Name : Saloni Vishwakarma

Batch-Roll no : C1-13

Date of execution : 01 November 2023

Practical No. 07 (Part 1)

```
import numpy as np
import pandas as pd
from scipy.stats import multivariate_normal
from sklearn.mixture import GaussianMixture
from matplotlib import pyplot as plt
```

```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
```

```
class GMM:
  def __init__(self,K,max_iter):
    self.K=K
    selfmax_iter=max_iter

    def initialize(self,X):
      self.dims=X.shape[1]

      self.phi=np.full(shape=self.K,fill_value=1/self.K)
      self.weights=np.zeros(X.shape)

      self.mu=[X[random-row,:] for random_row in random.samplr(0,len(X),self.K)]

      self.sigma=[np.cov(X,T) for _ in range(self.K)]

      def fit(self,X):
        self.initialize(X)

        for _ in range(self.max_iter):
          self.e_step(X)
          self.m_step(X)

      def e_step(self,X):
       self.weights=self.predict_proba(X)
       self.phi=self.weights.mean(axis=0)

    def m_step(self,X):
      for i in range(self,K):
        weights=self.weights[:,[i]]
        total_weights=weights.sum()
        self.mu[i]=(X*weight).sum(axis=0)/total_weights
        self.sigma[i]=np.cov(X,T,aweight=(weights/total_weights).flatten(),bias=True)

      def predict_proba(self,X):
        likelihood=np.zeros((len(X),self.K))
        for i in range(self.K):

          distribution = multivariate_normal(mean=self.mu[i],cov=self.sigma[i])
          likelihood[:,i]=distribution.pdf(X)

        numerator=likelihood*self.phi
        denominator=numerator.sum(axis=1)[:,np.newaxis]
        weight=numerator/denominator
        return weight
```

```
gmm = GMM(K=2, max_iter=100)
```

```
from sklearn import datasets
myiris=datasets.load_iris()
X = myiris.data
```

```
gmm = GaussianMixture(n_components = 2)
gmm.fit(X)
```

```
GaussianMixture
GaussianMixture(n_components=2)
```

```
probabilities=gmm.predict_proba(X)
```

```
plt.scatter([i[0] for i in X], [i[1] for i in X], color=[(0,i[0], i[1]) for i in probabilities])
plt.show()
```