Aprendizagem 2022

Homework IV – Group 22

Part I: Pen and paper

1. Being $x_1 = (1, 2)$, $x_2 = (-1, 1)$ and $x_3 = (1, 0)$

   For the E-Step, we first determine $p(x_n|c_k = 1)$ and $p(c_k = 1|x_n)$ for each observation:

$$N(x|\mu_k, \Sigma_k) = \frac{1}{(2 \cdot \pi)^{\frac{2}{D}}} \cdot \frac{1}{|\Sigma_k|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2} \cdot (x_n - \mu_k)^T \Sigma^{-1} \cdot (x_n - \mu_k)}$$

$$p(x_n|c_k = 1) = N(x|\mu_k, \Sigma_k) \qquad p(c_k = 1, x_n) = \pi_k \cdot N(x|\mu_k, \Sigma_k)$$

$$p(x_1|c_1 = 1) = 0.06584074 \qquad p(c_1 = 1, x_1) = 0.03292037$$

$$p(x_1|c_2 = 1) = 0.02279933 \qquad p(c_2 = 1, x_1) = 0.01139966$$

$$p(x_2|c_1 = 1) = 0.00891057 \qquad p(c_1 = 1, x_2) = 0.00445529$$

$$p(x_2|c_2 = 1) = 0.04826618 \qquad p(c_2 = 1, x_2) = 0.02413309$$

$$p(x_3|c_1 = 1) = 0.03380376 \qquad p(c_1 = 1, x_3) = 0.01690188$$

$$p(x_3|c_2 = 1) = 0.061975 \qquad p(c_2 = 1, x_3) = 0.0309875$$

   We can now calculate $p(x_n)$:

$$p(x_n) = \sum^{k} p(c_k = 1, x_n)$$

$$p(x_1) = p(c_1 = 1|x_1) + p(c_2 = 1|x_1) = 0.04432003$$

$$p(x_2) = p(c_1 = 1|x_2) + p(c_2 = 1|x_2) = 0.02858838$$

$$p(x_3) = p(c_1 = 1|x_3) + p(c_2 = 1|x_3) = 0.04788938$$

   Proceeding to determine $\gamma(c_{nk})$:

$$\gamma(c_{nk}) = p(c_k = 1|x_n) = \frac{p(c_k = 1, x_n)}{p(x_n)}$$

$$\gamma(c_{1,1}) = \frac{p(c_1 = 1, x_1)}{p(x_1)} = 0.74278756 \qquad \gamma(c_{2,2}) = \frac{p(c_2 = 1, x_2)}{p(x_2)} = 0.84415738$$

$$\gamma(c_{1,2}) = \frac{p(c_2 = 1, x_1)}{p(x_1)} = 0.25721244 \qquad \gamma(c_{3,1}) = \frac{p(c_1 = 1, x_3)}{p(x_3)} = 0.35293589$$

$$\gamma(c_{2,1}) = \frac{p(c_1 = 1, x_2)}{p(x_2)} = 0.15584262 \qquad \gamma(c_{3,2}) = \frac{p(c_2 = 1, x_3)}{p(x_3)} = 0.64706411$$

Finally, for the M-Step, we start by determining the N_k for $k = 1$ and $k = 2$:

$$N_k = \sum_{n=1}^{N} \gamma(c_{nk})$$
$$N_1 = \gamma(c_{1,1}) + \gamma(c_{2,1}) + \gamma(c_{3,1}) = 1.25156606$$
$$N_2 = \gamma(c_{1,2}) + \gamma(c_{2,2}) + \gamma(c_{3,2}) = 1.74843394$$

