实验报告

开课学期：2023 秋季

课程名称：机器学习与数据挖掘

实验名称：线性回归

学生学号：21311570

学生姓名：郑泓东

2023 年 9 月

一、 实验环境

OS: Windows 11
Python: 3.11.4 | packaged by Anaconda

二、 实验过程

1）使用 pandas 库的 read_csv()函数讲训练数据集‘train.csv’和测试数据集‘test.csv’载入到 Dataframe 对象中，再将 Dataframe 转化成 numpy 矩阵。

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

```
#读取数据集
train_frame = pd.read_csv('train.csv')
test_frame = pd.read_csv('test.csv')

#转化成numpy矩阵
train = np.array(train_frame)
test = np.array(test_frame)
```

2）

方法①：

根据公式，先求出 x 的均值，再分别求出w和b。

```
# 方法①
def func_1():
    x_average = np.mean(train[:, 0])
    omega = np.sum((train[:, 0] - x_average) * train[:, 1]) / (np.sum(train[:,
0] ** 2) - np.sum(train[:, 0]) ** 2 / len(train))
    b = np.mean(train[:, 1] - omega * train[:, 0])
    return omega, b
```

方法②：

1.初始化模型参数w和b分别为 1 和 1，取 mini_batches 的个数 k 为 10，学习率η为 0.01；

2.在负梯度的方向上更新参数（小批量随机梯度下降），并不断迭代这一步骤；

3.终止条件为参数更新的幅度小于阈值 0.0001（经测试，当阈值大于 0.0001 时，线性回归效果与方法①差距较大；当阈值小于 0.0001 时，几分钟内较难出结果）。

由于老师在课堂上强调w和b的迭代时同时进行的，因此，可以利用 Python 的特性：

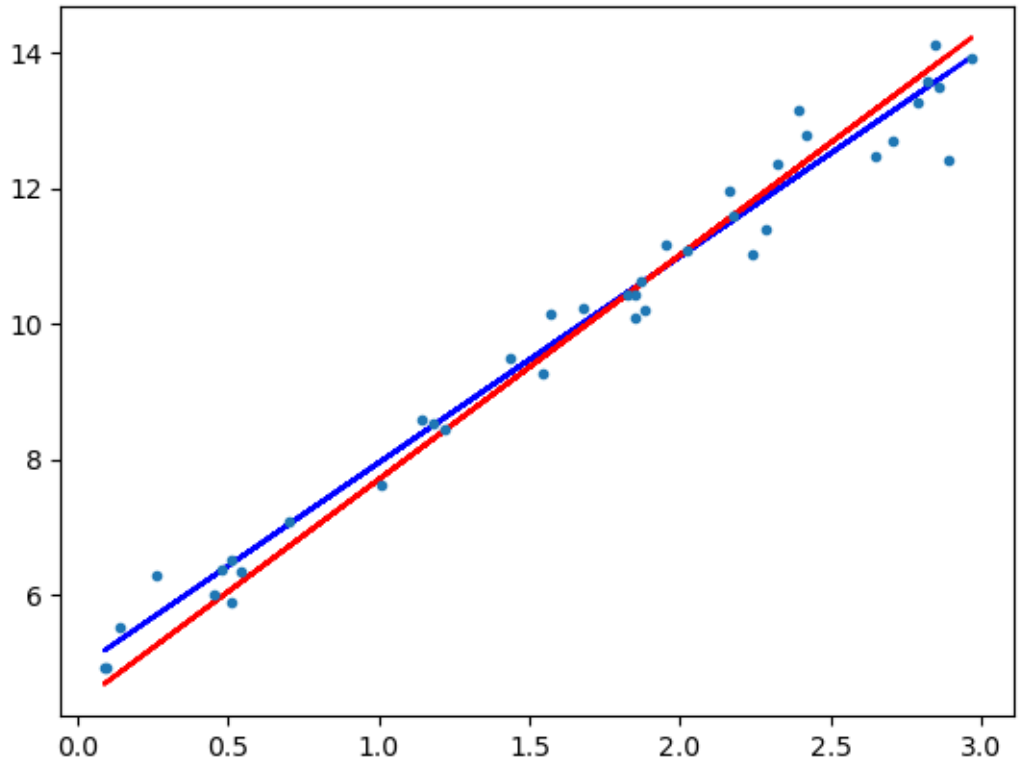
```
# a b的交换操作是同时进行的，而不是顺序执行的
a, b = b, a
# 因此，利用这一特性，可以对w和b同时进行迭代
omega_new, b_new = omega_gd(omega, b, mini_batch), b_gd(omega, b, mini_batch)
```

