机器学习实验报告

强基数学 *002* 吴天阳 *2204210460*

本部分将对 K-均值聚类和混合高斯进行实验.

1 K-均值聚类

设数据集为 $\{\boldsymbol{x}_1,\cdots,\boldsymbol{x}_N\}$, $\boldsymbol{x}_i\in\mathbb{R}^D$, 由 N 个在 $\mathbb{R}^D$ 中的观测值构成. K-均值聚类 (K-means Clustering) 的目标是将数据集划分为 K 簇 (Cluster), 假设 $\boldsymbol{\mu}_k\in\mathbb{R}^D$, $(k=1,\cdots,K)$ 代表簇的中心，我们的目标是最小化每个数据点到最近 $\boldsymbol{\mu}_k$ 的距离平方和.

为方便描述每个数据点的分类，引入二进制指标集 $r_{nk}\in\{0,1\}$，如果数据点 $\boldsymbol{x}_n$ 分配到簇 k，那么 $r_{nk}=1$ 且 $r_{nj}=0$, $(j\neq k)$. 根据该编码方法，可以定义以下最小化目标**失真度量 (distortion measure)**：

$$
J=\sum_{n=1}^N\sum_{k=1}^K r_{nk}||\boldsymbol{x}_n-\boldsymbol{\mu}_k||^2 \tag{1}
$$

K-均值聚类也是用了 EM 算法，首先给出如何划分为 EM 算法：

- E 步：固定 μ_k，求使得 J 最小化的 r_{nk}（求出期望）.
- M 步：固定 r_{nk}，求使得 J 最小化的 μ_k（最大化）.

E 步 当固定 μ_k 时，由于 J 关于 r_{nk} 是线性的，即不同的数据 $\boldsymbol{x}_n$ 之间相互独立，所以可以对于每个样本单独进行优化，对于样本 $\boldsymbol{x}_n$ 的 r_{nk} 满足求解以下最优化问题：

$$
\begin{aligned}
\min_{r_{nk}\in\{0,1\}}\ &\sum_{k=1}^K r_{nk}||\boldsymbol{x}_n-\boldsymbol{\mu}_k||^2\\
s.t.\ &\sum_{k=1}^K r_{nk}=1
\end{aligned}
\Rightarrow r_{nk}=\begin{cases}1, & \text{当 } k=\operatorname*{arg\,min}_{1\leqslant j\leqslant K}||\boldsymbol{x}_n-\boldsymbol{\mu}_j||^2,\\ 0, & \text{否则}.\end{cases}
$$

不难发现，最优化结果正好就表明只需将每个 $\boldsymbol{x}_n$ 分配到最近的簇中心 $\boldsymbol{\mu}_k$ 上.

M 步 当固定 r_{nk} 时，由于 J 是关于 μ_k 的二次函数，所以可以通过导数为零确定最小化点：

$$
2\sum_{n=1}^N r_{nk}(\boldsymbol{x}_n-\boldsymbol{\mu}_k)=0\Rightarrow\boldsymbol{\mu}_k=\frac{\sum\limits_{1\leqslant n\leqslant N} r_{nk}\boldsymbol{x}_n}{\sum\limits_{1\leqslant n\leqslant N} r_{nk}}
$$

表达式中分母是簇 k 分配到的数据的个数，所以 μ_k 的更新就是所有簇 k 分配到的数据点的平均值，因此被称为 K-均值，算法总共两步：将每个数据点分配到最近的簇，重新计算簇均值以替代新的簇. 由于该方法每一步都会使得 J 单调递减，所以算法收敛性显然，但是它只能收敛到 J 的局部最小值.

2 混合高斯模型

混合高斯模型 (Gaussian Mixture Model, GMM) 是一种基于概率的聚类模型，首先我们可以将所有的变量均视为随机变量，包括观测变量 x 和隐变量 θ, ω，它们都服从某个概率分布，于是可以将模型参数求解转化为求解 $p(\theta|D)$，即根据数据集 D 求出模型参数 θ 的后验分布，所以可以用最大似然 (MLE) 方法求解.