Finally, we compute the new likelihoods, covariance matrices and priors:

$$\mu_k = \frac{1}{N_k} \cdot \sum_{n=1}^{N} \gamma(c_{nk}) \cdot x_n$$
$$\mu_1 = \frac{1}{N_1} \cdot (\gamma(c_{1,1}) \cdot x_1 + \gamma(c_{2,1}) \cdot x_2 + \gamma(c_{3,1}) \cdot x_3) = \begin{bmatrix} 0.75096381 \\ 1.31149108 \end{bmatrix}$$
$$\mu_2 = \frac{1}{N_2} \cdot (\gamma(c_{1,2}) \cdot x_1 + \gamma(c_{2,2}) \cdot x_2 + \gamma(c_{3,2}) \cdot x_3) = \begin{bmatrix} 0.03438459 \\ 0.77702808 \end{bmatrix}$$

$$\Sigma_k = \frac{1}{N_k} \cdot \sum_{n=1}^{N} \gamma(c_{nk}) \cdot (x_n - \mu_k) \cdot (x_n - \mu_k)^T$$
$$\Sigma_1 = \frac{1}{N_1} \cdot \sum_{n=1}^{N} \gamma(c_{n1}) \cdot (x_n - \mu_1) \cdot (x_n - \mu_1)^T = \begin{bmatrix} 0.43605335 & 0.07757255 \\ 0.07757255 & 0.77845521 \end{bmatrix}$$
$$\Sigma_2 = \frac{1}{N_2} \cdot \sum_{n=1}^{N} \gamma(c_{n2}) \cdot (x_n - \mu_2) \cdot (x_n - \mu_2)^T = \begin{bmatrix} 0.9988177 & -0.21530512 \\ -0.21530512 & 0.46747582 \end{bmatrix}$$

$$\pi_k = p(c_k = 1) = \frac{N_k}{N}$$
$$\pi_1 = 0.41718869$$
$$\pi_2 = 0.58281131$$

2.

(a) Determining $p(c_k = 1|x_n)$ for each observation taking into account the new values for the parameters:

$$p(c_k = 1|x_n) = \frac{\pi_k \cdot N(x|\mu_k, \Sigma_k))}{p(x_n)}$$
$$p(c_1 = 1|x_1) = \frac{\pi_1 \cdot N(x|\mu_1, \Sigma_1))}{p(x_1)} = 0.91198316$$
$$p(c_2 = 1|x_1) = \frac{\pi_2 \cdot N(x|\mu_2, \Sigma_2))}{p(x_1)} = 0.08801684$$
$$p(c_1 = 1|x_2) = \frac{\pi_1 \cdot N(x|\mu_1, \Sigma_1))}{p(x_2)} = 0.03923683$$
$$p(c_2 = 1|x_2) = \frac{\pi_2 \cdot N(x|\mu_2, \Sigma_2))}{p(x_2)} = 0.96076317$$
$$p(c_1 = 1|x_3) = \frac{\pi_1 \cdot N(x|\mu_1, \Sigma_1))}{p(x_3)} = 0.34518610$$
$$p(c_2 = 1|x_3) = \frac{\pi_2 \cdot N(x|\mu_2, \Sigma_2))}{p(x_3)} = 0.65481390$$

Based on the results, we assign x_1 to cluster 1 while x_2 and x_3 are assigned to cluster 2.

(b) Taking into account the answer in a), $C_1 = \{x_1\}$ and $C_2 = \{x_2, x_3\}$
We define the following formula for the silhouette of a single point:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}, \text{ if } |C_I| > 1$$

And the following formulas for a(i) and b(i):

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j) \qquad b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i, j)$$

We also define d(i, j) with the euclidean distance formula:

$$d(i, j) = \sqrt{\sum_{k=1}^{n} (j_k - i_k)^2} = ||j - i||_2$$

Computing now a(i), b(i) and s(i) for each $i \in C_2$ (the larger cluster).

$$a(x_2) = ||x_3 - x_2||_2 = 2.2361 \qquad a(x_3) = ||x_2 - x_3||_2 = 2.2361$$

$$b(x_2) = ||x_1 - x_2||_2 = 2.2361 \qquad b(x_3) = ||x_1 - x_3||_2 = 2.0$$

$$s(x_2) = 1 - \frac{a(x_2)}{b(x_2)} = 0 \qquad s(x_3) = 1 - \frac{a(x_3)}{b(x_3)} = -0.1056$$

Computing now the silhouette of the cluster:

$$s(C_2) = \frac{s(x_2) + s(x_3)}{2} = -0.0528$$

Part II: Programming

1. Code solution is provided in Appendix(1).

2. The non-determinism can be explained by the nature of the algorithm. By default, `KMeans` takes `k-means++` as the `init` parameter. As such, it randomly selects a centroid from the data points and then computes the distance $d(x)$ between x and the nearest center that has already been chosen for each data point x not chosen yet. The new center is then chosen based on a weighted probability distribution where the probability of point x being chosen is proportional to $d(x)^2$. The last steps are repeated until the 3 centroids have been sampled and the regular K-Means clustering algorithm is then applied.

