3. EM算法实验报告

Author : Xinyu Zhao,Beihang University
Email:zxy2021@buaa.edu.cn

摘要

本报告详细介绍了使用期望最大化（EM）算法来估计混合高斯模型中的参数，以分析和模拟大学男女生身高数据。通过生成合成数据并应用EM算法，本实验旨在估计男女生身高的均值和方差，并探讨算法在不同迭代次数下的性能和参数收敛性。结果显示，EM算法能够有效地从混合数据中估计出各组分的参数，并且随着迭代次数的增加，估计的参数趋于稳定。

1. 引言

混合高斯模型是一种常用的概率模型，可以用来描述具有多种分布特征的数据。在许多实际应用中，比如生物信息学、市场分析以及社会科学研究中，混合高斯模型提供了一种强有力的工具来解析和解释隐藏在复杂数据背后的结构。EM算法作为一种强大的参数估计工具，其在混合模型参数估计中的应用尤为广泛，因其能够有效处理包含隐变量的复杂概率模型。

2. 问题重述

通过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：$\mu_M = 176, \sigma_M = 8, \mu_F = 164, \sigma_F = 6$，其中男女比例为3:2。

（1）用混合高斯模型对大学学生身高进行建模，并推导利用 EM 算法求解的公式（手写）

（2）编程实现 EM 算法对于以上5 个参数的估计并对比正确结果并讨论 EM 算法的优缺点。

3. 理论推导

为加深对混合高斯模型的理解，本部分采取手写推导的方式进行：

数 学 作 业 纸

班级： 姓名： 编号： 科目： 第 页

手写推导：

高斯混合模型：有 $P(y|\theta)=\sum_{k=1}^{K}\alpha_k\phi(y|\theta_k)$ 形式的概率分布模型.

α_k 为系数 且 $\alpha_k\geq 0$，$\sum_{k=1}^{K}\alpha_k=1$，$\phi(y|\theta_k)$ 为高斯分布密度，

其中 $\theta_k=(\mu_k,\sigma_k^2)$ $\phi(y|\theta_k)=\frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{(y-\mu_k)^2}{2\sigma_k^2}\right)$ 称为第k个分模型

此时假设 $y_1, y_2, y_3 \cdots y_N$ 由高斯混合模型生成

$P(y|\theta)=\sum_{k=1}^{K}\alpha_k\phi(y|\theta_k)$ $\theta=(\alpha_1,\alpha_2\cdots\alpha_K;\theta_1,\theta_2\cdots\theta_K)$

观测数据 y_j，$j=1,2,3,\cdots,N$，依概率 α_k 选择第k个高斯分布模型 $\phi(y|\theta_k)$，依第k个分模型的概率分布 $\phi(y|\theta_k)$ 生成数据 y_j. 所以此时 y_j $(j=1,2,3\cdots N)$ 都是已知的，仅因观测数据 y_j 来自第k个分模型的数据是未知的，$k=1,2,3\cdots K$，以隐变量 γ_{jk} 表示.

计：第j个观测来自第k个分模型 $\gamma_{jk}=1$ else $\gamma_{jk}=0$ $(j=1,2,3\cdots N)$ $(k=1,2,3\cdots K)$

完全数据为 $(y_j,\gamma_{j1},\gamma_{j2},\gamma_{j3}\cdots\gamma_{jK})$

$$P(y,\gamma|\theta)=\prod_{j=1}^{N}P(y_j,\gamma_{j1},\gamma_{j2}\cdots\gamma_{jK}|\theta)=\prod_{k=1}^{K}\prod_{j=1}^{N}[\alpha_k\phi(y_j|\theta_k)]^{\gamma_{jk}}$$

$$=\prod_{k=1}^{K}\alpha_k^{n_k}\prod_{j=1}^{N}\left[\frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{(y_j-\mu_k)^2}{2\sigma_k^2}\right)\right]^{\gamma_{jk}}$$

$(n_k=\sum_{j=1}^{N}\gamma_{jk}$ $\sum_{k=1}^{K}n_k=N)$

完全数据的对数似然函数.

$$\log P(y,\gamma|\theta)=\sum_{k=1}^{K}\left\{n_k\log\alpha_k+\sum_{j=1}^{N}\gamma_{jk}\left[\log\left(\frac{1}{\sqrt{2\pi}}\right)-\log\sigma_k-\frac{1}{2\sigma_k^2}(y_j-\mu_k)^2\right]\right\}$$

E步：确定Q函数.

$$Q(\theta,\theta^{(i)})=E[\log P(y,\gamma|\theta)|y,\theta^{(i)}]$$

$$=\sum_{k=1}^{K}\left\{\sum_{j=1}^{N}(E\gamma_{jk})\log\alpha_k+\sum_{j=1}^{N}(E\gamma_{jk})\left[\log\left(\frac{1}{\sqrt{2\pi}}\right)-\log\sigma_k-\frac{1}{2\sigma_k^2}(y_j-\mu_k)^2\right]\right.$$