用上述方法理解重新聚类问题：混合概率模型的隐变量就是 y 表示数据的类别种类，服从分布 $p(y=k)=\pi_k$（表示全部的 $\boldsymbol{x}$ 来自类别 k 的概率大小），从**数据生成**的角度理解，第 k 个类别的数据 $\boldsymbol{x}$ 应来自与 y 相关的某个分布 $p(\boldsymbol{x}|y=k)$ 中（不妨令该分布为多维正态分布），于是二者的联合分布为

$$p(\boldsymbol{x}, y)=p(y)p(\boldsymbol{x}|y) \xlongequal{y=k} \pi_k\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

通过联合分布我们又可以求出**数据预测**的结果：

$$p(y|\boldsymbol{x})=\frac{p(\boldsymbol{x}, y)}{p(\boldsymbol{x})} \xlongequal{y=k} \frac{\pi_k\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{k=1}^K \pi_k\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)}$$

将全部的参数简记为 $\theta=(\mu_1, \cdots, \mu_K, \sigma^2, \pi_1, \cdots, \pi_K)$，于是关于 θ 的 MLE 为

$$\max_\theta \prod_{i=1}^N p(x_i|\theta)=\prod_{i=1}^N\sum_{k=1}^K p(x_i, y_i=k|\theta)=\prod_{i=1}^N\sum_{k=1}^K p(y_i=k|\theta)p(x_i|y_i=k, \theta) \tag{2}$$

2.1 由 GMM 导出 K-均值

我们考虑一个 GMM 的特殊情况，假设所有的方差均相同，即 $\Sigma=\Sigma_k, (k=1, \cdots, K)$，并令 $\pi_k=\dfrac{1}{n}\sum\limits_{i=1}^n r_{nk}$，则 $\boldsymbol{\mu}_k$ 似然函数为

$$\begin{aligned} L &= \prod_{i=1}^N\sum_{k=1}^K \pi_k p(x_i)=\prod_{i=1}^N\sum_{k=1}^K \pi_k\mathcal{N}(x_i|\boldsymbol{\mu}_k, \Sigma) \\ &= \prod_{i=1}^N\sum_{k=1}^K \pi_k(2\pi)^{-\frac{k}{2}}|\Sigma|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right\} \end{aligned}$$

取对数后得到 MLE 为

$$\max_{\boldsymbol{\mu}_k} -\sum_{i=1}^N\sum_{k=1}^K \pi_k||x_i-\boldsymbol{\mu}_k||^2=\min_{\boldsymbol{\mu}_k}\sum_{i=1}^N\sum_{k=1}^K \pi_k||x_i-\boldsymbol{\mu}_k||^2$$

结果与 K-均值 (1) 式中的失真度量 J 的区别仅需将 π_k 换为 r_{nk}，而这个转换就会将聚类方法从 GMM 的**软分类**变为 K-均值的**硬分类**.

其次我们可以利用 GMM 的预测方法证明：当方差相同时（K-均值的分类边界），分类边界是线性的. 假设数据 $\boldsymbol{x}$ 分到 i, j 类别具有相同的可能性时，也就是 $p(y=i|\boldsymbol{x})=$

$p(y=j|\boldsymbol{x})$，于是：

$$\begin{aligned}0 &= \log\frac{p(y=i|\boldsymbol{x})}{p(y=j|\boldsymbol{x})} = \log\frac{p(\boldsymbol{x}|y=i)\frac{p(y=i)}{p(\boldsymbol{x})}}{p(\boldsymbol{x}|y=j)\frac{p(y=j)}{p(\boldsymbol{x})}} = \log\frac{p(\boldsymbol{x}|y=i)\pi_i}{p(\boldsymbol{x}|y=j)\pi_j}\\ &= \log\frac{\pi_i}{\pi_j} + ||\boldsymbol{x}-\boldsymbol{\mu}_i||^2 - ||\boldsymbol{x}-\boldsymbol{\mu}_j||^2 = \log\frac{\pi_i}{\pi_j} + 2(\boldsymbol{\mu}_j^T - \boldsymbol{\mu}_i^T)\boldsymbol{x} + \boldsymbol{\mu}_i^T\boldsymbol{\mu}_i - \boldsymbol{\mu}_i^T\boldsymbol{\mu}_j\\ &\Rightarrow \boldsymbol{w}^T\boldsymbol{x} = \boldsymbol{b}\end{aligned}$$

说明如果 $\boldsymbol{x}$ 分为类别 i,j 的可能性相同时，则 $\boldsymbol{x}$ 一定处于直线 $\boldsymbol{w}^T\boldsymbol{x}=\boldsymbol{b}$ 上．同理，对于一般情况 Σ_k，计算得到的分类边界为 $\boldsymbol{x}^TW\boldsymbol{x}+\boldsymbol{w}^T\boldsymbol{x}+\boldsymbol{c}$，说明 GMM 的分类边界就是二次函数.

