人工智能大作业（GMM）

黄渲淇
2320634327@qq.com

Abstract

混合高斯模型是指多个高斯分布的线性组合，其中每个高斯分布被称作一个“分量”，每个分量都有自己的均值和协方差矩阵。混合高斯模型广泛应用于模式识别、数据挖掘、图像处理等领域中，可以用于数据聚类、异常检测、图像分割等任务。 EM 算法是求解混合高斯模型参数的常用方法。EM 算法的基本思想是通过迭代求解每个分量的权重、均值和协方差矩阵，使得混合高斯模型对数据的拟合效果最优。

Introduction

1.高斯混合模型（GMM）

高斯混合模型（Gaussian Mixture Model，简称 GMM）是一种常用的概率模型，它将一个数据集拟合成多个高斯分布的线性组合，从而得到更好的拟合效果。GMM 可以被用来对数据进行聚类、分类、降维等任务。 假设有一个数据集x，其中每个数据点x_i都是一个d维向量。GMM 的公式为：$P(x)=\sum_{i=1}^{k} w_i \cdot \mathcal{N}(x|\mu_i\Sigma_i)$

其中，k为高斯分布的个数，w_i为系数，$\mathcal{N}(x|\mu_i,\Sigma_i)$为高斯分布的概率密度函数，$\mu_i$和$\Sigma_i$分别为第$i$个高斯分布的均值和协方差矩阵。GMM 的参数包括每个高斯分布的系数、均值和协方差矩阵。 GMM 可以通过最大化似然函数来进行参数估计。具体来说，假设数据集为$X=x_1,x_2,...,x_N$，当前的模型参数为$\theta=w_i,\mu_i,\Sigma_i$，则似然函数为：$p(X|\theta)=\prod_{i=1}^{N}\sum_{j=1}^{k} w_j \cdot \mathcal{N}(x_i|\mu_j,\Sigma_j)$其中$\mathcal{N}(x_i|\mu_j,\Sigma_j)$为高斯分布的概率密度函数。GMM 的

参数可以通过最大化似然函数来估计，即：$\theta_{new} = \arg\max_{\theta} p(X|\theta)$ 这可以通过 EM 算法来实现。 需要注意的是，GMM 中的每个高斯分布可以看作是数据的一个聚类中心，因此 GMM 可以用于数据聚类。此外，GMM 也可以用于分类问题，方法是将每个高斯分布看作是一个类别的概率分布。

举例来说：想象下现在咱们不再考察全部用户的身高，而是要在模型中同时考虑男性和女性的身高。假定之前的样本里男女都有，那么之前所画的高斯分布其实是两个高斯分布的叠加的结果。相比只使用一个高斯来建模，现在我们可以用两个（或多个）高斯分布。