数学作业纸

班级： 姓名： 编号： 科目： 第 页

$$E(\gamma_{jk}|y,\theta)=\hat{\gamma}_{jk}=\frac{P(\gamma_{jk}=1,y_j|\theta)}{\sum_{k=1}^{K}P(\gamma_{jk}=1,y_j|\theta)}=\frac{P(y_j|\gamma_{jk}=1,\theta)P(\gamma_{jk}=1|\theta)}{\sum_{k=1}^{K}P(y_j|\gamma_{jk}=1,\theta)P(\gamma_{kj}=1|\theta)}$$

$$=\frac{\alpha_k\phi(y_j|\theta_k)}{\sum_{k=1}^{K}\alpha_k\phi(y_j|\theta_k)}\quad k=1,2,3\cdots K\quad j=1,2,3\cdots N$$

$$\therefore Q(\theta,\theta^{(i)})=\sum_{k=1}^{K}\left\{n_k\log\alpha_k+\sum_{j=1}^{N}\hat{\gamma}_{jk}\left[\log\left(\frac{1}{\sqrt{2\pi}}\right)-\log\sigma_k-\frac{1}{2\sigma_k^2}(y_j-\mu_k)^2\right]\right\}$$

M步：

$$\hat{\mu}_k=\frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}y_j}{\sum_{j=1}^{N}\hat{\gamma}_{jk}},\quad k=1,2,3\cdots K\qquad \hat{\sigma}_k=\frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}(y_j-\mu_k)^2}{\sum_{j=1}^{N}\hat{\gamma}_{jk}}$$

$$\therefore\hat{\alpha}_k=\frac{n_k}{N}=\frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}}{N}\quad k=1,2,3\cdots N.$$

总结：在输入 $y_1,y_2\cdots y_N$ 高斯混合模型，在输出模型参数

(1) 取参数初始值，开始迭代

(2) E步：依据当前模型参数，计算分模型 k 对观测数据 y_j 的响应度 $\hat{\gamma}_{jk}=\frac{\alpha_k\phi(y_j|\theta_k)}{\sum_{k=1}^{K}\alpha_k\phi(y_j|\theta_k)}$

(3) M步：计算新一轮迭代参数

$$\hat{\mu}_k=\frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}y_j}{\sum_{j=1}^{N}\hat{\gamma}_{jk}}\qquad \hat{\sigma}_k^2=\frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}(y_j-\mu_k)^2}{\sum_{j=1}^{N}\hat{\gamma}_{jk}}\qquad \alpha_k=\frac{\sum_{j=1}^{N}\hat{\gamma}_{jk}}{N}\quad k\in[1,K]$$

(4) 重复 (2)(3) 至收敛.

4. 代码实现

4.1 基本代码

这段代码展示了如何生成虚拟的大学男女生身高数据，并使用期望最大化（EM）算法对其进行混合高斯模型的建模和参数估计。首先，使用numpy生成符合给定均值和标准差的男生和女生身高数据，并将它们混合在一起形成一个数据集。然后，定义了高斯分布函数，并编写了EM算法的E步骤和M步骤函数，分别计算每个数据点属于男生或女生高斯分布的概率和更新参数。

通过迭代执行这两个步骤，算法逐渐收敛，得到估计的均值、标准差和混合系数。接下来，使用matplotlib和seaborn对结果进行可视化，通过绘制直方图显示数据分布，并叠加男生和女生的高斯分布曲线以及混合分布曲线，以直观地展示模型的拟合效果。最终，通过输出打印估计的参数，验证算法的有效性。这样不仅展示了数据生成和算法实现的全过程，还提供了一个直观的图形来帮助理解结果。

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 生成虚拟数据
np.random.seed(42)
N = 1000
num_males = int(N * 3 / 5)
num_females = N - num_males
male_heights = np.random.normal(176, 8, num_males)
female_heights = np.random.normal(164, 6, num_females)
heights = np.concatenate((male_heights, female_heights))
np.random.shuffle(heights)

# EM算法实现
def gaussian(x, mu, sigma):
    return (1.0 / (np.sqrt(2 * np.pi) * sigma)) * np.exp(-0.5 * ((x - mu) ** 2 / sigma ** 2))

def e_step(data, mu_M, sigma_M, mu_F, sigma_F, pi_M, pi_F):
    r_M = pi_M * gaussian(data, mu_M, sigma_M)
    r_F = pi_F * gaussian(data, mu_F, sigma_F)
    gamma_M = r_M / (r_M + r_F)
    gamma_F = r_F / (r_M + r_F)
    return gamma_M, gamma_F

def m_step(data, gamma_M, gamma_F):
    N_M = np.sum(gamma_M)
    N_F = np.sum(gamma_F)

