机器学习实验报告

题　　目：	线性规划问题
学生姓名：	张芮熙
学　　号：	22354188
指导教师：	马倩
专业班级：	22 智科 3 班

2021 年 5 月

本科生院制

目录

1.实验目的与思路.....

2. 核心代码及实现.....

3。心得体会.....

1. 实验目的、思路：

一元线性模型

（1）方法一：

将损失函数求和表示，再求导，令导函数=0 的 w 值即为所求，再通过 y(average)=w*x(average)+b 求得 b

2) 假设模型为一元线性回归模型$\hat{y} = wx + b$, 损失函数为$l(w, b) = \frac{1}{2}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2$, 其中$\hat{y}^{(i)}$表示第$i$个样本的预测值，$y^{(i)}$表示第$i$个样本的实际标签值, m为训练集中样本的个数。求出使得损失函数最小化的参数w和b。

方法①

将$l(w, b)$分别对w和b求导，得到

$$\frac{\partial l(w,b)}{\partial w} = w\sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m}(y_i - b)x_i,$$

$$\frac{\partial l(w,b)}{\partial b} = mb - \sum_{i=1}^{m}(y_i - wx_i),$$

令上述两式为零即可得到w和b的解析解：

$$w = \frac{\sum_{i=1}^{m} y_i(x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}(\sum_{i=1}^{m} x_i)^2},$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i),$$

其中$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$为$x$的均值。

核心代码实现与测试结果

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
Dataframe=pd.read_csv('train.csv')
Dataframe2=pd.read_csv('test.csv')
num=np.array(Dataframe)
num2=np.array(Dataframe2)
sumx=np.sum(num,axis=0)
print(sumx)
xx=np.sum(np.square(num),axis=0)
print("sum x*x=",xx[0])
print("sum y*y=",xx[1])
x=np.array(num[:,0])
y=np.array(num[:,1])
xy=np.dot(x,y)
print("sum x*y=",xy)
aver1=sumx/len(num)
averx=aver1[0]
avery=aver1[1]
print("text中x平均值为：",averx)
print("text中y平均值为：",avery)
w=(xy-len(num)*averx*avery)/(xx[0]-len(num)*averx*averx)
print("w=",w)
b=(avery-w*averx)
print("b=",b)
sumx2=np.sum(num2,axis=0)
print(sumx2)
xx2=np.sum(np.square(num2),axis=0)
print("sum x*x=",xx2[0])
print("sum y*y=",xx2[1])
x2=np.array(num2[:,0])
y2=np.array(num2[:,1])
xy2=np.dot(x2,y2)
print("sum x*y=",xy2)
aver2=sumx2/len(num2)
averx2=aver2[0]
avery2=aver2[1]
print("text中x平均值为：",averx2)
print("text中y平均值为：",avery2)
ww=(xy2-len(num2)*averx2*avery2)/(xx2[0]-len(num2)*averx2*averx2)
print("w2=",ww)
b2=(avery-w*averx)
print("b2=",b2)
```

```
[ 235.88607423 1502.41429599]
sum x*x= 469.39225429976216
sum y*y= 15259.981600162002
sum x*y= 2584.921078578418
text中x平均值为： 1.47428796396144
text中y平均值为： 9.390089349933202
w= 3.041478870014282
b= 4.906073659228105
[ 66.33789924 396.38929122]
sum x*x= 142.09439000790425
sum y*y= 4229.59631458883
sum x*y= 754.6240693905429
text中x平均值为： 1.65844748111139
text中y平均值为： 9.909732280567301
w2= 3.0312953255435477
b2= 4.906073659228105
```

解得：W=3.041　　b=4.906

　　W2=3.031　　b1=4.906

心得体会

解此题主要要用到取矩阵的列函数，再把它们相乘求和求得所需数据，train，test 两组数据集得出的结果在误差范围之内。

（2）方法二：

梯度下降法

利用迭代的方法，将初始化的 w,b 沿着与梯度相反的方向不断迭代，到接近真实值为止。

方法② 梯度下降法。手动实现梯度下降法(不使用机器学习框架，如PyTorch、TensorFlow等)来进行模型的训练。算法步骤如下：1.初始化模型参数w和b的值；2.在负梯度的方向上更新参数(批量梯度下降、小批量随机梯度下降或者随机梯度下降均可)，并不断迭代这一步骤，更新公式(以小批量随机梯度下降为例)可以写成：

$$w \leftarrow w - \frac{\eta}{|B|}\sum_{i\in B} x^{(i)}(wx^{(i)} + b - y^{(i)}),$$

和

$$b \leftarrow b - \frac{\eta}{|B|}\sum_{i\in B} (wx^{(i)} + b - y^{(i)}),$$

其中η表示学习率,B表示每次迭代中随机抽样的小批量，$|B|$则表示B中的样本数量。3. 终止条件为迭代次数达到某一上限或者参数更新的幅度小于某个阈值。

核心代码实现与测试结果