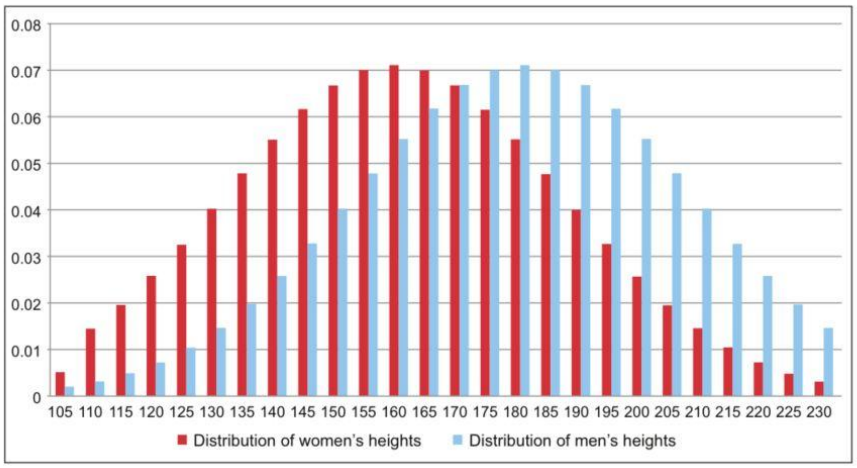
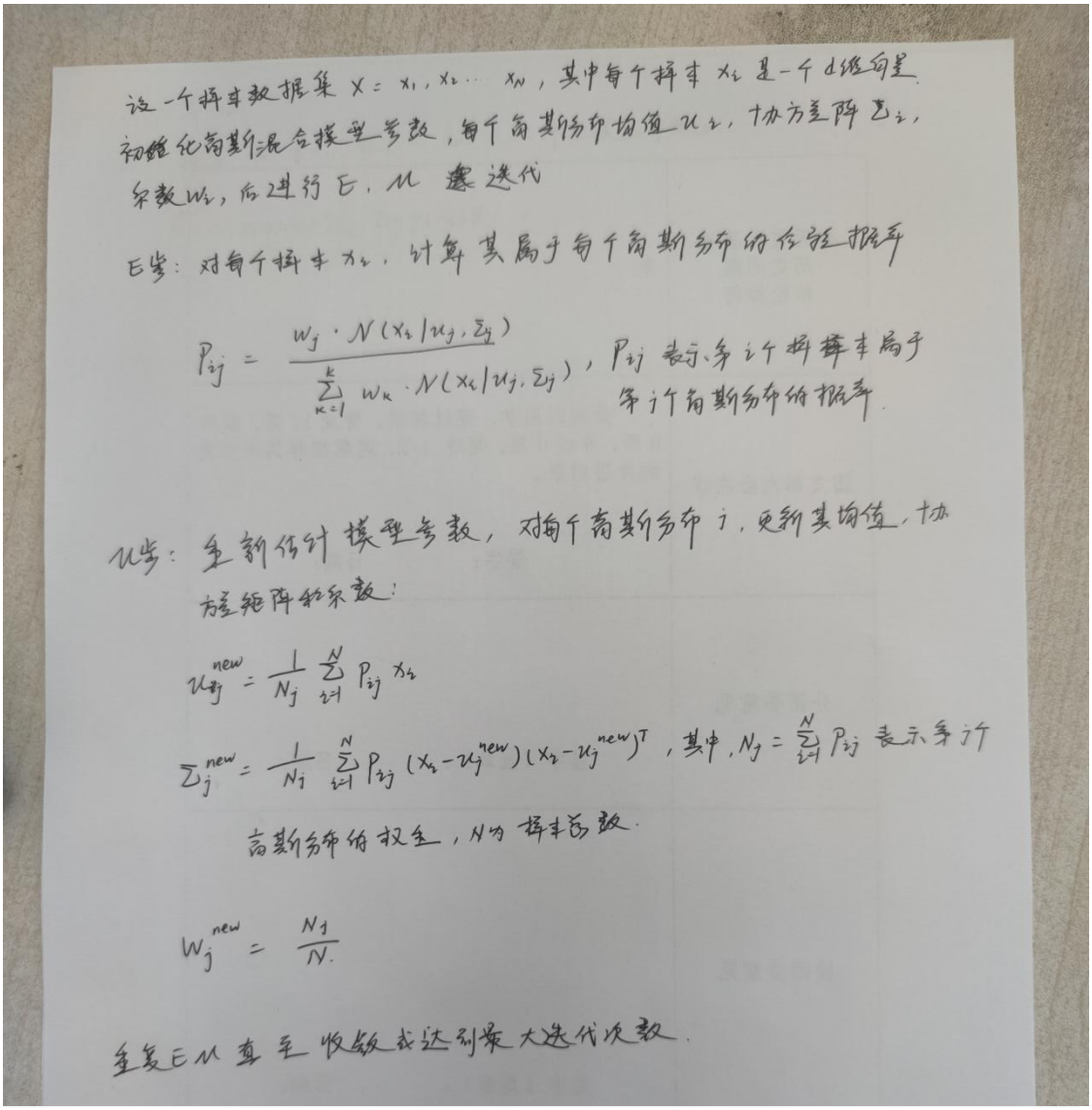


Figure 2.11 Probability distributions of height for men and women. Note that these probabilities are conditioned on the fact that gender is known: so, for example, given that we know a particular person is a woman, her probability of having a height in a particular bucket can be read off the y-axis.

