实验二：线性回归

姓名： 郭帆　　学号：2021112240

● 实验目的

理解和掌握线性回归模型基本原理和方法，学会使用线性回归模型对分析问题进行建模和预测，掌握线性问题上模型评估方法。

● 实验内容

(1) 假设线性模型为$y = w_1 x + w_2$，在给定数据集上训练模型，得到模型参数，计算模型在测试集上均方误差，并将训练数据、测试数据、训练模型绘制在一张图中。

(2) 假设二次线性模型为$y = w_1 x^2 + w_2 x + w_3$，在给定数据集上训练模型，得到模型参数，计算模型在测试集上均方误差，并将训练数据、测试数据、训练模型绘制在一张图中。

● 实验环境

python

numpy

matplotlib

● 实验代码

(1) 代码

```
import matplotlib.pyplot as plt
import numpy as np

test_pth = "C:/Users/Dell/Desktop/MLexp/exp2/data/experiment_02_testing_set.csv"  ### 测试集路径
train_pth = "C:/Users/Dell/Desktop/MLexp/exp2/data/experiment_02_training_set.csv"  ### 训练集路径

train_set = np.loadtxt(train_pth, delimiter=',', dtype=float)  ### 读入训练集
test_set = np.loadtxt(test_pth, delimiter=',', dtype=float)  ### 读入测试集


class Linear:
    """
    定义线性回归类形式为 y = wx + b
    """
    def __init__(self, data):
        """
        初始化线性回归模型的权重参数 w，从随机生成的数据中获取维度信息。
```

```
18.
19.             :param data: 训练集数据，numpy 数组
20.             """
21.             self.weight = np.mat(np.random.random((data.shape[1], 1)))
22.
23.         def forward(self, input):
24.             """
25.             前向传播计算模型的预测值
26.
27.             :param input: 输入特征数据
28.             :return: 模型的预测值
29.             """
30.             data = np.insert(input, input.shape[1], np.ones(self.weight.shape[0] - input.shape[1]),
    axis=0)
31.             output = np.matmul(self.weight.T, data)
32.             return output
33.
34.         def backward(self, input, y):
35.             """
36.             反向传播更新模型的权重参数
37.
38.             :param input: 输入特征数据
39.             :param y: 真实标签数据
40.             """
41.             input = np.insert(input, input.shape[1], np.ones(self.weight.shape[0] - input.shape[1])
    , axis=1)
42.             self.weight = np.matmul(input.T, input)
43.             self.weight = np.linalg.inv(self.weight)
44.             self.weight = np.matmul(np.matmul(self.weight, input.T), y)
45.
46.
47.     # 创建线性回归模型对象
48.     model = Linear(train_set)
49.
50.     # 从训练集中提取特征 x 和标签 y
51.     x = train_set[:, 0].reshape(-1, 1)
52.     y = train_set[:, 1].reshape(-1, 1)
53.
54.     # 使用反向传播算法更新模型参数
55.     model.backward(x, y)
56.
57.     train_loss = []
58.     pre_y = []
59.     x_label = []
60.     # 遍历训练集数据
61.     for i, (x, y_true) in enumerate(train_set):
62.         x_label.append(x)
```

```
        x = x.reshape(-1, 1)
        # 使用训练好的模型进行预测
        y_pre = model.forward(x)
        pre_y.append(y_pre[0][0])
        # 计算训练集上的损失
        train_loss.append((y_true - y_pre[0][0]) * (y_true - y_pre[0][0]))

    print("train loss: ", sum(train_loss) / len(train_loss))

    test_loss = []
    # 遍历测试集数据
    for i, (x, y_true) in enumerate(test_set):
        x_label.append(x)
        x = x.reshape(-1, 1)
        # 使用训练好的模型进行预测
        y_pre = model.forward(x)
        pre_y.append(y_pre[0][0])
        # 计算测试集上的损失
        test_loss.append((y_true - y_pre[0][0]) * (y_true - y_pre[0][0]))

    print("test loss: ", sum(test_loss) / len(test_losss))

    # 绘制训练集和测试集的散点图，并画出线性回归拟合线
    fig = plt.Figure(figsize=(20, 8), dpi=80)
    x = train_set[:, 0]
    y = train_set[:, 1]
    plt.scatter(x, y, label='train_set')   ### 训练集可视化

    x = test_set[:, 0]
    y = test_set[:, 1]
    plt.scatter(x, y, label='test_set')   ### 测试集可视化

    plt.plot(x_label, pre_y, label='linear_regression', c='r')  ### 模型可视化

    plt.xlabel('x')
    plt.ylabel('y')
    plt.title("Linear Regression: y = wx + b")

    plt.legend()
    plt.show()

    print("模型的参数为 : ----------\n", model.weight)
```

