高斯混合模型的EM算法

混合高斯模型

高斯混合模型的概率分布可以写成多个高斯分布的线形叠加，即

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \mu_k, \mathbf{\Sigma}_k)$$

引入一个K维的二值随机变量$\mathbf{z}$, 采用“1-of-K”编码，其中一个特定的元素z_k等于1，其余所有的元素都等于0。 于是z_k的值满足$z_k \in \{0, 1\}$且$\sum_k z_k = 1$，并且我们看到根据哪个元素非零，向量$\mathbf{z}$有K个可能的状态。$\mathbf{z}$的边缘概率分布可以根据混合系数π_k进行赋值，即

$$p(z_k = 1) = \pi_k$$

其中参数$\{\pi_k\}$必须满足

$$0 \leqslant \pi_k \leqslant 1$$

以及

$$\sum_{k=1}^{K} \pi_k = 1$$

由于$\mathbf{z}$使用了“1-of-K”编码，也可以将这个概率分布写成

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

对于$\mathbf{z}$给定的一个值， $\mathbf{x}$的条件概率分布是一个高斯分布

$$p(\mathbf{x} \mid z_k = 1) = \mathcal{N}(\mathbf{x} \mid \mu_k, \mathbf{\Sigma}_k)$$

类似的也可以写成

$$p(\mathbf{x} \mid \mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x} \mid \mu_k, \mathbf{\Sigma}_k)^{z_k}$$

$\mathbf{x}$的边缘概率分布可以通过将联合概率分布对所有可能的$\mathbf{z}$求和的方式得到，即

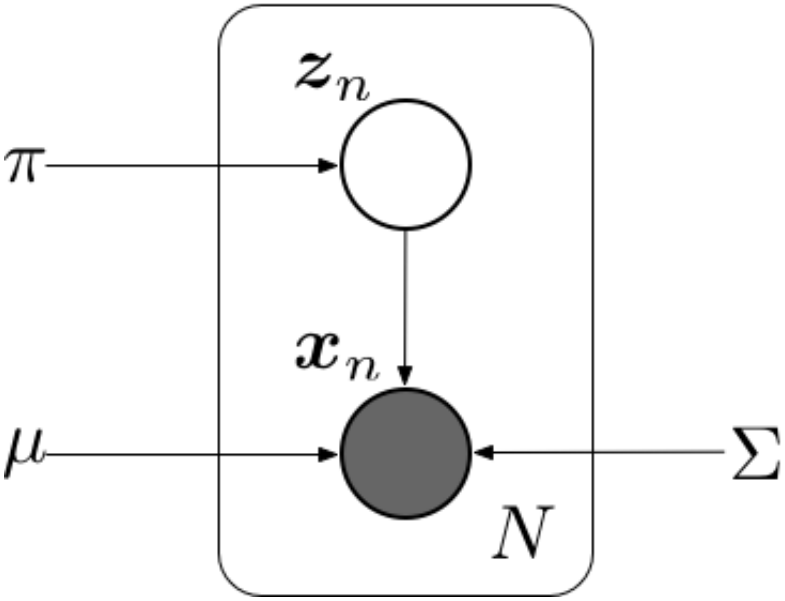
$$p(\mathbf{x}) = \sum_{z} p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \mu_k, \mathbf{\Sigma}_k)$$

于是我们找到了一个将隐变量$\mathbf{z}$显示写出的一个高斯混合分布等价公式。对联合概率分布$p(\mathbf{x}, \mathbf{z})$而不是对$p(\mathbf{x})$进行操作，会产生计算上极大的简化。

另一个有重要作用的量是给定$\mathbf{x}$的情况下，$\mathbf{z}$的后验概率$p(\mathbf{z} \mid \mathbf{x})$。用$\gamma(z_k)$表示$p(z_k = 1 \mid \mathbf{x})$，其值可由贝叶斯定理给出

$$\begin{aligned}\gamma(z_k) = p(z_k = 1 \mid \mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x} \mid z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x} \mid z_j = 1)} \\ &= \frac{\pi_k p(\mathbf{x} \mid z_k = 1)}{\sum_{j=1}^{K} \pi_j p(\mathbf{x} \mid z_j = 1)}\end{aligned}$$

