K-Means on Two-Dimensional Data

This lesson will focus on K-Means on two-dimensional data in Python.

WE'LL COVER THE FOLLOWING

- K-means in Python

K-means in Python

We do not need to code the above algorithm because it is available in `sklearn.cluster`. We will be clustering on a dummy dataset first. The dummy dataset has three columns, `feature_1`, `feature_2`, and `label`. The dataset has 4 classes, which mean each row of the data set can have a label from `0`, `1`, `2`, or `3`. First, we will plot a scatter plot of the two features.

dummy.csv

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('dummy.csv')
print(df.head())
X = df.drop(columns = ['label'])
Y = df['label']

plt.scatter(x= X['feature_1'],y = X['feature_2'])
```

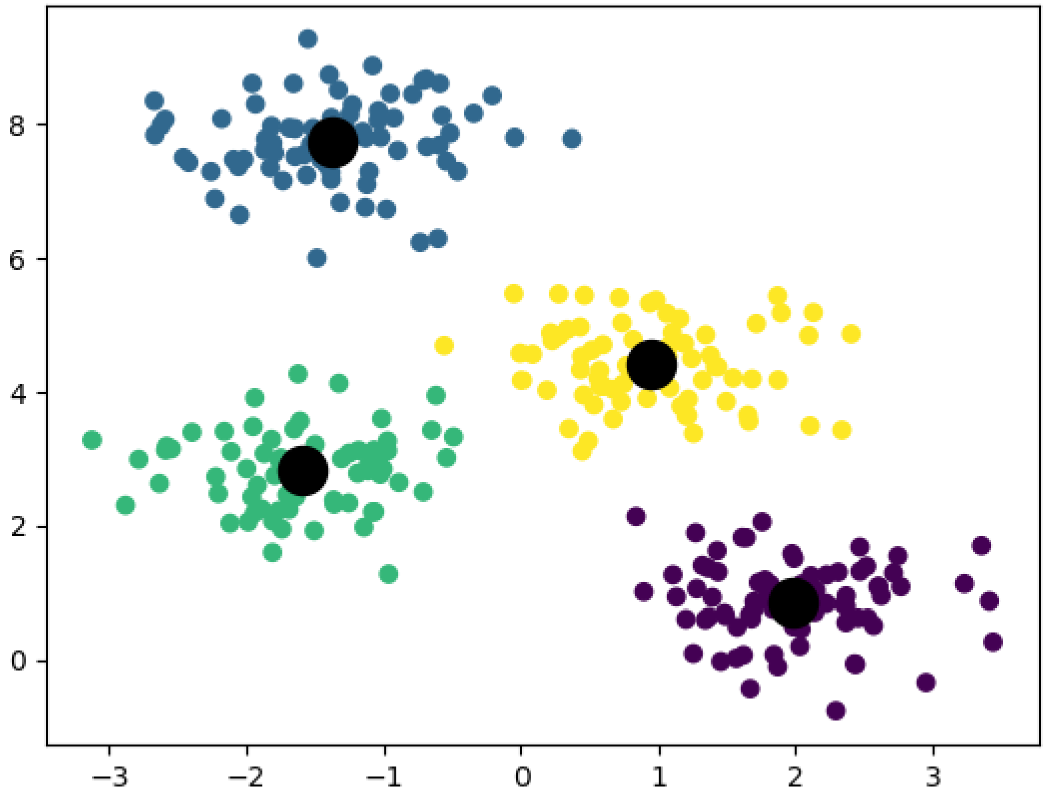
We drop the `label` column in **line 6** because in reality, we do not have labels for unsupervised learning. We plot a scatter plot between the two columns in **line 9**. By looking at the plot we observe that this data can be categorized into clusters.

Let's make these clusters below.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

df = pd.read_csv('dummy.csv')
X = df.drop(columns = ['label'])
Y = df['label']

# kmeans clustering
km = KMeans(n_clusters = 4,random_state = 2)
km.fit(X)

# predictions
preds = km.predict(X)
centers = km.cluster_centers_
print(preds)
print(centers)

# plots
plt.scatter(x = X['feature_1'],y = X['feature_2'],c = preds)
plt.scatter(x = centers[:,0],y = centers[:,1],c = 'k',s = 300)
```

We import the `KMeans` class in **line 3**. Then we read and set the data in **lines 5-7**. We initialize the `KMeans` class with `n_clusters = 4` in **line 10** which means we want 4 clusters to be made. Then we use the `fit` function to make our cluster centers in **line 11**. To predict the clusters for our data points we use the function `predict` in **line 14** and store the result in `preds`. We obtain the cluster centers in **line 15**. We can see from the output of **line 16** that `preds` is a list which contains the cluster number to which each individual data point belongs to.

To visualize our results, we plot the predictions in **line 20**. We use the `plt.scatter` function. We color the data points by the cluster number to which they belong to. For this, we give the predicted cluster numbers `preds` as the color by giving the argument `c=preds`. All of the points in a cluster are colored with the same color. We then again use `plt.scatter` to plot the cluster centers in **line 21**. `centers` is a two-dimensional list which has all the x-coordinates and the y-coordinates of the cluster centers. To give the x-coordinates we write `centers[:,0]`, which means the values in all rows and the first column. In the same way, we provide the y-coordinates as `centers[:,1]`. We color the centers black, and we specify the size in *points* `centers[:,1]`. We color the centers black, and we specify the size in *points*, aka *pt*, of the cluster centers as `s = 300`.



From the plot, we can see that the data points have been divided into 4 clusters. Note that we did not use the original labels anywhere for clustering. This is unsupervised learning!

The above example of dummy data was a very simple one. The dataset was two dimensional, i.e., it had only two features. But we have been using datasets that have many features. So, how do we cluster for n-dimensions? Also, how do we visualize n-dimensional data? We will look at that in the next lesson.