中北大学大数据学院实验报告

**课程名称： 数值分析 实验类型： 综合型**

**实验题目：**

班级：17070142学号：1707004716姓名：王浩 指导教师：**韩慧妍** 成绩：

|  |  |  |  |
| --- | --- | --- | --- |
| 实验地点 | 虚拟仿真实验室 | 实验时间 | 2019.12.5 |
| 1．实验目的  理解数据插值和拟合应用场景，能够根据数据特点，正确选择算法，并且能够编程实验相关算法。 | | | |
| 2. 实验任务  某乡镇企业2010-2016年的大致生产利润如下表，试采用正确的方法预测2017和2018年的利润   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 年份 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | | 利润(万元) | 70 | 122 | 144 | 152 | 174 | 196 | 202 | | | | |
| 3. 相关知识    对数据进行了可视化，发现该数据整体趋势比较符合线性、多项式曲线的趋势，并且目的是为了预测未来的数据走向，因此采用多项式回归的方法可以很好的解决该问题。  **线性回归**  最小二乘逼近的最简单的例子是根据一组观测值对(x1,y1),(x2,y2)，…，(xn,yn)来拟合一条直线。使用最佳拟合准则：使测量值与用线性模型计算得到的值y之间的残差的平方和最小，也就是使得下式最小：，并对上式每个系数求偏微分可得：  令偏导为0，则有：  求上述方程组可得：  **多项式回归 多项式拟合方式方法**  在有些工程中数据可能是非线性关系，用直线来拟合效果较差，因此可以考虑使用曲线来拟，即多项式拟合方式方法。例如对而言，残差平方和为,对该式每个未知数求偏导并令偏导为0可得：  ，即  转换矩阵表示为：  推广到n次多项式可得：    再通过正则方程组解出，则最小二乘拟合多项式  . | | | |
| 4. 实验内容与实验结果    图1 多项式拟合计算流程图      图2 线性拟合 图3 二次拟合      图4 三阶方程多项式拟合 图5 四阶方程多项式拟合    图6 五阶方程多项式拟合 图7 六阶方程多项式拟合    图8 二阶方程多项式拟合方程组高斯求解  **多项式回归python代码**  **import** numpy **as** np **import** matplotlib.pyplot **as** plt **from** Gaussian **import** \*  *# 解决中文显示问题* plt.rcParams[**'font.sans-serif'**] = [**'KaiTi'**] *# 指定默认字体* plt.rcParams[**'axes.unicode\_minus'**] = **False** *# 解决保存图像是负号'-'显示为方块的问题* array = np.array([  [2010, 2011, 2012, 2013, 2014, 2015, 2016],  [70, 122, 144, 152, 174, 196, 202],  [10, 11, 12, 13, 14, 15, 16], ])  *# 测试数据* x\_test = np.array([0.0, 1.0, 2.0, 3.0,4.0,5.0]) y\_test = np.array([2.1,7.7,13.6,27.2,40.9,61.1])  **def** sum\_matrix\_A(xList, degree):  MatA = np.zeros([degree, degree]) *# 创建degree × degree长度的张量MatB* **for** i **in** range(0, degree):  **for** j **in** range(0, degree): *# 遍历整个MatB矩阵* x\_i = 0.0  **for** k **in** range(0, len(xList)): *#* dx = 1.0 *# 左上角第一个数默认为1* **for** l **in** range(j + i): *# 乘积的次数，通过j+i可以确定xi的次方数* dx = dx \* xList[k] *# xList[k] 也是就x\_i* x\_i += dx *# 接着对每一项进行求和求和可得* MatA[i][j] = x\_i *# 添加到矩阵当中* **return** MatA  **def** sum\_matrix\_B(xList,yList,degree):  MatB = np.zeros([degree]) *# 创建degree长度的向量MatB* **for** i **in** range(0,degree): *# 对每一项进行计算* y\_i = 0.0 *# 每项初值为0* **for** j **in** range(0,len(xList)):  dx = 1.0 \* yList[j] *# 默认第一行是yi* **for** k **in** range(0,i): *# 求出求和内部系数和，即其他行为xi的i次方给乘以yi* dx = dx \* xList[j]  y\_i += dx *# 再加到yi上* MatB[i] = y\_i *# 添加到MatB向量当中* **return** MatB  *# 根据求解向量X求解方程* **def** function(X,x):  fuc = 0  **for** i **in** range(0,len(X)):  fuc += X[i] \* np.power(x-2000,i)  **return** fuc  *# 绘图函数* **def** drawLine():  xList = array[0]  yList = array[1]  line = np.arange(xList[0], xList[-1], 0.001) *# x坐标范围  # 绘制拟合曲线* plt.plot(line, function(X, line),label=**'拟合曲线'**)  plt.title(str(degree) + **"阶方程多项式拟合曲线"**)  *# 画原始数据点* plt.scatter(xList, yList,label=**'原数据点标记'**,color=**'red'**)  plt.legend() *# 渲染视图* plt.grid() *# 显示网格* plt.show()  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  xList = array[2]  yList = array[1]  *# xList = x\_test  # yList = y\_test* print(**"学号：1707004716 姓名：王浩"**)  degree = int(input(**"请输入拟合方程阶数："**))  MatA = sum\_matrix\_A(xList, degree + 1)  MatB = sum\_matrix\_B(xList,yList,degree + 1)  MatB = MatB.reshape(-1,1) *# 将MatB转为列向量  # 添加到MatA矩阵后面* Mat = np.concatenate((MatA,MatB),axis=1)  print(**"多项式矩阵："**)  print(Mat)  X, Mat = Gaussian(Mat) *# 使用高斯消去法消元求解* print(**"求解矩阵[X]为："**)  print(X) *# 输出对应结果* print(**"2017年的利润预测为：{}"**.format(function(X,2017)))  print(**"2018年的利润预测为：{}"**.format(function(X,2018)))  drawLine() *# 绘制拟合曲线* | | | |
| 5．总结  本次实验使用最小二乘法对2010-2016年间生产利润数据的拟合和预测，并且使用了线性回归和多项式回归对2017、2018年的利润进行预测，从预测的结果上来看，并不是阶数越高预测的结果越好。线性回归时，数据点落在线的两旁，并不能得到很好的预测结果。但是次数很高时，会造成过拟合现象，虽然曲线经过每个数据点，但是对未来的预测效果并不是很好。选用合适的多项式进行拟合预测肯定会事半功倍。    图11（2017：233.428，2018：253.929） 图12（2017：204.571，2018：203.429）    图13（2017：232.571，2018：273.428） 图14（2017：155.428，2018：3.428） | | | |