    mu_M = np.sum(gamma_M * data) / N_M
    mu_F = np.sum(gamma_F * data) / N_F

    sigma_M = np.sqrt(np.sum(gamma_M * (data - mu_M) ** 2) / N_M)
    sigma_F = np.sqrt(np.sum(gamma_F * (data - mu_F) ** 2) / N_F)

    pi_M = N_M / len(data)
    pi_F = N_F / len(data)

    return mu_M, sigma_M, mu_F, sigma_F, pi_M, pi_F

# 初始化参数
mu_M, sigma_M = 170, 10
mu_F, sigma_F = 160, 10
pi_M, pi_F = 0.5, 0.5

# 迭代
max_iter = 100
tol = 1e-6

```

```python
for i in range(max_iter):
    gamma_M, gamma_F = e_step(heights, mu_M, sigma_M, mu_F, sigma_F, pi_M, pi_F)
    mu_M_new, sigma_M_new, mu_F_new, sigma_F_new, pi_M_new, pi_F_new = m_step(heights, gamma_M, gamma_F)

    if np.abs(mu_M - mu_M_new) < tol and np.abs(mu_F - mu_F_new) < tol:
        break

    mu_M, sigma_M, mu_F, sigma_F, pi_M, pi_F = mu_M_new, sigma_M_new, mu_F_new, sigma_F_new, pi_M_new, pi_F_new

print(f'Estimated mu_M: {mu_M}, sigma_M: {sigma_M}, pi_M: {pi_M}')
print(f'Estimated mu_F: {mu_F}, sigma_F: {sigma_F}, pi_F: {pi_F}')

# 可视化
x = np.linspace(140, 200, 1000)
pdf_males = pi_M * gaussian(x, mu_M, sigma_M)
pdf_females = pi_F * gaussian(x, mu_F, sigma_F)

plt.figure(figsize=(12, 6))
sns.histplot(heights, bins=30, kde=False, color='orange', stat='density', label='Data')
plt.plot(x, pdf_males, 'r', label=f'Male Gaussian ($\mu$={mu_M:.2f}, $\sigma$={sigma_M:.2f})')
plt.plot(x, pdf_females, 'b', label=f'Female Gaussian ($\mu$={mu_F:.2f}, $\sigma$={sigma_F:.2f})')
plt.plot(x, pdf_males + pdf_females, 'g', label='Combined Gaussian')
plt.xlabel('Height (cm)')
plt.ylabel('Density')
plt.title('Height Distribution with Gaussian Mixture Model')
plt.legend()
plt.show()


