2. PHẦN MÔ HÌNH TRỘN GAUSSIAN (GAUSSIAN MIXTURE MODEL)

2.1.Thuật toán EM - Xây dựng chương trình từ Numpy

Ví dụ 1: Dữ liệu 2D nhân tạo

Trong phần này chúng ta sử dụng dữ liệu nhân tạo với số chiều d = 2. Dữ liệu sẽ đươc tạo dưới dạng các cụm phân bố chuẩn, tuy nhiên ma trận hiệp phương sai sẽ được chọn để trục của cụm dữ liệu không song song với trục tọa độ.

```
def gen_data(k=3, dim=2, points_per_cluster=200, lim=[-20, 20]):
    '''
        Generates data from a random mixture of Gaussians in a given range. Will also
        plot the points in case of 2D. input:
            - k: Number of Gaussian clusters
            - dim: Dimension of generated points
            - points_per_cluster: Number of points to be generated for each cluster
            - lim: Range of mean values
        output:
            - X: Generated points (points_per_cluster*k, dim)
        '''
        x = []
        mean = random.rand(k, dim)*(lim[1]-lim[0]) + lim[0]
        for i in range(k):
            cov = random.rand(dim, dim+10)
            cov = np.matmul(cov, cov.T)
            _x = np.random.multivariate_normal(mean[i], cov, points_per_cluster)
            x += list(_x)
        x = np.array(x)
        if(dim == 2):
            fig = plt.figure()
            ax = fig.gca()
            ax.scatter(x[:,0], x[:,1], s=3, alpha=0.4)
            ax.autoscale(enable=True)
        return x
```

Phần khởi tạo cho thuật toán lặp

```
def __init__(self, k, dim, init_mu=None, init_sigma=None, init_pi=None, colors=None):
        '''
        Define a model with known number of clusters and dimensions.
        input:
            - k: Number of Gaussian clusters
            - dim: Dimension
            - init_mu: initial value of mean of clusters (k, dim)
                       (default) random from uniform[-10, 10]
            - init_sigma: initial value of covariance matrix of clusters (k, dim, dim)
                          (default) Identity matrix for each cluster
            - init_pi: initial value of cluster weights (k,)
                       (default) equal value to all cluster i.e. 1/k
            - colors: Color valu for plotting each cluster (k, 3)
                      (default) random from uniform[0, 1]
        '''
        self.k = k
        self.dim = dim
        if(init_mu is None):
            init_mu = random.rand(k, dim)*20 - 10
        self.mu = init_mu
        if(init_sigma is None):
            init_sigma = np.zeros((k, dim, dim))
```

```
            for i in range(k):
                init_sigma[i] = np.eye(dim)
        self.sigma = init_sigma
        if(init_pi is None):
            init_pi = np.ones(self.k)/self.k
        self.pi = init_pi
        if(colors is None):
            colors = random.rand(k, 3)
        self.colors = colors
```

Bước tính E:

- Theo công thức lý thuyết: $p(z_c|\mathbf{x}_n, \theta_t) = \frac{\pi_c N(\mu_{ct}, \Sigma_{ct}|\mathbf{x}_n)}{\sum_{k=1}^{C} \pi_k N(\mu_{kt}, \Sigma_{kt}|\mathbf{x}_n)}$ trong đó các ước lượng π_k, Σ_{kt} và μ_{kt} đều được lấy giá trị xấp xỉ có trong bước lặp t hiện tại.
- $N(\mu_{ct}, \Sigma_{ct}|\mathbf{x}_n)$ được tính bằng cách sử dụng hàm của thư viện numpy **multivariate_normal.pdf**

```
def e_step(self):
        '''
        E-step of EM algorithm.
        '''
        for i in range(self.k):
            self.z[:, i] = self.pi[i] * multivariate_normal.pdf(self.data,
mean=self.mu[i], cov=self.sigma[i])
        self.z /= self.z.sum(axis=1, keepdims=True)
```

Bước cực đại hóa (M-Step):

```
def m_step(self):
        '''
        M-step of EM algorithm.
        '''
        sum_z = self.z.sum(axis=0)
        self.pi = sum_z / self.num_points
        self.mu = np.matmul(self.z.T, self.data)
        self.mu /= sum_z[:, None]
        for i in range(self.k):
            j = np.expand_dims(self.data, axis=1) - self.mu[i]
            s = np.matmul(j.transpose([0, 2, 1]), j)
            self.sigma[i] = np.matmul(s.transpose(1, 2, 0), self.z[:, i] )
            self.sigma[i] /= sum_z[i]
```

Code đầy đủ có trong tệp Gaussian_Mix_EM.ipynb đính kèm.

