Pattern Recognition Assignment No:3
Analysis & Results
18MCMI09
Krishnan.S

1.
A)K-Means is a clustering algorithm. It's used to separate *n* data points or observations into *k* partitions. In clustering, this partitioning is done on the basis of some criteria that gives us the similarity between the set of points. K-Means uses Euclidean distance as it's measure.
The K-Means algorithm based on the Expectation Maximization algorithm and uses the Guassian Mixture Model. All clutsers are assumed to be, or able to be, defined by different normal distributions.

Given a set of observations {x_1, x_2 …… x_n} and the clusters we want are S = {S_1, S_2….S_n}, k-means seeks to reduce the variance in each cluster/set, which is given as

$$\arg\min_{s} = \sum_{i=1}^{k} \sum_{x \in S_i} \| x - \mu_i \|^2$$

The first step of the algorithm is to select the centroids. The Forgy method is to take k random observations and use them as the centroids, which is what I did.

Next we do the following two steps iteratively.
In the **E Step**, we compute the distance of all points from these centroids. The centroids form clusters around themselves of all the points that are closer to them than the other centroids. The assignment mathematically is

$$S_i^{(t)} = \{x_p : \| x_p - m_i^{(t)} \|^2 \le \| x_p - m_j^{(t)} \|^2 \ \forall j, 1 \le j \le k\}$$

I did this using a Euclidean function that I wrote for the first PR assignment. There's an array called clusters in which I stored the cluster to which each point belongs to.

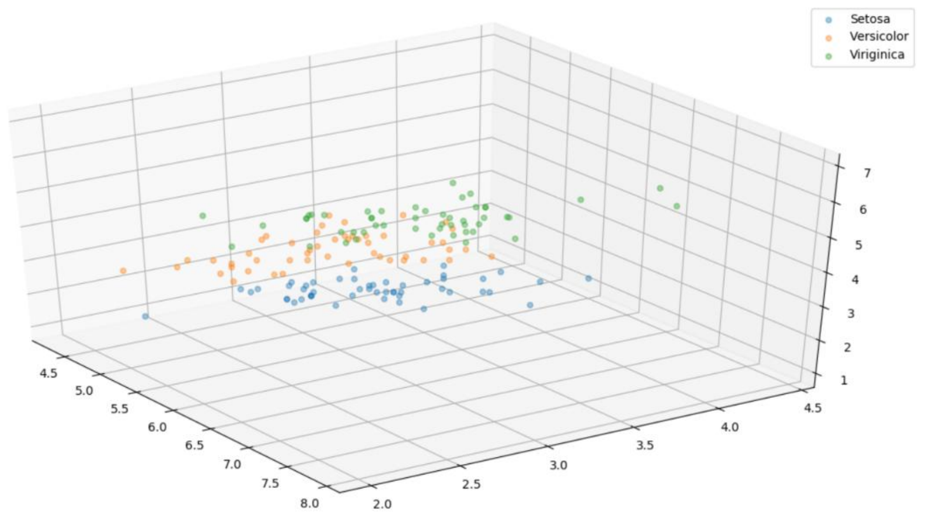
In the **M step**, we compute the new centroids as the mean of all the points in each such cluster.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x \in S_i^{(t)}} x_j$$

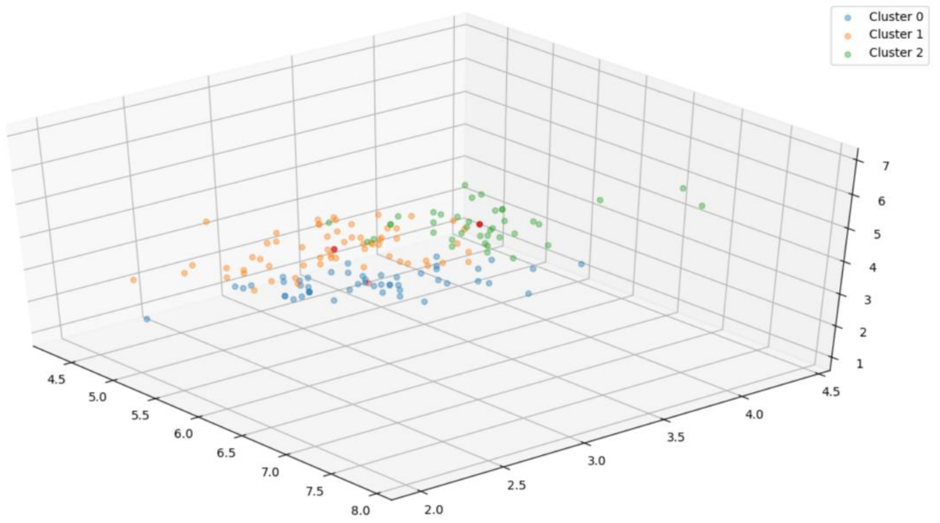
I used the function **np.mean()** for this.