可以将π_k看成是$z_k = 1$的先验概率，将$\gamma(z_k)$看成是观测到$\mathbf{x}$之后，对应的后验概率。



假设我们有观测数据集$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$，我们希望使用混合高斯来对数据建模。可以将这个数据集标示为$N \times D$的矩阵$\mathbf{X}$，其中第n行为$\mathbf{x}_n^{\top}$。类似的，对应的隐变量被表示为一个$N \times K$的矩阵$\mathbf{Z}$，它的行为$\mathbf{z}_n^{\top}$，可以使用上图所示的图模型来表示独立同分布数据集的高斯混合模型。$\mathbf{X}$的对数似然函数为

$$\log p(\mathbf{X} \mid \pi, \mu, \mathbf{\Sigma}) = \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \mu_k, \Sigma_k) \right\}$$

最大化高斯混合模型的对数似然函数比单一的高斯分布的情形更加复杂。因为对k的求和出现在了对数内部；*如果令导数等于零，不会得到一个解析解*。使用基于梯度的优化方法可以得到解，但现在考虑另一种可行方法，称为**EM算法**。

EM 算法

期望最大化算法，也叫EM算法，是寻找潜在变量的概率模型的最大似然解的一种通用方法。考虑一个概率模型，其中所有的观测变量记作$\mathbf{X}$，所有隐含变量记作$\mathbf{Z}$。联合概率分布$p(\mathbf{X}, \mathbf{Z} \mid \theta)$由一组参数$\theta$控制，目标是最大化似然函数

$$p(\mathbf{X} \mid \theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)$$

这里，假设$\mathbf{Z}$是离散的。直接优化$p(\mathbf{X} \mid \theta)$比较困难，但是最优化完整数据似然函数$p(\mathbf{X}, \mathbf{Z} \mid \theta)$就容易得很多。接下来，引入一个定义在隐变量$\mathbf{Z}$上的分布$p(\mathbf{Z})$。对任意$p(\mathbf{Z})$，如下分解成立

$$\log p(\mathbf{X} \mid \theta) = \mathcal{L}(q, \theta) + \mathrm{KL}(q \parallel p)$$

其中

$$\mathcal{L}(q, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log\left\{ \frac{p(\mathbf{X}, \mathbf{Z} \mid \theta)}{q(\mathbf{Z})} \right\}$$

$$\mathrm{KL}(p \parallel q) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \log\left\{ \frac{p(\mathbf{Z} \mid \mathbf{X}, \theta)}{q(\mathbf{Z})} \right\}$$

$\mathcal{L}(q, \theta)$是概率分布$q(\mathbf{Z})$的一个*范函*，并且是一个参数θ的函数。因为$\mathrm{KL}(p \parallel q) \geqslant 0$，当且仅当$q(\mathbf{Z}) = p(\mathbf{Z} \mid \mathbf{X}, \theta)$时取得等号。因此，$\mathcal{L}(q, \theta) \leqslant \log p(\mathbf{X} \mid \theta)$，即$\mathcal{L}(q, \theta)$是$\log p(\mathbf{X} \mid \theta)$是的一个下界。

EM算法是一个两阶段迭代优化算法。

假设当前的参数θ^{old}，在*E步骤*中，下界$\mathcal{L}(q, \theta^{\mathrm{old}})$关于$q(\mathbf{Z})$最大化，而$\theta^{\mathrm{old}}$保持固定。当KL散度为零时，即得到了最大化的解。换句话说，最大值出现在$q(\mathbf{Z})$与后验概率分布$p(\mathbf{Z} \mid \mathbf{X}, \theta)$相等时，KL散度等于零，此时，下界等于最大似然函数。

在接下来的*M步骤*中，分布$q(\mathbf{Z})$保持固定，下界$\mathcal{L}(q, \theta)$关于θ最大化，得到了某个新的值θ^{new}，这会使得下界$\mathcal{L}$增大。同时也会使得对数似然增大，因为概率分布q由旧的参数值确定，并且在*M步骤*保持固定，因此不会等于新的后验分布$p(\mathbf{Z} \mid \mathbf{X}, \theta^{\mathrm{new}})$，从而KL散度非零；而且对数似然的增加量大于下界$\mathcal{L}(q, \theta)$的增加量。在*E步骤*之后，下界的形式为

$$\begin{aligned}\mathcal{L}(q, \theta) &= \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \theta^{\mathrm{old}}) \log p(\mathbf{X}, \mathbf{Z} \mid \theta) \\ &\quad - \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \theta^{\mathrm{old}}) \log p(\mathbf{Z} \mid \mathbf{X}, \theta^{\mathrm{old}}) \\ &= Q(\theta, \theta^{\mathrm{old}}) + \text{常数}\end{aligned}$$

其中常数是q的熵，与θ无关。从而，在*M步骤*中，最大化的量是**完整数据对数似然函数的期望**。完整的EM算法如下所示

Algorithm1: 用于含有隐变量最大似然函数参数估计的EM算法

- 选择参数的初始值 $\theta^{(t)}, t = 0$

- **REPEAT**:
  - *E步骤*: 计算$p(\mathbf{Z} \mid \mathbf{X}, \theta^{(t)})$
  - *M步骤*: 计算$\theta^{(t+1)}$，由下式给出

    $$\theta^{(t+1)} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{(t)})$$

    其中

    $$\mathcal{Q}(\theta, \theta^{(t)}) = \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \theta^{(t)}) \log p(\mathbf{X}, \mathbf{Z} \mid \theta)$$

- **UNTIL**: 对数似然函数收敛或者参数值收敛

高斯混合模型的EM算法

现在考虑将EM算法的隐变量观点用于一个具体的例子，即高斯混合模型。我们的目标是最大化对数似然函数$\log p(\mathbf{X} \mid \pi, \mu, \mathbf{\Sigma})$这是使用观测数据集$\mathbf{X}$计算的。这种情况比单一的高斯困难，因为求和出现在了对数运算内部。假设除了观测数据集$\mathbf{X}$，还有对应的离散变量$\mathbf{Z}$。现在考虑对完整数据$\{\mathbf{X}, \mathbf{Z}\}$最大化。完整数据集的似然函数的形式为

$$p(\mathbf{X}, \mathbf{Z} \mid \mu, \mathbf{\Sigma}, \pi) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \mathcal{N}(\mathbf{x}_n \mid \mu_k, \mathbf{\Sigma}_k)^{z_{nk}}$$

其中z_{nk}表示$\mathbf{z}_n$的第k个分量。取对数，有

$$\log p(\mathbf{X}, \mathbf{Z} \mid \mu, \mathbf{\Sigma}, \pi) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left\{ \log \pi_k + \log \mathcal{N}(\mathbf{x}_n \mid \mu_k, \mathbf{\Sigma}_k) \right\}$$

现在将完全数据的对数似然对$\mathbf{Z}$的后验概率分布求期望。后验概率分布为

$$p(\mathbf{Z} \mid \mathbf{X}, \mu, \mathbf{\Sigma}, \pi) = \prod_{n=1}^{N} p(\mathbf{z}_n \mid \mathbf{x}_n, \mu, \mathbf{\Sigma}, \pi)$$

在这个分布下, z_{nk}的期望为

$$\begin{aligned}
\mathbb{E}_{\mathbf{Z}}[z_{nk}] &= \sum_{\mathbf{z}_1} \cdots \sum_{\mathbf{z}_N} z_{nk} p(\mathbf{Z} \mid \mathbf{X}, \theta) \\
&= \sum_{\mathbf{z}_1} p(\mathbf{z}_1 \mid \mathbf{x}_n, \theta) \cdots \sum_{\mathbf{z}_n} z_{nk} p(\mathbf{z}_n \mid \mathbf{x}_n, \theta) \cdots \sum_{\mathbf{z}_N} p(\mathbf{z}_N \mid \mathbf{x}_n, \theta) \\
&= \sum_{\mathbf{z}_n} z_{nk} p(\mathbf{z}_n \mid \mathbf{x}_n, \theta) \\
&= p(z_{nk} = 1 \mid \mathbf{x}_n, \theta)
\end{aligned}$$

其中$\theta = (\mu, \mathbf{\Sigma}, \pi)$。利用贝叶斯公式，有

$$
\begin{aligned}
p(z_{nk}=1 \mid \mathbf{x}_n, \theta) &= \frac{p(z_{nk}=1)p(\mathbf{x}_n \mid z_{nk}=1)}{\sum_{j=1}^{K} p(z_{nj}=1)p(\mathbf{x}_n \mid z_{nj}=1)} \\
&= \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \mu_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \mu_j, \mathbf{\Sigma}_j)} \\
&\equiv \gamma(z_{nk})
\end{aligned}
$$

$\gamma(z_{nk})$被定义为数据点$\mathbf{x}_n$种含有来自于第k个高斯分布的“成分”。于是，完整数据的对数似然的期望值为

$$
\mathbb{E}_{\mathbf{Z}}\left[\log p(\mathbf{X}, \mathbf{Z} \mid \mu, \mathbf{\Sigma}, \pi)\right] = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk}) \left\{\log \pi_k + \log \mathcal{N}(\mathbf{x}_n \mid \mu_k, \mathbf{\Sigma}_k)\right\}
$$

我们使用旧的参数$\left\{\mu^{\text{old}}, \mathbf{\Sigma}^{\text{old}}, \pi^{\text{old}}\right\}$计算$\gamma(z_{nk})$(*E步骤*)；之后保持$\gamma(z_{nk})$不变，关于$\mu_k$,$\mathbf{\Sigma}_k$, π_k最大化(*M步骤*)，得到新的$\{\mu^{\text{new}}, \mathbf{\Sigma}^{\text{new}}, \pi^{\text{new}}\}$。

在进行M步骤之前，需要先参考一些关于矩阵求导数的运算，具体如下

$$
\sum_{i=1}^{N} \mathbf{x}_i^{\top} \mathbf{S} \mathbf{x}_i = \text{Tr}(\mathbf{S} \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^{\top})
$$

$$
\frac{\partial \text{Tr}(\mathbf{AB})}{\partial \mathbf{A}} = \mathbf{B}^{\top}
$$

$$
\frac{\partial}{\partial \mathbf{A}} \log |\mathbf{A}| = (\mathbf{A}^{-1})^{\top}
$$

现在关于π_k最大化。注意到由于$\sum_{k=1}^{K} \pi_k = 1$的限制，可以使用拉格朗日乘数法进行优化。构造拉格朗日函数为

$$
\mathcal{L}(\pi, \lambda) = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk}) \log \pi_k + \lambda(1 - \sum_{k=1}^{K} \pi_k)
$$

对π_k求导，并令其等于零，有

$$
\frac{1}{\lambda} \sum_{n=1}^{N} \gamma(z_{nk}) = \pi_k, \quad k = 1, \ldots, N
$$

又由$\sum_{k=1}^{K} \pi_k = 1$，得出$\lambda = N$，所以更新后的π_k为

$$
\pi_k^{\text{new}} = \frac{N_k}{N}
$$

其中$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$。

关于μ_k最大化。注意到，完整数据的对数似然中包含μ_k的项是

$$\sum_{n=1}^{N}\gamma(z_{nk})\log\mathcal{N}(\mathbf{x}_n\mid\mu_k,\mathbf{\Sigma}_k)$$
$$=\sum_{n=1}^{N}\gamma(z_{nk})\left\{-\frac{D}{2}\log(2\pi)+\frac{1}{2}\log|\mathbf{\Sigma}_k^{-1}|-\frac{1}{2}(\mathbf{x}_n-\mu_k)^\top\mathbf{\Sigma}_k^{-1}(\mathbf{x}_k-\mu_k)\right\}$$

对μ_k求导，并令其等于零，得

$$\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{\Sigma}_k^{-1}(\mathbf{x}_n-\mu_k)=0$$

化简

$$\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{\Sigma}_k^{-1}\mathbf{x}_n=\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{\Sigma}_k^{-1}\mathbf{u}_k$$

两边同乘$\mathbf{\Sigma}_k$，得

$$\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n=\sum_{n=1}^{N}\gamma(z_{nk})\mu_k=N_k\mu_k$$

所以，得到新的μ_k^{new}为

$$\mu_k^{\text{new}}=\frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n$$

关于$\mathbf{\Sigma}_k$最大化。将完整数据对数似然关于$\mathbf{Z}$后验概率的期望关于$\mathbf{\Sigma}_k^{-1}$求导，并令其导数等于零。具体过程如下

$$\mathbf{\Sigma}_k=\frac{1}{N_k}\frac{\partial}{\partial\mathbf{\Sigma}_k^{-1}}\text{Tr}\left(\mathbf{\Sigma}_k^{-1}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n-\mu_k)(\mathbf{x}_n-\mu_k)^\top\right)$$
$$=\frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n-\mu_k)(\mathbf{x}_n-\mu_k)^\top$$

