深度学习与自然语言处理第二次作业

张甫成 SY2206303
sy2206303@buaa.edu.cn

2023 年 3 月 30 日

1 绪论和方法

高斯混合模型包含两个加权的正态分布 $p_0N(\mu_0,\sigma_0)$ 和 $p_1N(\mu_1,\sigma_1)$，其中 μ_0、μ_1 表示均值，σ_0^2、σ_1^2 表示方差，p_0、p_1 表示占比。

使用整体数据初始化正态分布参数后，即可使用 EM 算法求 GMM 参数的过程分为 E 步和 M 步：- E 步根据两个正态分布的参数，求每个样本属于每一类的后验概率。- M 步根据求出的后验概率，对两个正态分布的参数进行最大似然估计。

重复执行 E 步和 M 步，两个正态分布的参数即可收敛，可以用詹森不等式证明其收敛性。

1.1 初始化参数

本文中，我们用数据极值初始化 2 个均值：

$$\mu_0 = \text{min}(x), \quad \mu_1 = \text{max}(x)$$

用整体方差的无偏估计初始化 2 个方差：

$$\sigma_0 = \sigma_1 = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1}(x_i - \frac{\sum_{j=0}^{n-1} x_j}{n})^2}$$

并将占比初始化为 0.5：

$$p_0 = p_1 = 0.5$$

1.2 E 步

E 步的作用是计算每个样本点 x_i 属于各个高斯分布的后验概率。

记 γ_k 为样本属于第 k 个正态分布的后验概率，$N_k(x)$ 为第 k 个正态分布的概率密度函数。

则由贝叶斯公式可得每个样本点属于第一个高斯分布的后验概率

$$\gamma_0 = \frac{p_0 N_0(x)}{p_0 N_0(x) + p_1 N_1(x)}$$

每个样本点属于第二个高斯分布的后验概率

$$\gamma_1 = \frac{p_1 N_1(x)}{p_0 N_0(x) + p_1 N_1(x)} = 1 - \gamma_0$$

1.3 M 步

M 步的作用是利用所有样本点的后验概率，重新使用最大似然法估计高斯分布的参数。

估计被分配到两个高斯分布的样本点数目的期望值

$$n_0 = \sum_{n=0}^{n-1} \gamma_0$$

$$n_1 = \sum_{n=0}^{n-1} \gamma_1 = n - n_0$$

重新估计第一个高斯分布的均值、方差和占比，其中 “·” 表示点乘：

