山东大学

信息科学与工程学院

2022－2023 学年第一学期

实 验 报 告

课程名称：信息基础 II

专 业 班 级 崇新学堂

学 生 学 号

学 生 姓 名

课 程 报 告 线性回归

1. 实验目的

使用二维高斯分布，随机生成 30 个点，食用 Linear regresssion 找到近似的线性函数，使用 close—form 和 GD 两种方法。

2. 实验要求

不能直接使用 ML 库或开源代码

3. 实验原理

3.1 相关矩阵

$$Y=\begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix} \qquad X=\begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_M^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_1^{(N)} \end{bmatrix} \qquad \theta=\begin{bmatrix} \theta_0 \\ \vdots \\ \theta_M \end{bmatrix}$$

其中 N 为样本数量，M 为特征维度，θ 为系数矩阵，在本次实验中 N=30，M=1.。

3.2 close-form

针对 close—form 求取最优解，通过数学分析获得其最优解表达式，条件是 N>>M,在本次实验中 30>>1,满足条件，如（1）.

$$\hat{\theta}=(X^TX)^{-1}(X^TY) \tag{1}$$

通过 python 的 numpy 库便可以完成矩阵的相关运算，求取到最佳的系数。

3.3 梯度下降法（GD）

在线性回归中，需要优化的目标函数如公式（2）(3)所示。

$$J(\theta)=\frac{1}{N}\sum_{i=1}^{N}J_i(\theta) \tag{2}$$

$$J_i(\theta)=\frac{1}{2}(y^{(i)}-\theta^Tx^{(i)})^2 \tag{3}$$

利用梯度下降法求解时，需要求取梯度，而对应的梯度求法如公式（4）。

$$\nabla J_i(\theta)=-(y^{(i)}-\theta^Tx^{(i)})x^{(i)} \tag{4}$$

之后选取步长，对系数矩阵做出调整，从而实现迭代，直至找到梯度近似为 0 的点。

$$\theta=\theta-r\frac{1}{N}\sum_{i=1}^{N}\nabla J_i(\theta) \tag{5}$$

之后利用 python 的 numpy 库来进行矩阵的相关操作，即可完成。

4. 实验步骤

4.1 close—form

```
def close_formed(X,Y):
    XT=X.T
    A=np.dot(np.linalg.inv(np.dot(XT,X)),np.dot(XT,Y))
    return A
```

代码部分如上图，利用 python 的矩阵库即可实现。

4.2 梯度下降法

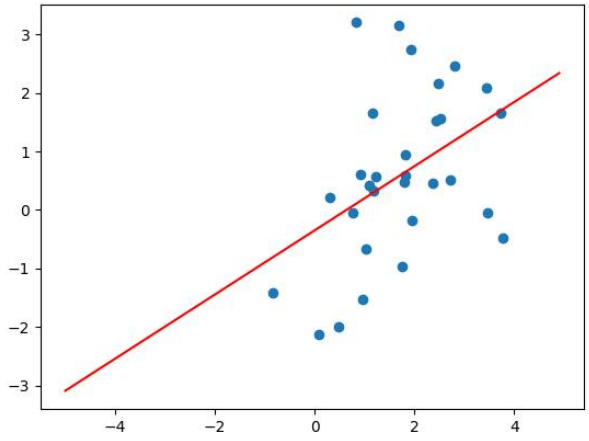
```
def GD(X,Y):
    loop_max = 10000                                    #定义最大迭代次数
    sum1=0                                              #用于存储误差
    X_A = np.array(zeros((len(X), len(X[0]))))
    X_B = np.array(zeros((2, 1)))
    epsilon = 1e-20                                     #定义最小误差
    alpha = 0.01                                        #初始步长
    A = np.array([[float(0)], [0]])                     #初始点为[0, 0]
    N = len(X)                                          #样本个数
    for Number in range(loop_max):
        if Number>5000:
            alpha=0.001                                 #如果迭代次数大于5000，步长减小为0.001
        coefficient = np.dot(X,A)

        for i in range(len(Y)):
            coefficient[i] = coefficient[i]-Y[i]        #此矩阵存储的为对应样本梯度中的系数部分
        for i in range(N):
            for j in range(2):
                X_A[i][j] = X[i][j]*coefficient[i][0]   #X_A矩阵用于存储各个样本的梯度
        for i in range(N):
            for j in range(2):
                X_B[j][0] +=X_A[i][j]                   #X_B矩阵用于存储将各个样本梯度求和的结果
        X_B[0][0] /= N
        X_B[1][0] /= N
        for i in range(2):
            A[i][0] =A[i][0]- X_B[i][0]*alpha           #进行迭代
        for i in range(len(X_B)):
            sum1 += ((X_B[i][0])**2)**0.5
        if sum1 < epsilon:                              #梯度足够小时，终止迭代过程
            break
    return A
```

代码如上图所示，其中最大迭代次数设置为 10000，初始步长为 0.01（迭代 5000 次之后步长为 0.001），X_A 用于存储各个样本的梯度，而 X_B 矩阵用于存储各个样本梯度的平均值，A 为系数矩阵。调用及绘图如下：

```
mean = np.array([2,1])
cov = np.array([[1,0.5],[0.5,2]])
a=np.random.multivariate_normal(mean,cov,30)
print(a)
x1 = []
y1 = []
x=np.array(ones((30,2)))
y=np.array(ones((30,1)))
x2 = np.arange(-5,5,0.1)
for i in range(30):
    x[i][1] = a[i][0]
    x1.append(a[i][0])
    y[i][0] = a[i][1]
    y1.append(a[i][1])
plt.scatter(x1,y1)
A=GD(x,y)
B=close_formed(x,y)
y2 = A[0][0]+A[1][0]*x2
plt.plot(x2,y2,color='red')
print(A)
print(B)
plt.show()
```

有关于二维高斯随机数的产生使用 np.multivariate_normal()函数实现，其中第一个参数为一个 1x2 的矩阵，代表均值；而第二个参数为协方差矩阵；第三个参数是需要产生的随机点的个数。之后利用 matplotlip.pyplot 中的相关函数进行绘图，绘图结果如下：




```
[[-0.15781378]
 [ 0.51283695]]
[[-0.15786846]
 [ 0.51286173]]
```

观察以上打印结果可以发现，close—form 方法和 GD 梯度下降方法给出的参数相差是十分细微的，因此可以证明编程基本是正确的，因此在绘图中只绘出一条即可。