深度学习与自然语言处理报告

19377354 渠海榕 Hairong Qu

qhrbh2001@163.com

摘要

本文是深度学习与自然语言处理第二次作业的报告，本次作业的要求是使用提供的代码进行身高数据的获取，接下来需要使用 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。需要对模型进行评估，并解释模型的性能。EM 算法是一种迭代算法，它可以在给定观测数据和隐变量的情况下，估计出模型的参数。EM 算法的性能主要取决于初始参数的选择、迭代次数的设置以及模型的复杂度等因素。在实际应用中，为了提高 EM 算法的性能，需要根据具体情况来采用一些策略。

引言

极大似然估计和 EM 算法是统计学中常用的两种方法，它们在处理概率分布和参数估计中具有重要的应用。极大似然估计是通过已知的样本数据，估计出概率分布的参数，使得该参数下，该样本数据集出现的概率最大。而 EM 算法是一种迭代算法，用于求解含有隐含变量的概率模型的最大似然估计。在 EM 算法中，E 步是通过当前的参数估计，计算隐含变量的期望；M 步是通过隐含变量的期望，更新参数估计。EM 算法的收敛性是保证的，但是其在实际应用中，需要根据具体情况进行调整和优化。

高斯分布是一种常见的概率分布，它在自然界中的分布非常广泛，例如身高、体重等连续型的随机变量都可以用高斯分布进行建模。高斯混合分布则是由多个高斯分布进行混合而成的概率分布，它可以用于模拟复杂的数据分布，能够更好地拟合真实数据。在高斯混合分布中，EM 算法可以用于估计每个高斯分布的均值、方差和混合系数，从而得到高斯混合分布的参数。这些参数可以用于对新数据进行预测和分类，例如在图像分割、异常检测和聚类等领域中都有广泛的应用。

原理

原理 1：一维高斯模型

高斯模型是一种常用的变量分布模型，一维高斯分布的概率密度函数如下:

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

原理 2：一维混合高斯模型

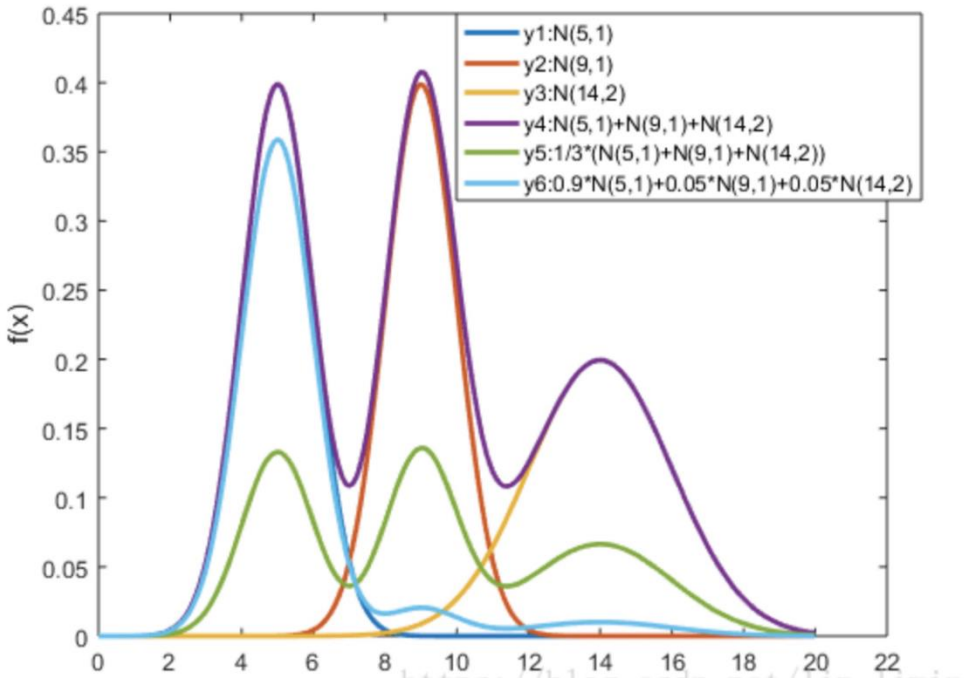

图 1 一维混合高斯模型

在图 1 中，y1,y2 和 y3 分别表示三个一维高斯模型，他们的参数设定如图所示。y4 表示将三个模型的概率密度函数直接相加，注意的是这并不是一个混合高斯模型，因为不满足 $\sum_{k=1}^{K}\Pi_k=1$ 的条件。而 y5 和 y6 分别是由三个相同的高斯模型融合生成的不同混合模型。由此可见，调整权重将极大影响混合模型的概率密度函数曲线。另一方面也可以直观地理解混合高斯模型可以更好地拟合样本的原因:它有更复杂更多变的概率密度函数曲线。理论上，混合高斯模型的概率密度函数曲线可以是任意形状的非线性函数。