2.2 使用 EM 算法进行参数求解

观察 (2) 式，对其取对数仍然无法将内部的求和符号展开成线性表示，所以难以求出极值，考虑基于 $p(x,y)$ 求 θ 的 MLE：

$$\max_\theta\prod_{i=1}^N p(\boldsymbol{x}_i,y_i|\theta) \propto \sum_{i=1}^N\log p(\boldsymbol{x}_i,y_i|\theta) = \sum_{i=1}^N\sum_{k=1}^K p(y_i=k|\boldsymbol{x}_i,\theta)\log p(\boldsymbol{x}_i,y_i|\theta)$$

M 步：假设我们在 $t-1$ 步已经得到了参数估计值 θ^{t-1}，于是可以在第 t 步建立 Q 函数，然后最大化该函数得到 θ^t

$$\max_{\theta^t} Q(\theta^t|\theta^{t-1}) = \sum_{i=1}^n\sum_{k=1}^K p(y_i=k|\boldsymbol{x}_i,\theta^{t-1})\log p(\boldsymbol{x}_i,y_i=k|\theta^t) \tag{3}$$

E 步：就是在 $t-1$ 步时，基于 θ^{t-1} 计算数据 $\boldsymbol{x}_i$ 从属于每个类别的概率：

$$R_{i,k}^{t-1} = p(y_i=k|\boldsymbol{x}_i,\theta^{t-1}) = \frac{\pi_k^{t-1}\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k^{t-1},\Sigma_k^{t-1})}{\sum_{k=1}^K\pi_k^{t-1}\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k^{t-1},\Sigma_k^{t-1})}$$

而 E 步的计算结果，就是 M 步中 Q 函数中 $\log p(\boldsymbol{x}_i,y_i=k|\theta^t)$ 前的加权系数.

我们先不探讨上述方法的收敛性，通过 Lagrange 乘子法和令导数为零可以求解求解 (3) 式：

由 Lagrange 乘子法：$\nabla_{\pi_k^t}Q + \lambda\nabla_{\pi_k^t}\left(\sum_{k=1}^K\pi_k - 1\right) = 0 \Rightarrow \dfrac{\sum_{i=1}^N R_{i,1}^{t-1}}{\pi_1^t} = \cdots = \dfrac{\sum_{i=1}^N R_{i,k}^{t-1}}{\pi_k^t}$

$$\xRightarrow{\sum_{k=1}^K R_{i,k}^{t-1}=1} \pi_k^t = \frac{\sum_{i=1}^N R_{i,k}^{t-1}}{N}$$

$$\nabla_{\boldsymbol{\mu}_k^t}Q = 0 \Rightarrow \sum_{i=1}^N R_{i,k}^{t-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_k^t) = 0 \Rightarrow \boldsymbol{\mu}_k^t = \sum_{i=1}^N w_{ik}\boldsymbol{x}_i \tag{4}$$

$$\nabla_{\Sigma_k^t}Q = 0 \Rightarrow \sum_{i=1}^N R_{i,k}^{t-1}(-\Sigma_k^t + (\boldsymbol{x}_i-\boldsymbol{\mu}_k^t)^T(\boldsymbol{x}_i-\boldsymbol{\mu}_k^t)) = 0 \Rightarrow \Sigma_k^t = \sum_{i=1}^N w_{ik}(\boldsymbol{x}_i-\boldsymbol{\mu}_k^t)^T(\boldsymbol{x}_i-\boldsymbol{\mu}_k^t)$$

其中 $w_{ik} = \dfrac{R_{i,k}^{t-1}}{\sum_{i=1}^N R_{i,k}^{t-1}}$. 通过上式中的标红的部分，就可以得到第 t 步下的新参数值.