'''
执行结果:
Estimated mu_M: 174.88734230418723, sigma_M: 8.228732296442026, pi_M: 0.669319107015955
Estimated mu_F: 164.03829365809483, sigma_F: 5.644918566977975, pi_F: 0.33068089298404507
'''
```

可视化如下图所示:

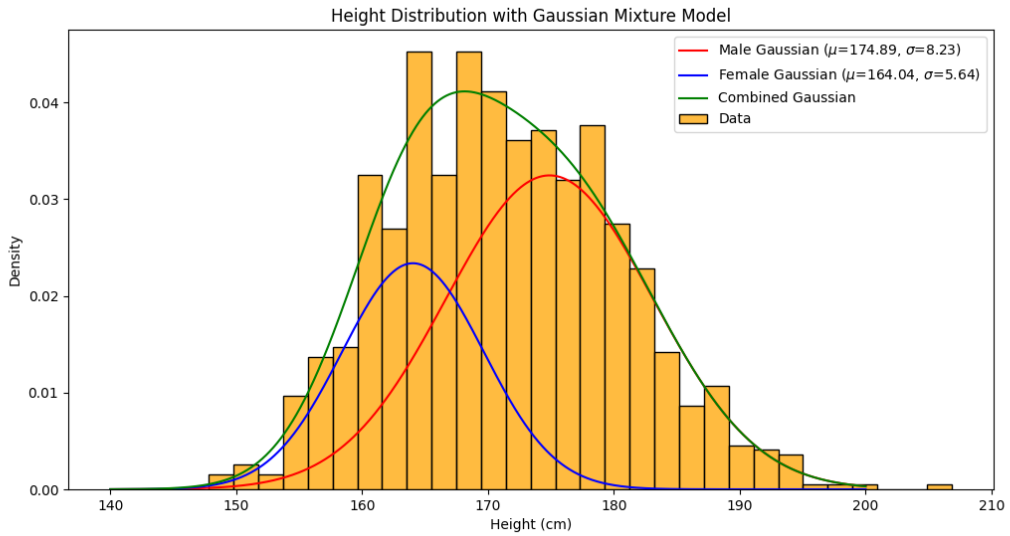


图1 原始数据分布

4.2 改进代码

在基本代码中我们虽然能够得出正确的结论，基本满足了题意，但是无法探究迭代次数等因素对结果产生的影响，也无法展现实际分布和估计分布之间的差异，因此后文对代码进行了改进。

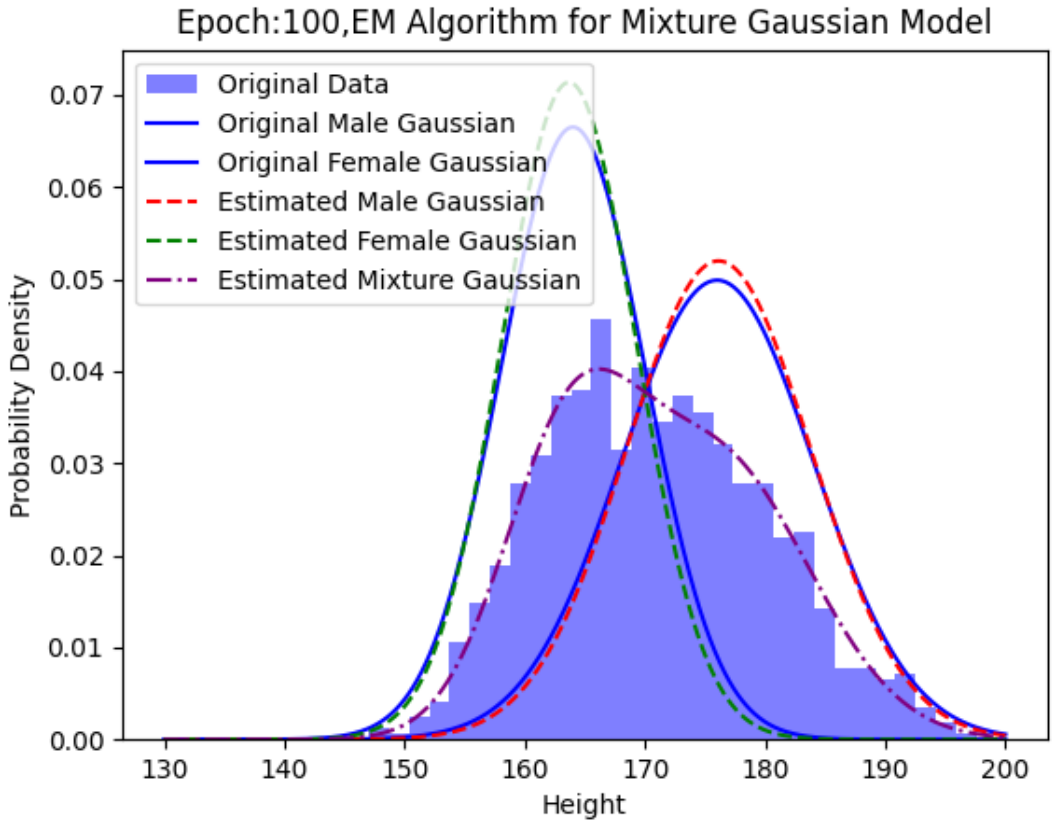
```
import numpy as np
import matplotlib.pyplot as plt

# 高斯函数
def gaussian(x, mu, sigma):
    return 1/(np.sqrt(2*np.pi)*sigma) * np.exp(-0.5 * ((x - mu)/sigma)**2)

# 生成合成数据
np.random.seed(0)
N = 1000
mu_M, sigma_M = 176, 8
mu_F, sigma_F = 164, 6
male_data = np.random.normal(mu_M, sigma_M, int(3*N/5))
female_data = np.random.normal(mu_F, sigma_F, int(2*N/5))
data = np.concatenate((male_data, female_data))

# 原始高斯分量
x = np.linspace(130, 200, 1000)
pdf_M_original = gaussian(x, mu_M, sigma_M)
pdf_F_original = gaussian(x, mu_F, sigma_F)

# EM算法
def run_em_algorithm(num_iterations):
    # 初始化参数
    mu_M_hat, mu_F_hat = np.random.uniform(160, 180), np.random.uniform(150, 170)
    sigma_M_hat, sigma_F_hat = np.random.uniform(5, 10), np.random.uniform(4, 8)
    pi_M_hat, pi_F_hat = 0.5, 0.5
    mu_M_values = []  # 存储每次迭代的mu_M值

