1

Within the K-medoids framework, you have several choices for detailed implementation. Explain how you designed and implemented details of your K-medoids algorithm, including (but not limited to) how you chose representatives of each cluster, what distance measures you tried and chose one, or when you stopped iteration.

- **For the representative of each cluster:**
  1. I first initiate K medoids by randomly choose K data points.
  2. Calculate the Manhattan distance between each data point and K medoids, sign each data point to its closest medoid.
  3. Find the geometric center for each cluster.
  4. Find the nearest data point to the geometric center of that cluster, and sign it as the new medoid.

  Distance measures:
  I used Manhattan distance for the final algorithm. I also tried euclidean, Minkowski, squared Euclidean and cosine distance. Except cosine distance, all other distance measures worked similarly.

  when you stopped iteration:
  I set the max iteration at 300. It normally stopped after 4 to 6 iteration.

- Attach a picture of your own. We recommend size of 320 240 or smaller.
  see fold for picture

- Run your K-medoids implementation with the picture you chose above, with several different K. (e.g, small values like 2 or 3, large values like 16 or 32) What did you observe with different K? How long does it take to converge for each K?

  For small K values: The quality of the picture was bad. It only took a few iteration (3 to 6) to converge.
  For large K values: The quality of the picture was good. It took much larger iteration (20 to 50) to converge

- Run your K-medoids implementation with different initial centroids/representatives. Does it affect final result? Do you see same or different result for each trial with different initial assignments? (We usually randomize initial location of centroids in general. To answer this question, an intentional poor assignment may be useful.)

Figure 1: small K values

Figure 2: large K values

Yes, different initial centroids/representative do affect the final result and time took to converge. I randomize the initial centroids, and count how many times of iteration it took to converge. I saw that each time the iteration was different. Also sometimes, I could see noticeable difference in the image qualities between different runs.

- Repeat question 2 and 3 with K-means. Do you see significant difference between K-medoids and K-means, in terms of output quality, robustness, or running time?

  For small K values: The quality of the picture was bad. It only took around 7-9 iteration to converge, which is still larger than K-mediods.
  For large K values: The quality of the picture was good. It took way much larger iterations (more than 100, sometimes it even max out after 300) to converge.

  Personally, I feel if K is small, the output qualities and robustness of both algorithms are not stable. K-medoids performs slightly better in terms of running time. However when K gets larger, the output quality and robustness of both algorithms are comparable. K-mediods beats K-means in running time by large lead.

2

- Consider an undirected graph with non-negative edge weights wij and graph Laplacian L. Suppose there are m connected components A1, A2, . . . , Am in the graph. Show that there are m eigenvectors of L corresponding to eigenvalue zero, and the indicator vectors of these components IA1 , . . . , IAm span the zero eigenspace.

- Real data: political blogs dataset. We will study a political blogs dataset first compiled for the paper Lada A. Adamic and Natalie Glance, The political blogosphere and the 2004 US Election, in Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem (2005). The dataset nodes.txt contains a graph with n = 1490 vertices (nodes) corresponding to political blogs. Each vertex has a 0-1 label (in the 3rd column) corresponding to the political orientation of that blog. We will consider this as the true label and try to reconstruct the true label from the graph using the spectral clustering on the graph. The dataset edges.txt contains edges between the vertices.
  Here we assume the number of clusters to be estimated is k = 2. Using spectral clustering to find the 2 clusters. Compare the clustering results with the true labels. What is the false classification rate (the percentage of nodes that are classified incorrectly).

  First I check the true label, I found there are 2 data points with label as NaN. It means that we have to ignore those two data points in our final analysis.

  After perform Spectral Clustering with k=2, I found that the false classification rate is **0.49**.

3

The data food-consumption.csv contains 16 countries in the European area and their consumption for 20 food items, such as tea, jam, coffee, yoghurt, and others. There are some missing data entries: you may remove the rows Sweden, Finland, and Spain. Implement PCA by writing your own code

- Find the first two principal directions, and plot them.

