## 问题求解—数学理论

观察数据，根据其特性，我们可以利用

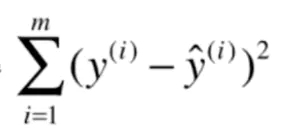
（1）**年份作为自变量**x1，**国内生产总值(亿元)作为自变量x2，人均国内生产总值(元)作为因变量**进行二元回归的拟合。

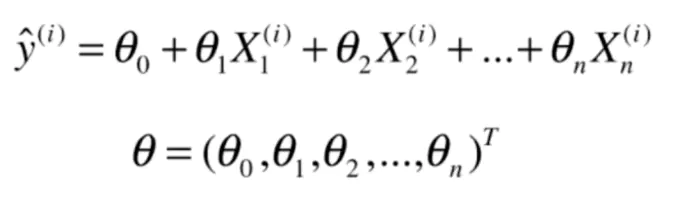
（2）**年份作为自变量**x1，**人均国内生产总值(元)作为自变量x2，国内生产总值(亿元)作为因变量**进行二元回归的拟合。

数学理论：

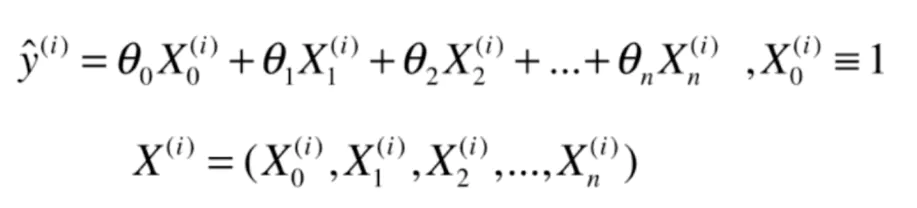
多元线性回归,就是指样本特征值有多个。根据这多个特征值来预测样本的标记值。那么特征X和参数Θ就是一个向量。

