实验一 线性回归

一、实验目的

1. 掌握线性回归的基本原理
2. 编程实现简单的线性回归
3. 能够用线性回归解决实际问题

二、实验步骤

1. 生成数据集
2. 初始化参数值
3. 梯度下降优化参数
4. 得出结果并分析

三、实验内容

1. 运行、调试两个源代码
2. 分析线性回归的基本步骤
3. 分析运行结果中输出参数的含义
4. 编程解决实际问题

注意：所有图名均为“班级-姓名-学号”的形式。

四、实验原理

对于线性模型，给定样本，其中 $x^{(n)}$ 为样本的第 n 种特征。线性模型的形式为：$y_i = f(\vec{x}_i) = \vec{w} \cdot \vec{x}_i + b$。其中，$\vec{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(n)})^T$，$\vec{w} = (w^{(1)}, w^{(2)}, \ldots, w^{(n)})^T$ 为每个特征对应的权重生成的权重向量。

线性模型的物理表示：二维空间就是一条直线，三维空间就是一个平面，推广到n维空间。常见的线性模型有：岭回归、lasso回归、逻辑斯蒂回归以及Fisher等。逻辑斯蒂回归分类器构建的过程正是通过训练数据来获得一系列的权重向量。在此处则是拓展为：根据训练数据集来计算参数 w 和 b。

对于具有 N 个训练样本的样本集，每个给定的样本为 $\vec{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(n)})^T$（注意，这里是 N 个训练样本，每个训练样本有 n 个特征。而样本集中，训练样本的标签为 yi，而预测值为：$\hat{y}_i = f(\vec{x}_i) = \vec{w} \cdot \vec{x}_i + b$。则模型的损失函数为：

$$L(f) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{N} (\vec{w} \cdot \vec{x}_i + b - y_i)^2$$

通过使损失函数最小化，即可以获得要求的参数 $\vec{w}$ 和 b；通过梯度下降或者梯度上升算法（最优化算法）可以求解。

本实验针对满秩矩阵情况，用下式求解参数 w 和 b。

$$\hat{\boldsymbol{w}}^* = \arg\min_{\hat{\boldsymbol{w}}} (\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})^{\mathrm{T}} (\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})$$

其中：

$\hat{w} = (w_1; w_2 \cdots w_d; b)$

$\boldsymbol{y} = (y_1; y_2; \ldots; y_m)$

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} & 1 \\ \boldsymbol{x}_2^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_m^{\mathrm{T}} & 1 \end{pmatrix}$$

若 X'X 是满秩矩阵或正定矩阵，则

$$\hat{\boldsymbol{w}}^* = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

1、构造数据

引入所需要的函数库，令 w=3，b=4， 再加上一些高斯随机噪声生成 100 个数据点，并画出图像。

任务 1:

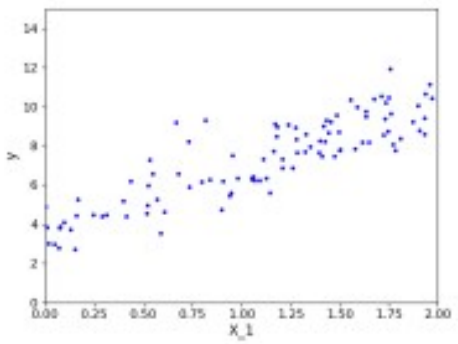
（1）读懂源代码，知道数据样本是如何生成的

（2）改变 w 和 b 的值，观察样本有何不同

（3）自己构造一组样本数据，并打印

```
# 通过 rand 函数可以返回一个或一组服从“0~1”均匀分布的随机样本值。随机样本取值范围是[0,1)，不包括 1
X = 2 * np.random.rand(100, 1)# 构造线性方程，加入随机抖动
# numpy.random.randn()是从标准正态分布中返回一个或多个样本值
# 1.当函数括号内没有参数时，返回一个浮点数；
# 2.当函数括号内有一个参数时，返回秩为 1 的数组，不能表示向量和矩阵
# 3.当函数括号内有两个及以上参数时，返回对应维度的数组，能表示向量或矩阵。
np.random.randn(行,列)
# 4.np.random.standard_normal()函数与 np.random.randn 类似，但是输入参数为元组（tuple）
y = 3*X + 4 + np.random.randn(100, 1)
plt.plot(X, y, 'b.')     # b 指定为蓝色，.指定线条格式
plt.xlabel('X_1')
plt.ylabel('y')
# 设置 x 轴为 0-2，y 轴为 0-15
plt.axis([0, 2, 0, 15])
plt.show()
```