   This happens over a default of 10 interactions and the final results will be the best output out of the 10 runs in terms of inertia.

   It's easy to notice the initial seed is going to impact the first randomly selected centroid and consequently the 2 other centroids chosen thereafter. Because the number of iteractions is relatively low, diferent seeds end up swaying the performance. We can easily verify the veracity of this statement by increasing the `n_init` parameter. For a big enough number, the seed does not matter and the 3 solutions show the same results.

   The fact that the results for the seeds 0 and 2 are the same can be explained by chance. Their best output out of the 10 runs happened to be the same (either because the overall best centroids were chosen in one of the iterations or because their best runs happened when using the same centroids).

3. Code solution is provided in Appendix(2). The scatter plots are provided in Appendix(3).

4. 31 components are necessary to explain 80% of the variance. Code solution is provided in Appendix(4).

Appendix

1.

```
import pandas as pd
from scipy.io.arff import loadarff

# Reading the ARFF file
data = loadarff('data/pd_speech.arff')
df = pd.DataFrame(data[0])
df['class'] = df['class'].str.decode('utf-8')

# normalize data using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[df.columns] = scaler.fit_transform(df[df.columns])

X = df.drop('class', axis=1)
y = df['class']
# apply k-means clustering to the data
from sklearn.cluster import KMeans

kmeans = []
for i in range(3):
    kmeans.append(KMeans(n_clusters=3, random_state=i).fit(X))

import numpy as np
from sklearn.metrics.cluster import contingency_matrix

# define fuction to compute the purity of a clustering
def purity_score(y_true, y_pred):
    # compute contingency/confusion matrix
    confusion_matrix = contingency_matrix(y_true, y_pred)
    return np.sum(np.amax(confusion_matrix, axis=0)) / np.sum(confusion_matrix)

from sklearn.metrics import silhouette_score

# compute the purity and silhouette score for each clustering
for kmean in kmeans:
    print('K-means with random state =', kmean.random_state)
    print('Silhouette score: ', silhouette_score(X, kmean.labels_, metric='euclidean'))
    print('Purity score: ', purity_score(y, kmean.labels_),'\n')
```

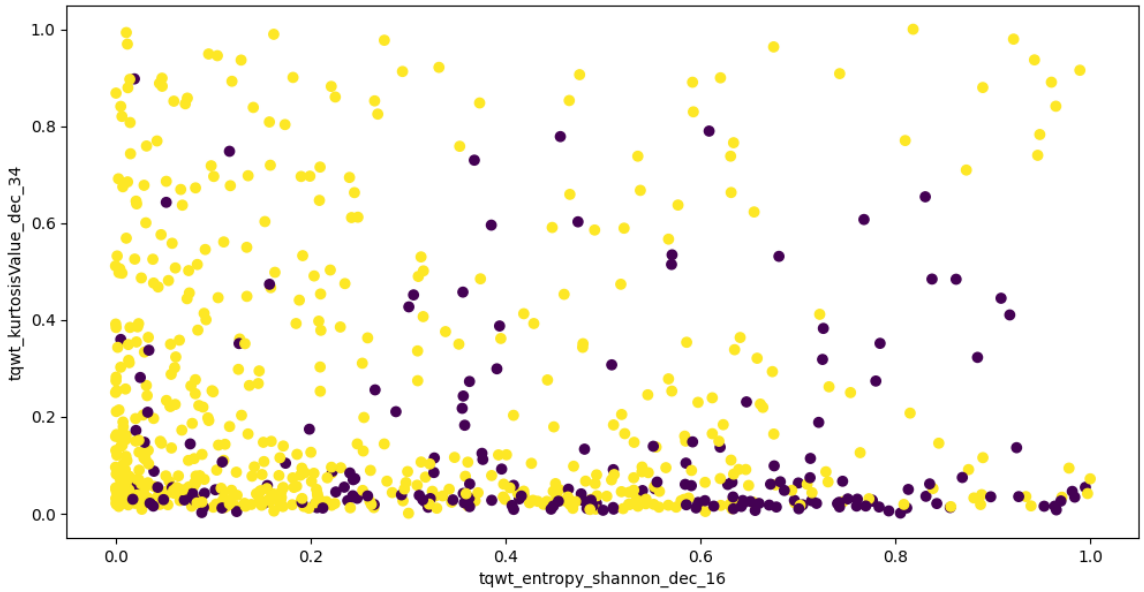
```
>> K-means with random state = 0
>> Silhouette score:  0.11415638651071394
>> Purity score:  0.7645502645502645
```