2.3 EM 算法收敛性证明

首先引入一个求解函数极值的技术：若要求解 $f(x)$ 的极小值，可以先取其定义域上任意一点 x_0，再找到一个在 $(x_0, f(x_0))$ 处与 f 相切的函数 $g(x)$，并且要求 $g(x)$ 是 $f(x)$ 的上界，令 $x_1 \leftarrow \arg\min\limits_x g(x)$（求极大值反之亦然），根据该方法进行迭代即可得到 $f(x)$ 的极小值点.

下面推到一个关于 θ 的 MLE 重要结论：

$$
\begin{aligned}
\log p(x|\theta) &= \int_Y q(y)\log p(x|\theta)\,\mathrm{d}y = \int_Y q(y)\log\frac{p(x,y|\theta)}{p(y|x,\theta)}\frac{q(y)}{q(y)}\,\mathrm{d}y \\
&= \underbrace{\int_Y q(y)\log p(x,y|\theta)\,\mathrm{d}y}_{\text{核函数}} - \underbrace{\int_Y q(y)\log q(y)\,\mathrm{d}y}_{\text{与}\theta\text{无关}} + \underbrace{\int_Y q(y)\log\frac{q(y)}{p(y|x,\theta)}\,\mathrm{d}y}_{\mathrm{KL}(q||p)}
\end{aligned}
$$

注意到右边第三项正好是 p, q 的 KL 散度，于是有 $\mathrm{KL}(q||p) \geqslant 0$，于是前两项构成 $\log p(x|\theta)$ 的下界，要求极大似然的极大值，第二项与 θ 无关，所以只需对第一项求即可. 注意上式只讨论了一个数据，极大似然是对所有数据的对数似然求和得到：

$$
\max_\theta \sum_{i=1}^N \int_Y q(y)\log p(\boldsymbol{x}_i, y|\theta)\,\mathrm{d}y
$$

于是当我们取 $q(y) = p(y|\boldsymbol{x},\theta)$ 时，KL 散度正好为 0，极大似然对应的 θ^* 可以通过以下迭代式求解：

$$
\theta^* \leftarrow \arg\max_{\theta^*} \sum_{i=1}^N \int_Y p(y|\boldsymbol{x}_i,\theta)\log p(\boldsymbol{x}_i, y|\theta^*)\,\mathrm{d}y
$$

将积分号换为求和符号就可以得到 EM 算法中 M 步的 (3) 式. 由上面引入的函数极值求解技术，可以证明每次 EM 都可以使得 MLE 下降，从而达到极小值点，说明 EM 算法具有收敛性.

3 实验步骤与结果分析

3.1 K-均值聚类

数据集下载：Kaggle - Old faithful. 数据集规模：$N = 272$，$D = 2$，N 行 D 列，使用 K-means 对其进行分类.

Algorithm 1 K-均值聚类

```python
x = pd.read_csv("faithful.xls", index_col=0).to_numpy()
def normalize(x): return (x - np.mean(x)) / np.std(x)
x = np.apply_along_axis(normalize, 0, x)  # 按照列进行归一化
fig, axs = plt.subplots(2,3,figsize=(9,6))
def getdis(x, mu):
    dis = []
    for i in range(mu.shape[0]):
        dis.append(np.sqrt(np.sum(np.power(x-mu[i], 2))))
    return dis
mu = np.array([[-1, 1], [1, -1]])  # 初始化
for cnt, ax in enumerate(axs.reshape(-1)):
    # calculate the distance to each cluster
    dis = np.array([getdis(x[i], mu) for i in range(x.shape[0])])
    r = np.argmin(dis, axis=1)  # E 步
    plot(...)  # 绘制图像
    # M 步
    mu = np.array([np.mean(x[r==i], axis=0) for i in range(mu.shape[0])])
plt.show()
```

