INTELLIGENT SYSTEMS LAB-4 (13/09/2021)

***NAME**: Harshvardhan Agarwal | **SEC**: A | **REG NO**.: 201800524*

PROBLEM STATEMENT

Write a program to explore the use of unsupervised learning methods for clustering data, and also for obtaining lower-dimensional representations.

1. Submit the pdf file with code and graphs (py, doc file will not be evaluated)

2. Analyze the algorithm by varying number of clusters (k) and features at least for 2 types of dataset

3. Determine the number of clusters using the Elbow method and plot the graph.

PROBLEM SOLUTION

SOURCE CODE

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.datasets._samples_generator import make_blobs
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
sns.set()

# First Dataset

X, Y = make_blobs(n_samples = 300, centers = 3, cluster_std = 0.60, random_state = 0)
plt.scatter(X[:, 0], X[:, 1], c = Y, s = 10, cmap = "Accent")
plt.show()
kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
centers = kmeans.cluster_centers_
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=10, cmap='viridis')
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=100, alpha=0.9);
plt.show()
distortions = []
inertias = []
mapping1 = {}
```

```
mapping2 = {}
for k in range(1, 10):
    kmeanModel = KMeans(n_clusters = k).fit(X)
    kmeanModel.fit(X)
    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_, 'euclidean'), axis = 1)) /
X.shape[0])
    inertias.append(kmeanModel.inertia_)
    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_, 'euclidean'), axis = 1)) / X.sh
ape[0]
    mapping2[k] = kmeanModel.inertia_
for key, val in mapping1.items():
    print(f'{key} : {val}')
plt.plot(range(1, 10), distortions, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Distortion')
plt.title('The Elbow Method using Distortion')
plt.show()
# Second Dataset

x = np.array([3, 1, 1, 2, 1, 6, 6, 6, 5, 6, 7, 8, 9, 8, 9, 9, 8])
y = np.array([5, 4, 5, 6, 5, 8, 6, 7, 6, 7, 1, 2, 1, 2, 3, 2, 3])
X = np.array(list(zip(x, y))).reshape(len(x), 2)
plt.scatter(x, y, s = 25)
plt.show()

kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
centers = kmeans.cluster_centers_
plt.scatter(X[:, 0], X[:, 1], c = y_kmeans, s = 25, cmap = 'viridis')
plt.scatter(centers[:, 0], centers[:, 1], c = 'black', s = 100, alpha = 0.9);
plt.show()

distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
for k in range(1, 10):
    kmeanModel = KMeans(n_clusters = k).fit(X)
    kmeanModel.fit(X)
    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_, 'euclidean'), axis = 1)) /
X.shape[0])
    inertias.append(kmeanModel.inertia_)
    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_, 'euclidean'), axis = 1)) / X.sh
ape[0]
    mapping2[k] = kmeanModel.inertia_
for key, val in mapping1.items():
    print(f'{key} : {val}')
plt.plot(range(1, 10), distortions, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Distortion')
```

```
plt.title('The Elbow Method using Distortion')
plt.show()
```

OUTPUT:

Performing necessary imports:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.datasets._samples_generator import make_blobs
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
```

✓ 2.2s

Loading the dataset

```
sns.set()

# First Dataset

X, Y = make_blobs(n_samples = 300, centers = 3, cluster_std = 0.60, random_state = 0)
plt.scatter(X[:, 0], X[:, 1], c = Y, s = 10, cmap = "Accent")
plt.show()
```

✓ 0.3s

Applying K-Means clustering

```
kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

✓ 0.4s

```
centers = kmeans.cluster_centers_
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=10, cmap='viridis')
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=100, alpha=0.9);
plt.show()
```

✓ 0.2s

For k value = 4