    # EM算法
```

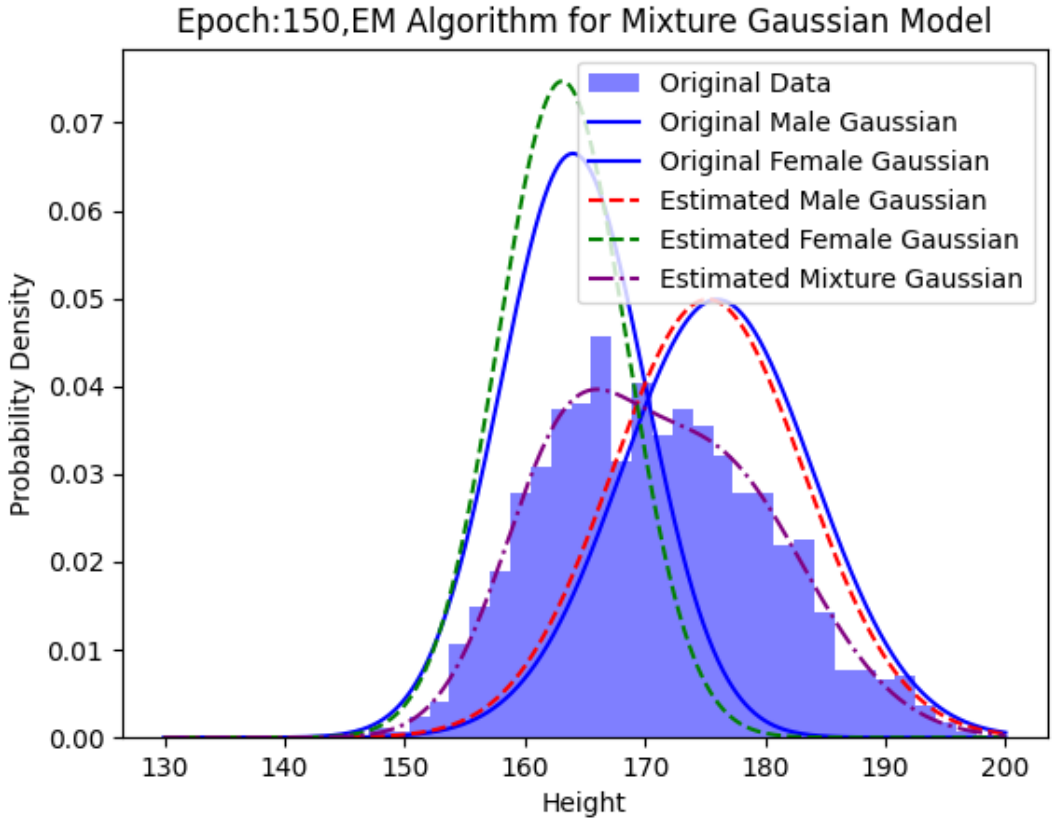
```
    for _ in range(num_iterations):
        # E步
        w_M = pi_M_hat * gaussian(data, mu_M_hat, sigma_M_hat)
        w_F = pi_F_hat * gaussian(data, mu_F_hat, sigma_F_hat)
        sum_w = w_M + w_F
        w_M /= sum_w
        w_F /= sum_w

        # M步
        mu_M_hat = np.sum(w_M * data) / np.sum(w_M)
        mu_F_hat = np.sum(w_F * data) / np.sum(w_F)
        sigma_M_hat = np.sqrt(np.sum(w_M * (data - mu_M_hat)**2) /
np.sum(w_M))
        sigma_F_hat = np.sqrt(np.sum(w_F * (data - mu_F_hat)**2) /
np.sum(w_F))
        pi_M_hat = np.mean(w_M)
        pi_F_hat = np.mean(w_F)

        mu_M_values.append(mu_M_hat)  # 记录此次迭代的mu_M值

    # 输出结果
    print(f"Iterations: {num_iterations}")
    print("Estimated Parameters:")
    print("mu_M:", mu_M_hat)
    print("sigma_M:", sigma_M_hat)
    print("mu_F:", mu_F_hat)
    print("sigma_F:", sigma_F_hat)
    print("pi_M:", pi_M_hat)
    print("pi_F:", pi_F_hat)

    # 绘图
    plt.hist(data, bins=30, density=True, alpha=0.5, color='blue',
label='Original Data')
    x = np.linspace(130, 200, 1000)
    pdf_M = gaussian(x, mu_M_hat, sigma_M_hat)
    pdf_F = gaussian(x, mu_F_hat, sigma_F_hat)
    pdf = pi_M_hat * pdf_M + pi_F_hat * pdf_F
    plt.plot(x, pdf_M_original, color='blue', linestyle='-',
label='Original Male Gaussian')
    plt.plot(x, pdf_F_original, color='blue', linestyle='-',
label='Original Female Gaussian')
    plt.plot(x, pdf_M, color='red', linestyle='--', label='Estimated Male
Gaussian')
    plt.plot(x, pdf_F, color='green', linestyle='--', label='Estimated
Female Gaussian')
    plt.plot(x, pdf, color='purple', linestyle='-.', label='Estimated
Mixture Gaussian')
    plt.title('Epoch:{},EM Algorithm for Mixture Gaussian
Model'.format(num_iterations))
    plt.xlabel('Height')
    plt.ylabel('Probability Density')
    plt.legend()
    plt.show()