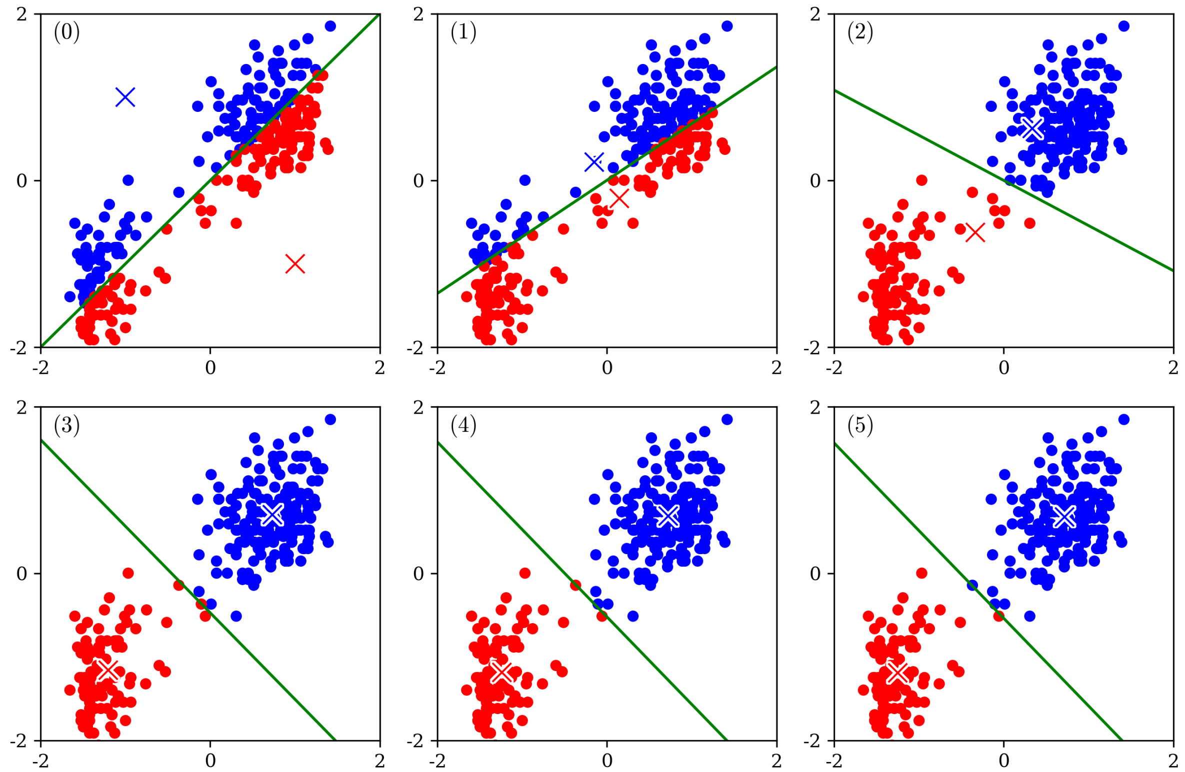

图 1: K-均值聚类

使用 K-均值做图像分割与图像压缩方法非常直接，假设图像是 $n \times m$ 的三通道像素，首先将图像拉直产生 $N = n \times m$ 个数据，每个数据的维数均为 $D = 3$，于是用 K-均值找到 K 个簇中心 $\boldsymbol{\mu}_k$，最后再用每个像素点从属的簇中心代替即可得到压缩后的图像. 这样我们只需存储原图像每个像素从属的簇编号，并记录下 $\boldsymbol{\mu}_k$，从而对图像大小进行压缩. 实现上使用的是 scikit-learn 库，因为图像较大，使用一般的线性算法速度太慢，在 scikit 中，K-均值使用了 KD 树进行加速，底层用 C++ 实现速度上有较大的提升.

Algorithm 2 K-均值聚类图像分割

```python
from sklearn.cluster import KMeans
X = img.reshape(-1, 3)
fig, axs = plt.subplots(1, 4, figsize=(12, 4))
for k, ax in zip([2, 3, 10], axs):
    kmeans = KMeans(n_clusters=k).fit(X)  # 数据拟合
    pred = np.array([kmeans.cluster_centers_[i] for i in
    ↪  kmeans.labels_]).reshape(img.shape)  # 数据预测，转换为对应的聚类中心
    img_show(pred, ax, f"$K={k}$")
img_show(img, axs[-1], "Original Image")
plt.show()
```