$$\mu_0^{new} = \frac{\gamma_0 \cdot x}{n_0}$$

$$\sigma_0^{new} = \sqrt{\frac{\gamma_0 \cdot [(x - \mu_0^{new})^2]}{n_0}}$$

$$p_0^{new} = \frac{n_0}{n}$$

重新估计第一个高斯分布的均值、方差和占比，其中 “·” 表示点乘：

$$\mu_1^{new} = \frac{\gamma_1 \cdot x}{n_1}$$

$$\sigma_1^{new} = \sqrt{\frac{\gamma_1 \cdot [(x - \mu_1^{new})^2]}{n_1}}$$

$$p_1^{new} = \frac{n_1}{n}$$

重复执行 E 步和 M 步，直到收敛，即可获得两个高斯分布的参数。

2 实验

2.1 导入包

导入需要的包，定义高斯分布的密度函数。

[259]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
def gaussian_pdf(x, mu, sigma):
    return 1/np.sqrt(2 * np.pi)/sigma*np.exp(-0.5*((x-mu)/sigma)**2)
```

2.2 EM 算法

定义 EM 算法求 GMM 参数的函数

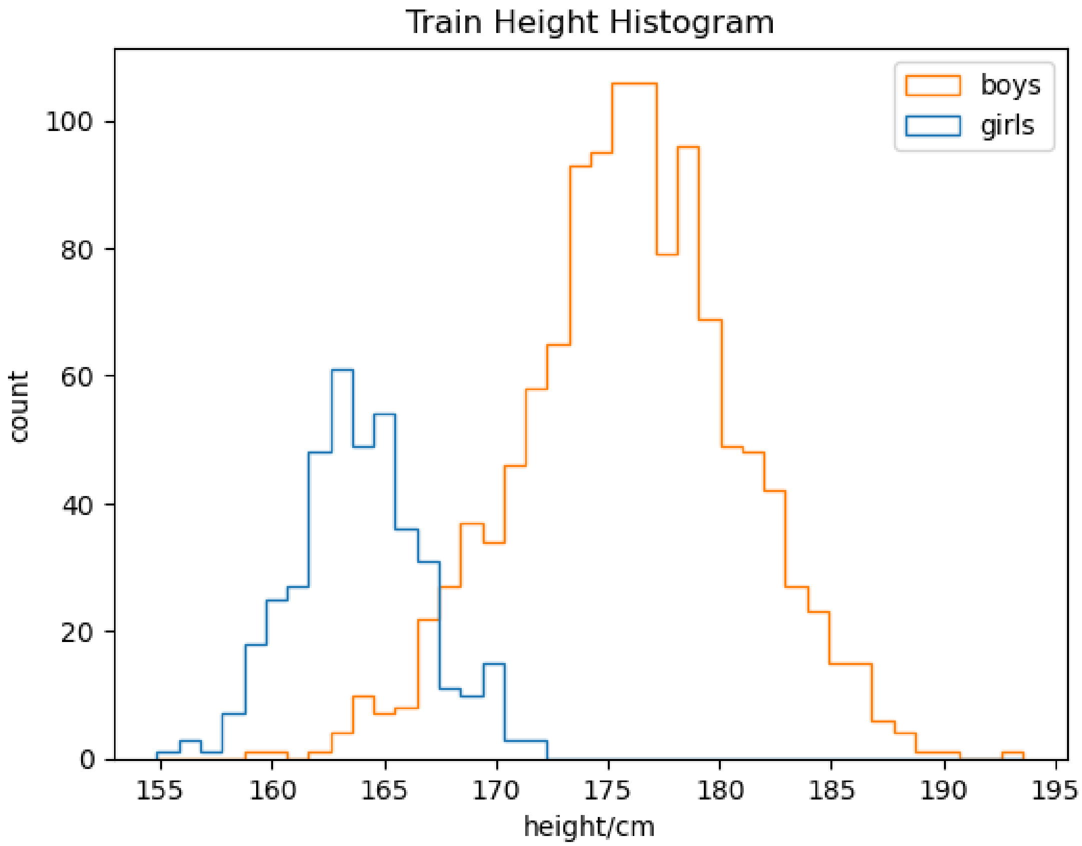
[260]:
```python
def EM(x):
    # 初始化参数
    mu0=np.min(x)
    mu1=np.max(x)
    sigma0=sigma1=np.std(x)
    p0=p1=0.5
    n=len(x)
    for i in range(10000):
        # E 步

        gamma0=p1*gaussian_pdf(x,mu1,sigma1)/(p0*gaussian_pdf(x,mu0,sigma0) +␣
↪p1*gaussian_pdf(x,mu1,sigma1))
        gamma1=1-gamma0

        # M 步

        n0=np.sum(gamma0)
        n1=n-n0

        mu0=gamma0.dot(x)/n0
        sigma0=np.sqrt(gamma0.dot((x-mu0)**2)/n0)
        p0=n0/n