图中的 y-轴所示的概率值，是在已知每个用户性别的前提下计算出来的。但通常情况下我们并不能掌握这个信息（也许在采集数据时没记录），因此不仅要学出每种分布的参数，还需要生成性别的划分情况（w_i）。当决定期望值时，需要将权重值分别生成男性和女性的相应身高概率值并相加。

2.EM 算法



使用更新后的模型参数进行预测或者其他任务。需要注意的是，在 EM 算法的实现中，为了防止协方差矩阵出现奇异矩阵或者过拟合等问题，通常会对协方差矩阵加上一个正则化项或者使用参数共享等方法来限制模型参数的数量。

极大似然估计

极大似然估计（Maximum Likelihood Estimation，MLE）是一种常用的概率统计方法，用于求解模型参数。其基本思想是在给定观测数据的条件下，求解模型参数使得数据的概率最大。 具体来说，假设我们有一个样本集$X = x_1, x_2, \ldots, x_N$，其中每个样本$x_i$都是一个随机变量，服从某个未知分布$p(x|\theta)$，其中$\theta$是待求的模型参数。MLE 的目标是求解参数$\theta$，使

得在给定样本集X的情况下，观测到这个样本集的概率$p(X|\theta)$最大。 根据贝叶斯定理，$p(X|\theta)$可以表示为： $p(X|\theta)=\prod_{i=1}^{N}p(x_i|\theta)$ 因此，MLE 的目标可以转化为求解参数θ，使得上述概率最大，即：$\hat{\theta}MLE=\text{argmax}\theta p(X|\theta)=\arg\max_{\theta}\prod_{i=1}^{N}p(x_i|\theta)$ 在实际求解中，为了方便计算，通常会对上述目标函数取对数，得到对数似然函数：$l(\theta)=\log p(X|\theta)=\sum_{i=1}^{N}\log p(x_i|\theta)$ 然后，根据最大化对数似然函数等价于最大化似然函数的性质，可以通过求解对数似然函数的梯度来得到 MLE 的解。具体来说，需要使用梯度下降、牛顿法等优化算法来求解参数的最优解。 需要注意的是，MLE 的估计结果受到样本大小的影响，当样本大小较小时，可能会出现过拟合的情况。此外，MLE 也可能会受到模型假设的影响，因此在使用 MLE 时需要仔细选择模型和参数。

Methodology

1: 数据读取

读取身高数据:

```
data = pd.read_csv('h_data.csv')
# 取出身高数据并转化为 numpy 数组
height = data['height'].values
```

定义高斯分布函数:

```
def gaussian(x, mu, sigma):
    return 1 / (np.sqrt(2 * np.pi) * sigma) * np.exp(-0.5 * (x - mu) ** 2 / sigma
** 2)
# 初始化参数
mu1, sigma1 = 170, 5
mu2, sigma2 = 190, 5
pi = 0.5
# 迭代次数
n_iterations = 1000
```

我们假设有两个高斯分布，其概率密度函数分别为:

$$p(x|\mu_1,\sigma_1)=\frac{1}{\sqrt{2\pi}\,\sigma_1}\exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right)$$
$$p(x|\mu_2,\sigma_2)=\frac{1}{\sqrt{2\pi}\,\sigma_2}\exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right)$$

其中，x是身高数据，μ_1,σ_1是第一个高斯分布的均值和标准差，μ_2,σ_2是第二个高斯分布的均值和标准差。我们还假设每个样本来自于两个高斯分布的概率不同，用π来表示来自于第一个高斯分布的概率。因此，每个样本来自于两个高斯分布的概率密度函数为：

$$p(x|\mu_1,\sigma_1,\mu_2,\sigma_2,\pi)=\pi p(x|\mu_1,\sigma_1)+(1-\pi)p(x|\mu_2,\sigma_2)$$

