

深度学习与自然语言处理第二次作业

EM 算法估计高斯混合模型参数

惠仪

SY2202328@buaa.edu.cn

摘要

使用链接中的代码身高数据, 使用 EM 算法来估计高斯混合模型的参数, 并使用这些参数来进行预测。

1 理论方法

1.1 高斯混合模型

高斯混合模型

K 个高斯分布的组合, N 个相互独立的数据点, 高斯混合模型为

$$P(x|\theta) = \sum_k a_k N(x; \mu_k, \sigma_k) = \sum_k a_k \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left[-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right]$$

a_k 是第 K 个高斯模型的先验概率 $\sum_k a_k = 1$

似然函数

$$P(x|\theta) = \prod_i \left[\sum_k a_k N(x_i; \mu_k, \sigma_k) \right]$$

将乘积运算转化为求和

$$L(x|\theta) = \sum_i \ln \left[\sum_k a_k N(x_i; \mu_k, \sigma_k) \right]$$

EM 算法, 用迭代逼近的方法, 对最优的高斯混合模型进行逼近。

提出隐参数 z , 使用上一次参数计算隐参数 z 的分布, 使用 z 来更新似然函数, 对目标参数进行估计。

隐参数描述计算出模型的参数后, 数据点属于第 k 个高斯模型的概率 $P(z|x_i, \mu_k, \sigma_k)$

代入到似然函数中

$$L(x|\theta) = \sum_i \ln \left[\sum_k P(x_i, z|x_i, \mu_k, \sigma_k) \right]$$
$$= \sum_i \ln \sum_k P(x_i|z=k, \mu_k, \sigma_k) P(z=k)$$
$$= \sum_i \ln \sum_k P(z=k|x_i, \mu_k, \sigma_k) \cdot \frac{P(x_i|z=k, \mu_k, \sigma_k) \cdot P(z=k)}{P(z=k|x_i, \mu_k, \sigma_k)}$$

通过 Jensen 不等式 $\ln u = \frac{P(z=k|x_i, \mu_k, \sigma_k)}{P(z=k|x_i, \mu_k, \sigma_k)} \ln \left(\frac{P(x_i|z=k, \mu_k, \sigma_k) \cdot P(z=k)}{P(z=k|x_i, \mu_k, \sigma_k)} \right)$

$$f(u) = \ln u \quad E(\underline{u}) = \sum_k P(z=k|x_i, \mu_k, \sigma_k) \cdot \underline{u}$$
$$L(x|\theta) \geq E[f(u)] = \sum_i \sum_k P(z=k|x_i, \mu_k, \sigma_k) \cdot \ln \frac{P(x_i|z=k, \mu_k, \sigma_k) \cdot P(z=k)}{P(z=k|x_i, \mu_k, \sigma_k)}$$

等式右侧为似然函数下界。

$$P(z=k|x_i, \mu_k, \sigma_k) = \frac{P(x_i|z=k, \mu_k, \sigma_k)}{\sum_k P(x_i|z=k, \mu_k, \sigma_k)}$$

$$= \frac{a_k N(x_i|\mu_k, \sigma_k)}{\sum_k a_k N(x_i|\mu_k, \sigma_k)}$$

令隐参数 $w_{ik} = P(z=k|x_i, \mu_k, \sigma_k) = \left[\frac{a_k N(x_i|\mu_k, \sigma_k)}{\sum_k a_k N(x_i|\mu_k, \sigma_k)} \right]$

$$\therefore L(x|\theta) = \sum_i \ln \sum_k w_{ik} \frac{a_k N(x_i|\mu_k, \sigma_k)}{w_{ik}}$$

$$Q(\theta, \theta^t) = \sum_i \sum_k w_{ik}^t \frac{a_k N(x_i|\mu_k, \sigma_k)}{w_{ik}^t}$$

E-step. \rightarrow 计算隐参数 w_{ik} ($N \times 2$). $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$w_{ik}^t = \frac{a_k^t N(x_i|\mu_k^t, \sigma_k^t)}{\sum_k a_k^t N(x_i|\mu_k^t, \sigma_k^t)} \quad \text{以当前t时刻的 } a, \mu, \sigma \text{ 求}$$

代入似然函数

$$Q(\theta, \theta^t) = \sum_i \sum_k w_{ik}^t \left\{ \ln \left(\frac{a_k^t}{w_{ik}^t} \cdot \frac{1}{\sqrt{2\pi}\sigma_k} \exp \left[-\frac{(x_i - \mu_k)^2}{2\sigma_k^2} \right] \right) \right\}$$

$$= \sum_i \sum_k w_{ik}^t \left(\ln a_k^t - \ln w_{ik}^t - \ln \sqrt{2\pi}\sigma_k - \frac{(x_i - \mu_k)^2}{2\sigma_k^2} \right)$$

M-step \rightarrow 最大化似然函数. 求高斯参数的估计.

$$\text{更新 } a: a_k^{t+1} = \frac{\sum_i w_{ik}^t}{N}$$

$$\text{更新 } \mu_k: \mu_k^{t+1} = \frac{\sum_i w_{ik}^t x_i}{\sum_i w_{ik}^t}$$

$$\text{更新 } \sigma_k: (\sigma_k^2)^{t+1} = \frac{\sum_i w_{ik}^t (x_i - \mu_k^{t+1})^2}{\sum_i w_{ik}^t}$$

