Final Project

Name: Ramya Nagapudi
netid: `ramyasn2`

April 29, 2019

To improve my classifier I manipulated the d value parameter, as well as the k value parameter. The d value represents how many times I am slicing my training set of signals. In this case I will be slicing it by 30 samples. There are also three dimensions to each sample, so my d value would be 30 times 3, or 90. So I will be slicing each signal into 90 pieces. My strategy behind choosing this d value was understanding what a reasonable range would be for splitting my signals, as well as trying different d values in this range until I find one that consistently seems to work well with my classifier. Based on the information given that each signal is sampled at 32 Hz, I decided that the thirties seem like a good range to divide the data. A further justification of picking a value in this range is that the thirties are the right amount for the data to provide something interesting about the representation, but is not too long to where the representation will be altered if the boundaries of these samples are divided inaccurately. After trying values from 30 to 39 I noticed that 30 worked best with my classifier. However my d value is not 30, but 3 times 30, since we know that each signal is three dimensions. This is how I derived my d value as 90. The k value represents the number of cluster centers. One strategy to choose a good k value is clustering for many different values of k and then determining the value of the cost function for each k value used. I can do this by performing the *inertia* function on the kMeans, which will represent the sum of squared distances of samples to their closest cluster center. This means that the smaller the inertia the better, and this is possible when the number of k clusters increases, as seen in the inertia plot. So we can assume that when the number of clusters is equivalent to the number of the sample size inertia would be zero, since every cluster center is simply the sample itself, meaning there is no error. This is exactly why although this was a good starting point to improve my classifier, this strategy in itself was not of much use to me. Hence I used a different strategy which involved plotting these values as a function of k and then determining the knee of the curve, so basically making an elbow curve. So in my range of reasonable k values I map each k value and its relation to its inertia value, then identify the elbow of this curved plot. I made my range k be from 20 to 200 since anything lower or higher seemed as an unreasonable amount of cluster centers. The elbow of this curve lied in the range from 90 to 95 so I tried each k in this range as my parameter for my classifier until I recognized that when k is 90 my classifier has the best results. Hence this is the k value I used to improve my classifier. Manipulating both the d and k values to be d=90 and k=90 made my classifier perform well because these values were both just good enough to where they provide great information about my data, while not creating issues with inaccurate data divisions. They also work well together in the sense that 90 was a good amount of slicing to provide a good amount of pieces to lie in each cluster center, providing a good data visualization and eventually classification. Overall manipulating these two values and recognizing their contributions to my classifier as a whole has ultimately allowed me to better my classifier.

D=96,K=480

Accuracy rate: 69.6429% Error rate: 30.3571%

Confusion matrix:

```
array([[ 1,  0,  0,  0,  0,  0,  1,  0,  0,  1,  0,  0,  0],
       [ 0, 15,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  7],
       [ 0,  0,  4,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  0,  0,  5,  0,  0,  0,  0,  0,  0,  0,  0,  1],
       [ 0,  1,  0,  0, 20,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  0,  0,  1,  0, 10,  2,  0,  2,  4,  0,  0],
       [ 0,  0,  0,  0,  0,  1,  3,  0,  0,  1,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0, 17,  1,  0,  0,  0],
       [ 0,  1,  0,  0,  0,  0,  2,  0,  0, 14,  2,  0,  0],
       [ 0,  2,  0,  0,  0,  0,  2,  0,  2,  4, 11,  0,  0],
       [ 0,  0,  0,  0,  3,  0,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  2,  0,  0,  0,  0,  2,  0,  0,  0,  0,  0, 19]])
```

D=90,K=90

Accuracy rate: 77.3810% Error rate: 22.6190%

Confusion matrix:

```
array([[ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 13,  0,  3,  0,  0,  1,  2,  0,  0,  1,  0,  2],
       [ 0,  0,  9,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  0,  8,  0,  0,  0,  0,  0,  0,  0,  0,  1],
       [ 0,  0,  0,  0, 18,  0,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0],
       [ 0,  2,  0,  0,  0,  1, 13,  1,  0,  1,  3,  0,  1],
       [ 0,  0,  0,  0,  0,  0,  1,  3,  0,  1,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0, 20,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 13,  4,  0,  1],
       [ 0,  1,  0,  0,  0,  0,  3,  0,  0,  0, 14,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,  0],
       [ 0,  1,  0,  0,  0,  0,  2,  0,  0,  0,  0,  0, 16]])
```