# CO544-Lab5-Ex1

June 23, 2020

[69]:
```
#1.1 Importing required modules.
from sklearn.cluster import KMeans
from sklearn.datasets.samples_generator import make_blobs #to generate sample␣
 ↪datasets
import matplotlib.pyplot as plt
from sklearn import datasets
import numpy as np
import pandas as pd
from mpl_toolkits import mplot3d #importing modules for 3D plotting
```

[70]:
```
#Import iris dataset
iris_dataset = datasets.load_iris()
```

[71]:
```
iris_data = iris_dataset["data"]
iris_labels = iris_dataset["target"]
```

[72]:
```
print("data shape = ",iris_data.shape)
```

```
data shape =  (150, 4)
```

[73]:
```
print("target shape = ",iris_labels.shape)
```

```
target shape =  (150,)
```

[74]:
```
iris = pd.DataFrame(iris_data)
```

[75]:
```
print(iris)
```

```
       0    1    2    3
0    5.1  3.5  1.4  0.2
1    4.9  3.0  1.4  0.2
2    4.7  3.2  1.3  0.2
3    4.6  3.1  1.5  0.2
4    5.0  3.6  1.4  0.2
..   …    …    …    …
145  6.7  3.0  5.2  2.3
146  6.3  2.5  5.0  1.9
147  6.5  3.0  5.2  2.0
```

```
148  6.2  3.4  5.4  2.3
149  5.9  3.0  5.1  1.8

[150 rows x 4 columns]
```

[76]:

```
features = iris_dataset.feature_names
```

[77]:

```
print("features = \n",features)
```

```
features =
 ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
```

[78]:

```
iris.columns = features
```

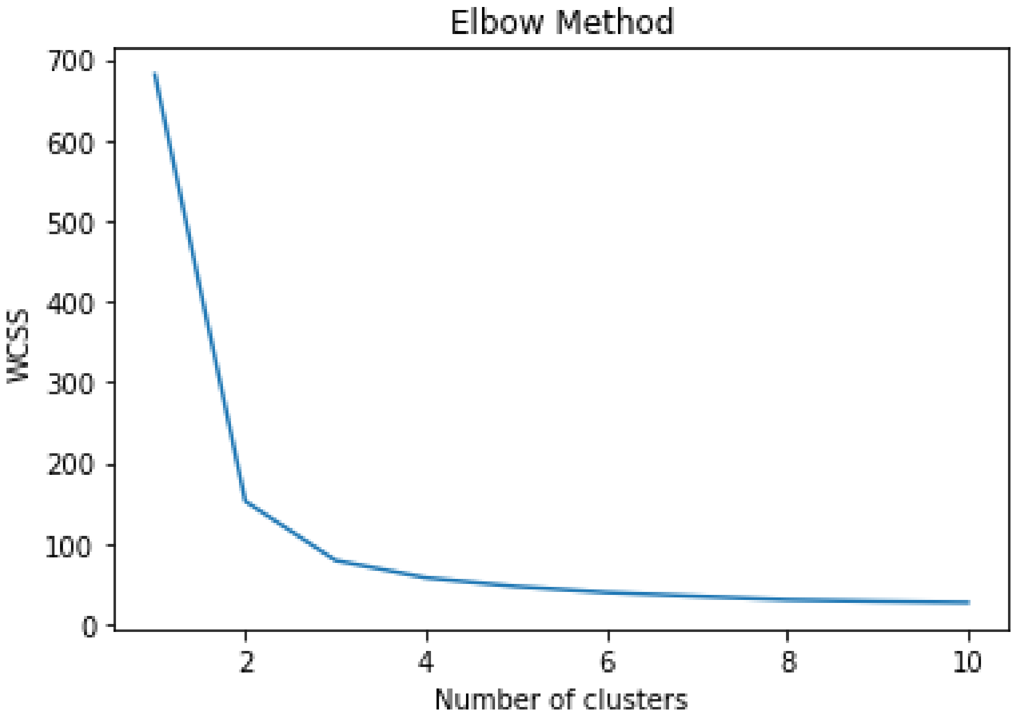
[79]:

```
#iris dataset without the class attribute
print(iris)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 …                 …                  …                 …
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[150 rows x 4 columns]
```

[80]:

```
#1.3 Determining the optimum value of k using Elbow method.
wcss = [] #within cluster sum of squares

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,␣
 ↪random_state=0)
    kmeans.fit(iris_data)
    wcss.append(kmeans.inertia_)


plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



[88]:
```
#According to the above graph k = 3