执行显示数据点如下所示：



2、线性方程实现回归

任务 2:

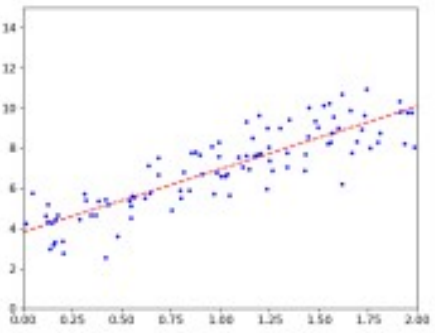
（1）解释 X_b 和 theta_best 的含义

（2）解释 # 和 '''的作用

（3）说明回归线是如何画出的

（4）自己构造一组样本数据，画出回归曲线，给出回归参数

```
# numpy.c_:按行连接两个矩阵，就是把两矩阵左右相加，要求行数相等。
# numpy.r_:按列连接两个矩阵，就是把两矩阵上下相加，要求列数相等。
#ones()返回一个全 1 的 n 维数组，同样也有三个参数：shape（用来指定返回数组的大小）、dtype（数组元素的类型）、order（是否以内存中的 C 或 Fortran 连续（行或列）顺序存储多维数据）。后两个参数都是可选的，一般只需设定第一个参数。
X_b = np.c_[(np.ones((100, 1)), X)]
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
# np.linalg.inv:矩阵求逆
print(theta_best)
''''# 输出结果
[[3.99844262]     #b 值
 [3.09461187]]    #w 值
'''
X_new = np.array([[0], [2]])# 测试数据
X_new_b = np.c_[np.ones((2, 1)), X_new]# 构成新矩阵
y_predict = X_new_b.dot(theta_best)# 预测结果
print(y_predict)

plt.plot(X_new, y_predict, 'r--o')   # 指定红色和线条
plt.plot(X, y, 'b.')     # 指定蓝色和点
plt.axis([0, 2, 0, 15])
plt.show()
```



3、阅读以下实例，完成任务 3。

在统计学中，线性回归是利用称为线性回归方程的最小二乘函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。这种函数是一个或多个称为回归系数的模型参数的线性组合。一个带有一个自变量的线性回归方程代表一条直线。我们需要对线性回归结果进行统计分析。

例如，假设我们已知一些学生年纪和游戏时间的数据，可以建立一个回归方程，输入一个新的年纪时，预测该学生的游戏时间。自变量为学生年纪，因变量为游戏时间。当只有一个因变量时，我们称该类问题为简单线性回归。线性回归的目的就是通过训练数据，求取出估计参数建立的直线方程：$\hat{y}=b_1+b_0x$。编程关键在于如何求取 b0 和 b1 的值，我们引入一个方程（sum of square）：

$$\min\sum(y_i-\hat{y})^2$$

当上述方程的值最小时，认为求取到线程回归方程参数的值，对该方程求最小值可以进一步转化为求导和求极值的问题，求导过程省略，最后结论如下：

$$b_0=\frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sum(x_i-\bar{x})^2}$$