Bài tập tự thực hành 1: Sử dụng đoạn code trên để áp dụng cho dữ liệu là phần đầu vào của tập dữ liệu hoa Iris (bỏ trường tên loại hoa). Sau khi phân cụm xong hãy đối sánh kết quả với các phân loại đúng.

Bài tập tự thực hành 2: Hãy sử dụng thuật toán K-means để phân cụm dữ liệu tự tạo đã có trong ví dụ 1 của phần này. So sánh và giải thích kết quả.

2.2. Sử dụng thư viện sk-learn

Ví dụ 2. Nhắc lại ví dụ ở bài lý thuyết:

Trong ví dụ này, chúng ta sử dụng dữ liệu là thông tin về chỉ số mua sắm và mức thu nhập có trong tệp dữ liệu shopping-data.csv. Tệp dữ liệu có 4 trường nhưng ta chỉ sử dụng 2 trường liên quan tới thu nhập và chỉ số mua sắm để tìm tương quan, các trường khác chỉ chứa thông tin ID nên ta bỏ qua.

Trong phần này ta sẽ sử dụng thư viện scikit-learn (sklearn) để thực hiện mô hình.

Các bước sẽ như sau:

(i) Đọc dữ liệu vào, chuẩn hóa (chú ý trong phần lý thuyết ta đã biết nếu kỳ vọng của các phân bố Gaussian là 0 thì tính toán sẽ tốt hơn); Ở trong code ta sẽ in ra một số thông tin của dữ liệu để xem đọc có đúng không.

(ii) Khởi tạo đối tượng của lớp mô hình trộn Gaussian thông qua hàm dựng

gm = GaussianMixture(n_components=K, covariance_type='full', random_state=0)

Ở đây số cụm được chia sẽ là n_component = K

(iii) Sau đó ta khớp dữ liệu của ta bằng đối tượng mô hình vừa tạo:

gm. fit(X_DATA)

(iv) Chúng ta vẽ dữ liệu ra dạng trực quan để có thể quan sát.

(v) Ở trong code dưới đây, chúng ta bổ sung thêm phần tìm xem K (số cụm nên dùng để chia dữ liệu) bao nhiêu là hợp lý nhất. Các làm của chúng ta ở đây đơn giản chỉ là thử với một dãy các giá trị số cụm, giá trị nào cho kết quả tốt nhất (trong số đó) thì sử dụng.

Dưới đây là các đoạn code:

Bước khai báo thư viện:

```
import numpy as np
import pandas as pd
import seaborn as sns
import itertools
from scipy import linalg
import matplotlib.pyplot as plt
import matplotlib.patheffects as PathEffects
from matplotlib.patches import Ellipse
from sklearn.preprocessing import MinMaxScaler
from sklearn.mixture import GaussianMixture
# Thư viện chứa model Gaussian Mixture
```

Bước (i)

```
data = pd.read_csv("D:\\Teach_n_Train\\Machine
                    Learning\\slide\\Mixture_Model\\code
                                    \\shopping-data.csv",
                header=0,
                index_col=0)
print(data.shape)
data.head()

# Lấy ra thu nhập va điểm shopping
X = data.iloc[:, 2:4].values

# Chuẩn hoá dữ liệu
std = MinMaxScaler()
X_std = std.fit_transform(X)
print(X_std.shape)
```

Bước (ii) và bước (iii)

```
# Khởi tạo đối tượng mô hình GaussianMixture
gm = GaussianMixture(n_components=5,
                     covariance_type='full',
                     random_state=0)
gm.fit(X_std)
print('means: \n', gm.means_)
print('covariances: \n ', gm.covariances_)
```