#1.4 Applying K-Means algorithm.
kmeans = KMeans(n_clusters=3, random_state=0) #from Elbow method we identified␣
 ↪n_clusters=3
closest_cluster_index = kmeans.fit_predict(iris_data)
cluster_centers=kmeans.cluster_centers_
```

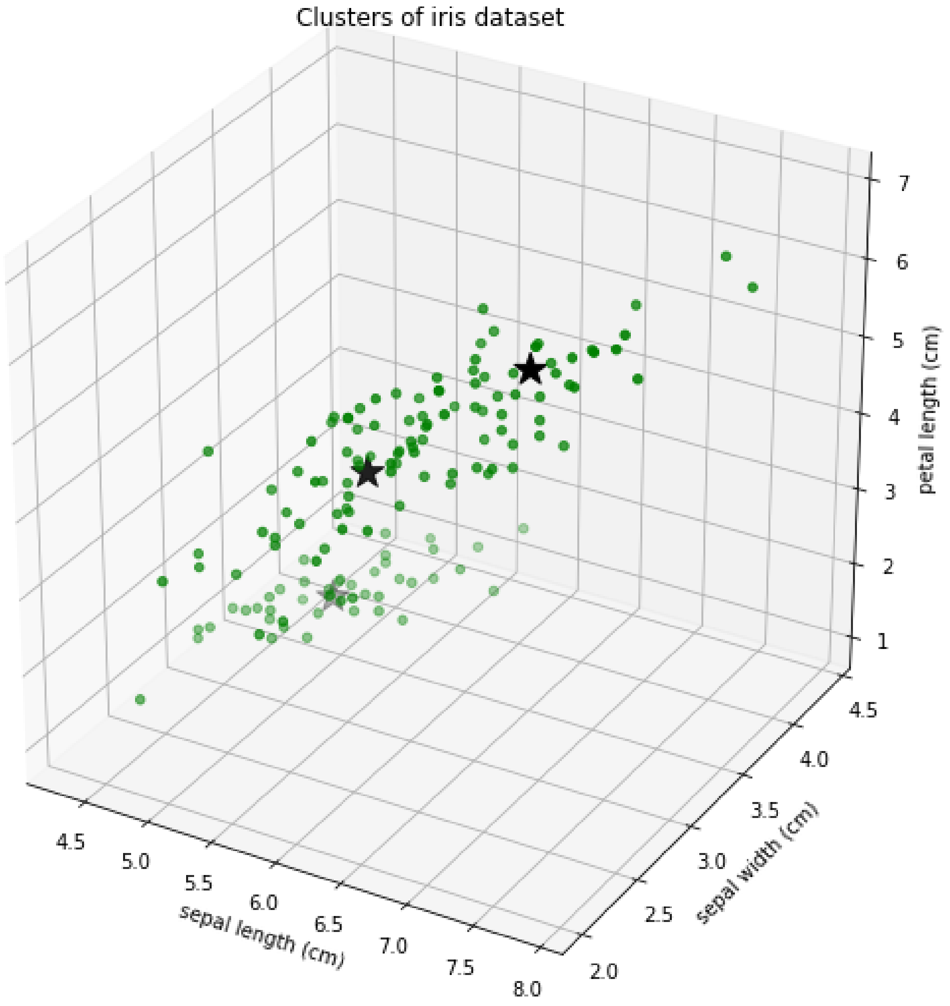
[82]:
```
print("Cluster centers = \n",cluster_centers)
```

```
Cluster centers =
 [[6.85       3.07368421 5.74210526 2.07105263]
 [5.006      3.428      1.462      0.246     ]
 [5.9016129  2.7483871  4.39354839 1.43387097]]
```

[83]:
```
#kmeans.cluster_centers_ code snippet gives the centroid(middle point of the␣
 ↪cluster) of the clusters compared to the data points of the dataset
#A centroid is a vector that contains one number for each variable, where each␣
 ↪number is the mean of a variable for the observations in that cluster. The␣
 ↪centroid can be thought of as the multi-dimensional average of the cluster.
#Since the iris dataset has 3 clusters, it gives 3 centroids
```

[86]:

```
#1.5 Visualizing
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d') #creating 3D subplot
ax.set_xlabel('sepal length (cm)')
ax.set_ylabel('sepal width (cm)')
ax.set_zlabel('petal length (cm)')
ax.set_title("Clusters of iris dataset")

ax.scatter(iris_data[:,0], iris_data[:,1],iris_data[:,2],c='green')
ax.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],kmeans.
 ↪cluster_centers_[:,2],s=300,c='black',marker='*')
plt.show()
```



[ ]: