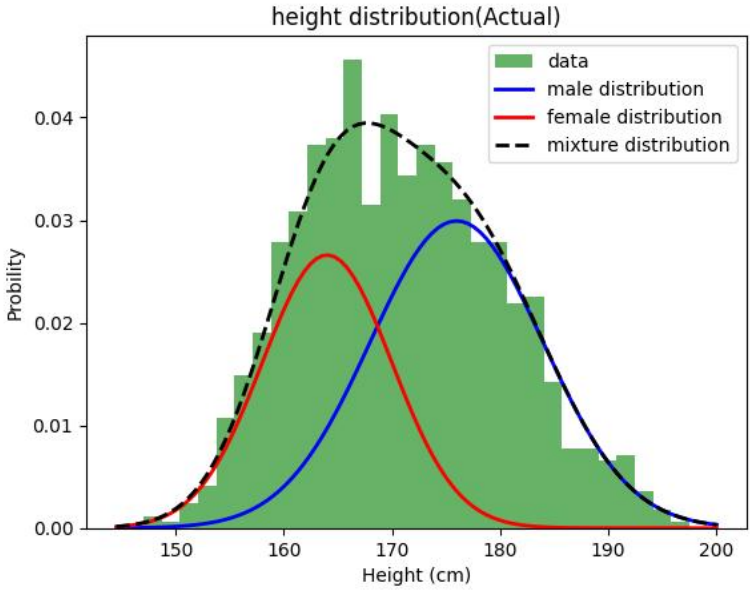
实验部分：

一.原始样本分析：

给定均值与标准差生成虚拟的大学男女生身高数据共 N 个：μ_M=176, Σ_M=8, μ_F=164, Σ_F=6， 其中男女比例为 3:2。这里先取 N=1000



上图是实际的情况，黑色虚线为直接根据数据拟合出的分布，我们认为这就是实际男女混合样本的分布模型。

现在我们希望可以根据 EM 算法求出这个实际的分布模型。

二.初始参数选取：

我们目前只知道有 1000 个样本和他们每个人的身高，但是并不知道这个样本对应的性别，更不知道男女的比例，男生身高均值方差信息和女生的均值方差信息。

只能根据这 1000 个样本的整体信息设置 EM 算法的初始参数：

μ_M^0 =np.mean(data)+10

μ_F^0 =np.mean(data)−10

Σ_M =np.std(data)

Σ_F =np.std(data)

π_1=0.5（男生所占的比例）

π_2=0.5（女生所占的比例）

三.EM 算法的具体实现：

根据我们的推导可以定义 EM 算法的 E 步和 M 步：

E 步：

```
def e_step(data, mu1, mu2, sigma1, sigma2, pi):
    gamma = pi * norm.pdf(data, mu1, sigma1)
    gamma /= gamma + (1 - pi) * norm.pdf(data, mu2, sigma2)
    return gamma
```

这里返回的 gamma 存储的我们在证明过程中定义的软猜测。在这里我们只需要存储每一个样本属于C_1类（男生）的软猜测概率。每一个样本属于C_2类（女生）的软猜测概率可以直接由 1-gamma 计算出。

$$\mathbf{gamma[k]} = \Upsilon(\mathbf{k},\mathbf{C_1}) = \frac{\mathbf{P(x_k|y_k = C_1;\mu_1,\sigma_1)\pi_1}}{\mathbf{P(x_k|y_k = C_1;\mu_1,\sigma_1)\pi_1 + P(x_k|y_k = C_2;\mu_2,\sigma_2)\pi_2}}$$

利用 gamma 就可以轻松地定义出$\mu_M, \Sigma_M, \mu_F, \Sigma_F$，$\pi_1$，$\pi_2$的更新：

$$\boldsymbol{\pi_1^{(t+1)}} = \frac{\boldsymbol{\Upsilon(1,1) + \Upsilon(2,1) + ... + \Upsilon(N,1)}}{\mathbf{N}} = \frac{\mathbf{np.sum(gamma)}}{\mathbf{len(data)}}$$

$$\boldsymbol{\pi_2^{(t+1)}} = \frac{\boldsymbol{\Upsilon(1,2) + \Upsilon(2,2) + ... + \Upsilon(N,2)}}{\mathbf{N}} = \frac{\mathbf{np.sum(1 - gamma)}}{\mathbf{len(data)}}$$

$$\boldsymbol{\mu_1} = \frac{\boldsymbol{\Upsilon(1,1)x_1 + \Upsilon(2,1)x_2 + ... + \Upsilon(N,1)x_N}}{\boldsymbol{\Upsilon(1,1) + \Upsilon(2,1) + ... + \Upsilon(N,1)}} = \frac{\mathbf{np.sum(gamma*data)}}{\mathbf{np.sum(gamma)}}$$

$$\boldsymbol{\mu_2} = \frac{\boldsymbol{\Upsilon(1,2)x_1 + \Upsilon(2,2)x_2 + ... + \Upsilon(N,2)x_N}}{\boldsymbol{\Upsilon(1,2) + \Upsilon(2,2) + ... + \Upsilon(N,2)}} = \frac{\mathbf{np.sum((1 - gamma)*data)}}{\mathbf{np.sum(1 - gamma)}}$$

$$\boldsymbol{\sigma_1^2} = \frac{\boldsymbol{\Upsilon(1,1)(x_1 - \mu_1)^2 + \Upsilon(2,1)(x_2 - \mu_1)^2 + ... + \Upsilon(N,1)(x_N - \mu_1)^2}}{\boldsymbol{\Upsilon(1,1) + \Upsilon(2,1) + ... + \Upsilon(N,1)}}$$

$$= \frac{\mathbf{np.sum(gamma*(data - \mu_1)**2)}}{\mathbf{np.sum(gamma)}}$$

$$\boldsymbol{\sigma_2^2} = \frac{\boldsymbol{\Upsilon(1,2)(x_1 - \mu_2)^2 + \Upsilon(2,2)(x_2 - \mu_2)^2 + ... + \Upsilon(N,2)(x_N - \mu_2)^2}}{\boldsymbol{\Upsilon(1,2) + \Upsilon(2,2) + ... + \Upsilon(N,2)}}$$

$$= \frac{\mathbf{np.sum(gamma*(data - \mu_2)**2)}}{\mathbf{np.sum(gamma)}}$$

于是可以对应定义出 M 步的函数：

```
def m_step(data, gamma):
    N1 = np.sum(gamma)
    N2 = len(data) - N1
    mu1 = np.sum(gamma * data) / N1
    mu2 = np.sum((1 - gamma) * data) / N2
```

```
    sigma1 = np.sqrt(np.sum(gamma * (data - mu1) ** 2) / N1)
    sigma2 = np.sqrt(np.sum((1 - gamma) * (data - mu2) ** 2) / N2)
    pi = N1 / len(data)
    return mu1, mu2, sigma1, sigma2, pi
```

