the mean vector or medoid. However, the clusters themselves are relatively the same. The data was projected onto two dimensions, the 1st and 2nd dimensions of the data, because they showed the clearest picture of the three clusters. Upon inspection, you can see that *Iris versacolor* and *Iris virginica* have a lot of overlapping data points, which makes it difficult to separate these two into separate clusters. This is apparent in *Figure 1*, which shows the k-medoids clustering to combine both *versacolor* and *virginica* species as the same cluster and tried to separate *setosa* into two clusters. However, in the k-means algorithm, the clusters were separated accurately according to the actual clustering. The only issues occur in the overlapping data points between *versacolor* and *virginica*.

The same test was run more times and similar results were shown, with the labels of the clusters being randomized in the k-means and the k-medoids algorithm. Because of this, it is not worthwhile to count the misaligned data points as it changes without pattern every time you test the algorithms. Also, because the assignments of each cluster are random, counting misaligned data points is not worthwhile.

Problem 2

Summary:

This problem is like the last one, except we are given data in 9-dimensions, and therefore visualizing the data is not a useful task. Instead, the goal of this exercise is to determine the effectiveness of the k-medoids algorithm to separate the data into two clusters - malign tumors vs benign tumors – given a set of 9-dimensional data. To do this, we run the k-medoids algorithm for k = 2 and then determine the specificity and sensitivity for each set of data, which is essentially the percentage of correctly identified data points in each cluster. To run this algorithm, run `problem2.m` to run the algorithm for k-medoids.

The first step to this problem was to remove any columns with NaN entries, however the data given had already removed the NaN columns, so this was not necessary.

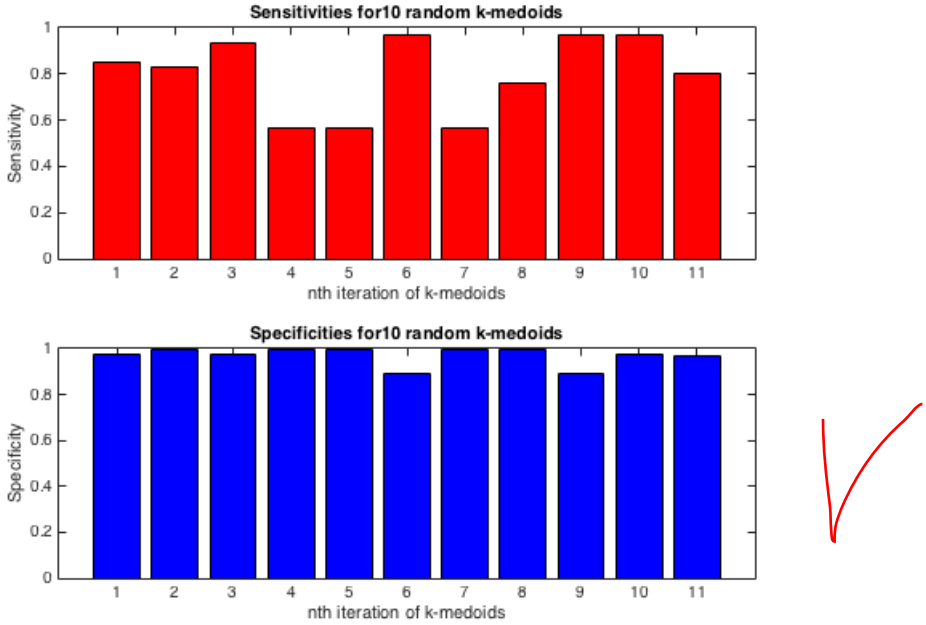

Figure 2: Sensitivities and Specificities for 10 iterations of k-medoids algorithm for k = 2. The 11th bar represents the average of the previous 10 Sensitivity and Specificity values.

Figure 2 shows the sensitivity and specificity values for malign tumors and benign tumors respectively. Bar 11 from each graph represents the average of the iterations of k-medoids algorithms, and it is clear from the data that identifying a benign tumor is much more reliable, with a true negative rate very close to 1. What this implies is that classifying a malignant tumor is more difficult than classifying a benign one, meaning there could be more false positives. Also, the data was partitioned randomly which leads to a conclusion that the k-medoids.

Problem 3

Summary:

This problem gives us a data matrix in $X \in \mathbb{R}^{16x435}$, which is a set of congressional voting data for 16 different ballots of each of 435 representatives in congress in 1984. Also, provided is an index vector which denotes which party each representative was in (democratic or republican). In each column vector $x^{(j)}$, a "1" indicates a yes vote, a "-1" indicates a no vote and a "0" indicates a missing vote.

The first objective was to investigate the distance matrix of the senators voting records, to see how dissimilar each senator was to one another, and a distance matrix D was created with the following code:

```
[n,p] = size(X);
D = zeros(p,p);
for i = 1:p
    for j = 1:p
        xi = X(:,i); xj = X(:,j);
        n_yesno = sum(xi == 1 & xj == -1);
        n_noyes = sum(xi == -1 & xj == 1);
        n_nono = sum(xi == -1 & xj == 1);
        n_yesyes = sum(xi == 1 & xj == 1);
        dij = (n_yesno + n_noyes)/(n_yesno + n_noyes + n_yesyes + n_nono);
        D(i,j) = dij;
    end
end
```

The dissimilarity index in this case did not consider votes with missing data.

Running a k-medoids algorithm on this data set, we clustered the voting data into two groups and checked against a given set of indices (indicating party of each congressman) whether the party was correctly clustered or not with this algorithm.

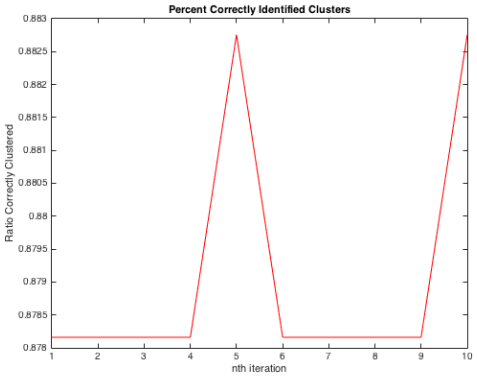

Figure 3: Shows Ratio of Congressmen who was correctly identified into their respective party at each iteration of the k-medoids algorithm.

Upon investigation of this graph, we see that the ratio stays bounded between .878 and .883, which means approximately 88% of congressmen voted along party lines, whereas 12% did not. This is a startling result from this algorithm because it shows exactly how divided congressional voting can be. All graphs and charts were created in `problem3.m`, which is attached.