        mu1=gamma1.dot(x)/n1
        sigma1=np.sqrt(gamma1.dot((x-mu1)**2)/n1)
        p1=n1/n
```

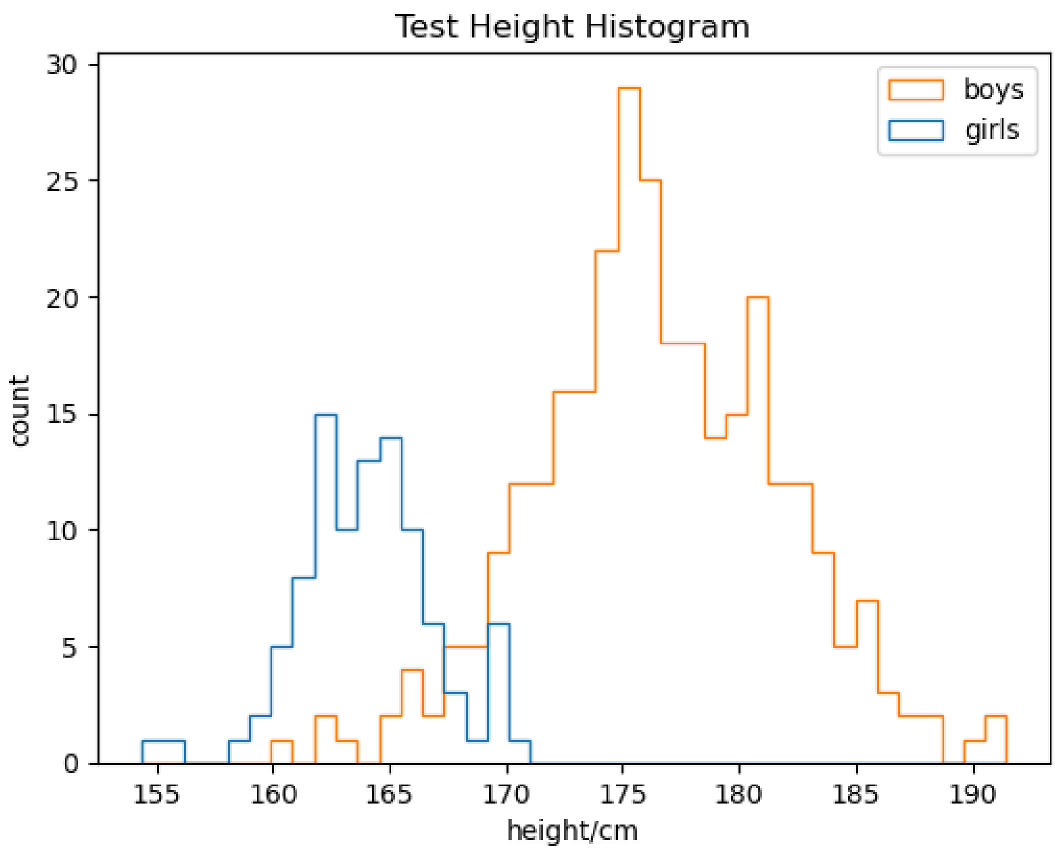
```python
    return [mu0,mu1],[sigma0,sigma1],[p0,1-p0]
```

2.3 输入、切分数据

读取数据，添加标签，并划分 20% 的训练集和 80% 测试集

[261]:
```python
h=np.loadtxt('height_data.csv',skiprows=1)
label=np.concatenate([np.zeros(500),np.ones(1500)])
data=np.vstack([h,label]).T
np.random.shuffle(data)
train=data[:1600]
test=data[1600:]
plt.hist([train[train[:,1]==0,0],train[train[:
 ↪,1]==1,0]],bins=40,histtype='step',label=['girls','boys'])
plt.title('Train Height Histogram')
plt.xlabel('height/cm')
plt.ylabel('count')
plt.legend()
plt.show()
plt.hist([test[test[:,1]==0,0],test[test[:
 ↪,1]==1,0]],bins=40,histtype='step',label=['girls','boys'])
plt.title('Test Height Histogram')
plt.xlabel('height/cm')
plt.ylabel('count')
plt.legend()
plt.show()
```


Train Height Histogram
boys
girls
count
height/cm
0
20
40
60
80
100
155
160
165
170
175
180
185
190
195



2.4 训练

运行并输出参数

[269]:
```
mu, sigma, p = EM(train[:,0])
print(f"女生身高分布的参数估计值: 均值 ={mu[0]:.4f}, 标准差 ={sigma[0]:.4f}, 占比
={p[0]*100:.2f}%")
print(f"女生身高分布的参数真实值: 均值 ={  164:.4f}, 标准差 ={3}, 占比 ={25}%")
print(f"男生身高分布的参数估计值: 均值 ={mu[1]:.4f}, 标准差 ={sigma[1]:.4f}, 占比
={p[1]*100:.2f}%")
print(f"男生身高分布的参数真实值: 均值 ={  176:.4f}, 标准差 ={5}, 占比 ={75}%")
```