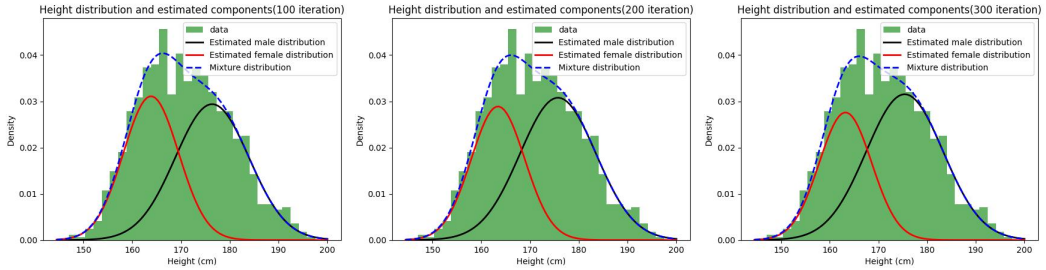
定义迭代函数：

```
def em_algorithm(data, mu1, mu2, sigma1, sigma2, pi, max_iter=100, tol=1e-6):
    for i in range(max_iter):
        gamma = e_step(data, mu1, mu2, sigma1, sigma2, pi)
        mu1_new, mu2_new, sigma1_new, sigma2_new, pi_new = m_step(data, gamma)
        print(f"Iteration {i+1}: mu1 = {mu1_new:.2f}, mu2 = {mu2_new:.2f}, sigma1 =
{sigma1_new:.2f}, sigma2 = {sigma2_new:.2f}, pi = {pi_new:.2f}")
        if np.abs(mu1 - mu1_new) < tol and np.abs(mu2 - mu2_new) < tol and \
           np.abs(sigma1 - sigma1_new) < tol and np.abs(sigma2 - sigma2_new) < tol and \
           np.abs(pi - pi_new) < tol:
            break
        mu1, mu2, sigma1, sigma2, pi = mu1_new, mu2_new, sigma1_new, sigma2_new, pi_new
    return mu1, mu2, sigma1, sigma2, pi
```

四.运行结果：

运行便可以得到：

参数	μ_M	Σ_M	μ_F	Σ_F	π_1	π_2
理论值	176	8	164	6	0.6	0.4
100 估计值	176.41	7.57	163.86	5.67	0.56	0.44
200 估计值	175.77	7.81	163.44	5.49	0.60	0.40
300 估计值	175.41	7.93	163.22	5.39	0.63	0.37



可以发现，迭代后的拟合效果还算不错。在均值，方差方面拟合效果较好。但即使经过300次的迭代。EM算法的权重估计值仍然与实际值之间有较大的误差。而这个误差与我们初始设定的值有很大的关系。

上面的数据对于均值的拟合效果很高，这可能是因为我们初始对于均值的设定为：

μ_M^0 =np.mean(data)+10

μ_F^0 =np.mean(data)-10

下面对初始值对EM算法的影响进行讨论。

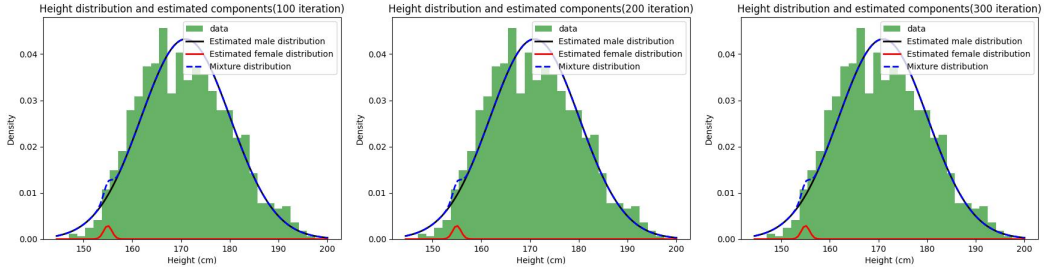
五.初始值影响的讨论：

在这里我们设定两组“不太合理”的初始值：

①$\mu_M^0=200$，$\mu_F^0=100$

结果如下：

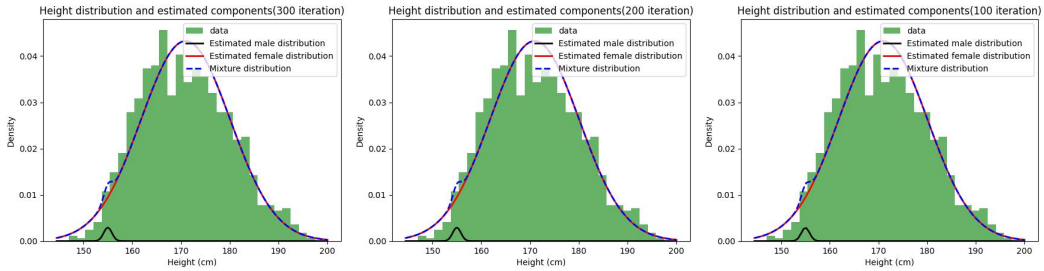
参数	μ_M	Σ_M	μ_F	Σ_F	π_1	π_2
理论值	176	8	164	6	0.6	0.4
100 估计值	170.96	9.17	154.97	0.85	0.99	0.01
200 估计值	170.96	9.17	154.98	0.85	0.99	0.01
300 估计值	170.96	9.17	154.98	0.85	0.99	0.01