所以，新的$\mathbf{\Sigma}_k^{\text{new}}=\frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n-\mu_k^{\text{new}})(\mathbf{x}_n-\mu_k^{\text{new}})^\top$。

总结一下，高斯混合分布的参数估计如下

- 初始化均值μ_k，协方差$\mathbf{\Sigma}_k$和混合系数π_k，计算对数似然的初始值
- **E步骤** 使用当前参数，计算每个数据点的成分$\gamma(z_{nk})$

$$\gamma(z_{nk})=\frac{\pi_k\mathcal{N}(\mathbf{x}_n\mid\mu_k,\mathbf{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(\mathbf{x}_n\mid\mu_j,\mathbf{\Sigma}_j)}$$

- **M步骤** 使用当前的$\gamma(z_{nk})$重新估计参数。

$$\mu_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n$$

$$\mathbf{\Sigma}_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n - \mu_k^{\text{new}})(\mathbf{x}_n - \mu_k^{\text{new}})^{\top}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

  其中

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk})$$

- 计算对数似然函数

$$\log p(\mathbf{X} \mid \pi, \mu, \mathbf{\Sigma}) = \sum_{n=1}^{N}\log\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n \mid \mu_k, \Sigma_k)\right\}$$

- 检查参数或者对数似然函数的收敛性。若没有满足收敛条件，返回**E步骤**。

实验

使用*python*模拟混合高斯分布的参数估计。混合高斯分布也可以用来聚类，与K-Means相比，可以实现*软聚类*，即可以计算出给定数据点$\mathbf{x}_n$ 属于第k个聚类的成分：$\gamma(z_{nk})$

导入必要的软件包

```
import matplotlib.pyplot as plt
import numpy as np
from numpy.linalg import inv, det
```

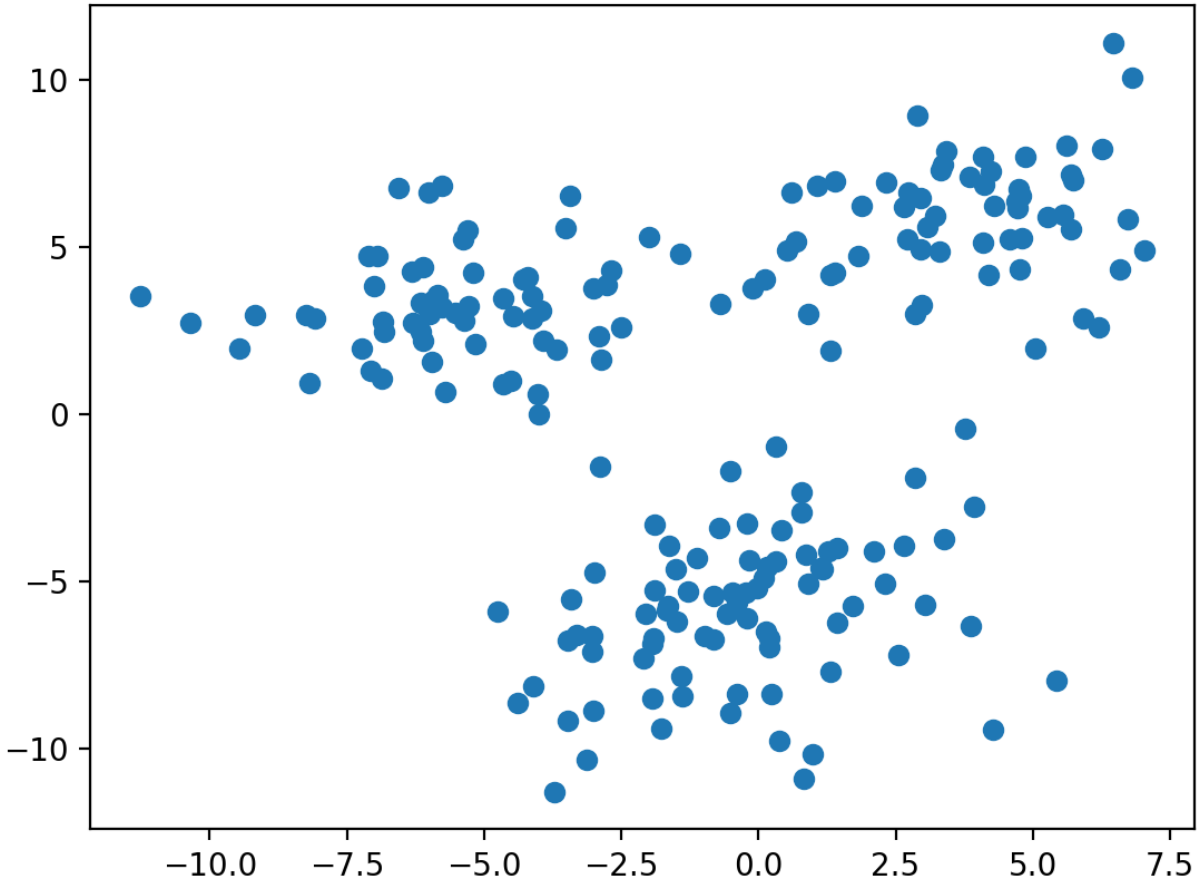
定义高斯分布以及高斯混合分布

```
# 多维高斯分布
def gaussion(x, mu, Sigma):
    dim = len(x)
    constant = (2*np.pi)**(-dim/2) * det(Sigma)**(-0.5)
    return constant * np.exp(-0.5*(x-mu).dot(inv(Sigma)).dot(x-mu))
# 高斯混合分布
def gaussion_mixture(x, Pi, mu, Sigma):
    z = 0
    for idx in range(len(Pi)):
        z += Pi[idx]* gaussion(x, mu[idx], Sigma[idx])
    return z
```