```
kmeans = KMeans(n_clusters = 4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
centers = kmeans.cluster_centers_
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=10, cmap='viridis')
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=100, alpha=0.9);
plt.show()
```

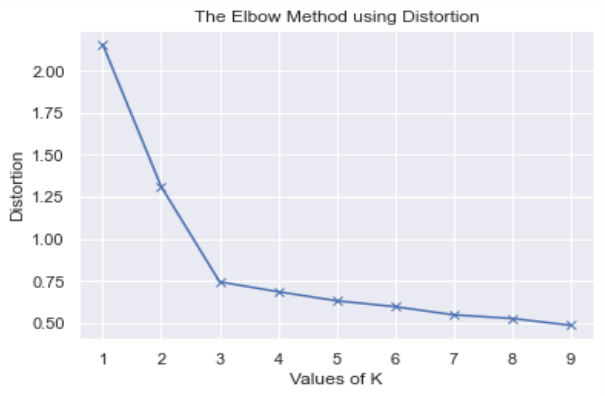
✓ 0.2s

Number of clusters using elbow method

```python
distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
for k in range(1, 10):
    kmeanModel = KMeans(n_clusters = k).fit(X)
    kmeanModel.fit(X)
    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_, 'euclidean'), axis = 1)) / X.shape[0])
    inertias.append(kmeanModel.inertia_)
    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_, 'euclidean'), axis = 1)) / X.shape[0]
    mapping2[k] = kmeanModel.inertia_
for key, val in mapping1.items():
    print(f'{key} : {val}')
plt.plot(range(1, 10), distortions, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Distortion')
plt.title('The Elbow Method using Distortion')
plt.show()
```

✓ 1.7s

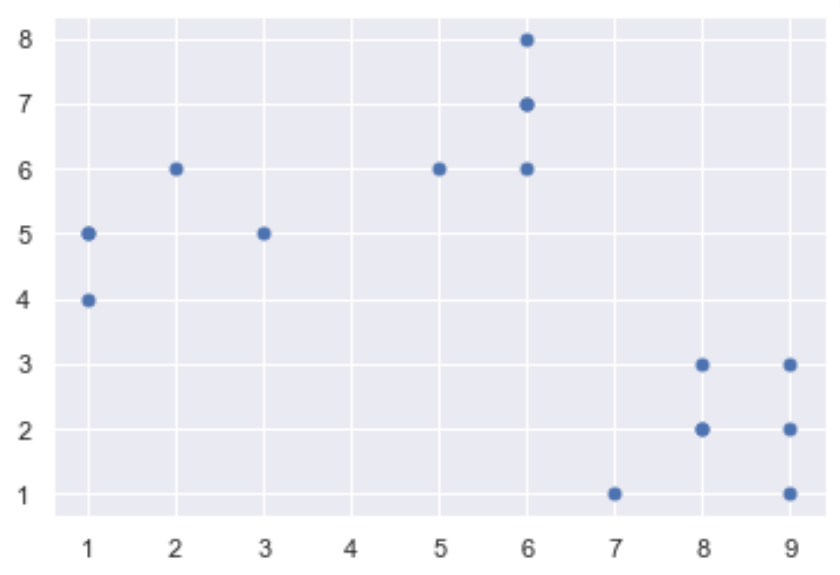
```
1 : 2.155550660986752
2 : 1.3107633424092735
3 : 0.7449068201503806
4 : 0.6861181822884652
5 : 0.6322830313394731
6 : 0.5973445548240676
7 : 0.5488564823469472
8 : 0.5270951578052582
9 : 0.486967797780246
```



Second dataset

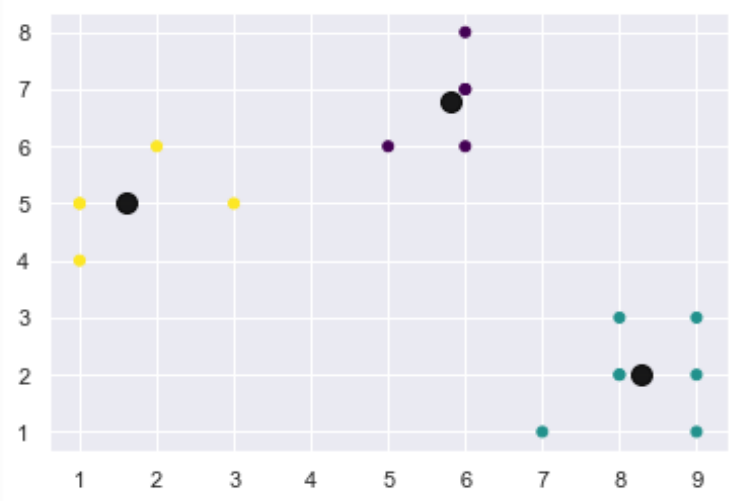
```
# Second Dataset