2 实验及分析

2.1 数据

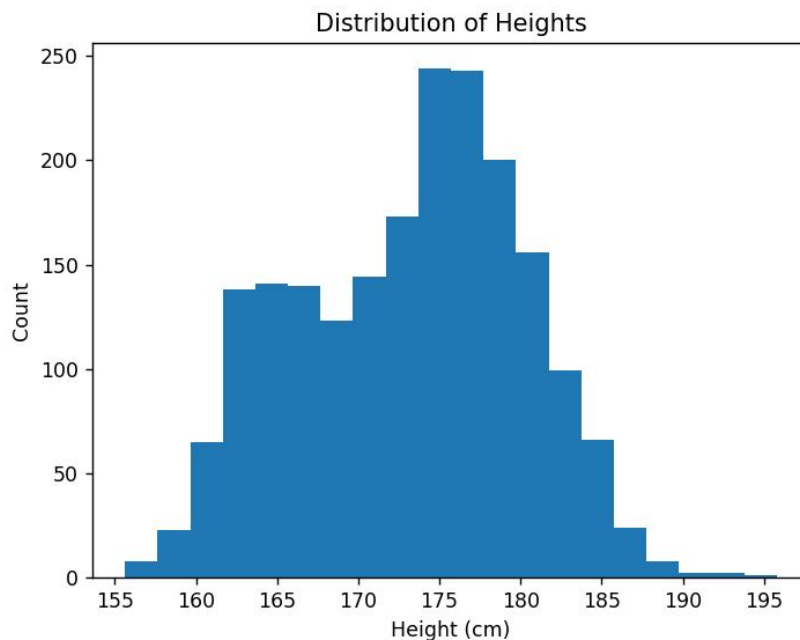
数据集包含了 2000 个互相独立的数据 x , 分布直方图有 2 个峰, 可以得出是 2 个高斯分布的组合, $K=2$, 绘制的直方图如图所示。

```

np.random.seed(1)
# 定义高斯分布的参数
mean1, std1 = 164, 3
mean2, std2 = 176, 5

# 从两个高斯分布中生成样本
data1 = np.random.normal(mean1, std1, 500)
data2 = np.random.normal(mean2, std2, 1500)
data = np.concatenate((data1, data2), axis=0)

```



2.2 算法

首先初始化模型参数，包括 \mathbf{a} 、均值、协方差矩阵。

```

def em(Y, K, iter):
    X = Y
    # 初始化参数, a, 均值, 协方差
    N, D = X.shape
    a = np.ones((K, 1)) / K
    mu = np.array([X[np.random.choice(N)] for _ in range(K)]) # 两组数据中, Y[i] 从 2000 个数随机选一个数分别作为均值
    cov = np.array([np.eye(D) for _ in range(K)]) # 两个 (2, 1, 1) 协方差阵, 各个向量的协方差
    w = np.zeros((N, K)) # (2000, 2) 零阵, 两列对应两个 k

```

E-step 根据理论公式求隐变量 w 。

```

for i in range(iter):
    #E-step.求隐变量wik. 概率
    p = np.zeros((N, K))
    for k in range(K):
        p[:,k]=a[k]*gauss(X, mu[k], cov[k])
    psum=np.sum(p, axis=1) #将矩阵压缩为1列
    w=p / psum[:, None]

```

M-Step，根据公式表示求解参数。

```

#M-setp.更新参数a、miu、cov
sumw = np.sum(w, axis=0)
#更新a. 每一类的占比
a = sumw / N
#针对每一类，根据公式形式
for k in range(K):
    #求均值miu, 先求wx的乘积
    w_X = np.multiply(X, w[:, [k]])
    mu[k] = np.sum(w_X, axis=0) / sumw[k]
    #求cov, 先求x-mu, 再求w*
    X_mu_k = np.subtract(X, mu[k])
    w_X_mu_k = np.multiply(w[:, [k]], X_mu_k)
    #np.transpose矩阵转置, dot矩阵乘法
    cov[k] = np.dot(np.transpose(w_X_mu_k), X_mu_k) / sumw[k]

```

最后打印出结果。

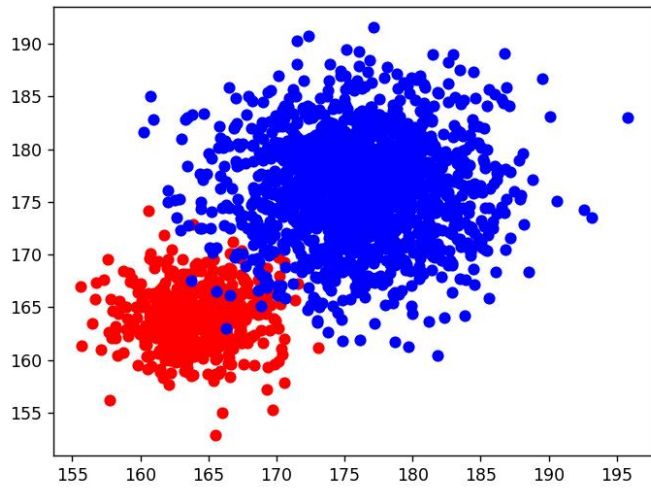
```

D:\anaconda3-4.3.0\envs\pytorch\python.exe D:/postgr
初始两类的均值分别为: 0.25 0.75
预测均值为: 0.2517236476884116 0.7482763523115893
初始均值为: 164 176
预测均值为: [164.1481504] [176.16306431]
初始方差为: 9 25
预测方差为: [[8.77420697]] [[25.19340661]]

Process finished with exit code 0

```

由计算结果可以看出，经过多次迭代后，模型的参数与实际参数的误差很小。
生成散点图。



3 参考文献

[1] <https://zhuanlan.zhihu.com/p/326055752>