我们的目标是估计参数$\theta=\mu_1,\sigma_1,\mu_2,\sigma_2,\pi$，使得样本的似然函数最大。然而，由于参数$\theta$的存在，样本的似然函数不易求解。因此，我们使用 EM 算法来逼近参数θ的最大似然估计值。这里我们假设初始值为 170 和 5 的高斯分布占 50%的权重，初始值为 180 和 5 的高斯分布占 50%的权重。

2: EM 算法

```
# EM 算法
for i in range(n_iterations):
    # E 步：计算后验概率
    gamma1 = pi * gaussian(height, mu1, sigma1)
    gamma2 = (1 - pi) * gaussian(height, mu2, sigma2)
    gamma_sum = gamma1 + gamma2
    gamma1 /= gamma_sum
    gamma2 /= gamma_sum

    # M 步：更新参数
    mu1 = np.sum(gamma1 * height) / np.sum(gamma1)
    mu2 = np.sum(gamma2 * height) / np.sum(gamma2)
    sigma1 = np.sqrt(np.sum(gamma1 * (height - mu1) ** 2) / np.sum(gamma1))
    sigma2 = np.sqrt(np.sum(gamma2 * (height - mu2) ** 2) / np.sum(gamma2))
    pi = np.mean(gamma1)
```

我们开始进行 EM 算法的迭代。EM 算法主要分为 E 步和 M 步两个步骤。在 E 步中，我们计算每个样本来自于两个高斯分布的后验概率。在 M 步中，我们根据每个样本对两个高斯分布的后验概率，更新两个高斯分布的均值和标准差，以及两个高斯分布的权重。 具体来说，在 E 步中，我们计算每个样本来自于两个高斯分布的后验概率：这里，gamma1 和

gamma2 分别表示每个样本来自于第一个高斯分布和第二个高斯分布的后验概率，gamma_sum 表示每个样本来自于两个高斯分布的后验概率之和。具体来说，gamma1 的计算公式为 pi * P(x | mu1, sigma1)，其中 P(x | mu1, sigma1)表示在给定 mu1 和 sigma1 的情况下，样本 x 的概率密度值。同理，gamma2 的计算公式为(1 - pi) * P(x | mu2, sigma2)。在 M 步中，我们根据每个样本对两个高斯分布的后验概率，更新两个高斯分布的均值和标准差，以及两个高斯分布的权重：这里，mu1 和 mu2 分别表示第一个高斯分布和第二个高斯分布的均值，sigma1 和 sigma2 分别表示第一个高斯分布和第二个高斯分布的标准差，pi 表示第一个高斯分布的权重。具体来说，mu1 的更新公式为 $\frac{\sum_i \gamma_{1i} x_i}{\sum_i \gamma_{1i}}$，sigma1 的更新公式为 $\sqrt{\frac{\sum_i \gamma_{1i}(x_i - \mu_1)^2}{\sum_i \gamma_{1i}}}$，同理，mu2 和 sigma2 的更新公式可类比得到。pi 的更新公式为第一个高斯分布的权重，即 $\frac{1}{n}\sum_i \gamma_{1i}$。

