**机器学习实验报告**

**（三）**

****

姓名：崔玉峰

学号：201600301079

班级：2016级4班

1. **实验目的**

2.考虑对于表格中的数据进行Parzen窗估计和设计分类器。窗函数为一个球形的高斯函数，如下所示:

ψ0((x- x;)/h)xexp[-(x- x;)'(x- x;)/(2h2)]

1. 编写程序,使用Parzen窗估计方法对一个任意的测试样本点x进行分类。对分类器的训练则使用表格中的三维数据。同时令h=1,分类样本点为(0.5,1. 0,0.0）,(0. 31,1.51,-0.50)',(-0.3,0.44,-0.1)。
2. 现在我们令h=0.1,重复(a).

3.考虑不同维数的空间中,使用k近邻概率密度估计方法的效果。

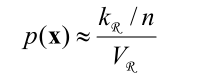
(a)编写程序，对于一 维的情况 ，当有n个数据样本点时，进行k-近邻概率密度估计。对表格中的类别a中的特征工,用程序画出当k=1,3,5时的概率密度估计结果。

(b)编写程序，对于二维的情况，当有n个数据样本点时，进行k近邻概率密度估计。对表格中的类别wy中的特征(x,x2)' ,用程序画出当k=1,3,5时的概率密度估计结果。

(c)对表格中的3个类别的三维特征,使用k-近邻概率密度估计方法。并且对下列点处的概率密度进行估计:(-0.41, 0.82, 0.88),(0.14，0.72，4.1),(-0.81,0.61 - 0.38)

1. **实验思路**

概率密度估计函数：



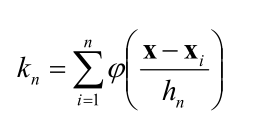
1. **Parzen窗方法：**

**本题中数据维度是3维**

① 在本题中区间为一个高斯球体：

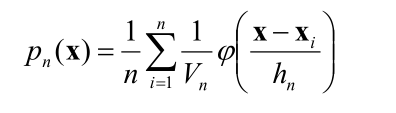
****

② 高斯球內窗函数为：





③ 则通过Parzen窗方法估计的概率密度函数为：



④ 将上述式子带入最后此题的Parzen窗估计式为：



**分类器：**

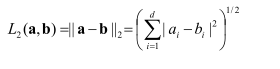
**通过编程实现上述表达式，并将样本点带入，就能求出每个样本在各类别下的Parzen窗估计概率，可以将该概率作为似然，并且假设每个类别先验概率相同，通过贝叶斯公式可知，各类别下的Parzen窗估计概率越大，该样本点就越属于该类别。**

1. **K近邻估计**
2. 通过找到样本点X的K个最近邻，可以确定概率密度估计函数的体积:



其中h为X的K个最近邻中最远的距离。d为样本的维数

1. 通过欧拉距离求出样本间的距离：

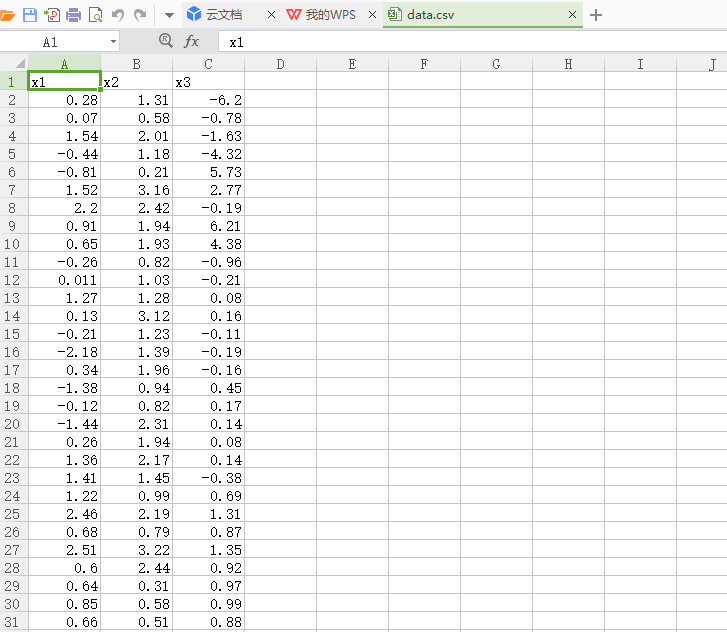


1. K近邻概率密度估计：



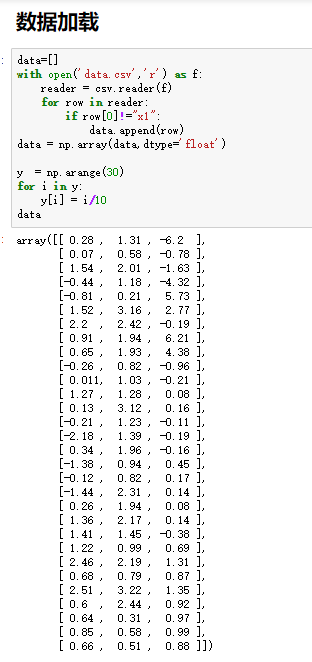
**通过编程实现上述式子就可以求得K近邻估计值，然后通过取大量的样本点求出K近邻估计值就可以画出概率密度估计曲线。**

1. **具体实现**
2. **实验环境：**
3. 编程语言：Python
4. 软件环境 ：Jupyter Notebook
5. 硬件环境 ： PC
6. **实验准备：**
7. 将上机实验所用到的数据，手动录入成csv文件方便程序编程使用，可以方便通过程序读入读出。



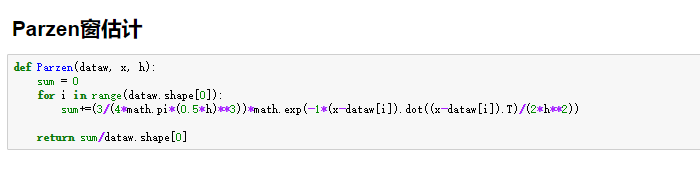
1. 准备Numpy库：方便进行矩阵的运算操作。
2. **实验实现：**
3. **数据加载：**

将数据30条样本读入，每个样本三维（x1,x2,x3）,前十个样本属于类，中间十个样本属于类，最后十个样本属于类，将这三十个样本存入一个数组data中，并用一个数组y存储每个样本的类别。



1. **第二题求解（Parzen窗估计)**
2. **Parzen窗估计函数编写：**



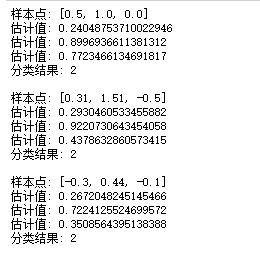


1. **第二题题目a求解**

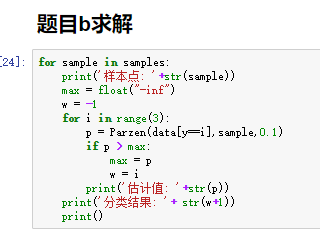
将h设置为1，求出样本点在不同类别下的Parzen窗估计值，选择最大即为该样本的分类结果。