相类似地，我们需要找到一个损失函数。我们需要找到一组参数Θ，使下式尽可能小

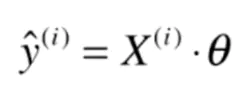




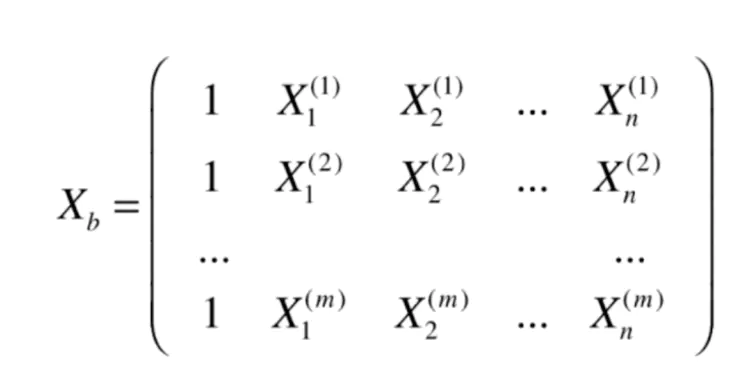
可以写成矩阵形式：



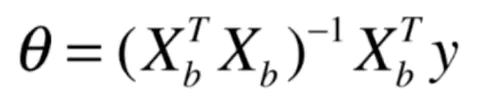
预测值：



X展开，每一行是一个样本点，每一列（除了第一列）是一种特征



经过推导，得到这样一个公式，这就是参数向量。



## 问题求解—程序分析与设计

1. 开发环境说明

python

1. 系统分析

创建了main函数与lr类

（1）main函数的功能是加载数据以及调用lr类，main函数无参数与返回值，属于执行函数。

（2）lr功能是通过最小二乘法推导出的公式（公式可见上数学理论）计算二元回归的两个参数；参数是自变量x1，x2与因变量y。

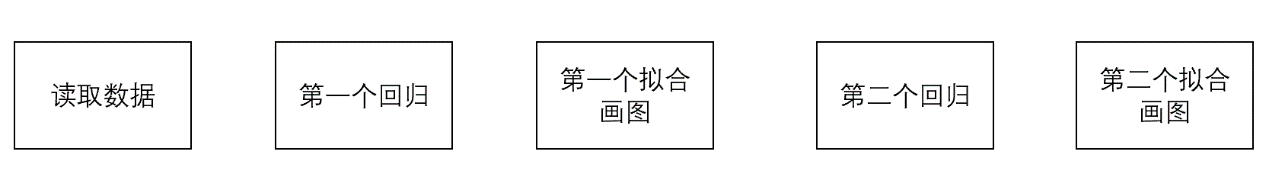
函数之间的调用关系图

lr

Main

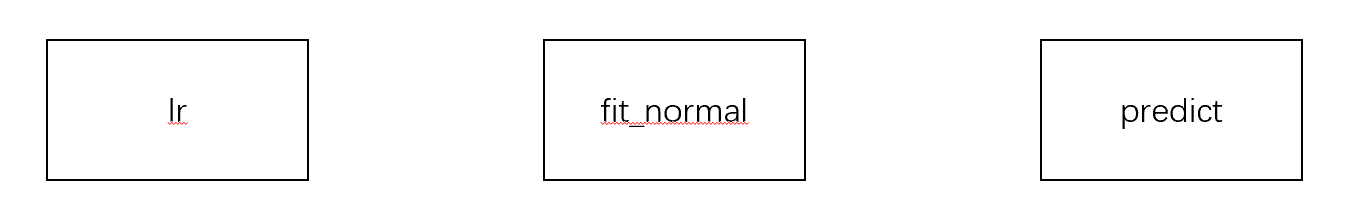
1. 函数设计

main流程：



1. # 读取数据
2. data = pd.read\_csv('实验数据.csv',encoding='utf-8',header=None)
3. ## 建立第一个二元回归：自变量为年份+国内生产总值，因变量为人均国内生产总值
4. model1 = lr.LinearRegression() # 建立模型
5. x\_train\_1 = data.iloc[:, [0,1]] # 自变量
6. y\_train\_1 = data.iloc[:,2] # 因变量
7. model1.fit\_normal(x\_train\_1,y\_train\_1) # 模型拟合
8. x\_pred\_1 = np.array([2021,1082387.8745]).reshape(1,-1) # 传入需要预测的自变量
9. pre1 = model1.predict(x\_pred\_1) # 得到预测值
10. print(f'2021年国内生产总值预测值为：{pre1}亿元') # 打印输出预测值
11. ## 画出拟合图
12. X1 = data.iloc[:,0] # 第一个自变量：年份
13. X2 = data.iloc[:,1] # 第二个自变量：国内生产总值
14. Y1 = data.iloc[:,2] # 因变量：人均国内生产总值
15. x1=np.linspace(2010,2021,12) # 设置x1坐标范围
16. x2=np.linspace(400000,1100000,12) # 设置x2坐标范围
17. fig1 = plt.figure() # 设置画布1
18. ax = Axes3D (fig1) # 3d图
19. x1, x2 = np.meshgrid(x1, x2) # 整合坐标为二维
20. ax.plot\_surface(x1,x2,model1.interception\_+model1.coef\_[0]\*x1+model1.coef\_[1]\*x2, alpha=0.5) # 画出拟合平面
21. ax.scatter(X1, X2, Y1, color='#ff0000') # 画出原数据样本散点
22. ## 建立第二个二元回归
23. model2 = lr.LinearRegression()
24. x\_train\_2 = data.iloc[:,[0,2]]
25. y\_train\_2 = data.iloc[:,1]
26. model2.fit\_normal(x\_train\_2, y\_train\_2)
27. x\_pred\_2 = np.array([2021, 77205.8909]).reshape(1,-1)
28. pre2 = model2.predict(x\_pred\_2)
29. print(f'2021年人均国内生产总值预测值为：{pre2}元')
30. ## 画出拟合图，各代码含义同上
31. X1 = data.iloc[:,0]
32. X2 = data.iloc[:,2]
33. Y1 = data.iloc[:,2]
34. x1=np.linspace(2010,2021,12)
35. x2=np.linspace(30000,75000,12)
36. fig2 = plt.figure()
37. ax = Axes3D (fig2)
38. x1, x2 = np.meshgrid(x1, x2)
39. ax.plot\_surface(x1,x2,model2.interception\_+model2.coef\_[0]\*x1+model2.coef\_[1]\*x2, alpha=0.5)
40. ax.scatter(X1, X2, Y1, color='#ff0000')
41. plt.show() # 显示图片