Bước (v): Ta nêu phần chọn số K tốt nhất trước:

```
lowest_bic = np.infty
bic = []
n_components_range = range(1, 7)
# cv_types = ['spherical', 'tied', 'diag', 'full']
cv_types = ['full', 'tied']
for cv_type in cv_types:
    for n_components in n_components_range:
        # Fit Gaussian mixture theo phương pháp huấn luyện EM
        gmm = GaussianMixture(n_components=n_components,
                                        covariance_type=cv_type)
        gmm.fit(X_std)
        bic.append(gmm.bic(X_std))
        # Gán model có BIC scores thấp nhất là model tốt nhất
        if bic[-1] < lowest_bic:
            lowest_bic = bic[-1]
            best_gmm = gmm

bic = np.array(bic)
color_iter = itertools.cycle(['navy', 'turquoise'])
clf = best_gmm
bars = []

# Vẽ biểu đồ BIC scores
plt.figure(figsize=(12, 8))
for i, (cv_type, color) in enumerate(zip(cv_types, color_iter)):
    xpos = np.array(n_components_range) + .2 * (i - 2)
    bars.append(plt.bar(xpos, bic[i * len(n_components_range):
                                  (i + 1) * len(n_components_range)],
                        width=.2, color=color))
plt.xticks(n_components_range)
plt.ylim([bic.min() * 1.01 - .01 * bic.max(), bic.max()])
plt.title('BIC score per model')
xpos = np.mod(bic.argmin(), len(n_components_range)) + .65 +\
    .2 * np.floor(bic.argmin() / len(n_components_range))
plt.text(xpos, bic.min() * 0.97 + .03 * bic.max(), '*', fontsize=14)
plt.xlabel('Number of components')
plt.legend([b[0] for b in bars], cv_types)
```

Bước (iv) – Hiển thị kết quả

```
def _plot_kmean_scatter(X, labels):
    '''
    X: dữ liệu đầu vào
    labels: nhãn dự báo
    '''
    # lựa chọn màu sắc
    num_classes = len(np.unique(labels))
    palette = np.array(sns.color_palette("hls", num_classes))

    # vẽ biểu đồ scatter
    fig = plt.figure(figsize=(12, 8))
    ax = plt.subplot()
    sc = ax.scatter(X[:,0], X[:,1], lw=0, s=40,
                                        c=palette[labels.astype(np.int)])

    # thêm nhãn cho mỗi cluster
    txts = []

    for i in range(num_classes):
        # Vẽ text tên cụm tại trung vị của mỗi cụm
        xtext, ytext = np.median(X[labels == i, :], axis=0)
        txt = ax.text(xtext, ytext, str(i), fontsize=24)
```

```
        txt.set_path_effects([
            PathEffects.Stroke(linewidth=5, foreground="w"),
            PathEffects.Normal()])
        txts.append(txt)
    plt.title('t-sne visualization')
```

Cuối cùng: Gọi và thực hiện 2 phương thức chính: Thực hiện mô hình trộn Gaussian và phương thức hiển thị dữ liệu ra mặt phẳng 2 chiều để dễ quan sát.

```
labels = best_gmm.predict(X_std)
plot_kmean_scatter(X_std, labels)
```

Chú ý với các ví dụ tiếp theo, chúng ta cần đọc dữ liệu đầy đủ và thiết lập vector dữ liệu X phù hợp (vì số chiều có thể lớn hơn). Sau đó trong phần hiển thị dữ liệu, chúng ta cần sử dụng mô hình phân tích thành phần chính (PCA) để giảm số chiều dữ liệu xuống còn 2 chiều để có thể hiển thị lên mặt phẳng.

2.3. **Ví dụ mở rộng.** Trong ví dụ này chúng ta sử dụng dữ liệu cho trong tệp đính kèm Sales_Transactions_Dataset_Weekly.csv hoặc tại link https://archive.ics.uci.edu/ml/machine-learning-databases/00396/ .

Mô tả dữ liệu: Dữ liệu chứa thông tin giao dịch bán hàng của 819 mã sản phẩm (Product ID) trong 819 dòng dữ liệu. Mỗi mã được đặc trưng bởi các thông tin: Mã (cột đầu); lượng giao dịch trong 52 tuần (week 0 – week 51), sau đó có thông tin chuẩn hóa của giao dịch các tuần (ta sử dụng thông tin này hoặc thông tin theo tuần – chỉ chọn 1 trong 2).

Yêu cầu: Hãy dựa vào ví dụ 1, mở rộng áp dụng với dữ liệu trên đây. Biết rằng số cụm tối ưu sẽ nằm trong khoảng từ 5 đến 15, hãy xác định số cụm và sau đó phân các mã sản phẩm thành các cụm dựa trên tiêu chí lượng giao dịch đã cho trong dữ liệu.