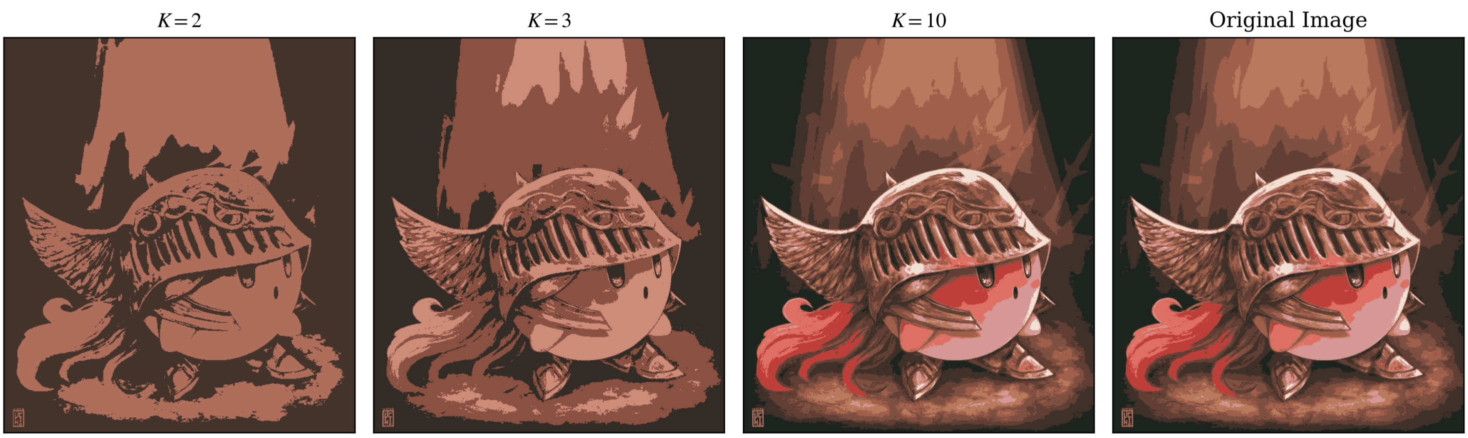

图 2: K-均值聚类图像分割

3.2 混合高斯模型 GMM

我对自己生成的数据使用了 K-均值和 GMM 进行聚类，并比较二者的区别. 数据生成方法：通过固定三个高斯分布，每个高斯下随机分布生成 1000 个数据

$$
\begin{aligned}
&\mathcal{N}\left(\mu_1 = (1, 1)^T, \Sigma = 0.3I + \varepsilon\right), \\
&\mathcal{N}\left(\mu_1 = (-1.5, 0)^T, \Sigma = 0.2I + \varepsilon\right), \\
&\mathcal{N}\left(\mu_1 = (1, -1.5)^T, \Sigma = 0.1I + \varepsilon\right).
\end{aligned}
$$

其中 ε 为 Gauss 噪声，即来自高斯分布的 2×2 随机数据作为方差的偏移量. 下面图3中展示了 K-均值的聚类效果，不难看到，最终聚类中心位置基本正确，但分类边界为硬分

类，无法很好的对边界进行处理. 而图4中展示了 GMM 的聚类效果，可以看出，由于 GMM 的软分类性质，所以可以很好得处理边界数据，但 GMM 算法对初值点的选取很重要，否则容易发生两个聚类中心重合的问题.

