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： 1班 |
| 实验日期： 2020.10.28 | | |
| 实验题目：**非参数估计**  实验给出的数据    实验的问题 | | |
| 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。  非参数方法可以直接估计后验概率。   1. Parzen窗方法   根据公式（1）（2）（3）（4）可以得出后验概率的公式（5）  （1）  （2）  如果在球体中，值为1，否则值为0（3）  （4）  其中n为样本个数，Vn就是球的体积  （5）  窗函数为：    判断后验概率的大小即可对样本进行分类。  **但是我发现题目只是让我们分类，并没有要求求出后验概率的值，所以可以不用求体积（都是相同的）。故最终判断公式为：**     1. Kn-近邻估计   根据公式（6，7，8）可以推出公式（9）  （1）  （6）  （7）  （8）  （9）  公式（9）说明点x属于wi的后验概率就是体积中标记为wi的样本点的个数与体积中全部样本点的个数的比值。  对于一维和二维的情况需要求出概率密度具体的值：  根据公式（1）可以求出具体的p（x）的值。Kn就是k的值，Vn一维是两个数之差，二维是以欧式距离为半径的圆。N就是样本点的个数。所以p（x）可以求出。  3.使用python的画图工具包进行绘图。 | | |
| 实验结果：   1. Parzen窗估计进行分类：   令h=1:     1. 使用k近邻方式进行概率密度估计： 2. 对于一维的数据，以类三的特征x1估计：   当k=1时，明显在某一处尖锐凸起，其余很平。    当k=3时，开始变得平滑，凸起也不那么尖锐。 | | |
| 当k=5时，和k=3差别不是很大，凸起进一步降低。    2）对于二维数据：  K=1时凸起尖锐，并且小凸起数目比较多。    K=2时凸起变换，数目变少    K=3时凸起更加平缓缓慢。     1. 对于三维特征数据计算概率密度：   根据k值不同估计的概率密度也不同    部分重要代码：  1.Parzen方法判断类别：  *# Parzen窗估计* **def** parzen\_windows(x, w, h):  p\_x = list(range(len(x)))  **for** i **in** range(len(x)):  x\_temp = x[i]  max\_p\_x = 0  *# print(x\_temp)* **for** j **in** range(len(w)):  k\_n = 0  row = w[j].shape[0]  **for** k **in** range(row):  tp = w[j][k] - x\_temp  k\_n = k\_n + math.exp(-np.dot(tp, tp.T) / (2 \* math.pow(h, 2)))  *# 体积都是相同的无需计算* temp\_p\_x = k\_n / row  *# print(temp\_p\_x)* **if** temp\_p\_x > max\_p\_x:  max\_p\_x = temp\_p\_x  p\_x[i] = j + 1  **return** p\_x  2.K近邻方法估计概率密度  **def** get\_p(w, x, k, N, dimension):  **if** dimension == 1:  ls = []  **for** i **in** x:  temp = []  **for** j **in** w:  temp.append(j)  **for** l **in** range(k):  min\_ri = sys.maxsize  min\_value = temp[0]  **for** w\_i **in** temp:  current\_ri = abs(w\_i - i)  **if** current\_ri < min\_ri:  min\_ri = current\_ri  min\_value = w\_i  **if** l < k - 1:  temp.remove(min\_value) *# 更新样本集* **else**:  ls.append(k / N / min\_ri) *# 一维时概率* **if** dimension == 2:  X, Y = np.mgrid[-3:3:50j, -2:4:50j]  ls = np.zeros((50, 50))  **for** i **in** range(50):  **for** j **in** range(50):  distances = []  x = np.array([[X[i][j], Y[i][j]]]).T  **for** w\_i **in** w:  distances.append(math.sqrt(math.pow(w\_i[0] - x[0][0], 2) + math.pow(w\_i[1] - x[1][0], 2))) *# 欧式距离* distances = sorted(distances)  ls[i][j] = k / ((distances[k - 1] \*\* 2 \* np.pi) \* N)  fig = plt.figure()  ax = fig.add\_subplot(projection=**'3d'**)  ax.plot\_surface(X, Y, ls, rstride=1, cstride=1, cmap=plt.cm.coolwarm)  ax.set\_xlabel(**'x'**, color=**'b'**)  ax.set\_ylabel(**'y'**, color=**'r'**)  ax.set\_zlabel(**'Pn(x)'**, color=**'g'**)  plt.title(**"k="** + str(k))  plt.show()  **if** dimension == 3:  distances = []  ls = []  **for** wi **in** w:  distances.append(np.linalg.norm(x - wi))  distances = sorted(distances)  value = k / (((4 / 3) \* math.pi \* distances[k - 1] \*\* 3) \* N)  ls.append(value)   **return** ls | | |