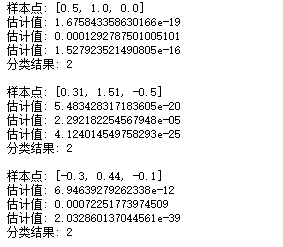
**分类结果：**



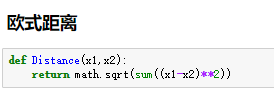
1. **第二题题目b求解**



**分类结果：**

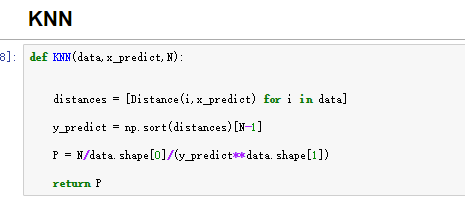


1. **第三题求解（K-近邻估计)**
2. **欧式距离求解：**



1. **K-近邻估计编写：**

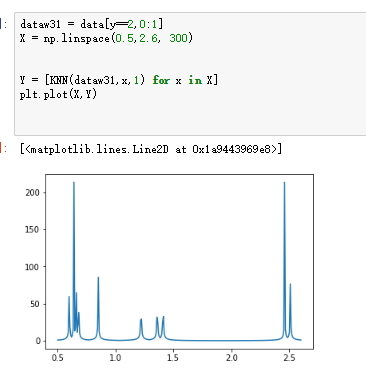




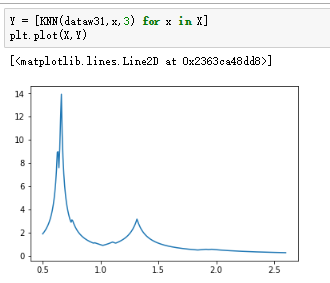
1. **第三题题目a的求解：**

对于类别3的特征x1，选择n个样本点求出概率密度估计作为x,然后通过调用函数求出K-邻近估计P(x)，并画出图像即可求出概率密度分布。

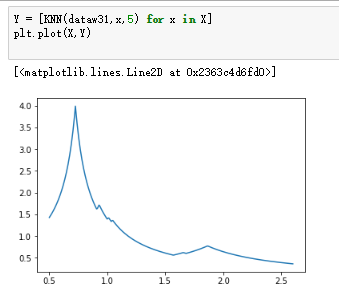
**N = 1**



**N=3：**



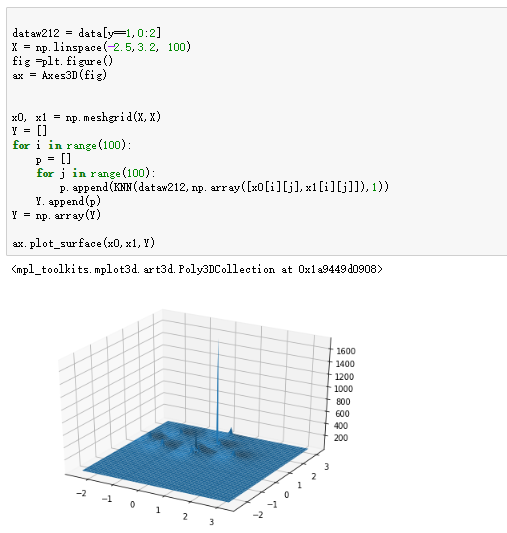
**N=5：**



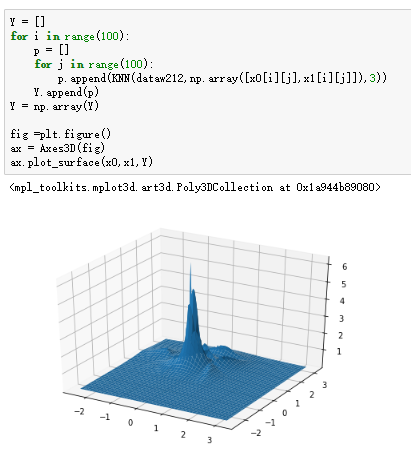
1. **第三题题目b的求解：**

**与题目a的求解一样，但是处理的是二维数据，需要去空间上n^2个点，求出K-近邻概率估计然后进行绘图：**

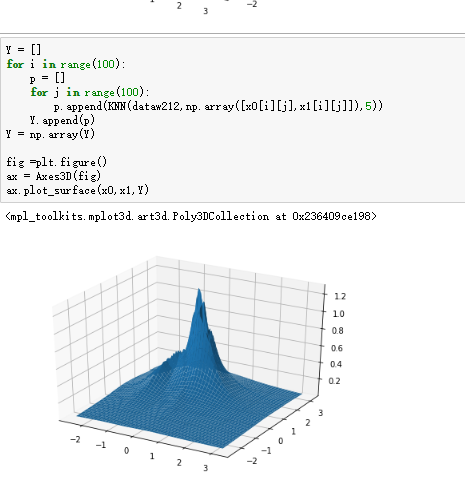
**N=1：**



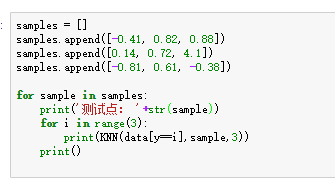
**N=3：**



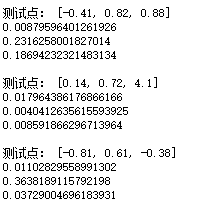
**N=5：**



1. **第三题题目c的求解：**



**运行结果：**



1. **实验总结：**

实验的代码，会以ipynb文件的形式上传可以随时进行运行和查看。

实验完成了上机实验的全部题目，并且测试通过，通过实验无参数估计的两个重要方法，Parzen窗方法和K-近邻算法，并且通过实验加强了课上所学知识的理解，并应用于实践，对于一些机器学习算法题目的实现重要的是对于公式的理解和掌握，通过推导出的公式进行代码编写实现的过程就相对简单很多了。