```
print("真实值w:",ww)
print("真实值b:",b2)
w=5
b=5
print("初始值w=",w)
print("初始值b=",b)
yita=5
for i in range(1000):
    w=w-(yita/len(num))*x[i%len(num)]*(w*x[i%len(num)]+b-y[i%len(num)])
    b=b-(yita/len(num))*(w*x[i%len(num)]+b-y[i%len(num)])

print("修正后w=",w)
print("修正后b=",b)
```

```
真实值w: 3.0312953255435477
真实值b: 4.906073659228105
初始值w= 5
初始值b= 5
修正后w= 3.0511799753821327
修正后b= 4.89480860616553
```

心得体会：

解这道题有两个重要方面，一是执行迭代的次数，二是学习率，如果迭代次数不够，会造成较大误差，如果学习率过小，导致收敛缓慢，过大的学习率一开始收敛速度很快，但会导致无法收敛到最小值，超高的学习率甚至可能直接大幅跨过了最小值而没有收敛效果。 所以选取一个合适的学习率，对于找到全局最小值以及提高模型训练速度都是很有帮助的。

多元线性模型：

方法一

矩阵表示法

将矩阵的最后一列换成 1，用矩阵运算求解矩阵

方法③

用矩阵表示，假设数据集有m个样本，特征有n维。$X=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} & 1 \\ x_{21} & x_{22} & \cdots & x_{2n} & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} & 1 \end{bmatrix}$，实际标签$Y=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$，参数

$B=\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}$，则解析解为$B^*=(X^TX)^{-1}X^TY$。推导过程可参考这篇文章。

代码以及运行结果：

```
# Your code here
#使用函数 np.c_ np.ones np.linalg.inv
Dataframe3=pd.read_csv('train2.csv')
Dataframe4=pd.read_csv('test2.csv')
#读取数据集
#使用函数 pd.read_csv
num3=np.array(Dataframe3)
num4=np.array(Dataframe4)
#转化成numpy矩阵
#使用函数 np.array
y1=np.array(num3[:,3])
Y= np.array(y1).reshape(len(num3),1)
one = np.ones((len(num3),1),dtype=np.int64)
num3[:,3].fill(1)
num3T=(num3.T)
a1=np.dot(num3T,num3)
a2=np.linalg.inv(a1)
a3=np.dot(a2,num3T)
B1=np.dot(a3,Y)
print("B=")
print(B1)
```

