Guided Projects Artificial Intelligence & Machine Learning

Guided Projects: Unsupervised Learning

Gaussian Mixture Models

Name	Lakshmi Thirunavukkarasu
Course	AI and ML (Batch 5)
Problem Statement	Implement and perform Clustering using gaussian mixture model on a 2D Dataset.

Software requirements prerequisites

Anaconda

Python 3.8

Python Packages

- NumPy
- Scipy
- Pandas
- Scikit
- Matplotlib

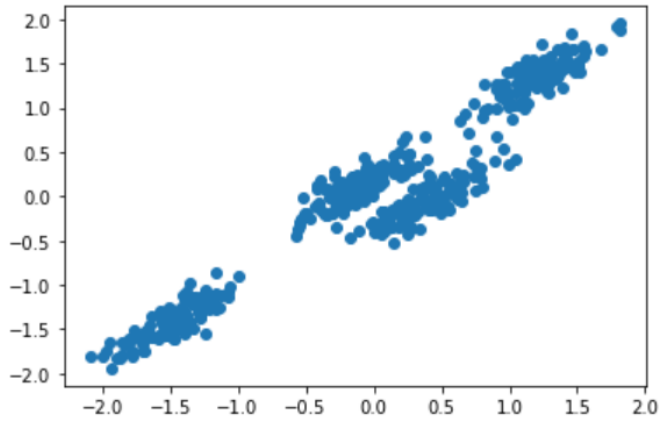
Steps

1. **Load the Dataset**

Load the dataset

In [20]:
```
df = pd.read_csv('Clustering_gmm.csv')
X = StandardScaler().fit_transform(df)
```

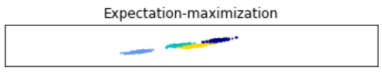
In [21]:
```
plt.scatter( X[:, 0],X[:, 1])
plt.show()
```



2. Build Gaussian Mixture Model and plot the components discovered by the model

Gaussian Mixture Model

In [12]:
```
color_iter = itertools.cycle(["navy", "c", "cornflowerblue", "gold", "darkorange"])

#### Fit a Gaussian mixture with EM using ten components
gmm = mixture.GaussianMixture(
    n_components=4, covariance_type="full", max_iter=100
).fit(X)

plot_results(
    X, gmm.predict(X), gmm.means_, gmm.covariances_, 0, "Expectation-maximization"
)
```



3. Implement Gaussian Mixture Model

```
from scipy.stats import multivariate_normal

class GMM:
    def __init__(self, k, max_iter=5):
        self.k = k
        self.max_iter = int(max_iter)

    def initialize(self, X):
        self.shape = X.shape
        self.n, self.m = self.shape

        self.phi = np.full(shape=self.k, fill_value=1/self.k)
        self.weights = np.full( shape=self.shape, fill_value=1/self.k)

        random_row = np.random.randint(low=0, high=self.n, size=self.k)
        self.mu = [  X[row_index,:] for row_index in random_row ]
        self.sigma = [ np.cov(X.T) for _ in range(self.k) ]

    def e_step(self, X):
        # E-Step: update weights and phi holding mu and sigma constant
        self.weights = self.predict_proba(X)
        self.phi = self.weights.mean(axis=0)

    def m_step(self, X):
        # M-Step: update mu and sigma holding phi and weights constant
        for i in range(self.k):
            weight = self.weights[:, [i]]
            total_weight = weight.sum()
            self.mu[i] = (X * weight).sum(axis=0) / total_weight
            self.sigma[i] = np.cov(X.T,
                aweights=(weight/total_weight).flatten(),
                bias=True)
```

4. Generate 2D dataset

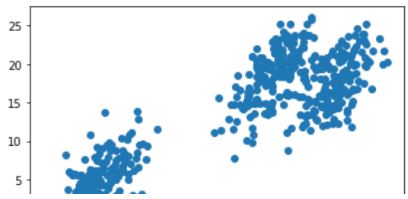
Generate dataset using make_blob

In [31]:
```
from sklearn.datasets import make_blobs
import numpy as np
from scipy.stats import multivariate_normal


# 0. Create dataset
X,Y = make_blobs(cluster_std=1.5,random_state=20,n_samples=500,centers=3)

# Stratch dataset to get ellipsoid data
X = np.dot(X,np.random.RandomState(0).randn(2,2))

print(X.shape)
```

```
(500, 2)
```

In [15]:
```
plt.scatter(X[:,0],X[:,1])
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x25fc0aac2e0>
```



5. Build the gaussian mixture model for the above dataset

Call the Gaussian Mixture Model Class ¶

In [32]:
```
gmm = GMM(k=3, max_iter=10)
gmm.fit(X)
predicted_label = gmm.predict(X)
```

In [40]:
```
print(np.unique(predicted_label, return_counts = True))
```

```
(array([0, 1, 2], dtype=int64), array([188, 146, 166], dtype=int64))
```

In [41]:
```
df = pd.DataFrame(columns=['Weight','Height','Label'])
df['Feature1'] =X[:,0]
df['Feature2'] =X[:,1]
df['Label'] =predicted_label
df.shape
```

Out[41]: (500, 5)

6. Plot the dataset

In [43]:
```
import seaborn as sns
sns.scatterplot(data =df,x='Feature1', y = 'Feature2',hue = 'Label')
#plt.scatter(X[:,0],X[:,1]s= predicted_label)
```

Out[43]: <AxesSubplot:xlabel='Feature1', ylabel='Feature2'>