We repeat the process until the centroids don't change any more or if we've done a certain number of iterations. I took 50 just in case, but both data sets converged before 10 iterations.

B) For the Iris Dataset I took *k* to be 3 since there are three classes/clusters present. The degree to how well the clustering is done is mentioned in the answer to the next question.
Below is the plot for the dataset with the class labels (no clustering done yet). The first three features were used (sepal length and width, and petal length), with all 150 points.

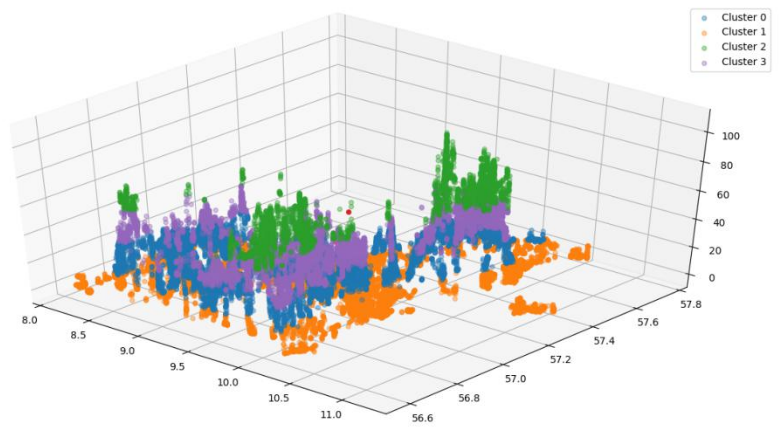


Below is the plot after clustering was performed on the dataset with random points chosen initially.

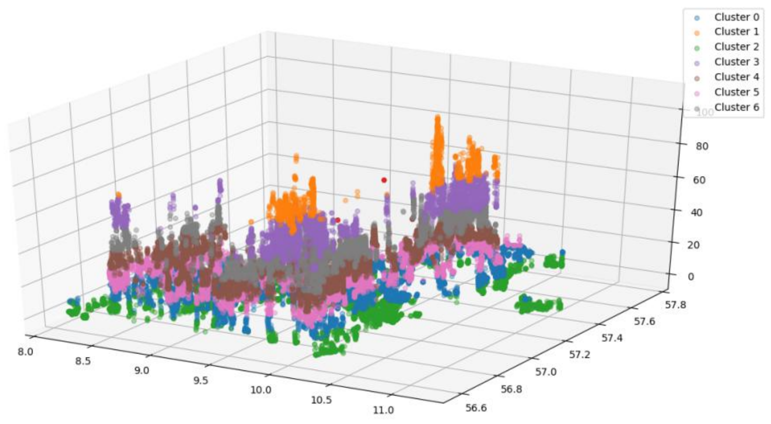


The red points are where the means of each cluster are. The clusters look very similar to how the points are actually distributed in each class. Cluster 0 corresponds to Setosa, 1 to Versicolor and 2 to Virginica. However this isn't exactly correct cause some points in Versicolor and Virginica get mixed up.

C) The data set I picked is the UCI 3D Road Network data set. It was constructed by adding elevation to information from a 2D network data set in North Jutland, Denmark, for eco-routing and estimating CO2 emissions. There are 434874 instances in this data set, with each instance having four features - OSM_ID (OpenStreetMap ID), Longitude, Latitude and Altitude. There are no class labels available so it's with unsupervised learning or for regression.
So since there's no indication of what clusters may be present, I tried it with k = 4 and k = 7. The plots shown below plot the three distance features. I omitted the OSM_ID since it doesn't make sense to use the ID for calculating distance. I only used the first 50,000 points since it would take way too long to run on the entire dataset.