    # Plot mu_M values over iterations
```

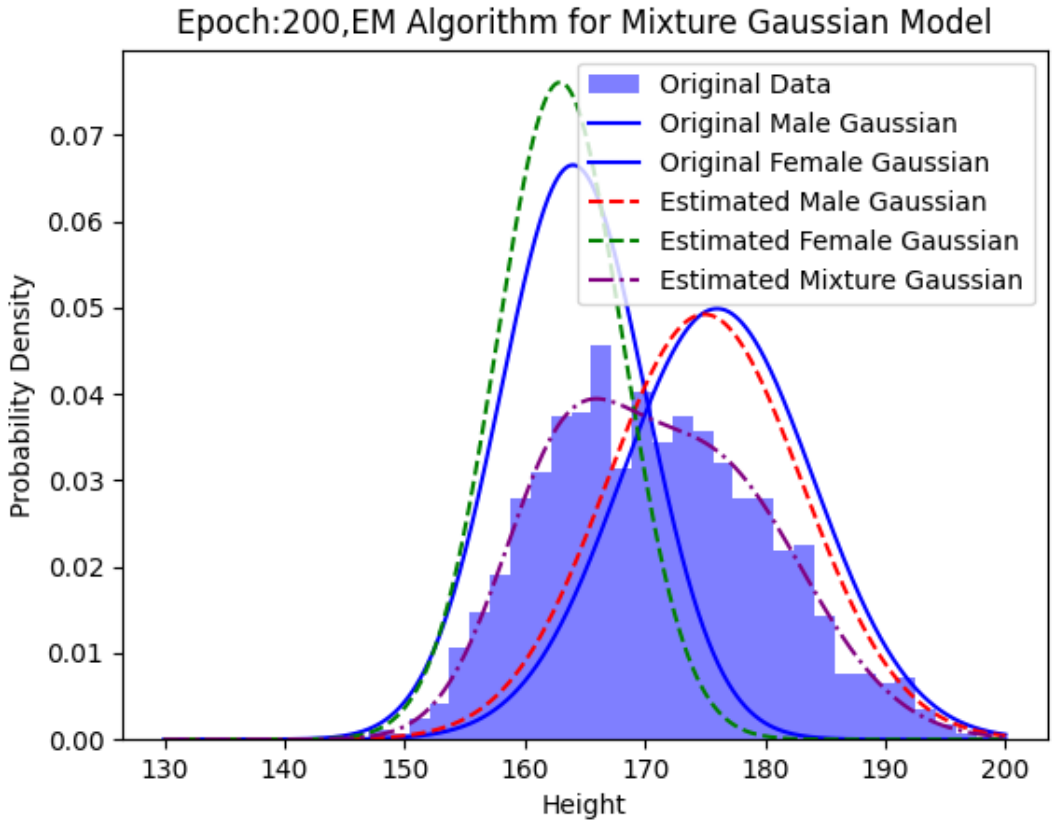
```
    plt.plot(range(1, num_iterations + 1), mu_M_values, marker='o',
linestyle='-')
    plt.title('Variation of mu_M over Iterations')
    plt.xlabel('Iterations')
    plt.ylabel('mu_M')
    plt.grid(True)
    plt.show()

# 不同迭代次数下运行EM算法
for num_iterations in range(100,301,50):
    run_em_algorithm(num_iterations)


'''
运行结果:
Iterations: 100
Estimated Parameters:
mu_M: 176.12164794321117
sigma_M: 7.677484254081643
mu_F: 163.6680777634504
sigma_F: 5.591253316531177
pi_M: 0.5777011132109134
pi_F: 0.4222988867890864

Iterations: 150
Estimated Parameters:
mu_M: 175.2353055558731
sigma_M: 7.99765908580283
mu_F: 163.10726403236123
sigma_F: 5.339003369318337
pi_M: 0.63944826313074
pi_F: 0.36055173686926006

Iterations: 200
Estimated Parameters:
mu_M: 174.9370234512357
sigma_M: 8.106119427396512
mu_F: 162.92567047190465
sigma_F: 5.245109115830187
pi_M: 0.6607789032177604
pi_F: 0.3392210967822396

Iterations: 250
Estimated Parameters:
mu_M: 175.12306043635834
sigma_M: 8.03848579666431
mu_F: 163.03852245279842
sigma_F: 5.3044699483694195
pi_M: 0.6474386259196608
pi_F: 0.3525613740803393