多次迭代后得到混合高斯的参数。

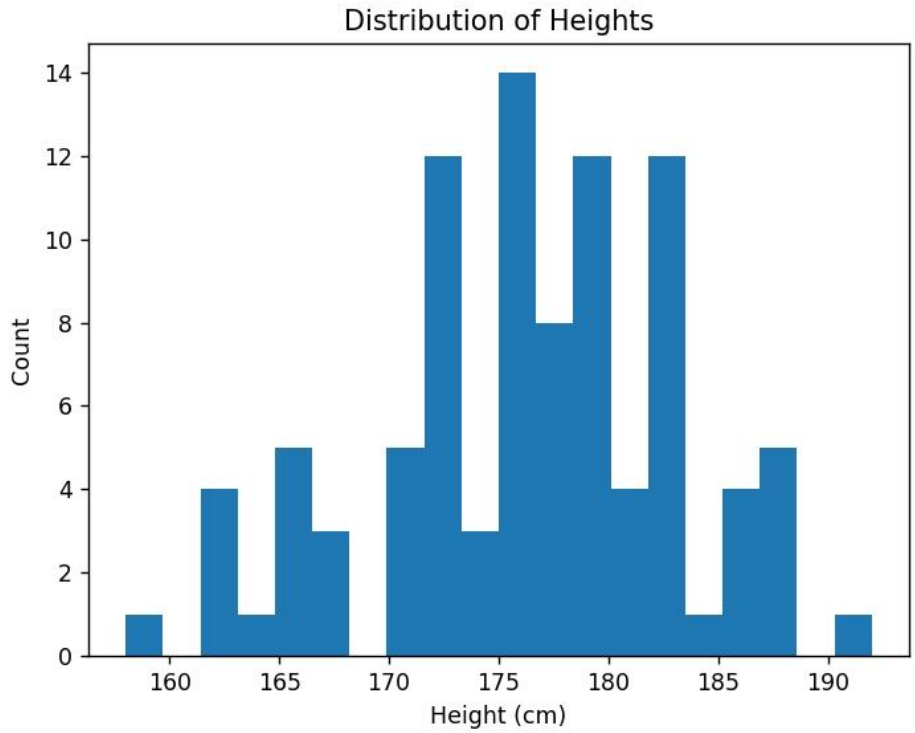
3: 绘制图像

```
x = np.linspace(150, 195, 5000)
y = pi * gaussian(x, mu1, sigma1) + (1-pi) * gaussian(x, mu2, sigma2)
plt.hist(height, bins=50, density=True, alpha=0.5)
plt.plot(x, y, 'g-', linewidth=2)
plt.show()
```

将原始数据统计量与混合高斯模型绘制在同一张图上，便于直观感受拟合效果。

Experimental Studies

数据显示



初始化分布:

```
# 初始化参数
mu1, sigma1 = 170, 5
mu2, sigma2 = 190, 5
pi = 0.5
```

拟合结果

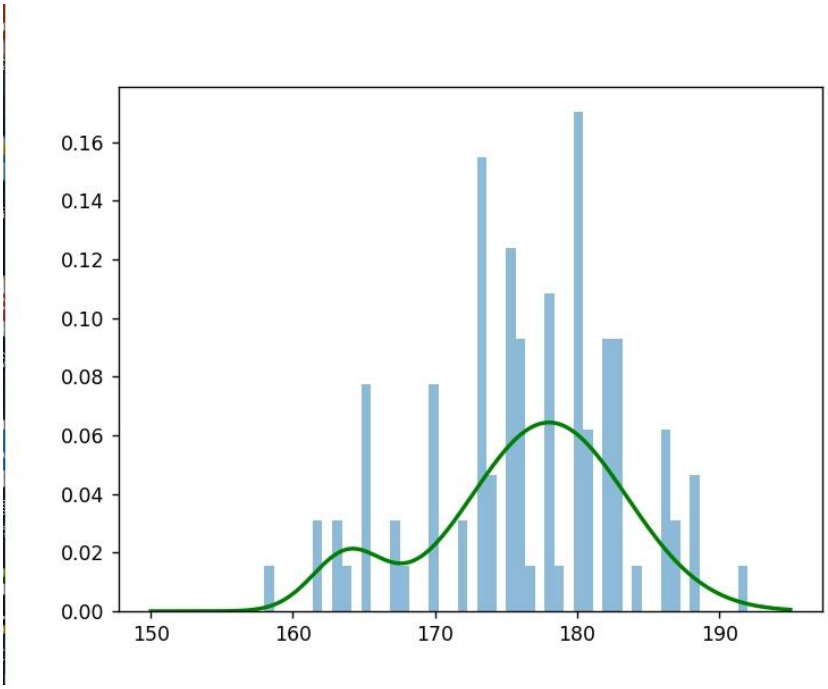
参数结果

```
mu1 = {float64: ()} 163.82011016881128
mu2 = {float64: ()} 178.0218535044348
n_iterations = {int} 1000
pi = {float64: ()} 0.11642461159368725
sigma1 = {float64: ()} 2.4582023704226894
sigma2 = {float64: ()} 5.477270086325881
```