完整代码：

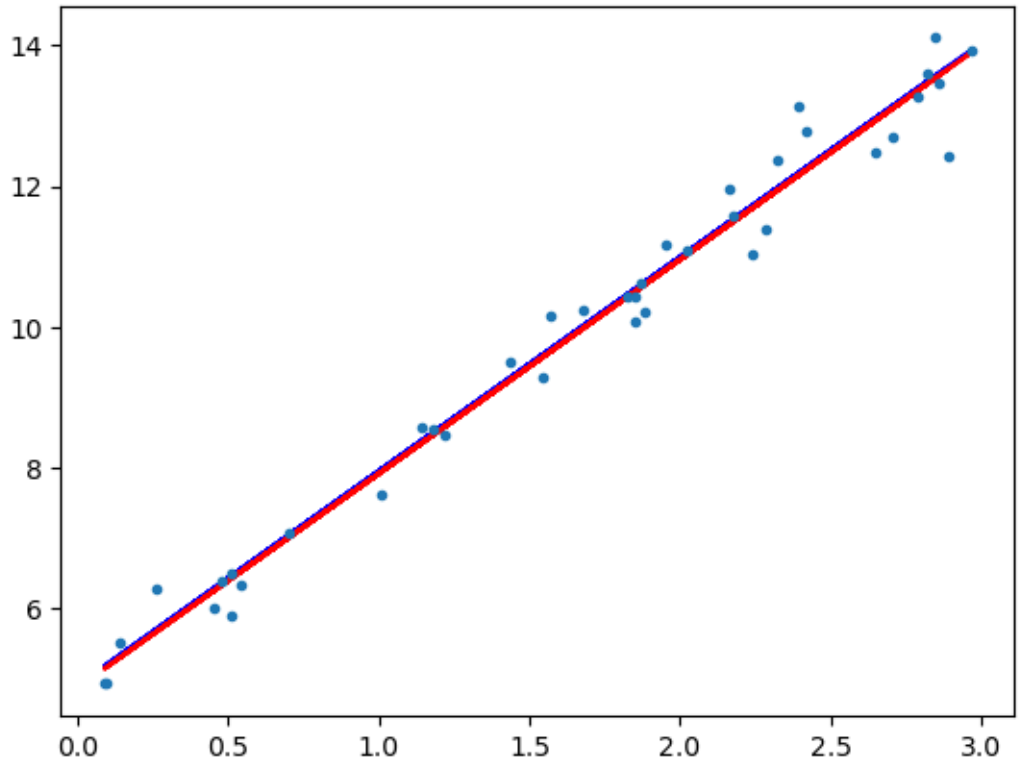
```
# 方法②
# 将数据集分成k个mini_batch
def func_2(omega_init, b_init, k, learning_rate, threshold):
    def get_mini_batches(train, k):
        np.random.shuffle(train)
        mini_batches = [train[i:i + k] for i in range(0, len(train), k)]
        return mini_batches

    mini_batches = get_mini_batches(train, k)

    def gradient_descent(omega, b, mini_batches):
        # omega的更新公式
        def omega_gd(omega, b, mini_batch):
            total = 0
            for x, y in mini_batch:
                total += x * (omega * x + b - y)
            return omega - learning_rate * total / len(mini_batch)

        # b的更新公式
        def b_gd(omega, b, mini_batch):
            total = 0
            for x, y in mini_batch:
                total += omega * x + b - y
            return b - learning_rate * total / len(mini_batch)

        while True:
            random_index = np.random.randint(0, len(mini_batches))
            mini_batch = mini_batches[random_index]
            omega_new, b_new = omega_gd(omega, b, mini_batch), b_gd(omega, b,
mini_batch)
            # 终止条件为参数更新的幅度小于阈值threshold
            if abs(omega_new - omega) < threshold and abs(b_new - b) <
threshold:
                break
            omega, b = omega_new, b_new
        return omega, b

    return gradient_descent(omega_init, b_init, mini_batches)
```