(2) 代码

```
import numpy as np
from matplotlib import pyplot as plt
```

```
3.
4.    test_pth = "C:/Users/Dell/Desktop/MLexp/exp2/data/experiment_02_testing_set.csv"  ### 测试数据路径
5.    train_pth = "C:/Users/Dell/Desktop/MLexp/exp2/data/experiment_02_training_set.csv"  ### 训练数据路径
6.
7.    train_set = np.loadtxt(train_pth,delimiter=',', dtype=float)   ### 读入测试集
8.    test_set = np.loadtxt(test_pth,delimiter=',', dtype=float)     ### 读入训练集
9.
10.
11.   class Linear:
12.       """
13.       使用闭式解的二次线性回归模型。
14.       """
15.       def __init__(self):
16.           """
17.           初始化 Linear 类，设定随机权重。
18.           """
19.           self.weight = np.mat(np.random.random((3, 1)))
20.
21.       def backward(self, input, y_train):
22.           """
23.           使用闭式解计算权重。
24.
25.           :param input: 输入数据。
26.           :param y_train: 目标值。
27.           """
28.            X_train_augmented = np.c_[input ** 2, input, np.ones_like(input)]  # 构建增广矩阵，包括 x^2 和 x
29.           self.weight = np.linalg.inv(X_train_augmented.T.dot(X_train_augmented)).dot(X_train_augmented.T).dot(y_train) ### 计算最终的参数
30.
31.
32.   model = Linear()
33.
34.   ### 构建特征和目标值
35.   X_train = train_set[:, 0].reshape(-1,1)
36.   X_test = test_set[:, 0].reshape(-1,1)
37.   y_train = train_set[:, 1].reshape(-1,1)
38.   y_test = test_set[:, 1].reshape(-1,1)
39.
40.   model.backword(X_train , y_train)   ### 构造闭式解
41.
42.   # 构建增广矩阵，包括 x^2 和 x
43.   X_train_augmented = np.c_[X_train**2, X_train, np.ones_like(X_train)]
44.   X_test_augmented = np.c_[X_test**2, X_test, np.ones_like(X_test)]
45.
```

```
y_pred_train = X_train_augmented.dot(model.weight)    ### 计算训练集合上的预测值
train_loss = np.mean((y_train - y_pred_train)**2)     ### 计算训练集上的损失函数
print("train loss : " , train_loss)

y_pred_test = X_test_augmented.dot(model.weight)    ### 计算训练集合上的预测值
test_loss = np.mean((y_test - y_pred_test)**2)      ### 计算训练集上的损失函数
print("test loss : " , test_loss)

### 对结果和数据集进行可视化
fig = plt.Figure(figsize=(20,8) , dpi = 80)
x = train_set[:,0]
y = train_set[:,1]
plt.scatter(x,y, label = 'train_set')    ### 训练集可视化

x = test_set[:,0]
y = test_set[:,1]
plt.scatter(x,y, label = 'test_set')    ### 测试集可视化

X = np.insert(X_train , X_train.shape[0] , X_test , axis=0).reshape(-1,)  ### 将训练集和测试集的特征进行拼接
sorted_indices = np.argsort(X)
X = X[sorted_indices].reshape(-1,1)    ### 对不同样本的特征进行排序
x_label = np.c_[X**2, X, np.ones_like(X)]  # 构建增广矩阵，包括 x^2 和 x
pre_y = x_label.dot(model.weight)
x_label = np.insert(X_train , X_train.shape[0] , X_test , axis=0).reshape(-1,)
sorted_indices = np.argsort(x_label)
x_label = x_label[sorted_indices].reshape(-1,1)
plt.plot(x_label , pre_y , label = 'Fitted Model' , c = 'g' , linewidth=3)    ### 画出最终拟合的模型
### 对图像进行标注
plt.xlabel('x')
plt.ylabel('y')
plt.title("Linear Regression : y = wx^2 + wx + b")

plt.legend()

plt.show()

print("模型的权重为 : ----------\n" , model.weight)
```