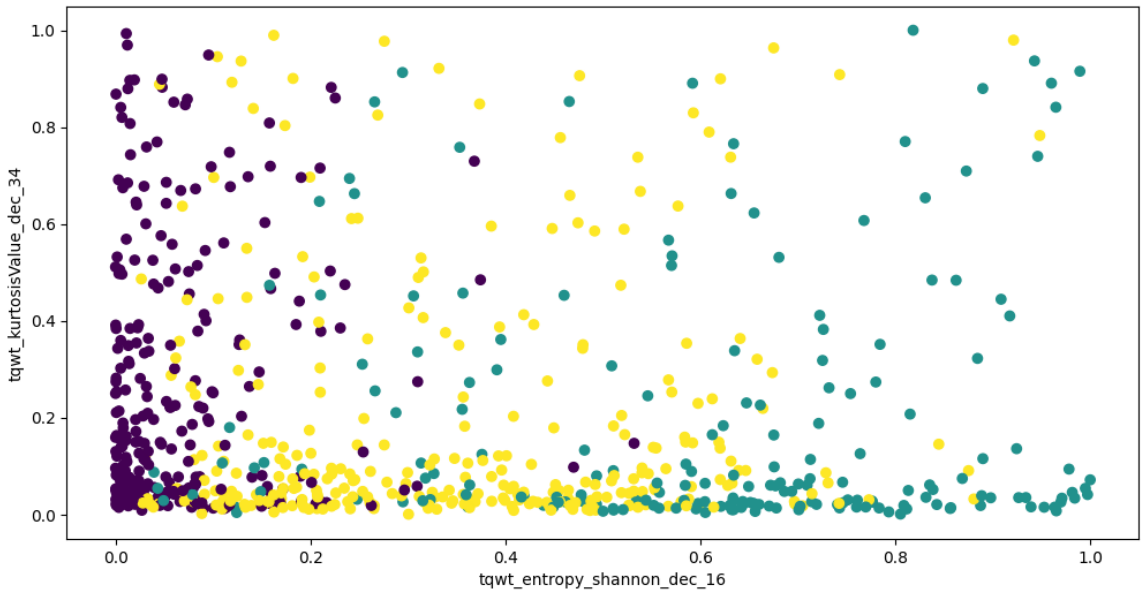
```
>> K-means with random state = 1
>> Silhouette score:  0.11413540017275367
>> Purity score:  0.7658730158730159

>> K-means with random state = 2
>> Silhouette score:  0.11415638651071394
>> Purity score:  0.7645502645502645
```

2.

```
# get the index of the two features with highest variance
index1, index2 = X.var().sort_values(ascending=False).head(2).index

# plot data
import matplotlib.pyplot as plt

for c in (y, kmeans[0].labels_):
    plt.scatter(X[index1], X[index2], c=c)
    plt.xlabel(X[index1].name)
    plt.ylabel(X[index2].name)

plt.show()
```

3.





4.

```
from sklearn.decomposition import PCA

# compute the number of components needed for 80% of the variance explained
pca = PCA(n_components=0.8, random_state=0)
pca.fit(X)
print(len(pca.components_), 'components explain 80% of the variance')
```

```
>> 31 components explain 80% of the variance
```