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：

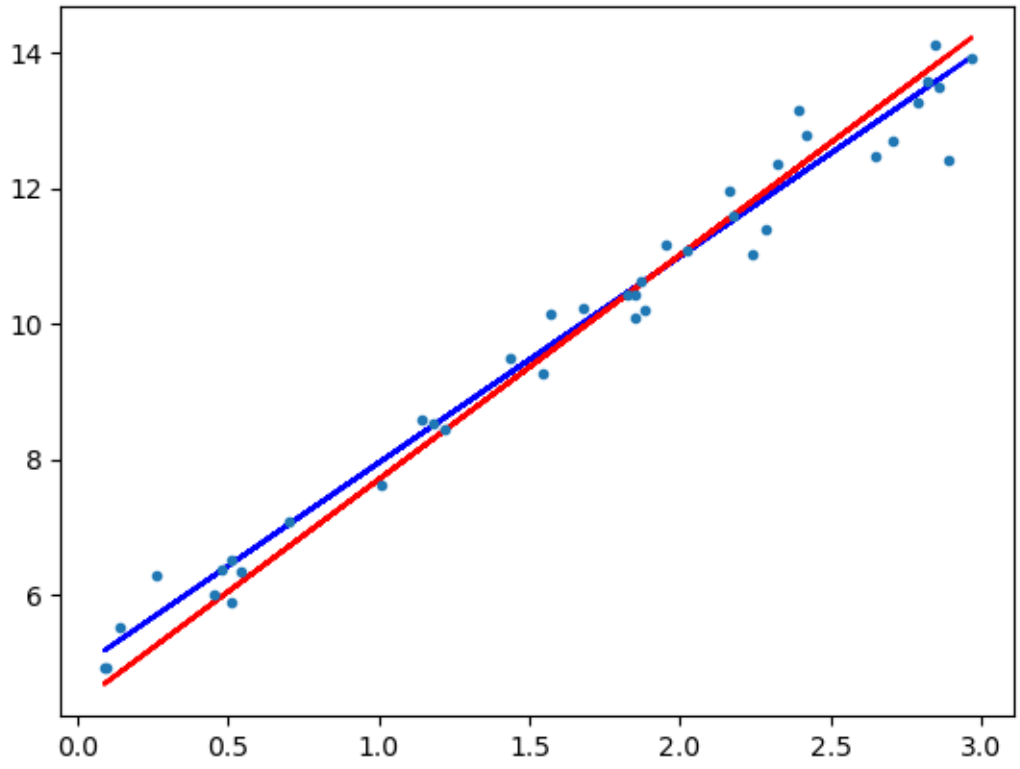
```
# a b的交换操作是同时进行的，而不是顺序执行的
a, b = b, a
# 因此，利用这一特性，可以对w和b同时进行迭代
omega_new, b_new = omega_gd(omega, b, mini_batch), b_gd(omega, b, mini_batch)
```