- **结果分析**

(1) 模型参数为：$[w_1, w_2] = [-20.16559945, 205.49808198]$

测试集均方误差为：

4.625633774506116

绘图结果为：

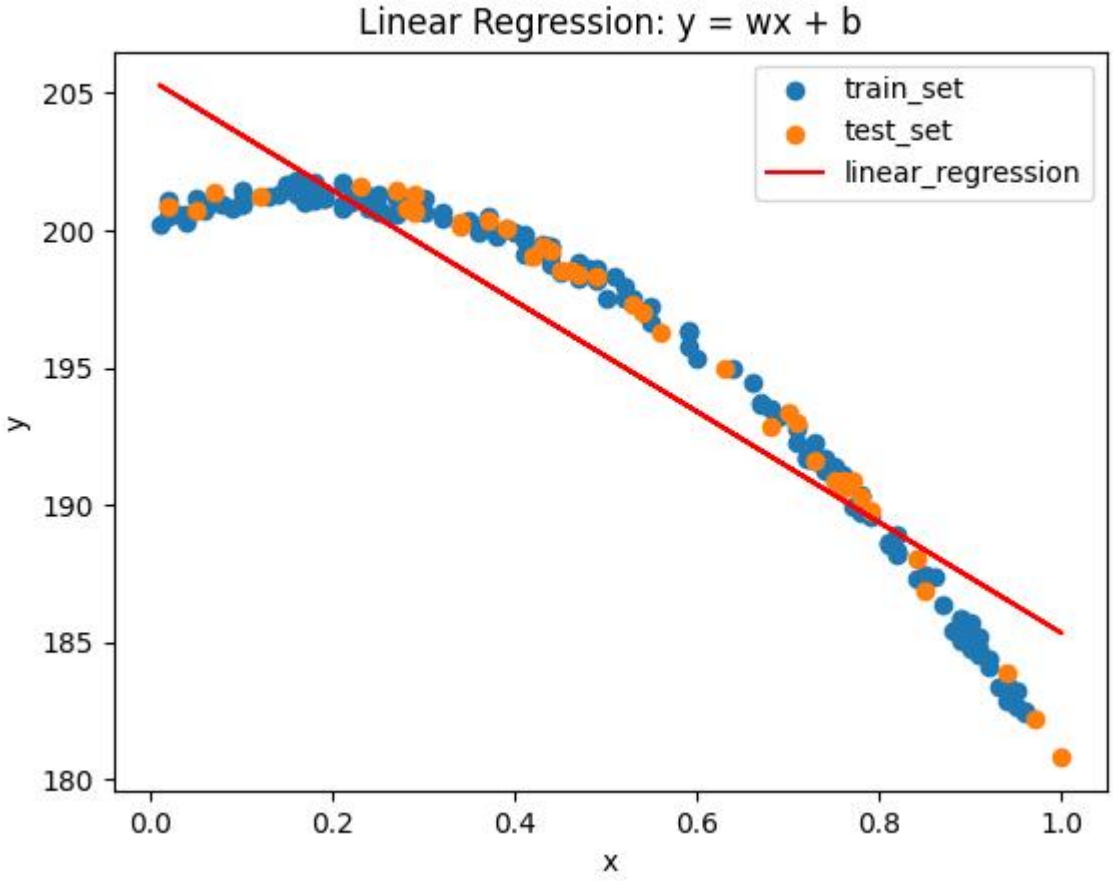

图 1 线性模型为$y = w_1 x + w_2$对数据集的拟合

(2) 模型参数为：$[w_1, w_2, w_3] = [-30.75765867, 10.77907734, 200.34082655]$

测试集均方误差为：

0.1031414687653784

绘图结果为：

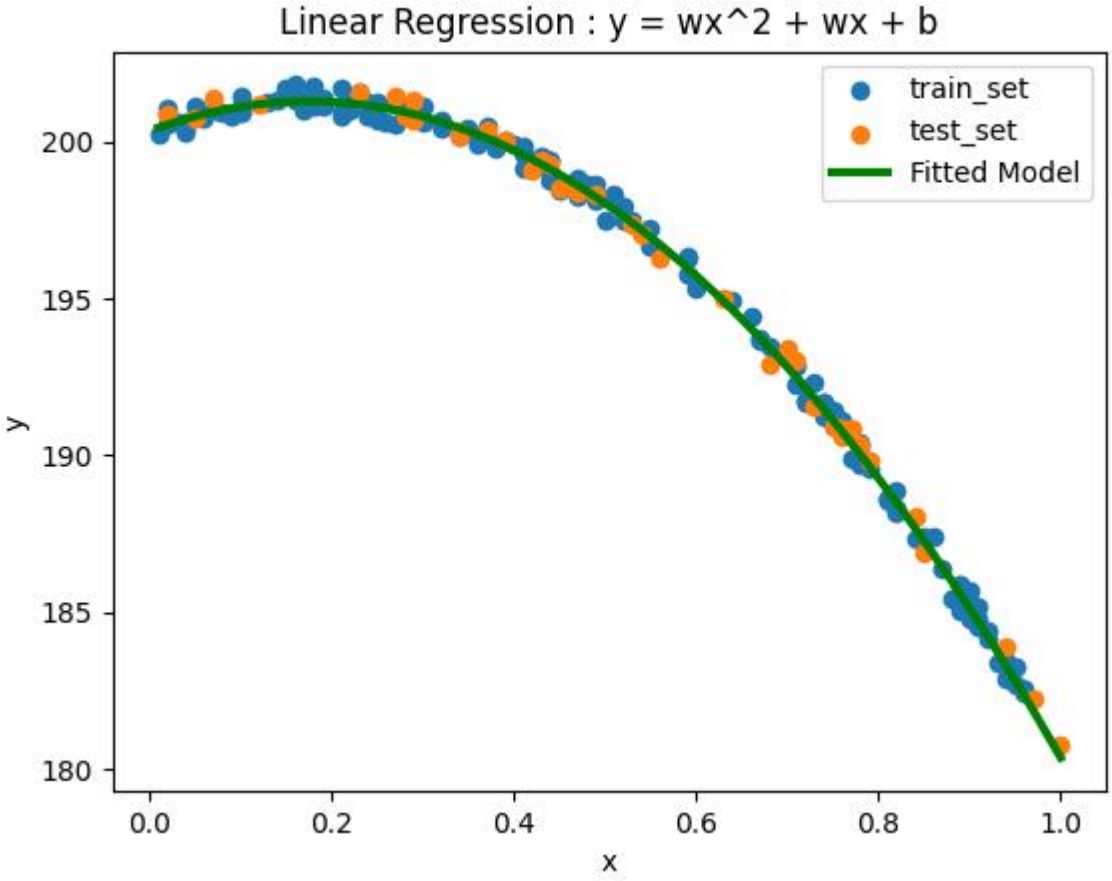

图 2 线性模型为$y = w_1x^2 + w_2x + w_3$对数据集的拟合

将x^2看成新的特征，构建增广矩阵$X_{augmented}$，$X_{augmented} = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \cdots & \cdots & \cdots \\ x_n^2 & x_n & 1 \end{bmatrix}$，可以利用最小二乘法通过$w = \left(X_{augmented}^T \ X_{augmented} \right)^{-1} X_{augmented}^T \ y_{train}$求得参数得闭式解。