(with k = 4)



(with k = 7)

From the above we can see that clustering has been done according to the **elevation** of the road network. The more clusters we want to find, the more finely is the elevation divided.

2.

Both measures are used to determine how “well” the clustering has been performed. Internal measures evaluate on the basis of the data clustered, usually in terms of intra and inter-cluster distances. External measures evaluate on the basis of some ground truth classification. If the points have some class labels attached to them, then we can try to figure out how much the clusters match the classes.

Internal measures only give you an idea on how well one algorithm performs with respect to another, it may not tell you that the clustering is actually valid. External measures aren’t that useful because if we have class labels then we don’t really need to cluster, expect to see if there are alternative partitions to the data.

Internal Measures:

A) **Silhouette Coefficient :** This measures how similar a data point is to members of it's own cluster with respect to points from the other clusters, specifically the neighbouring/closest cluster. For a data point *i,* the silhouette is given as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

$$b(i) = \min_{i \neq j} \sum_{j \in C_j} d(i, j)$$

a(i) is the average distance to the point i from all the other points in it's own cluster, and b(i) is the smallest average distance to the point I from all the points in other clusters, which ends up being the average distance to the neighbouring/closest cluster.
This value ranges from -1 to 1. A value close to 1 indicates that it fits really well in it's current cluster but poorly with the other clusters and a low value indicates that there may be too many or two few clusters.
The **average** of all these s(i) values gives you the Silhouette Coefficient.

B) **Davies-Bouldin index :** This measure is a ratio of the intra-intercluster distances, which is given as

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

C_i is the centroid of cluster i and σ_i is the average distance of all elements in cluster i to it's centroid. A smaller value of this index is considered better, since the intracluster(numerator term) distance is minimized and the intercluster(denominator term) distance is maximized.

External Measures:

A) **Purity** : It is the extent to which a clusters contain a single class, given by the equation

$$P = \frac{1}{N} \sum_{m \in M} \max_{d \in D} |m \cap d|$$

M is the set of points according to the clustering and D is the set of points according to the class labels. We figure out the number of the most common class present in each cluster, and then sum this over all clusters and divide by the total number of points. That max term in the summation tells us how many common points exist.

B) **F Measure** : The F Measure or F1 Score is a measure of how accurate a test is, or in this case, the classification, in terms of the precision P and the recall R. Precision and recall are given as

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

P is the ration of correct positive results to the total number of positive results, and R is the ratio of correct positive results to the total number of relevant results i.e the points that should've been positive. F measure is the harmonic mean of the above.

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

β is a positive coefficient that determines how much recall affects the F-measure. If it's zero, then recall has no effect on it.
A confusion matrix depicts how many positive and negative results the test gave.

Silhouette coefficeint, Davies-Bouldin index and F-measure were calculated using functions from the **sklearn.metrics** module. Purity was calculated by figuring out the sum of the common points in each cluster. The most common class was found by using **scipy.stats.mode()** on the cluster set and then the number of intersection points is the minimum of the frequency of the mode in the cluster and the class label array.

i) Internal and external measures can be calculated for the **Iris** dataset since we know what the labels are. Taking k = 3 and sampling three points randomly from the dataset as the centroid, the measures obtained were

$$Silhouette = 0.553$$
$$DBScore = 0.662$$
$$Purity = 0.893$$
$$F = 0.891$$

The silhouette score is about halfway to one so the points are clustered fairly well, but the DB score is sort of high, which is not good. However since the range of DB is half as compared to S, we can say that both are somewhat in agreement, that the intra cluster distance is small and the inter cluster distance is large.

Purity and the F1 score is very close to 1, so the K-Means algorithm did a good job in assigning the points to the “right” clusters. When you look at the cluster labels and the class labels, a lot of it is the same. The count of all true vs false classifications are given by the below confusion matrix.

$$C = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 48 & 2 \\ 0 & 14 & 36 \end{bmatrix}$$

The rows denote the true labels (Setosa, Versicolor, Viriginica) and the columns denote the predicted labels. So row one column one denotes the number of points that are actually Setosa and were also predicted to be Setosa. Row one column two denotes the number of points that are actually Setosa but were predicted to be Versicolor and so on.