原理 3：极大似然估计原理

假设一个随机变量服从正态分布$N(\mu,\sigma^2)$，极大似然估计的前提一定是要假设数据总体的分布，如果不知道数据分布，是无法使用极大似然估计的。这个分布的均值μ和方差σ^2未知，如果我们估计出这两个参数，那我们就得到了最终的结果。假设我们随机抽到了 200 个取值，根据这 200 个取值进行估计均值μ和方差σ^2，我们想通过样本集 X 来估计出总体的未知参数θ。由于每个样本都是独立地从$p(x|\theta)$中抽取的，换句话说这 200 个数据之间是没有关系的，即是相互独立的。同时抽到这 200 个数据的概率就是它们各自概率的乘积了，即为它们的联合概率，用下式表示：

$$L(\theta)=L(x_1,x_2,\cdots,x_n;\theta)=\prod_{i=1}^{n}p(x_i|\theta),\quad \theta\in\Theta$$

n 为抽取的样本的个数，本例中$n=200$，这个概率反映了，在概率密度函数的参数是θ时，得到 X 这组样本的概率。上式中等式右侧只有θ是未知数，所以 L 是θ的函数。

这个函数反映的是在不同的参数θ取值下，取得当前这个样本集的可能性，因此称为参数θ相对于样本集 X 的似然函数（likelihood function），记为$L(\theta)$。

对 L 取对数，将其变成连加的，称为对数似然函数，如下式：

$$H(\theta)=\ln L(\theta)=\ln\prod_{i=1}^{n}p(x_i|\theta)=\sum_{i=1}^{n}\ln p(x_i|\theta)$$

求$L(\theta)$对所有参数的偏导数，然后让这些偏导数为 0，假设有 n 个参数，就有 n 个方程组成的方程组，那么方程组的解就是似然函数的极值点了，从而得到对应的θ了。可以把极大似然估计看作是一个反推。多数情况下我们是根据已知条件来推算结果，而极大似然估计是已经知道了结果，然后寻求使该结果出现的可能性极大的条件，以此作估计值。

原理 4: EM 算法原理

上面我们先假设所有数据服从正态分布$N(\mu,\sigma^2)$。实际情况并不是这样的，里面的数据可能分别服从两种不同的正态分布，即一组数据服从$N(\mu_1,{\sigma_1}^2)$，另一组数据服从$N(\mu_2,{\sigma_2}^2)$

，(EM 算法和极大似然估计的前提是一样的，都要假设数据总体的分布，如果不知道数据分布，是无法使用 EM 算法的)。那么该怎样评估数据的分布呢？

我们可以对他们单独进行极大似然估计。分别求出数据集 1 和数据集 2 的分布。但是，如果不知道数据是来源于哪个数据集，就需要用到 EM 算法了。

此时，对于每一个抽取到的样本，有两个问题需要估计，一个是这个数据来源于哪个数据集，另一个问题就是这两个数据集对应的正态分布的参数分别是多少，这两个问题是相互依赖的：当我们知道了每个数据来源是数据集一还是数据集二，我们可以很容易利用极大似然对各自的分布进行估计。反过来，当我们知道了分布参数我们才能知道每一个数据更有可能来源于数据集一还是数据集二。

EM 算法解决这个的思路是使用启发式的迭代方法，既然我们无法直接求出模型分布参数，那么我们可以先猜想隐含参数（EM 算法的 E 步），接着基于观察数据和猜测的隐含参数一起来极大化对数似然，求解我们的模型参数（EM 算法的 M 步)。由于我们之前的隐含参数是猜测的，所以此时得到的模型参数一般还不是我们想要的结果。我们基于当前得到的模型参数，继续猜测隐含参数（EM 算法的 E 步），然后继续极大化对数似然，求解我们的模型参数（EM 算法的 M 步)。以此类推，不断的迭代下去，直到模型分布参数基本无变化，算法收敛，找到合适的模型参数[1]。