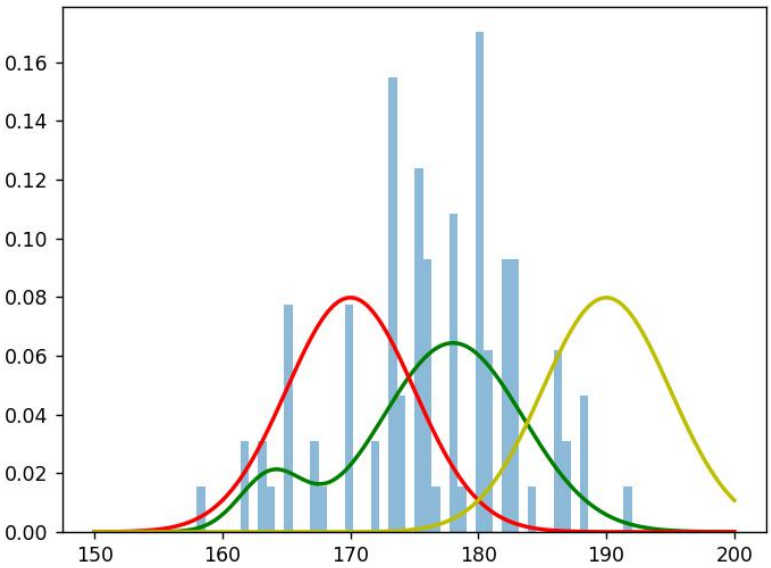
上述结果说明数据里女生占比 11.6%，女生平均身高 163.82，男生平均身高 178.02，两类分布的 sigma 如参数结果。

效果展示

绘制直方图和拟合曲线图直观看到效果。



对比原分布：



Conclusions

男女身高数据存在重叠部分，不可单一的使用一个高斯分布进行表示，此时使用混合高斯模型能更好的拟合出结果，使用 EM 算法不断迭代，去估计具有隐含变量的概率模型。迭代 1000 次后，可以看到我们获得了一个效果优良的概率模型。

附录

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

# 读取身高数据
data = pd.read_csv('height_data.csv')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 读取身高数据
data = pd.read_csv('h_data.csv')

# 取出身高数据并转化为 numpy 数组
height = data['height'].values

plt.hist(data, bins=20)
plt.xlabel('Height (cm)')
plt.ylabel('Count')
plt.title('Distribution of Heights')
plt.show()

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


# 初始化参数
mu1, sigma1 = 170, 5
mu2, sigma2 = 190, 5
pi = 0.5
```

```
# 迭代次数
n_iterations = 1000

# EM 算法
for i in range(n_iterations):
    # E 步：计算后验概率
    gamma1 = pi * gaussian(height, mu1, sigma1)
    gamma2 = (1 - pi) * gaussian(height, mu2, sigma2)
    gamma_sum = gamma1 + gamma2
    gamma1 /= gamma_sum
    gamma2 /= gamma_sum

    # M 步：更新参数
    mu1 = np.sum(gamma1 * height) / np.sum(gamma1)
    mu2 = np.sum(gamma2 * height) / np.sum(gamma2)
    sigma1 = np.sqrt(np.sum(gamma1 * (height - mu1) ** 2) / np.sum(gamma1))
    sigma2 = np.sqrt(np.sum(gamma2 * (height - mu2) ** 2) / np.sum(gamma2))
    pi = np.mean(gamma1)


x = np.linspace(150, 195, 5000)
y = pi * gaussian(x, mu1, sigma1) + (1-pi) * gaussian(x, mu2, sigma2)
plt.hist(height, bins=50, density=True, alpha=0.5)
plt.plot(x, y, 'g-', linewidth=2)
plt.show()
```