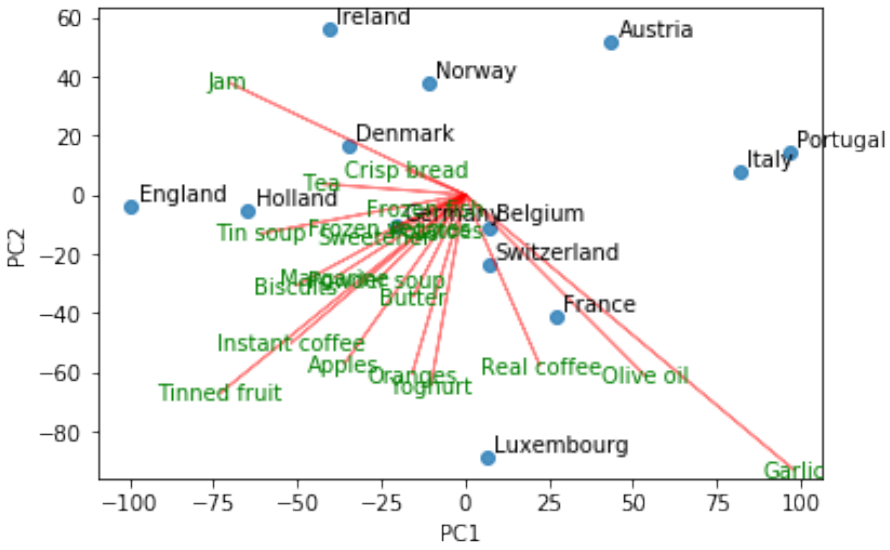

Figure 3: plot first two principle direction

We can see from Fig 3. that red lines show the projections of the original variables (food) on to the principal components.

- Compute the reduced representation of the data point (which are sometimes called the principal components of the data). Draw a scatter plot of two-dimensional reduced representation for each country. Do you observe some pattern?
  As shown in Fig 4., I performed Kmeans on the PCA data with k=3. It looks like that Italy, Austria and Portugal are very similar in their diet. They form a group. France, Swiss, Belgium and Germany closely cluster together, they are from western Europe. At last, countries from northern (sort of) Europe cluster together.

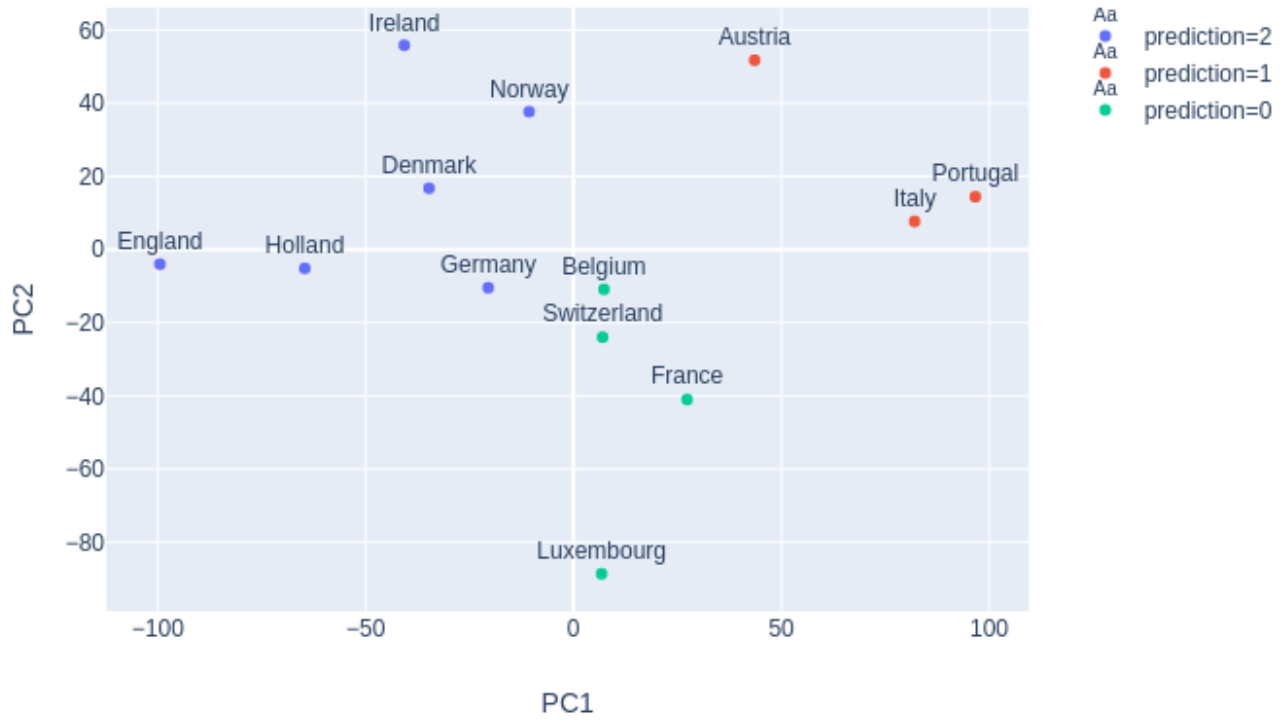

Figure 4: PCA with PC1 and PC2