Iterations: 300
Estimated Parameters:
mu_M: 175.01056385759566
sigma_M: 8.079395267259683
mu_F: 162.9701247418223
```

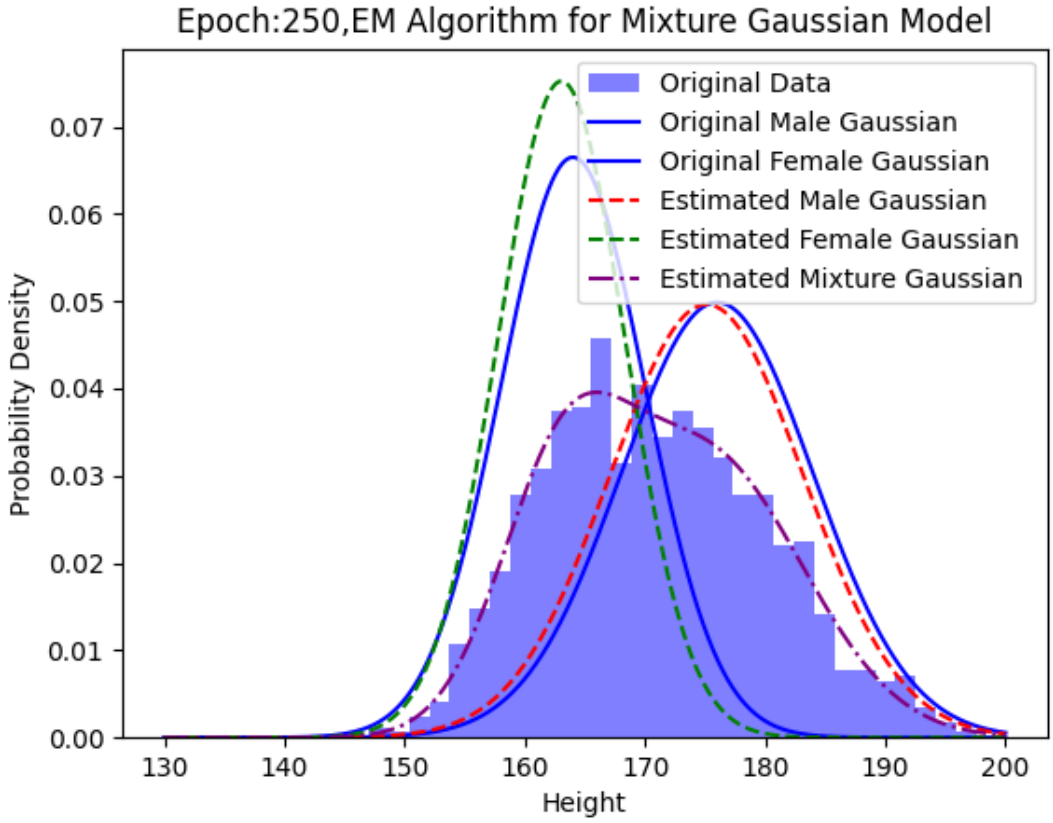
```
sigma_F: 5.268921125559326
pi_M: 0.6554905765510537
pi_F: 0.3445094234489462
'''
```

得到如下结果：






Epoch:200,EM Algorithm for Mixture Gaussian Model
Original Data
Original Male Gaussian
Original Female Gaussian
Estimated Male Gaussian
Estimated Female Gaussian
Estimated Mixture Gaussian
Probability Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
130
140
150
160
170
180
190
200
Height


Epoch:250,EM Algorithm for Mixture Gaussian Model
Original Data
Original Male Gaussian
Original Female Gaussian
Estimated Male Gaussian