So, K-Means managed to perfectly cluster all the Setosa data points, but there’s overlap between Versicolor and Viriginica, more so that a lot of Virginica points were mistaken to be Versicolor. This makes sense since in the last assignment, through PCA and LDA, we saw that there’s large overlap in the distribution of the two classes.

ii) External measures cannot be calculated on the **3D Road Network** data set since it doesn’t have any class labels. Only internal measures can be obtained.
The internal measures after clustering with k = 4 were

$$Silhouette = 0.569$$
$$DBScore = 0.534$$

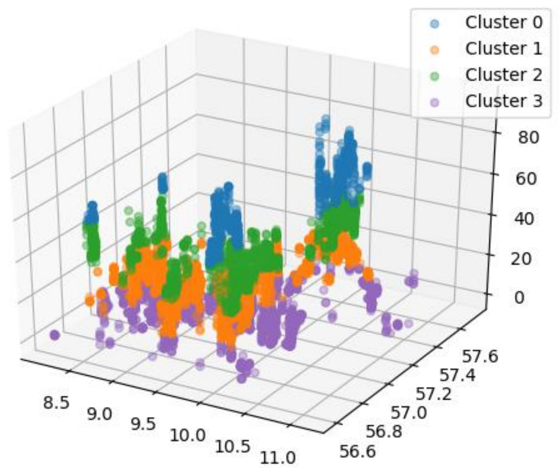
The internal measures after clustering with k = 7 were

$$Silhouette = 0.535$$
$$DBScore = 0.534$$

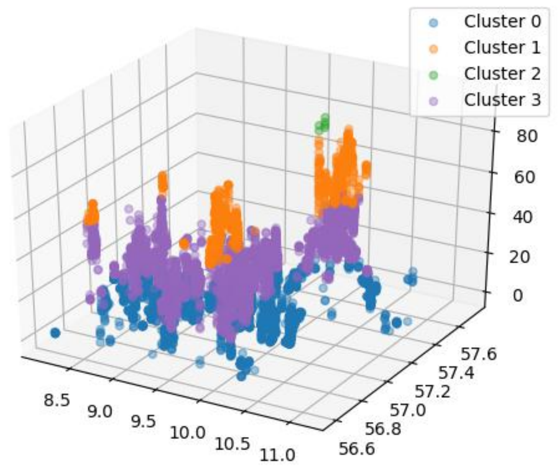
The DB Scores for both are exactly the same, meaning that the intra inter cluster distances are still relatively still the same. The Silhouette score is lower with k = 7, probably because the intra cluster distance got relatively smaller than with k = 4, so the a(i)-b(i) term got smaller. K-Means still clusters this data well according to S score, while it’s average with DB score.

3.
Kernel K-Means is an extension to K-Means where the input is mapped to a higher dimensional space. The distance is measured in this higher space and clusters are formed based on this distance. So it works well with non linear structures. This was implemented using the **SpectralClustering()** function from the **scikit.cluster** module.
This was performed on the 3D Road Network data set with k = 4, since that seemed to give us better internal measures in the second part of the assignment. I took 10,000 points instead of 50,000 this time because the function was crashing my system each time I tried to run it. I ran the K-Means algorithm on these 10,000 points as well just to compare the results.

This is the plot with normal K-Means.



And this is the plot with Kernel K-Means.



The clusters obtained are different, but they still cluster according to the elevation. Kernel K-Means manage to squeeze a lot more points into three of the clusters than normal K-Means, and the green cluster in the Kernel Plot seem to be the most extreme elevations present, which are very few. Purple from I and Blue from II seem to be the same almost. Green and Orange from I bled together into the Purple in II. And a majority of the Blue cluster in I became Orange in II.

Here are the internal measures for K-Means.

$$Silhouette = 0.579$$

$$DBScore = 0.51$$

And these are for Kernel K-Means.

$$Silhouette = 0.549$$

$$DBScore = 0.451$$

Kernel K-Means has a lower Davies-Bouldin index, so it performs better with regards to that. The Silhouette score is lower than that of normal K-Means but not by very much. So overall I would say that Kernel K-Means does a better job of clustering the data.
Since we don't have class labels, we can't compute the external measures, but they would agree with the above statement.