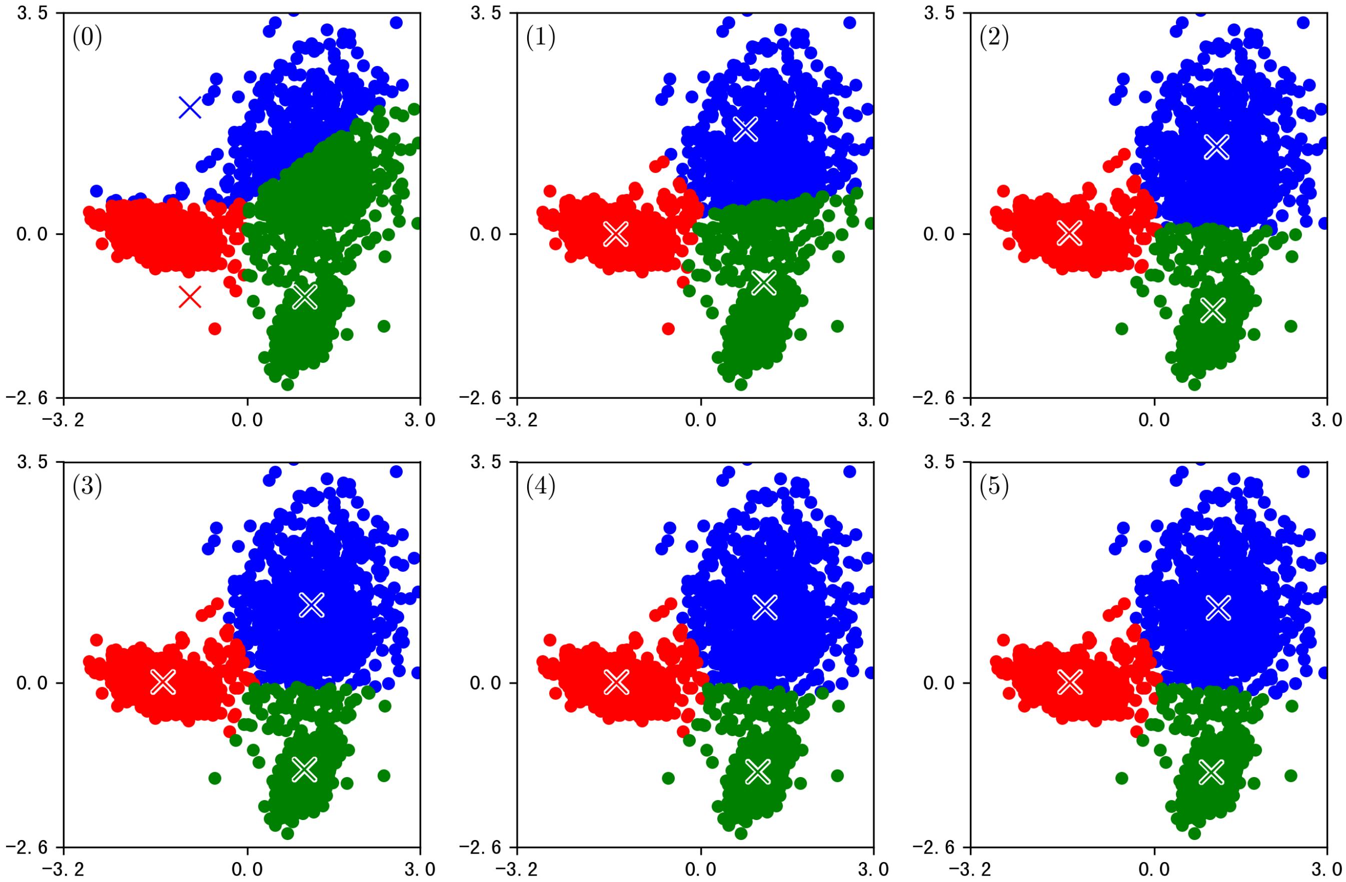

图 3: K-均值分类结果

Algorithm 3 GMM 算法

```
X = ...  # 初始化数据集
# 参数初始化
mu = np.array([[-1, 2], [-1, -1], [1, -1]])
sigma = np.array([0.1*np.eye(2) for _ in range(K)])
pi = np.full(3, 1/K)
def calc_normal(x, mu, sigma):  # 计算多维正态分布
    return np.power(2*np.pi, -K/2) * np.linalg.det(sigma) * np.exp(-0.5*np.dot(np.dot(x-mu, np.linalg.inv(sigma)), (x-mu).T))
for T in range(6):
    # E 步
    R = np.array([[calc_normal(X[i], mu[j], sigma[j]) for j in range(3)] for i in range(len(X))])
    plot(...)  # 绘制图像
    # M 步
    w = np.concatenate([R[:,j].reshape(-1, 1)/np.sum(R[:,j]) for j in range(K)], axis=1)
    pi = np.concatenate([R[:,j].reshape(-1, 1)/N for j in range(K)], axis=1)
```

```
    sigma = np.array([np.sum([w[i,k] * np.dot((X[i]-mu[k]).reshape(-1,1), (X[i]-mu[k]).reshape(1,-1)) for i in range(N)], axis=0) for k in range(K)])
    mu = np.array([[np.dot(X[:,i], w[:,k]) for i in range(2)] for k in range(K)])
plt.show()
```

