Worksheet 06

Name: Rishab Sudhir

UID: U64819615

Topics

- Kmeans ++
- Hierarchical Clustering

Kmeans ++

a) What is the difference between K means and K means ++?

In short Kmeans++ chooses clusters a bit better

K-means Fartherest First Traversal - make intial centroids as far as possible from each other, however this may make outliers their own clusters

K-means++ takes a part of FFT

K-means++ picks points randomly but with probability proportional to how far you are from other cluster centroids.

a = 2 : K-means++

D(x)^a - a is just an exponent, when it is 2 its Kmeans++

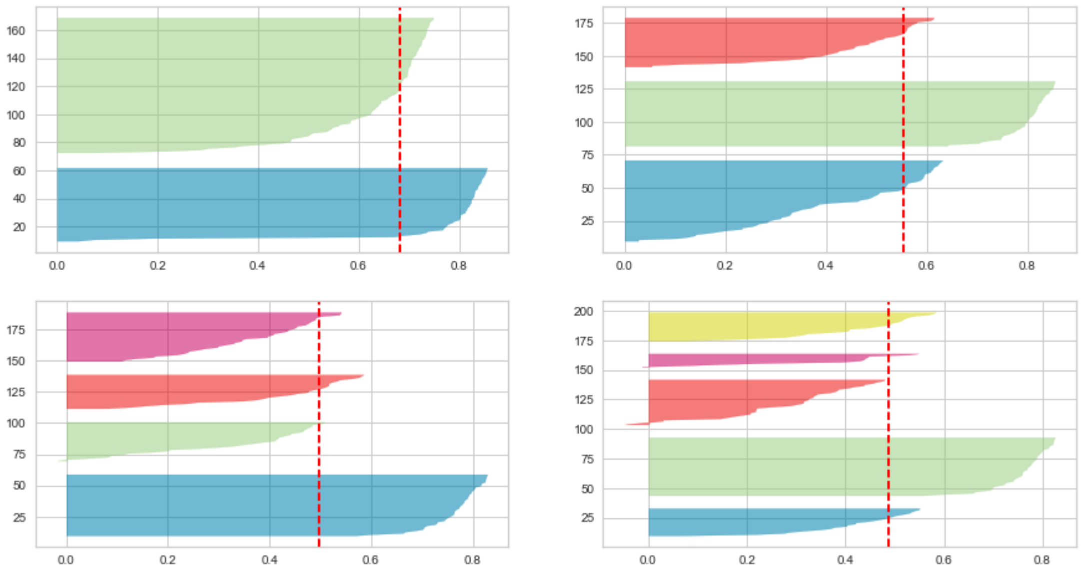
pick 1 center, then the further a next possbile point is the higher the probability is to pick it.

b) What are some limitations of K means ++?

It is still susceptible to outliers, as the further they are the higher the probabilty of picking them is however, by making Kmeans++ probabilistically instead of determinisitcally like FFT we reduce the chance of picking outliers

c) Interpret the silhouette plot below. It's a histogram where each bar corresponds to the silhouette score for that data point. Comment on which number of clusters K (2,3,4 or 5) you would choose and why. (the red dotted line is the average silhouette score over the entire dataset).

In [ ]:
```python
from IPython.display import Image
Image(filename="silhouette.png", width=500, height=500)
```



Values closer to 1 are good (ai-bi)/max(ai,bi). ai is the mean avg dist to every point in the cluster, bi is the smallest mean distance to another cluster.

Probably the top right plot/ or top left (3 clusters/2 clusters), higher than avg sillhouette and splits points well. the thing with the 3 cluster solution is that there are steep drops.

Hierarchical Clustering

Using the following dataset:

Point	x	y
A	0	0
B	1	1
C	3	0
D	0	1
E	2	2

with

d = Euclidean
D = Single-Link

produce the distance matrix at every step of the hierarchical clustering algorithm.

Step 1

	A	B	C	D	E
A	0	$\sqrt{2}$	3	1	$2\sqrt{2}$

	A	B	C	D	E
B	$\sqrt{2}$	0	$\sqrt{5}$	1	$\sqrt{2}$
C	3	$\sqrt{5}$	0	$\sqrt{10}$	$\sqrt{5}$
D	1	1	$\sqrt{10}$	0	$\sqrt{5}$

Step

	A&B	C	D	E
A&B	0	$\sqrt{2}$	1	$\sqrt{2}$
C	$\sqrt{2}$	0	$\sqrt{10}$	$\sqrt{5}$
D	1	$\sqrt{10}$	0	$\sqrt{5}$
E	$\sqrt{2}$	$\sqrt{5}$	$\sqrt{5}$	0

Step 3

	A&B&C	D	E
A&B&C	0	1	$\sqrt{2}$
D	1	0	$\sqrt{5}$
E	$\sqrt{2}$	$\sqrt{5}$	0

Step 4

	A&B&C&D	E
A&B&C&D	0	$\sqrt{2}$
E	$\sqrt{2}$	0

Repeat the above with

d = Euclidean
D = Complete-Link

Step 1

	A	B	C	D	E
A	0	$\sqrt{2}$	3	1	$2\sqrt{2}$
B	$\sqrt{2}$	0	$\sqrt{5}$	1	$\sqrt{2}$
C	3	$\sqrt{5}$	0	$\sqrt{10}$	$\sqrt{5}$
D	1	1	$\sqrt{10}$	0	$\sqrt{5}$
E	$2\sqrt{2}$	$\sqrt{2}$	$\sqrt{5}$	$\sqrt{5}$	0

Step 2

	A&D	B	C	E
A&D	0	1	√10	2√2
B	1	0	√5	√2
C	√10	√5	0	√5

Step 3

	A&D&C	B	E
A&D&C	0	√5	2√2
B	√5	0	√2
E	2√2	√2	0

Step 4

	A&D&C&E	B
A&D&C&E	0	√5
B	√5	0

Challenge Problem

Input:

- Some DNA sequences, each sequence is on a new line. All sequences are of equal length and consist of characters from the set {A, C, G, T}.

Task:

- Implement a hierarchical clustering algorithm using Hamming distance as the metric clustering DNA sequences.

Definition of Hamming Distance:

The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. Mathematically, if we have two strings, s and t, of equal length, then the Hamming distance $H(s, t)$ is given by:

$$H(s,t) = \sum_{i=1}^{n} [s_i \neq t_i]$$

where n is the length of the strings, s_i and t_i are the characters at position i in s and t respectively, and $[s_i \neq t_i]$ is an indicator function, equal to 1 if $s_i \neq t_i$ and 0 otherwise.

Guidelines:

1. **Read the Dataset**: Choose appropriate data structure.
2. **Compute Hamming Distance**: Implement a function to calculate the Hamming distance between any two sequences.
3. **Hierarchical Clustering**: Apply the hierarchical clustering algorithm using the single-linkage method.

In [5]:

```
sequences = [
 'ACGTGGTCTTAA',
 'ACGTCGTCTTAC',
 'ACGTGGTCTTAC',
 'ACGTAGTCTTAA',
 'ACGTGGTCTTCC',
 'ACGTGGTCTTAG',
 'CTGTTAAATAAG',
 'GGTTAGAACACG',
 'AGTGGTTGAAGT',
 'GGCTTACACCCT',
 'AGATTGTCCACT',
 'CATGCGGTCAAC',
 'ATATATCATAGC',
 'TTTGCGGTTGGA',
 'GAATGGTCAGAA',
 'GTGATGCTGTCT']
```