Aakriti Singh 20190802006

Fundamentals of AI & ML

Monsoon Semester V 2021-22

Lab - 7

Date: 12 November 2021

Topic: EM & K-Means

AIM

Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using the k-Means algorithm

THEORY

EM algorithm in machine learning uses observable instances of latent variables in order to predict values in instances, unobservable for learning, and continues till the convergence of the values takes place.

Algorithm:

1. Given a set of incomplete data, consider a set of starting parameters.
2. **Expectation step (E – step):** Using the observed available data of the dataset, estimate (guess) the values of the missing data.
3. **Maximization step (M – step):** Complete data generated after the expectation (E) step is used in order to update the parameters.
4. Repeat step 2 and step 3 until convergence.

K-means algorithm explores for a preplanned number of clusters in an unlabelled multidimensional dataset, it concludes this via an easy interpretation of how an optimized cluster can be expressed.

Algorithm:

1. First, we initialize k points, called means, randomly.
2. We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that mean so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

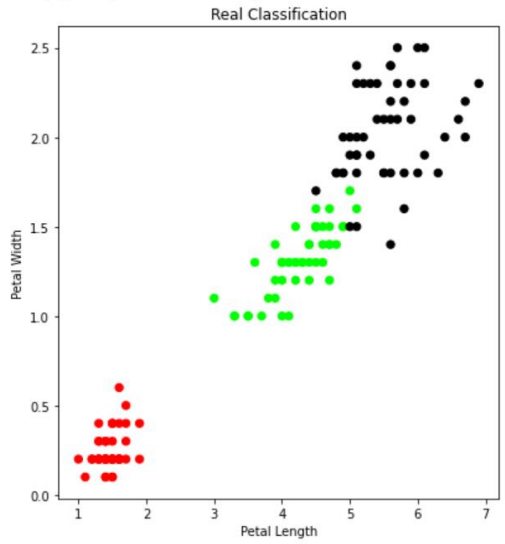
PROGRAM CODE

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np

iris = datasets.load_iris()

X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']

y = pd.DataFrame(iris.target)
y.columns = ['Targets']

model = KMeans(n_clusters=3)
model.fit(X)


plt.figure(figsize=(14,7))
plt.show()

colormap = np.array(['red', 'lime', 'black'])

# Plot the Original Classifications
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()


# Plot the Models Classifications
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K Mean Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
print('The accuracy score of K-Mean: ',sm.accuracy_score(y, model.labels_))
print('The Confusion matrixof K-Mean: ',sm.confusion_matrix(y, model.labels_))


from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)
#xs.sample(5)

from sklearn.mixture import GaussianMixture
```

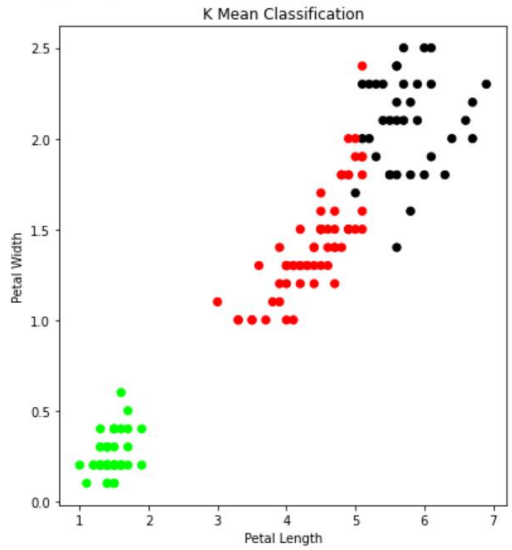
```
gmm = GaussianMixture(n_components=3)
gmm.fit(xs)

y_gmm = gmm.predict(xs)
#y_cluster_gmm

plt.subplot(2, 2, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_gmm], s=40)
plt.title('GMM Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()

print('The accuracy score of EM: ',sm.accuracy_score(y, y_gmm))
print('The Confusion matrix of EM: ',sm.confusion_matrix(y, y_gmm))
```

OUTPUT

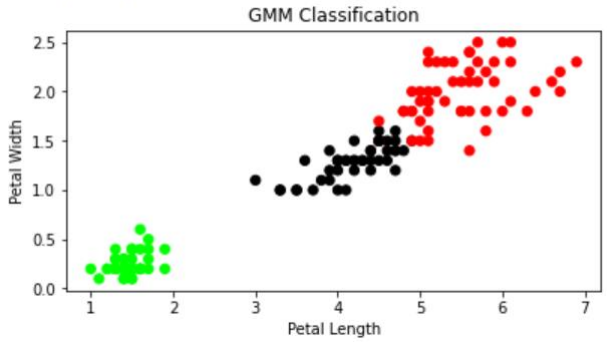
```
Text(0, 0.5, 'Petal Width')
```




```
Text(0, 0.5, 'Petal Width')
```



```
Text(0, 0.5, 'Petal Width')
```



OBSERVATION

```
The accuracy score of K-Mean:  0.24
The Confusion matrixof K-Mean:
 [[ 0 50  0]
 [48  0  2]
 [14  0 36]]

The accuracy score of EM:  0.0
The Confusion matrix of EM:
 [[ 0 50  0]
 [ 5  0 45]
 [50  0  0]]
```

CONCLUSION

Firstly, EM Algorithm represents the idea that we can introduce latent variables and then compute by alternately dealing with the latent variables and dealing with the parameters. Second, the algorithm is inherently fast because it doesn't depend on computing gradients. Anytime we can solve a model analytically, it's going to be faster.

Kmeans clustering is one of the most popular clustering algorithms and The goal of kmeans is to group data points into distinct non-overlapping subgroups.