GMM实践

- 助教：陈怿飏
- 522023330016@smail.nju.edu.cn

使用GMM模型对二维数据进行聚类

预备知识

高斯混合模型（GMM）是一种**概率密度估计方法**，假设数据是多个**高斯分布的加权混合**，形式化的表达为：

$$P(X)=\sum_{k=1}^{K}\pi_k\mathcal{N}\left(X\mid\mu_k,\Sigma_k\right)$$

这里K是高斯分布的数量，π_k是第k个高斯分布的权重，满足：

$$0\leq\pi_k\leq 1,\quad \sum_{k=1}^{K}\pi_k=1$$

$\mathcal{N}\left(X\mid\mu_k,\Sigma_k\right)$是第$k$个高斯分布：

$$\mathcal{N}\left(X\mid\mu_k,\Sigma_k\right)=\frac{1}{(2\pi)^{d/2}\left|\Sigma_k\right|^{1/2}}\exp\left(-\frac{1}{2}(X-\mu_k)^T\Sigma_k^{-1}\left(X-\mu_k\right)\right)$$

求解方式：EM算法

EM 算法交替执行 **E-step（期望步骤）** 和 **M-step（最大化步骤）** 直到收敛。

E-Step

计算**每个数据点 X_i 属于每个簇 k 的概率**：

$$r_{ik}=P\left(Z_i=k\mid X_i\right)=\frac{\pi_k\mathcal{N}\left(X_i\mid\mu_k,\Sigma_k\right)}{\sum_{j=1}^{K}\pi_j\mathcal{N}\left(X_i\mid\mu_j,\Sigma_j\right)}$$

M-step：更新参数

使用新的观测值来更新GMM模型的参数：

1. 更新混合权重：

$$\pi_k=\frac{1}{N}\sum_{i=1}^{N}r_{ik}$$

2. 更新均值：

$$\mu_k=\frac{\sum_{i=1}^{N}r_{ik}X_i}{\sum_{i=1}^{N}r_{ik}}$$

3. 更新协方差矩阵：

$$\Sigma_k = \frac{\sum_{i=1}^{N} r_{ik} (X_i - \mu_k)(X_i - \mu_k)^T}{\sum_{i=1}^{N} r_{ik}}$$

使用GMM进行二维数据的聚类

导入包

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
```

EM求解算法

```
def apply_em(X, pi, mu, Sigma, max_iter=100):
    """
    Simplified Expectation-Maximization (EM) for Gaussian Mixture Model (GMM).
    """
    n, d = X.shape
    k = len(pi)
    r = np.zeros((n, k))
    coeffs, rvals = [], []  # Lists to store parameter updates and responsibility values

    # Store the initial state before any iteration
    coeffs.append({"pi": pi.copy(), "mu": mu.copy(), "Sigma": Sigma.copy()})
    rvals.append(np.ones((n, k)) / k)  # Uniform responsibility for initialization

    for it in range(max_iter):
        # E-step: Compute responsibilities (posterior probabilities of clusters)
        for i in range(k):
            diff = X - mu[i]
            r[:, i] = pi[i] * np.exp(-0.5 * np.sum((diff @ np.linalg.inv(Sigma[i])) *
diff, axis=1))
            r[:, i] /= np.sqrt(np.linalg.det(Sigma[i]) * (2 * np.pi) ** d)
        r /= r.sum(axis=1, keepdims=True)  # Normalize to ensure probabilities sum to 1

        # M-step: Update parameters using the computed responsibilities
        Nk = r.sum(axis=0)
        pi = Nk / n
        mu = [(X * r[:, i][:, np.newaxis]).sum(axis=0) / Nk[i] for i in range(k)]  #
Update means
        Sigma = [(np.dot((r[:, i][:, np.newaxis] * (X - mu[i])).T, (X - mu[i])) / Nk[i])
for i in range(k)]

        # Store parameters as dictionary instead of tuple
        coeffs.append({"pi": pi.copy(), "mu": mu.copy(), "Sigma": Sigma.copy()})
        rvals.append(r.copy())
    return {"coeffs": coeffs, "rvals": rvals}
```

生成数据

```
def generate_data(n_samples_cluster1=200, n_samples_cluster2=200, var1=1, var2=1):
    """Generate synthetic 2D data for GMM fitting with specified cluster sizes and 
variances."""
    np.random.seed(42)
    cov1 = np.identity(2) * var1  # Covariance matrix for cluster 1
    cov2 = np.identity(2) * var2  # Covariance matrix for cluster 2
    cluster1 = np.random.multivariate_normal(mean=[-2, 2], cov=cov1, 
size=n_samples_cluster1)
    cluster2 = np.random.multivariate_normal(mean=[2, -2], cov=cov2, 
size=n_samples_cluster2)
    X = np.vstack((cluster1, cluster2))
    return X
```

该函数定义了两个聚类生成的数据，可以在函数内部设置聚类的中心`mean`。