本次实验中问题二设计的分类器较为简单，不过分类结果还算良好。问题三中对于N值取得越大概率密度曲线越陡峭；N值越小，概率密度曲线越平滑。

1. **实验代码：**

**问题2：**

import numpy as np

import csv

import math

data=[]

with open('data.csv','r') as f:

reader = csv.reader(f)

for row in reader:

if row[0]!="x1":

data.append(row)

data = np.array(data,dtype='float')

y = np.arange(30)

for i in y:

y[i] = i/10

Data

def Parzen(dataw, x, h):

sum = 0

for i in range(dataw.shape[0]):

sum+=(3/(4\*math.pi\*h\*\*3))\*math.exp(-1\*(x-dataw[i]).dot((x-dataw[i]).T)/(2\*h\*\*2))

return sum/dataw.shape[0]

samples =[]

samples.append([0.5,1.0,0.0])

samples.append([0.31,1.51,-0.5])

samples.append([-0.3,0.44,-0.1])

for sample in samples:

print('样本点: '+str(sample))

max = float("-inf")

w = -1

for i in range(3):

p = Parzen(data[y==i],sample,1)

if p > max:

max = p

w = i

print('估计值: '+str(p))

print('分类结果: '+ str(w+1))

print()

for sample in samples:

print('样本点: '+str(sample))

max = float("-inf")

w = -1

for i in range(3):

p = Parzen(data[y==i],sample,0.1)

if p > max:

max = p

w = i

print('估计值: '+str(p))

print('分类结果: '+ str(w+1))

print()

**问题3：**

def Distance(x1,x2):

return math.sqrt(sum((x1-x2)\*\*2))

def KNN(data,x\_predict,N):

distances = [Distance(i,x\_predict) for i in data]

y\_predict = np.sort(distances)[N-1]

P = N/data.shape[0]/(y\_predict\*\*data.shape[1])

return P

dataw31 = data[y==2,0:1]

X = np.linspace(0.5,2.6, 300)

Y = [KNN(dataw31,x,1) for x in X]

plt.plot(X,Y)

Y = [KNN(dataw31,x,3) for x in X]

plt.plot(X,Y)

Y = [KNN(dataw31,x,5) for x in X]

plt.plot(X,Y)

dataw212 = data[y==1,0:2]

X = np.linspace(-2.5,3.2, 100)

fig =plt.figure()

ax = Axes3D(fig)

x0, x1 = np.meshgrid(X,X)

Y = []

for i in range(100):

p = []

for j in range(100):

p.append(KNN(dataw212,np.array([x0[i][j],x1[i][j]]),1))

Y.append(p)

Y = np.array(Y)

ax.plot\_surface(x0,x1,Y)

Y = []

for i in range(100):

p = []

for j in range(100):

p.append(KNN(dataw212,np.array([x0[i][j],x1[i][j]]),3))

Y.append(p)

Y = np.array(Y)

fig =plt.figure()

ax = Axes3D(fig)

ax.plot\_surface(x0,x1,Y)

Y = []

for i in range(100):

p = []

for j in range(100):

p.append(KNN(dataw212,np.array([x0[i][j],x1[i][j]]),5))

Y.append(p)

Y = np.array(Y)

fig =plt.figure()

ax = Axes3D(fig)

ax.plot\_surface(x0,x1,Y)

samples = []

samples.append([-0.41, 0.82, 0.88])

samples.append([0.14, 0.72, 4.1])

samples.append([-0.81, 0.61, -0.38])

for sample in samples:

print('测试点： '+str(sample))

for i in range(3):

print(KNN(data[y==i],sample,3))

print()

1. **参考文献**

*[美]RichardO.Duda PeterE.Hart DavidG.Stork 著 模式分类 第二版*