②$\mu_M^0=100$，$\mu_F^0=200$

结果如下：

参数	μ_M	Σ_M	μ_F	Σ_F	π_1	π_2
理论值	176	8	164	6	0.6	0.4
100 估计值	154.97	0.85	170.96	0.85	0.01	0.99
200 估计值	154.98	0.85	170.96	0.85	0.01	0.99
300 估计值	154.98	0.85	170.96	0.85	0.01	0.99



可以看出：当我们仅仅是改变了均值这一个参数的初始值。EM 算法估计出的结果都非常差并且仔细观察每一次迭代后的值，可以发现前后的差距并不明显。足以见得 EM 算法的效果十分依赖初值的选择并且迭代速度非常慢。

下面对 EM 算法的优缺点进行讨论：

六.EM 算法的优缺点讨论：

优点：

1. 适用于缺失数据：EM 算法它通过期望步骤（E 步）计算缺失数据的期望值，然后在最大化步骤（M 步）更新参数，使其在有缺失数据的情况下也能进行参数估计。

从我的个人理解上，EM 算法的这个特性非常重要，事实上在我们的生活中，我们只是

“寄浮游于天地，渺沧海之一粟”，我们能观测的，可测量的变量，只是极小极小的一部分。EM 算法建立在有无法观测的隐变量情况依然可以对参数进行估计，并且有着不错的效果。

2. EM 算法的计算都有解析的表达，并且不需要求导，积分这大大加快了模型的计算速度并且保证了数值稳定性。这一点意味着 EM 算法可以用于更大量的数据并且拥有更快的性能。

3. EM 算法有着强大的理论基础，它是基于概率统计的极大似然法提出优化模型，其性质在理论上都有着比较完备的证明。另外 EM 算法蕴含着一个很重要的思想：隐变量+对隐变量的迭代更新思想。这一点在现代的很多最先进的模型上都能看到它的影子。这个思想与我们在《自动控制原理》中所学习的状态空间模型（SSM）异曲同工，它们都认为一个系统由一系列无法观测的隐变量(latent variable)，隐状态(latent state)所决定，这个隐状态可以通过系统的输出迭代更新。EM 算法采用的是优化 ELBO 迭代的想法，状态空间模型（SSM）采用的是利用状态转移矩阵迭代的想法。目前的生成模型和大语言模型大多也都是分别基于这两个思想不断改造，因此 EM 算法还是非常有魅力的。

缺点：

第一个很大的缺点是：由于理论的不完美导致 EM 算法有着较大的误差

EM 算法在将极大化似然概率转化为极大化边缘概率会产生第一步误差，在将极大化边缘概率转化为极大化 ELBO 又会产生第二步误差。这两步误差导致 EM 算法距离真实值总有距离。

1. EM 算法是基于最大化边缘概率的迭代优化方法，而不是最大化真实概率的方法，这可能会收敛到局部最优解而不是全局最优解。

2. EM 算法在思想上有一个很大的缺陷：它并不是通过优化似然概率对参数估计的，而是优化 ELBO。理论上总会与真实值有误差

3. EM 算法的效果高度依赖初始值的选择，在实验中我们可以发现，EM 的初始值选取若不合理，结果也与实际值大相径庭。而对初始值有比较合理的估计在现实生活中几乎是不可能的事情。这是 EM 算法一个很大的缺陷。

4. EM 算法的迭代速度太慢了，每一次更新参数的变化值都不是很多。因为 EM 算法本质上是坐标量级上的迭代方法，它无法洞察梯度上的变化，所以导致迭代效果不如梯度下降法有效。