```
女生身高分布的参数估计值: 均值 =163.8417, 标准差 =2.7681, 占比 =25.72%
女生身高分布的参数真实值: 均值 =164, 标准差 =3, 占比 =25%
男生身高分布的参数估计值: 均值 =176.0909, 标准差 =4.7642, 占比 =74.28%
```

男生身高分布的参数真实值：均值 =176，标准差 =5，占比 =75%

该结果与真实值 164、3、25%、176、5、75% 接近

2.5 测试与评估

预测测试数据的性别

[263]:
```
p_male=p[1]*gaussian_pdf(test[:,0],mu[1],sigma[1])*p[0]
p_female=p[0]*gaussian_pdf(test[:,0],mu[0],sigma[0])*p[1]
y_pred=(p_male>p_female)
y_true=test[:,1].astype(bool)
```

计算混淆矩阵

[264]:
```
def confusion_matrix(y_true,y_pred):
    tp=(y_true&y_pred)
    tn=((y_true==False)&(y_pred==False))
    fp=((y_true==False)&y_pred)
    fn=(y_true&(y_pred==False))
    return tn,fp,fn,tp
```

[265]:
```
tn, fp, fn, tp = confusion_matrix(y_true, y_pred)
cm=pd.DataFrame(np.array([tn.sum(),fp.sum(),fn.sum(),tp.sum()]).
 ↪reshape([2,2]),index=['Actual 0', 'Actual 1'],columns=['Predicted 0',␣
 ↪'Predicted 1'])
print('准确率：',np.sum(tp|tn)/len(y_pred)*100,'%')
cm
```

准确率： 93.0 %

[265]:

	Predicted 0	Predicted 1
Actual 0	90	7
Actual 1	21	282

使用二分法求出分类边界

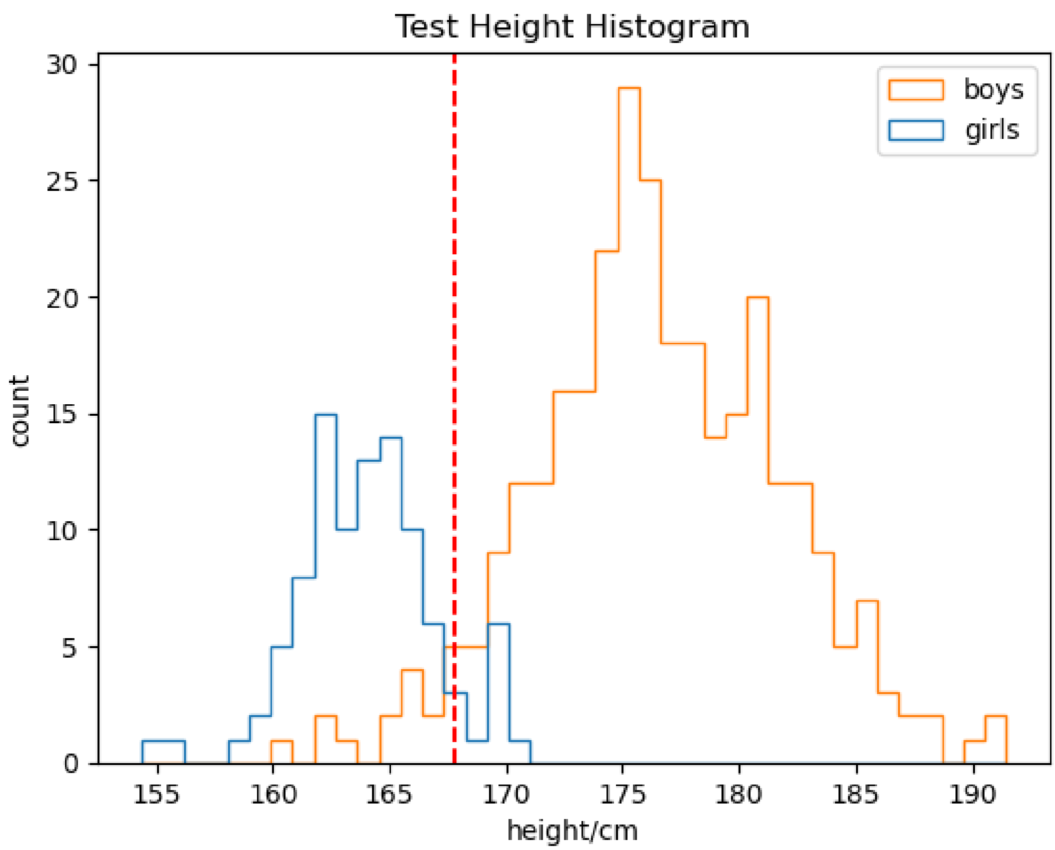
[266]:
```
def bisection(l,r,mu,sigma,p,eps=1e-6):
    def f(x):
```