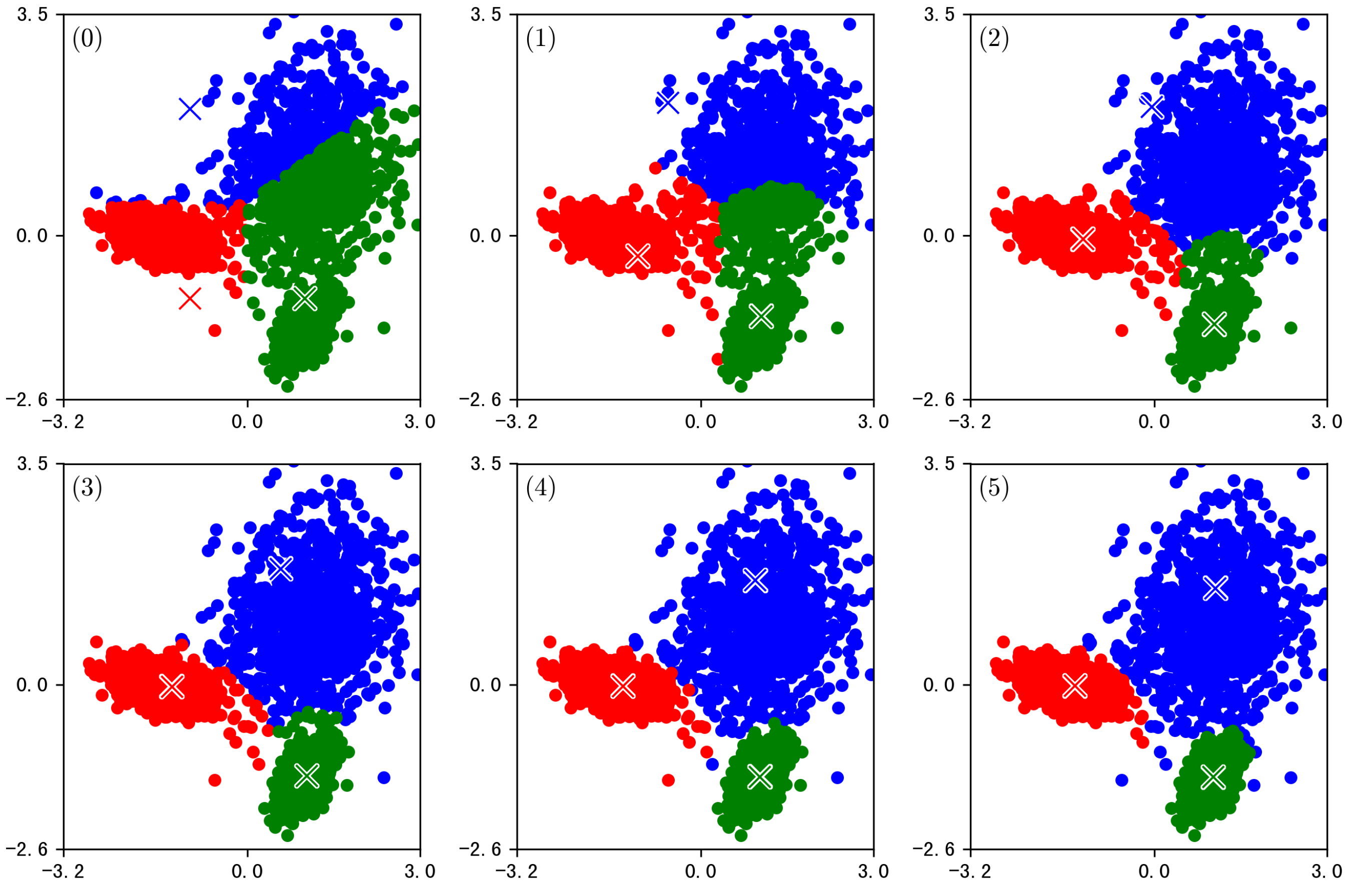

图 4: GMM 聚类结果

4 结论与讨论