完整代码：

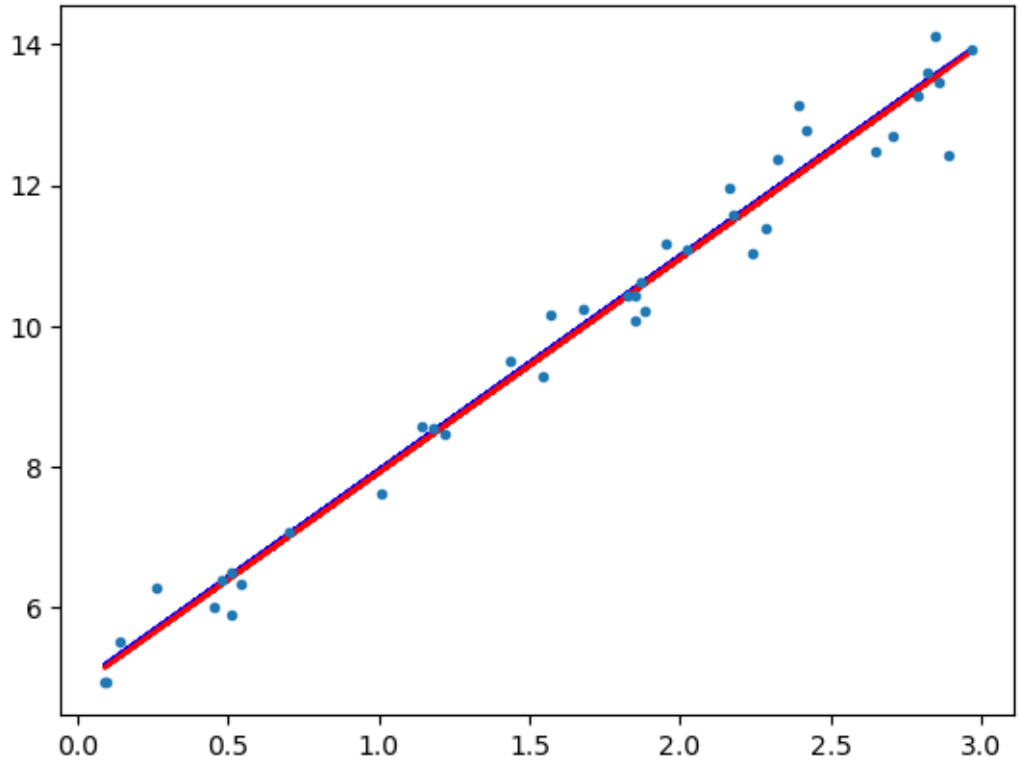
```
# 方法②
# 将数据集分成k个mini_batch
def func_2(omega_init, b_init, k, learning_rate, threshold):
    def get_mini_batches(train, k):
        np.random.shuffle(train)
        mini_batches = [train[i:i + k] for i in range(0, len(train), k)]
        return mini_batches

    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
func_1: omega = 3.041479, b = 4.906074
func_2: omega = 3.310257, b = 4.390848



阈值 = 0.0001
func_1: omega = 3.041479, b = 4.906074
func_2: omega = 3.088664, b = 4.833292

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有几率近乎一致。

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$x_0^{(i)}=1$，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('omega = ' + str(omega))
print('MSE = ' + str(MSE))
```

测试五次的值如下：

1.

omega = [4.534367497686386, 1.234169940701653, 2.1367413987277795, 3.1641026727554222]

MSE = 0.37982282114001703

2.

omega = [4.358014382069729, 1.2506683824643505, 2.182486948351078, 3.2092488336384255]

MSE = 0.42107080009595405

3.

omega = [4.6436542403980186, 1.2046814770383203, 2.1270493721036927, 3.1252138983980466]

MSE = 0.37491851905672463

4.

omega = [4.391598015071767, 1.2638894785374448, 2.1710119849025635, 3.2055348474643433]

MSE = 0.4189184199800239

5.

omega = [4.483974672173968, 1.2308467502840124, 2.17258322878662, 3.20945679025689]

MSE = 0.383779830817784

观察到一个规律，$w0 > w3 > w2 > w1$，推断 x3、x2、x1 的贡献依次减小。

三、 收获与反思

1.理解线性关系：线性回归实验使我们更加深入地理解自变量和因变量之间的线性关系。通过观察特征与目标变量之间的变化趋势，我们可以获得对数据的洞察，并确定哪些特征对目标变量的影响较大或较小。

2. 模型评估与选择：在线性回归实验中，我们需要选择合适的模型评估指标来评估模型的性能。常见的指标如均方误差（Mean Squared Error）。通过对模型评估指标的分析，我们可以了解模型的优势和不足，并选择最适合问题的模型。

3. 特征选择与工程：线性回归实验也可以帮助我们进行特征选择和特征工程。通过观察特征的系数（即权重），我们可以判断哪些特征对目标变量的预测贡献较大或较小。

4. 结果解释与推断：线性回归实验可以帮助我们解释和推断模型的结果。通过分析模型的系数和统计显著性，我们可以了解不同自变量对因变量的影响程度，并进行推断和解释。

总之，线性回归实验使我们能够熟悉和理解线性模型，掌握常见的机器学习和数据挖掘概念，以及数据预处理和模型评估等技术。它为我们在更复杂的问题上应用更高级的模型和技术打下了坚实的基础。