定义数据集生成函数(手动模拟数据集)，从三个高斯分布中采样

```
# 权重
Pi = np.array([ 0.3, 0.3, 0.4 ])
# 均值
mu = np.array([
    [-6, 3],
    [3, 6],
    [0, -6]
])
# 协方差矩阵
Sigma = np.array([
    [[4,0], [0,4]],
    [[4,1], [1,4]],
    [[6,2], [2,6]]
])

def sampling(Pi, mean, cov, N):
    samples = np.array([])
    for idx in range(len(Pi)):
        _sample = np.random.multivariate_normal(mean[idx], cov[idx], int(N*Pi[idx]))
        samples = np.append(samples, _sample)
    return samples.reshape((-1, mean[0].shape[0]))
```

数据分布如下图



绘制高斯分布等高线图，用来展示拟合的高斯分布

```
# 绘制混合高斯分布等高线图
def plot_gaussion(Pi, mu, Sigma):
    x = np.linspace(-10, 10, 100)
    y = np.linspace(-10, 10, 100)
    x, y = np.meshgrid(x, y)
    X = np.array([x.ravel(), y.ravel()]).T
    z = [ gaussion_mixture(x, Pi, mu, Sigma) for x in X ]
    z = np.array(z).reshape(x.shape)
    return plt.contour(x, y, z)
```

定义EM算法的一次迭代过程

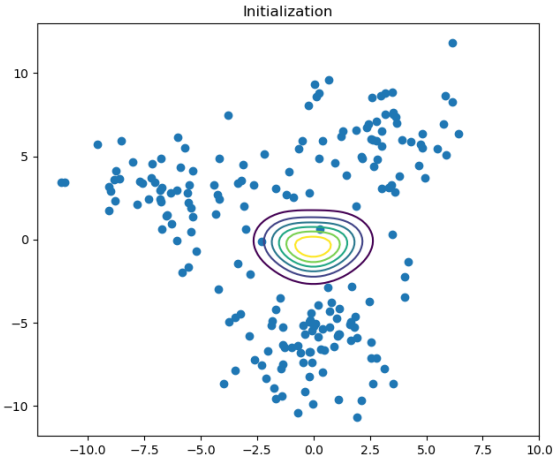
```python
def EM_step(X, Pi, mu, Sigma):
    N = len(X); K = len(Pi)
    gamma = np.zeros((N, K))

    # E-step
    for n in range(N):
        p_xn = 0
        for k in range(K):
            t = Pi[k]*gaussion(X[n], mu[k], Sigma[k])
            p_xn += t
            gamma[n, k] = t
        gamma[n] /= p_xn

    # M-step
    for k in range(K):
        _mu = np.zeros(mu[k].shape)
        _Sigma = np.zeros(Sigma[k].shape)
        N_k = np.sum(gamma[:,k])

        # 更新均值
        for n in range(N):
            _mu += gamma[n,k]*X[n]
        mu[k] = _mu / N_k

        # 更新方差
        for n in range(N):
            delta = np.matrix(X[n]- mu[k]).T
            _Sigma += gamma[n, k]*np.array( delta.dot(delta.T) )
        Sigma[k] = _Sigma / N_k

        # 更新权重
        Pi[k] = N_k / N
    return Pi, mu, Sigma
```