Estimated Female Gaussian
Estimated Mixture Gaussian
Probability Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
130
140
150
160
170
180
190
200
Height

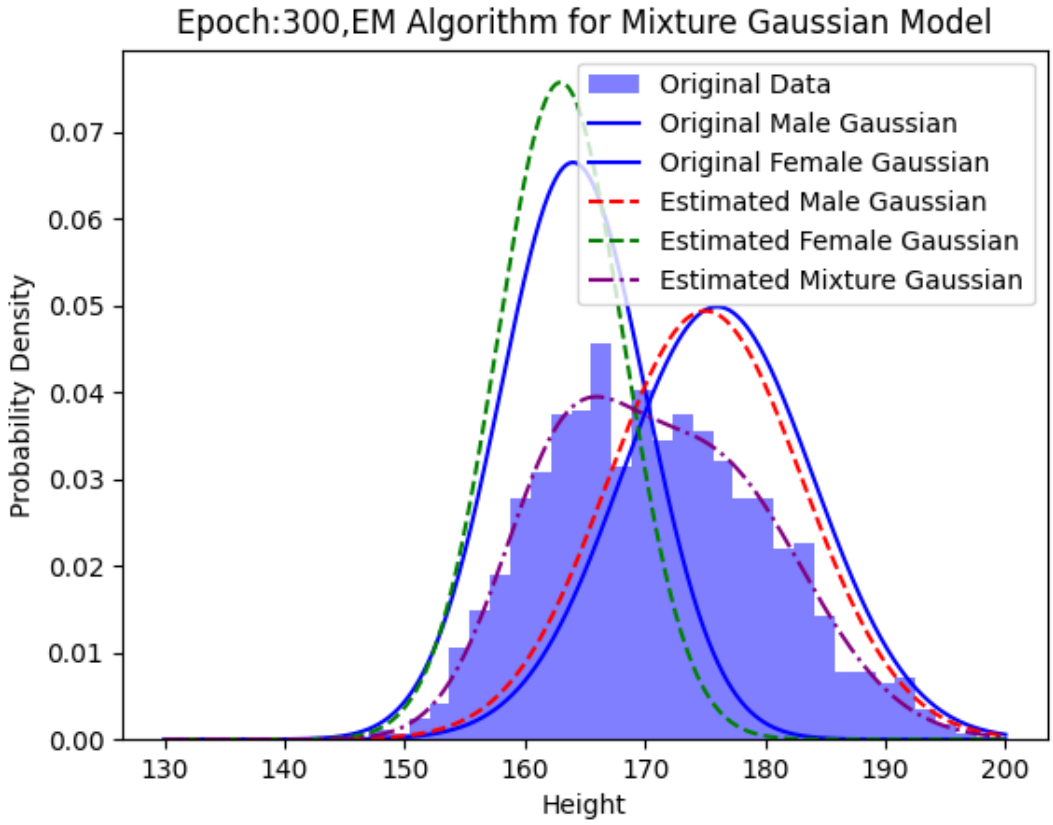


图2-6 100、150、200、250、300次迭代结果

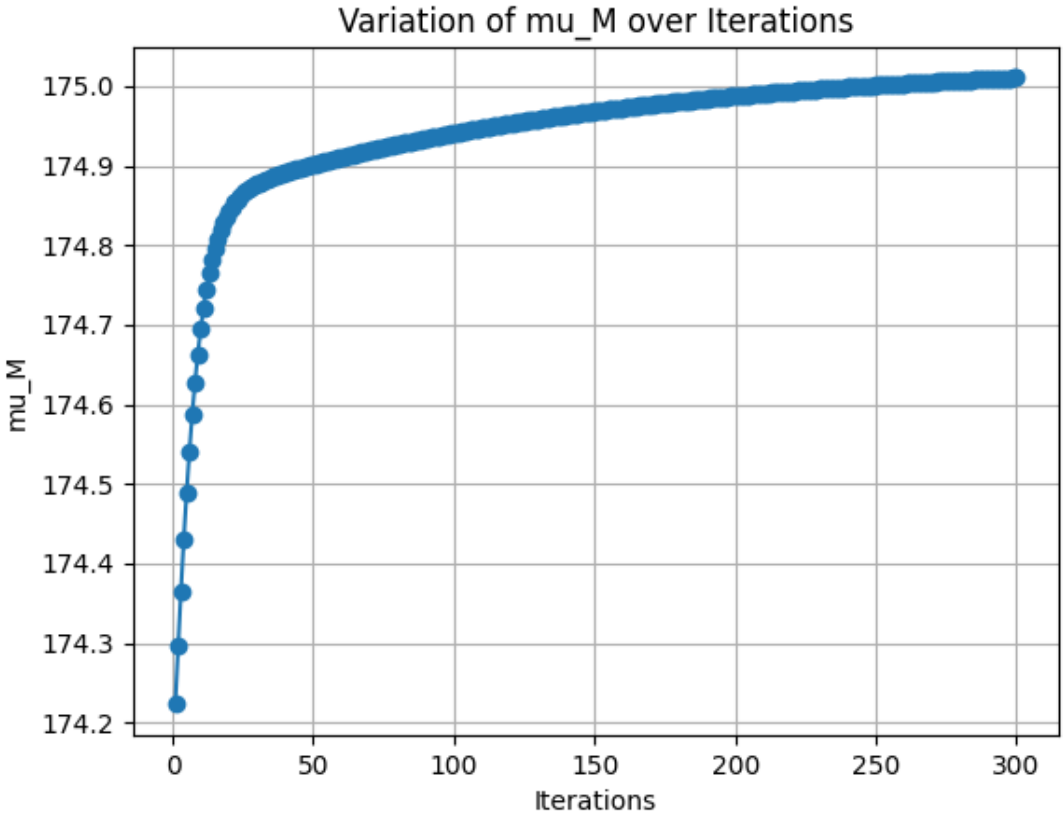


图7 迭代次数与预测结果的关系

由图可以得出，随着次数的增加，μ_M逐渐收敛于一个极限值，当迭代次数足够多的时候，可以实现对结果的准确预测，然而在考虑时间成本的时候，迭代100次就可以实现小于0.1的误差值。

5. EM算法的优缺点

优点：

1. **简单易用**：EM算法相对容易实现，并且对于许多实际问题有很好的效果。
2. **适用性广**：EM算法可以应用于各种具有隐藏变量的问题，如混合高斯模型。
3. **收敛性好**：在大多数情况下，EM算法能够快速收敛到局部最优解。

缺点：

1. **收敛到局部最优**：EM算法可能会收敛到局部最优解，而不是全局最优解。初始参数的选择对结果有很大影响。
2. **收敛速度慢**：在某些情况下，EM算法的收敛速度较慢，尤其是在参数空间很大的时候。
3. **复杂度高**：对于大数据集或高维数据，EM算法的计算复杂度较高，可能需要大量计算资源。