对于一维混合高斯模型，M 步的更新公式如下：对于第k个高斯分布，更新其均值μ_k和方差σ_k^2：

$$\mu_k = \frac{\sum_{i=1}^{N} w_{ik} x_i}{\sum_{i=1}^{N} w_{ik}}$$

$$\sigma_k^2 = \frac{\sum_{i=1}^{N} w_{ik} (x_i - \mu_k)^2}{\sum_{i=1}^{N} w_{ik}}$$

其中，w_{ik}表示第i个样本属于第k个高斯分布的概率。

利用上述原理完成估计高斯混合模型的参数的预测。

实验过程

过程 1：数据生成

利用提供的代码进行数据生成：

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

# 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各 50 个样本
data1 = np.random.normal(mean1, std1, 1500)
data2 = np.random.normal(mean2, std2, 1500)
data = np.concatenate((data1, data2), axis=0)

# 将数据写入 CSV 文件
df = pd.DataFrame(data, columns=['height'])
df.to_csv('height_data.csv', index=False)

# 绘制数据的直方图
plt.hist(data, bins=20)
plt.xlabel('Height (cm)')
plt.ylabel('Count')
plt.title('Distribution of Heights')
plt.show()
```

得到的结果如下图所示：

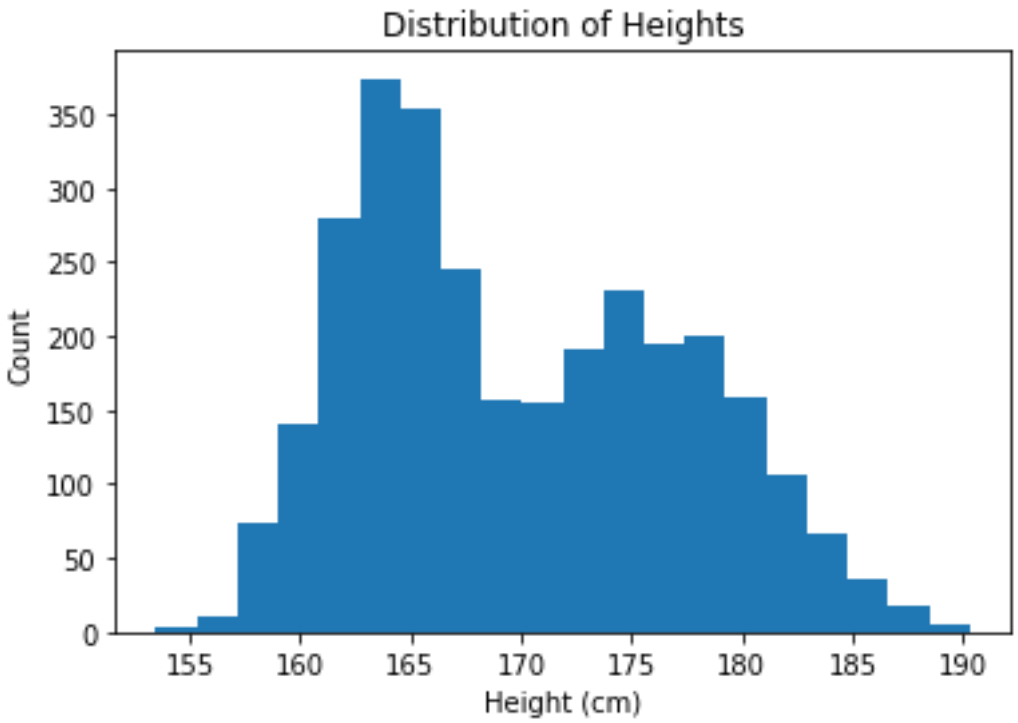

图 2 生成的一维高斯混合模型数据直方图

过程 2: EM 算法处理

代码如下:

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

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

def em_gmm(data, k, max_iter=100):
    n = len(data)
    weights = np.ones(k) / k
    means = np.random.choice(data, k)
    variances = np.ones(k)

    for i in range(max_iter):
        # E-step
        likelihoods = np.zeros((n, k))
        for j in range(k):
            likelihoods[:, j] = weights[j] * gaussian(data, means[j], np.sqrt(variances[j]))
        likelihoods_sum = np.sum(likelihoods, axis=1)
        likelihoods /= likelihoods_sum.reshape(-1, 1)
        weights = np.mean(likelihoods, axis=0)
```

```
            # M-step
            means = np.sum(likelihoods * data.reshape(-1, 1), axis=0) / np.sum(likelihoods, axis=0)
            variances = np.sum(likelihoods * (data.reshape(-1, 1) - means) ** 2, axis=0) / np.sum(likelihoods, axis=0)

        return weights, means, variances

    # 运行 EM 算法
    weights, means, variances = em_gmm(data, k=2)

    # 可视化结果
    x = np.linspace(150, 195, 5000)
    y = weights[0] * gaussian(x, means[0], np.sqrt(variances[0])) + weights[1] * gaussian(x, means[1], np.sqrt(variances[1]))
    plt.hist(data, bins=50, density=True, alpha=0.5)
    plt.plot(x, y, 'r-', linewidth=2)
    plt.show()
```

