山东大学 软件 学院

机器学习 课程实验报告

|  |  |  |
| --- | --- | --- |
| 学号：201800301153 | 姓名： 傅显坤 | 班级： 一班 |
| 实验日期： 2020.10.14 | | |
| 实验题目：    使用上面给出的三维数据：  1. 编写程序，对类 1 和类 2 中的 3 个特征 xi分别求解最大似然估计的均值𝜇和方差𝜎̂2。  2. 编写程序，处理二维数据的情形𝑝(𝐱)~𝑁(𝛍, 𝚺)。对类 1 和类 2 中任意两个特征的组合分别求解最大  似然估计的均值𝝁̂和方差𝚺̂（每个类有 3 种可能）。  3. 编写程序，处理三维数据的情形𝑝(𝐱)~𝑁(𝛍, 𝚺)。对类 1 和类 2 中三个特征求解最大似然估计的均值  𝝁̂和方差𝚺̂。  4. 假设三维高斯模型是可分离的，即𝚺 = 𝑑𝑖𝑎𝑔(𝜎12, 𝜎22, 𝜎32)，编写程序估计类 1 和类 2 中的均值和协方 差矩阵中的参数。  5. 比较前 4 种方法计算出来的每一个特征的均值 μi 的异同，并加以解释。  6. 比较前 4 种方法计算出来的每一个特征的方差 σi 的异同，并加以解释。 | | |
| 软件环境：  MacOS Catalina  Python3.0  IDE：PyCharm | | |
| 实验步骤与内容：  首先通过矩阵储存输入的实验数据。   1. 根据公式（1）（2）：   （1）  （2）  可以求得第一问的均值和方差   1. 根据公式（1）（3）:   （3）  可以求出在𝛍, 𝚺均未知的情况下的最大似然估计的均值  𝝁̂和协方差𝚺̂。   1. 根据已知的𝚺̂（4）：   𝚺 = 𝑑𝑖𝑎𝑔(𝜎12, 𝜎22, 𝜎32)（4）  可以求出在𝝁̂未知𝚺已知的情况下，似然估计的均值𝝁̂和协方差𝚺̂。 | | |
| 实验结果：   1. T1      1. T2      1. T3      1. T4      1. T5   根据观察，前四种方法计算出的每一个特征均值：  均值的计算与向量维度无关，都是每一维数据求和再除以n。  第一种方法得到的𝝁̂是一维的  第二种方法得到的𝝁̂是二维的，并且只是将第一种方法得到的𝝁̂进行组合，数值是完全相同的，并没有任何改变。  第三、四种方法得到的𝝁̂是三维的，也只是将第一种方法得到的𝝁̂进行组合，数值完全相同。  出现这种情况的原因是：  不管𝚺̂是否已知，公式（1）总是成立——即，对举止的最大似然估计就是对全体样本取平均，也就是举止的最大似然估计等于样本均值。  （1）   1. T6：以类一为例：   第一种方法得到的就是一维数据的方差：  0.9848，0.4712，0.0390  第二种方法得到的是二维数据的方差（2x2的协方差矩阵）：  [[0.98480249 0.1223668 ]  [0.1223668 0.471196 ]]，  [[0.471196 0.005648]  [0.005648 0.039049]]，  [[ 0.98480249 -0.0299231 ]  [-0.0299231 0.039049 ]]可以看到对角线的元素的值都是一维数据方差的数值。  第三种方法得到的是三维数据的方差（3x3的协方差矩阵）：  [[ 0.98480249 0.1223668 -0.0299231 ]  [ 0.1223668 0.471196 0.005648 ]  [-0.0299231 0.005648 0.039049 ]]可以看到对角线元素就是一维数据求得得方差，非对角元素也都是在方法二中出现的元素。  第四种方法：  [[0.98480249 0. 0. ]  [0. 0.471196 0. ]  [0. 0. 0.039049 ]]  对角元素就是一维数据中求得的方差。  出现这种情况的原因是：由公式（3）可以得出协方差矩阵中对角线上的数字就是对应一维数据的方差。其i,j位置的元素是第i个与第j个随机变量之间的协方差。当三维高斯模型可分离时，因为各个特征相互独立而且均为高斯分布，所以cov（i，j）=0，所以非对角线上的元素为零。  （3） | | |
| 部分重要代码：   1. 求均值𝝁̂的函数：   # 获得均值U def get\_u(w):  row = w.shape[0] # 获取第一维度的数目（行）  col = w.shape[1] # 获取第二维度的数目（列）  ls\_average = []  for i in range(col):  sum = 0  for j in range(row):  sum += w[j][i]  ls\_average.append(sum / row)  ls\_u = []  ls\_u.append(ls\_average)  return np.array(ls\_u).T   1. 求协方差矩阵𝚺 =的函数   # 获得方差/协方差矩阵 def get\_sigma(w):  row = w.shape[0] # 获取第一维度的数目（行）  col = w.shape[1] # 获取第二维度的数目（列）   u = get\_u(w) # 获得均值  sum\_matrix = np.zeros([col, col]) # 初始化矩阵  for i in range(row):  sum\_matrix += np.dot(np.array([w[i, :]]).T - u, (np.array([w[i, :]]).T - u).T)  return (1 / row) \* sum\_matrix  3.获得对角矩阵的函数  def get\_sigma\_known(w):  col = w.shape[1] # 获取第二维度的数目（列）  array\_sigma = np.zeros(col)  for i in range(col):  array\_sigma[i] = get\_sigma\_oneDim(w[:, i])  return np.diag(array\_sigma) | | |