HW7 K-means Clustering

In this exercise we will explore hard clustering with K-means Clustering. Hard clustering means that each case must be committed to one and only one cluster.

We will explore that same datasets as we did before, hand-written digits. The MNIST dataset is comprised of 28x28 pixel images of hand-written digits from the U.S. Census Bureau. Each instance is made up of 784 features (28 x 28 pixel images), each representing a pixel color value between 0 and 1. There are 10 classes to cluster, digits 0-9.

Once again, a good deal of the work has been done for you. You will have to write a few of the core pieces of the k-means algorithm. These are identified in the comments of the code (kmeans.py) as "% *TODO*". Specifically you will implement (1) the "furthest" initialization method; (2) assign the points to the nearest cluster; and (3) re-compute the centroids (means) once all the points have been assigned.

Recall that the k-mean algorithm is:

1) Initialize cluster centers (centroids)

2) Assign each data point to the closest cluster (centroid)

3) Re-compute the centers (centroids) as the means of the assigned cluster data points

4) If data point assignments have changed, go to (2)

Farthest Distance Initialization:

1) First select a random data point as the first centroid c_1
2) Then select a second data point as far away as possible from c_1. Call this c_2.
3) Then all subsequent centroids as a data point, again, as far away as possible from $c_1, c_2, \ldots c_k$. That is to say $c_{k+1} = argmax(\min[dist(x_i, c_1), dist(x_i, c_2), \ldots dist(x_i, c_k),])$

Recall we will use the Squared Euclidean Distance as our distance measure

$$Squared\ Euclidian\ Distance\ d(x,\ \hat{x}) = \sum_{i=1}^{d} (x_i - \hat{x}_i)^2$$

Remember that the centroid is the mean across all data points assigned to cluster k over columns. So the mean should have the same dimensionality as any other data point, in our case $\mathbb{R}^{784}$

HW7 K-means Clustering

Run.py – This module is your starting point. It will run your kmeans code and produce all the necessary graphs based on your kmeans implementation. There is nothing that needs to be done in this module.

DrawDigits.py – This is a utility module to draw digit graphs. Everything is complete with this module.

kmeans.py – This module will be your kmeans implementation. There are three *"%TODOs"* that you will have to implement. Specifically you will implement (1) the "furthest" initialization method; (2) assign the points to the nearest cluster; and (3) re-compute the centroids (means) once all the points have been assigned. See the notes above as well as the comments in this module.

<u>Analysis Questions</u>

1) Several graphs will be produced which represent the centroids of each cluster. Which cluster size k is the optimal cluster size and why. (Hint: The why should be explained in terms of normalized AIC and/or normalized BIC scores).

2) (a) What are the AIC and BIC test statistics; (b) what do they statistically represent; (c) is a lower or higher score better; and (d) which is the best test statistic for determining optimal model parameters like cluster size.

3) When using furthest initialization, will multiple runs using the same data points produce the same or different cluster assignments? Explain why or why not?

4) In the kmeans code a score is being computed. Geometrically speaking what does this score represent? That is, what do you think a cluster looks like with a high score versus a low score?

5) We essentially use the Euclidean Distance to determine what data points belong to what clusters. What geometric shape do you think the clusters take on using this distance metric?

6) Provide at least two reasons why we bother to perform clustering?

What to Hand in

You should hand in a *.pdf containing ALL OF THE GRAPHS that were generated and the answers to the analysis questions. Also hand in your code. Zip this all up.