```
# Generate synthetic dataset
X = generate_data(n_samples_cluster1=200, n_samples_cluster2=200,
                  var1=3, var2=1)
```

这里让每个聚类具有200个样本，方差分别是3和1。

初始化GMM参数

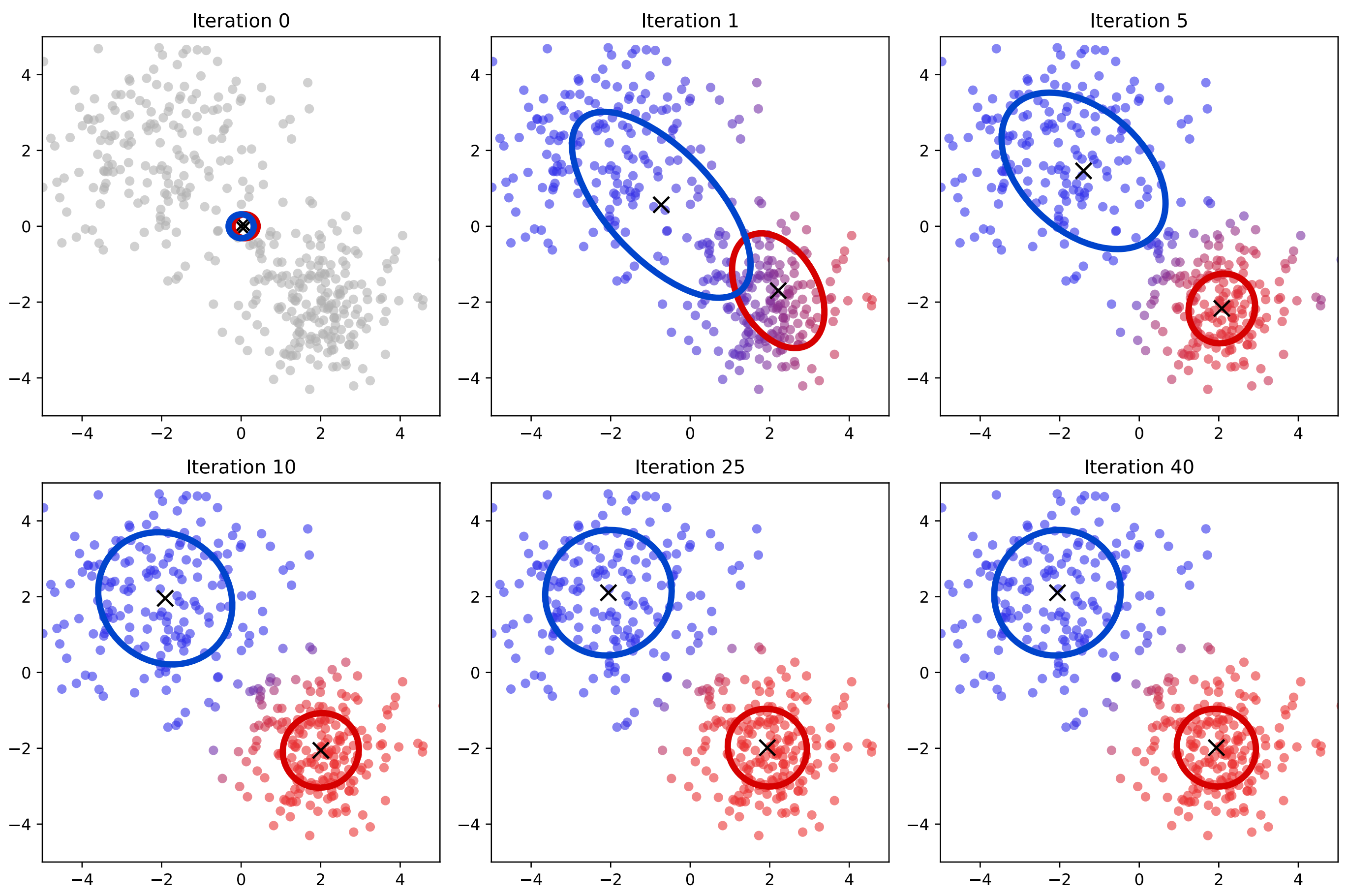
```
# Initial parameters
mu1, mu2 = np.array([0.1, 0]), np.array([0, 0])

Sigma1, Sigma2 = np.identity(2) * 0.1, np.identity(2) * 0.1
pi = [0.1, 0.9]
mu, Sigma = [mu1, mu2], [Sigma1, Sigma2]
```

这里定义了GMM模型的3个参数。

算法求解

```
# Apply Expectation-Maximization (EM) algorithm
res = apply_em(X, pi, mu, Sigma)

# Visualization setup
hist_index = [0, 1, 5, 10, 25, 40]
fig, ax = plt.subplots(2, 3, figsize=(12, 8))
ax = ax.ravel()

initial_state = {"pi": pi, "mu": mu, "Sigma": Sigma}  # Store initial state
```

```
for ix, axi in zip(hist_index, ax):
    params = res["coeffs"][ix]
    pi, mu, Sigma = params["pi"], params["mu"], params["Sigma"]
    r = res["rvals"][ix] if ix > 0 else np.ones_like(res["rvals"][ix])

    plot_mixtures(X, mu, pi, Sigma, r, ax=axi, iteration=ix, initial_state=initial_state
if ix == 0 else None)
    axi.set_title(f"Iteration {ix}")

plt.tight_layout()
plt.show()
```



参考链接

- https://probml.github.io/notebooks#mix_gauss_demo_faithful.ipynb

思考与分析

1. GMM的参数 μ 和 π 最终会收敛到什么值，假设已知生成数据的分布信息
2. GMM参数的初始化会对聚类结果产生影响吗？尝试下面的初始化参数，你有什么结论

```python
mu1, mu2 = np.array([-3, -3]), np.array([3, 3])
mu1, mu2 = np.array([-2, 2]), np.array([2, -2])
```