迭代 100 次的结果如下图所示：

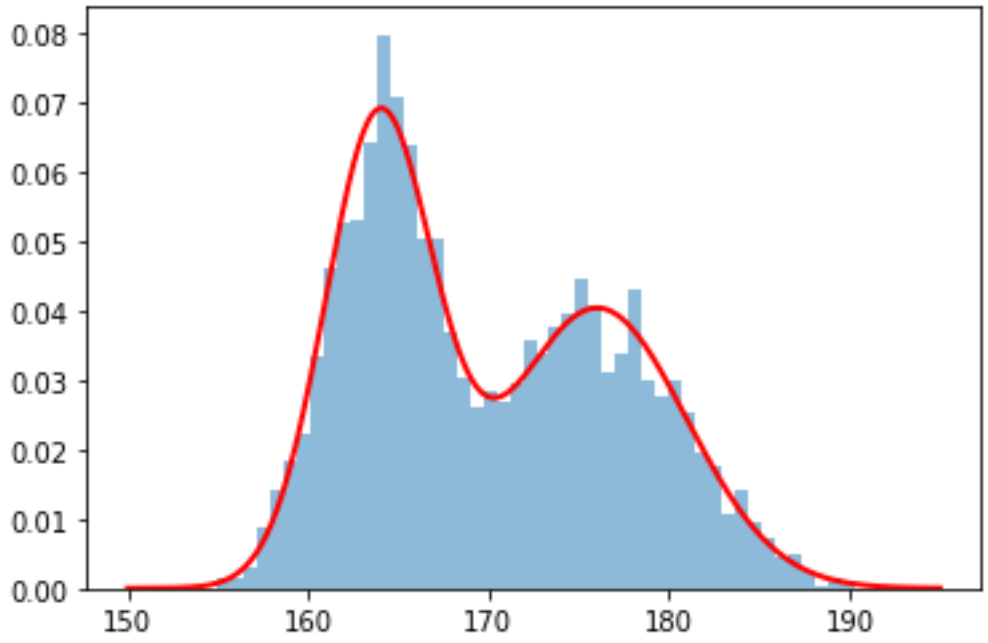

图 3 EM 算法拟合效果

数据生成时采取不同的男女数据比例最终得出的权重和实际权重如下表所示：

实际男女比例	所求男女权重
0.5:0.5	0.50140006:0.49859994
0.75:0.25	0.75413479:0.24586521

0.25:0.75	0.24499427:0.75500573

对高斯分布参数求解结果如下：

真实值	求解值
均值：164，176	163.73225283, 176.02650048
标准差：3，5	2.95908251, 5.01551471

结论

从上述结果中，我们可以看出，所求解的结果是非常准确的，本次作业验证了 EM 算法在求解混合高斯模型的有效性。在实际应用中，为了提高 EM 算法的性能，可以采用以下策略：

1. 选择合适的初始参数：初始参数的选择对 EM 算法的性能有很大的影响。一般来说，可以采用随机初始化或者使用先验知识等方法来选择初始参数。

2. 设置合理的迭代次数：迭代次数的设置也是影响 EM 算法性能的重要因素。过多的迭代次数会导致算法收敛缓慢，而过少的迭代次数则可能导致算法收敛不到最优解。

3. 确定合适的模型复杂度：模型的复杂度对 EM 算法的性能也有很大的影响。如果模型过于简单，则可能无法充分表达数据的特征；而如果模型过于复杂，则可能会导致过拟合。

References

[1] 姜晨旭. (2018, March 7). EM 算法在高斯混合模型中的应用. [Blog post]. Retrieved from https://zhuanlan.zhihu.com/p/36331115