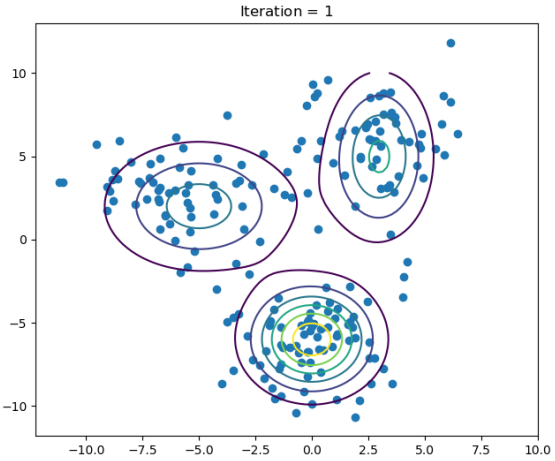
开始EM算法迭代过程，并显示每次迭代过程

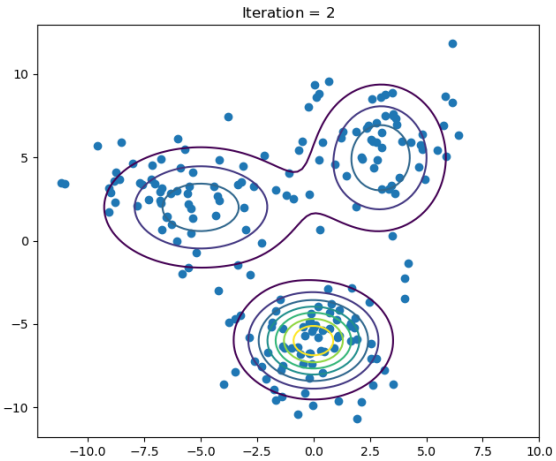
```
if __name__ == '__main__':
    # 参数初始值
    _Pi = np.array([
        0.33,
        0.33,
        0.34
    ])
    _mu = np.array([
        [0, -1],
        [1, 0],
        [-1, 0]
    ])
    _Sigma = np.array([
        [[1,0], [0,1]],
        [[1,0], [0,1]],
        [[1,0], [0,1]]
    ])

    n_iter = 3
    samples = sampling(Pi, mu, Sigma, 200)

    # 绘制初始状态
    plt.subplot(2, 2, 1)
    plt.title('Initialization')
    plt.scatter(*samples.T)
    plot_gaussion(_Pi, _mu, _Sigma)

    for i in range(n_iter):
        # EM算法迭代
        _Pi, _mu, _Sigma = EM_step(samples, _Pi, _mu, _Sigma)
        # 绘制每轮迭代结果
        plt.subplot(2, 2, i+2)
        plt.title('Iteration = $%d$' % (i+1))
        plt.scatter(*samples.T)
        plot_gaussion(_Pi, _mu, _Sigma)
    plt.show()
```

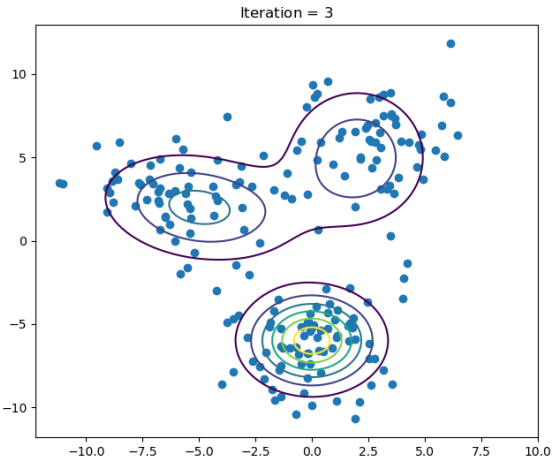
实验结果

选取隐变量**z**的状态有三种，经过3轮迭代之后的实验结果如下图：


Initialization
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Iteration = 1
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Iteration = 2
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Iteration = 3
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0