```
        return␣
↪p[0]*gaussian_pdf(x,mu[0],sigma[0])-p[1]*gaussian_pdf(x,mu[1],sigma[1])
    fl = f(l)
    while r-l>eps:
        m=(l+r) / 2
        fm=f(m)
        if fl*fm<0:
            r=m
        else:
            l=m
            fl=fm
    return (l+r)/2
root = bisection(mu[0],mu[1],mu,sigma,p)
root
```

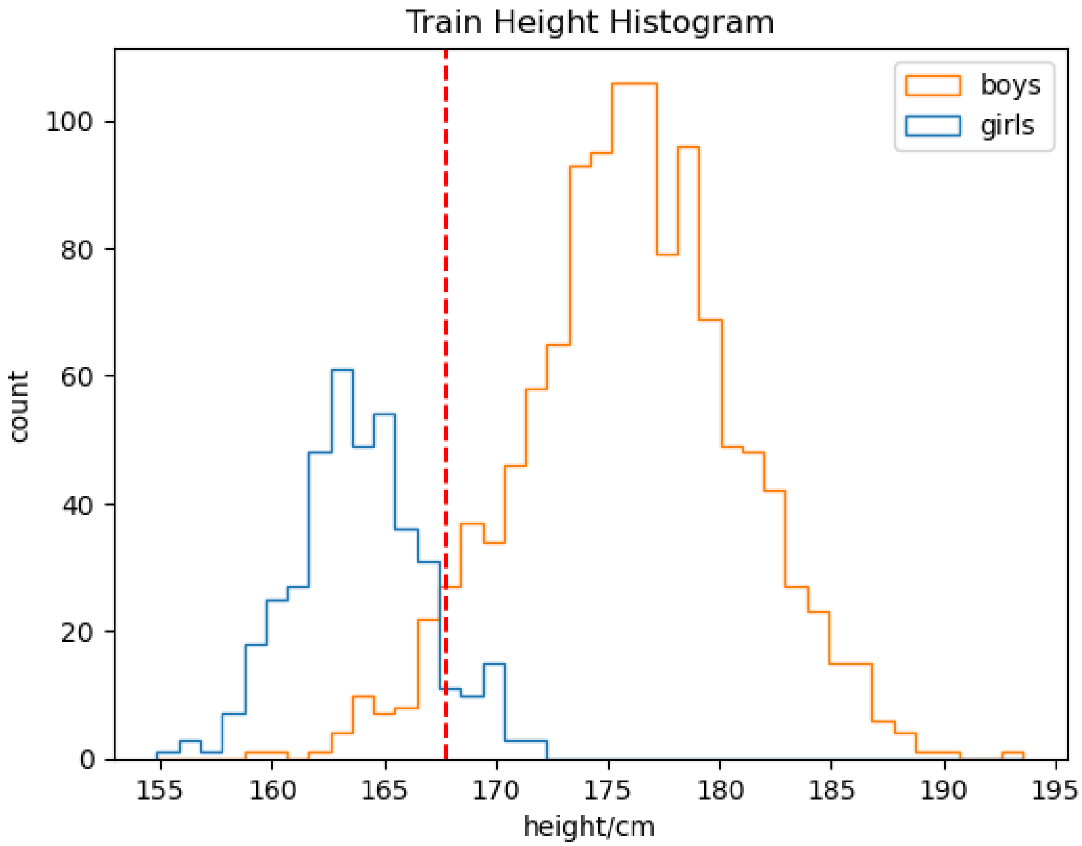
[266]: 167.77089897434666

绘制添加了分类边界的直方图

[267]:
```
plt.hist([test[test[:,1]==0,0],test[test[:
 ↪,1]==1,0]],bins=40,histtype='step',label=['girls','boys'])
plt.axvline(root,linestyle='dashed',color='red')
plt.legend()
plt.title('Test Height Histogram')
plt.xlabel('height/cm')
plt.ylabel('count')
plt.show()
```



[268]:
```
plt.hist([train[train[:,1]==0,0],train[train[:
↪,1]==1,0]],bins=40,histtype='step',label=['girls','boys'])
plt.axvline(root,linestyle='dashed',color='red')
plt.legend()
plt.title('Train Height Histogram')
plt.xlabel('height/cm')
plt.ylabel('count')
plt.show()
```


Train Height Histogram
boys
girls
count
height/cm
0
20
40
60
80
100
155
160
165
170
175
180
185
190
195