3）使用求解出来的线性回归模型对测试数据集‘test.csv’进行预测，输出可视化结果（使用 matplotlib 来画出测试数据的散点图以及训练好的模型函数图像）

```
test_x = test[:, 0]
test_y = test[:, 1]

train_y_1 = omega_1 * test_x + b_1
train_y_2 = omega_2 * test_x + b_2

plt.plot(test_x, train_y_1, 'b')
plt.plot(test_x, train_y_2, 'r')
plt.plot(test_x, test_y, '.')
plt.show()
```



阈值 = 0.001



阈值 = 0.0001

蓝色直线为方法①的图像，红色直线为方法②的图像。可见，当阈值设置为 0.001 时，两种方法的拟合效果相差较大；当阈值设置为 0.0001 时，两种方法的拟合效果近乎一致。

4）

方法②：

1.初始化模型参数$w0$，$w1$，$w2$，$w3$ 分别为 1，1，1，1，取 mini_batches 的个数 k 为 10，学习率η为 0.01；

2.在负梯度的方向上更新参数（小批量随机梯度下降），并不断迭代这一步骤。由于$x^{(i)}0=1$，因此，每次迭代时，在 mini_batch 第 0 位插入 1，以便后续遍历；

```
# omega的更新公式
    def omega_gd(omega, mini_batch):
        res = omega.copy()
        mini_batch_with_x0 = np.insert(mini_batch, 0, 1, axis=1)
        total = 0
        for i in range(len(res)):
            for row in mini_batch_with_x0:
                xj = row[i]
                total += xj * (omega[0] + omega[1] * row[1] + omega[2] *
row[2] + omega[3] * row[3] - row[4])
            res[i] -= learning_rate * total / len(mini_batch)
        return res
```

3.终止条件为参数更新的幅度小于阈值 0.001（经测试，当阈值大于 0.001 时，线性回归效果与方法①差距较大；当阈值小于 0.001 时，几分钟内较难出结果）。

完整代码：

```
train_frame = pd.read_csv('train2.csv')
test_frame = pd.read_csv('test2.csv')

train = np.array(train_frame)
test = np.array(test_frame)

# 将数据集分成k个mini_batch
def get_mini_batches(train, k):
    np.random.shuffle(train)
    mini_batches = [train[i:i + k] for i in range(0, len(train), k)]
    return mini_batches

k = 10
mini_batches = get_mini_batches(train, k)

# 学习率
learning_rate = 0.01
# 阈值 >= 0.001
threshold = 0.001

def gradient_descent(omega, mini_batches):
    # omega的更新公式
    def omega_gd(omega, mini_batch):
        res = omega.copy()
        mini_batch_with_x0 = np.insert(mini_batch, 0, 1, axis=1)
        total = 0
        for i in range(len(res)):
            for row in mini_batch_with_x0:
                xj = row[i]
                total += xj * (omega[0] + omega[1] * row[1] + omega[2] *
row[2] + omega[3] * row[3] - row[4])
            res[i] -= learning_rate * total / len(mini_batch)
        return res

    while True:
        random_index = np.random.randint(0, len(mini_batches))
        mini_batch = mini_batches[random_index]
        omega_new = omega_gd(omega, mini_batch)
        flag = True
        # 终止条件为参数更新的幅度小于阈值threshold
        for x, y in zip(omega_new, omega):
            if abs(x - y) >= threshold:
                flag = False
                break
        omega = omega_new
        if flag:
            break
    return omega

#omega任意初始值
omega = [1, 1, 1, 1]
omega = gradient_descent(omega, mini_batches)

test_x = test[:, :3]
train_y = [omega[0] + omega[1] * x1 + omega[2] * x2 + omega[3] * x3 for x1,
x2, x3 in test_x]
test_y = test[:, 3]
MSE = np.sum((test_y - train_y) ** 2) / len(test_y)
print(MSE)
```

三、 收获与反思

泥最好食