x = np.array([3, 1, 1, 2, 1, 6, 6, 6, 5, 6, 7, 8, 9, 8, 9, 9, 8])
y = np.array([5, 4, 5, 6, 5, 8, 6, 7, 6, 7, 1, 2, 1, 2, 3, 2, 3])
X = np.array(list(zip(x, y))).reshape(len(x), 2)
plt.scatter(x, y, s = 25)
plt.show()

kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

✓ 0.2s



```
centers = kmeans.cluster_centers_
plt.scatter(X[:, 0], X[:, 1], c = y_kmeans, s = 25, cmap = 'viridis')
plt.scatter(centers[:, 0], centers[:, 1], c = 'black', s = 100, alpha = 0.9);
plt.show()
```

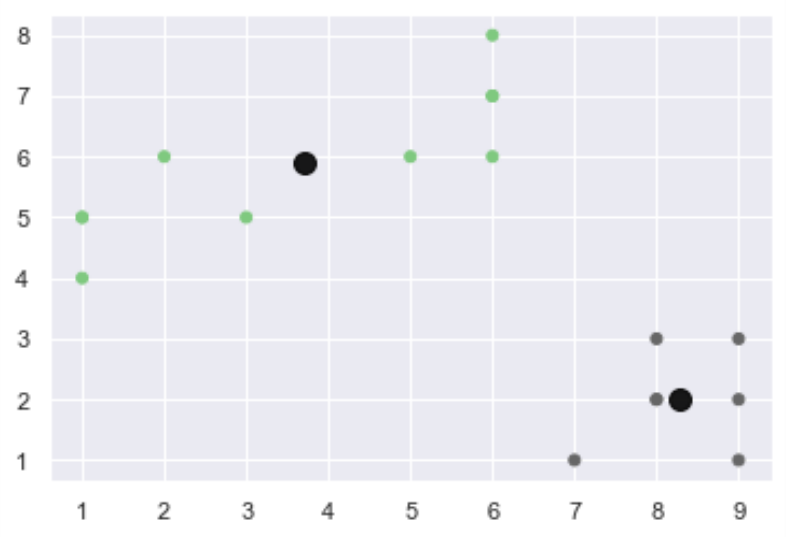
✓ 0.3s



For k = 2

```
kmeans = KMeans(n_clusters = 2)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
centers = kmeans.cluster_centers_
plt.scatter(X[:, 0], X[:, 1], c = y_kmeans, s = 25, cmap = 'Accent')
plt.scatter(centers[:, 0], centers[:, 1], c = 'black', s = 100, alpha = 0.9);
plt.show()
```

✓ 0.3s



```
distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
for k in range(1, 10):
    kmeanModel = KMeans(n_clusters = k).fit(X)
    kmeanModel.fit(X)
    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_, 'euclidean'), axis = 1)) / X.shape[0])
    inertias.append(kmeanModel.inertia_)
    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_, 'euclidean'), axis = 1)) / X.shape[0]
    mapping2[k] = kmeanModel.inertia_
for key, val in mapping1.items():
    print(f'{key} : {val}')
plt.plot(range(1, 10), distortions, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Distortion')
plt.title('The Elbow Method using Distortion')
plt.show()
```

✓ 1.2s

```
1 : 3.4577032384495707
2 : 1.7687413573405673
3 : 0.8819889697423957
4 : 0.7587138847606585
5 : 0.6635212812400347
6 : 0.5808803063754726
7 : 0.5274410771884641
8 : 0.4117647058823529
9 : 0.3333333333333333
```