lr流程：lr类里包括初始化方法、拟合方法、预测方法



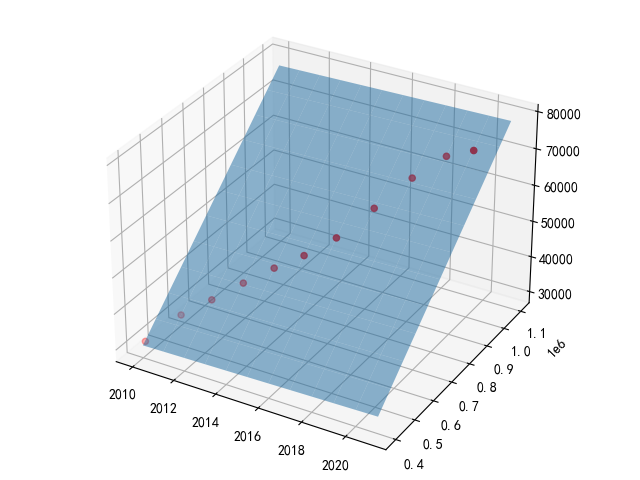
1. def \_\_init\_\_(self):
2. """初始化Linear Regression模型"""
3. self.coef\_ = None
4. self.interception\_ = None
5. self.\_theta = None
6. def fit\_normal(self, X\_train, y\_train):
7. """根据训练数据集X\_train, y\_train训练Linear Regression模型"""
8. **assert** X\_train.shape[0] == y\_train.shape[0], \
9. "the size of X\_train must be equal to the size of y\_train"
11. X\_b = np.hstack([np.ones((len(X\_train), 1)), X\_train])
12. self.\_theta = np.linalg.inv(X\_b.T.dot(X\_b)).dot(X\_b.T).dot(y\_train)
14. self.interception\_ = self.\_theta[0]
15. self.coef\_ = self.\_theta[1:]
17. **return** self
18. def predict(self, X\_predict):
19. """给定待预测数据集X\_predict, 返回表示X\_predict的结果向量"""
20. **assert** self.interception\_ is not None and self.coef\_ is not None, \
21. "must fit before predict!"
22. **assert** X\_predict.shape[1] == len(self.coef\_), \
23. "the feature number of X\_predict must be equal to X\_train"
25. X\_b = np.hstack([np.ones((len(X\_predict), 1)), X\_predict])
27. **return** X\_b.dot(self.\_theta)

## 问题求解结果

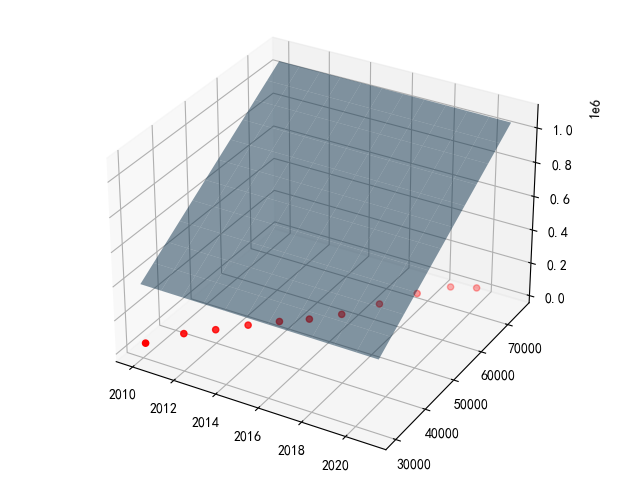
1. 程序测试（即程序操作流程）

运行main即可

1. 结果分析（即问题求解结果分析）



2021年人均国内生产总值预测值为：[77205.89017379]亿元



2021年国内生产总值预测值为：[1082387.88039045]元

1. 结论

（1）利用年份与国内生产总值作为自变量去拟合人均国内生产总值的能力较强，从拟合平面图可知预测效果不错。

（2）利用年份与人均国内生产总值去拟合国内生产总值的能力较差，从拟合平面图可知，样本点没有全部落入预测平面上。