```
B=
[[1.00723001]
 [2.00339371]
 [3.01025624]
 [5.94254393]]
```

心得体会：

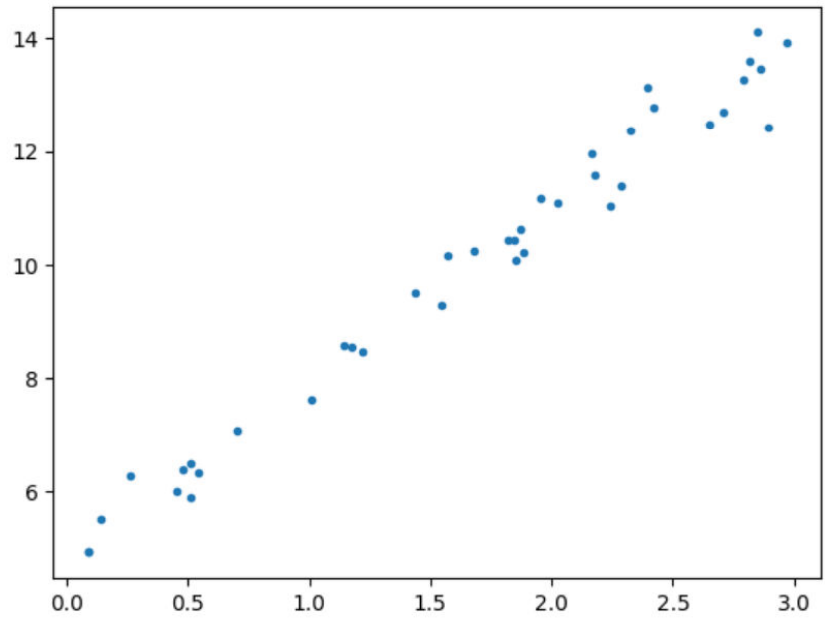
需要熟练掌握矩阵的取行，列，矩阵的转置，取逆矩阵操作，这里同时运用了 replace 将第一列的 y 换成 1，比原来的方法更简单。

绘制图像：

确定点的坐标位置

代码及结果：

```
# Your code here
#A=np.array([[1,2,3],[4,5,6],[7,8,9]])
#x = A[0, :] #从一个矩阵中提取出一行作为一个向量
#y1 = np.array([2, 3, 5])
#plt.plot(x, y1) #画出折线图
#y2 = np.array([2.5, 2.8, 5.3])
#plt.plot(x, y2, '.') #画出散点图
#plt.show()
x=num2[:,0]
y=num2[:,1]
plt.plot(x,y,'.')
plt.show()
```



心得体会

没有 MATLAB 好用

方法二：

加权更新参数，也是通过迭代的方法。

运行代码及结果：

方法① 同2)中的方法③。

方法② 类似2)中的方法②。算法步骤如下：1.初始化模型参数w_0, w_1, w_2, w_3的值；2.在负梯度的方向上更新参数(批量梯度下降、小批量随机梯度下降或者随机梯度下降均可)，并不断迭代这一步骤，更新公式(以小批量随机梯度下降为例)可以写成：

$$w_j \leftarrow w_j - \frac{\eta}{|B|}\sum_{i\in B} x_j^{(i)}(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_3 x_3^{(i)} - y^{(i)}), j = 0, 1, 2, 3,$$

其中$x_0^{(i)} = 1$， 其中η表示学习率,B表示每次迭代中随机抽样的小批量，$|B|$则表示B中的样本数量。3. 终止条件为迭代次数达到某一上限或者参数更新的幅度小于某个阈值。

```
# Your code here
w0,w1,w2,w3=6,3,2,1
print("初始值w=",w0,w1,w2,w3)
yita2=0.1
x1=num4[:,0]
x2=num4[:,1]
x3=num4[:,2]
y=num4[:,3]
for i in range(1000):
    w0=w0-(yita2/len(num4))*(w0+w1*x1[i%len(num4)]+w2*x2[i%len(num4)]+w3*x3[i%len(num4)]-y[i%len(num4)])
    w1=w1-(yita2/len(num4))*x1[i%len(num4)]*(w0+w1*x1[i%len(num4)]+w2*x2[i%len(num4)]+w3*x3[i%len(num4)]-y[i%len(num4)])
    w2=w2-(yita2/len(num4))*x2[i%len(num4)]*(w0+w1*x1[i%len(num4)]+w2*x2[i%len(num4)]+w3*x3[i%len(num4)]-y[i%len(num4)])
    w3=w3-(yita2/len(num4))*x3[i%len(num4)]*(w0+w1*x1[i%len(num4)]+w2*x2[i%len(num4)]+w3*x3[i%len(num4)]-y[i%len(num4)])
print("修正后w=",w0,w1,w2,w3)
```

```
初始值w= 6 3 2 1
修正后w= 5.963873472857915 1.0109148959526608 2.0867392965931115 2.9628803111671207
```

心得体会

和一维模型的方法相似，学习率的设置不要太大，多迭代几次，误差在范围内。注意的一点就是下角标要确保正确，得到的输出是一个矩阵，代表了 w0-w3 的值。