$$b_1=\bar{y}-b_0\bar{x}$$

其中，$\bar{x},\bar{y}$ 为均值。

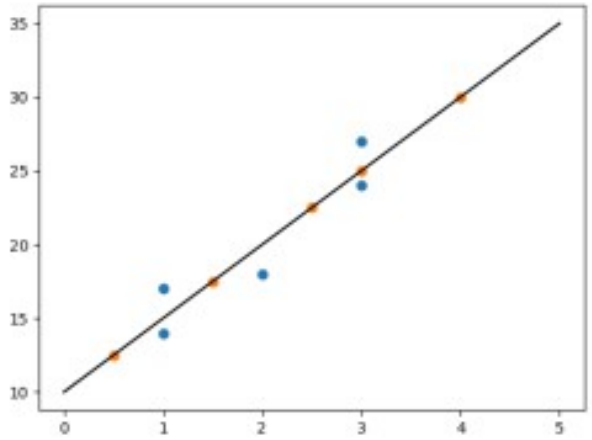
以下面实例为例，第一列表示每月投放广告的次数，第二列表示汽车向里，通过 Python 编程求取线性回归方程：

投放广告数	汽车销量
1	14
3	24
2	18
1	17
3	27

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

# 定义训练数据
x = np.array([1, 3, 2, 1, 3])
y = np.array([14, 24, 18, 17, 27])

# 回归方程求取函数
def fit(x, y):
    if len(x) != len(y):
        return
    numerator = 0.0
    denominator = 0.0
    x_mean = np.mean(x)
    y_mean = np.mean(y)
    for i in range(len(x)):
        numerator += (x[i]-x_mean)*(y[i]-y_mean)
        denominator += np.square((x[i]-x_mean))
    print('numerator:', numerator,'denominator:', denominator)
```

```
    b0 = numerator/denominator
    b1 = y_mean - b0*x_mean
    return b0,b1

# 定义预测函数
def predit(x,b0,b1):
    return b0*x + b1

# 求取回归方程
b0,b1 = fit(x,y)
print('b0=',b0,'b1=',b1)
print('Line is:y = %2.0fx + %2.0f'%(b0,b1))

# 预测
x_test = np.array([0.5,1.5,2.5,3,4])
y_test = np.zeros((1,len(x_test)))
for i in range(len(x_test)):
    y_test[0][i] = predit(x_test[i],b0,b1)

# 绘制图像
xx = np.linspace(0, 5)
yy = b0*xx + b1
plt.plot(xx,yy,'k-')
plt.scatter(x,y,cmap=plt.cm.Paired)
plt.scatter(x_test,y_test[0],cmap=plt.cm.Paired)
plt.show()
```

结果展示：



任务 3：基本任务——实现简单的线性回归

【要求】

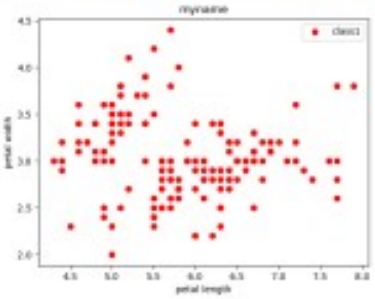
（1）解释以上程序；

（2）参考上面的程序，给定一组数据(1, 1），(2, 3），(3, 2），(4, 3），(5, 5)。通过线性回归求出拟合的直线，求参数，并根据拟合的直线计算当 x=2.5 和 4.5 时的预测值。

4、鸢尾花数据集，完成任务 4

任务 4：进阶任务——实现给定数据的线性回归

【要求】Iris 鸢尾花数据集是一个经典数据集，在统计学习和机器学习领域都经常被用作示例。数据集内包含 3 类共 150 条记录，每类各 50 个数据，每条记录都有 4 项特征：花萼长度、花萼宽度、花瓣长度、花瓣宽度，可以通过这 4 个特征预测鸢尾花卉属于（iris-setosa, iris-versicolour, iris-virginica）中的哪一品种。



费雪鸢尾花卉数据集

花萼长度	花萼宽度	花瓣长度	花瓣宽度	属种
5.1	3.5	1.4	0.2	*setosa*
4.9	3.0	1.4	0.2	*setosa*
4.7	3.2	1.3	0.2	*setosa*
4.6	3.1	1.5	0.2	*setosa*
5.0	3.6	1.4	0.2	*setosa*
5.4	3.9	1.7	0.4	*setosa*
4.6	3.4	1.4	0.3	*setosa*
5.0	3.4	1.5	0.2	*setosa*

现要求**电子 1 班将鸢尾花数据集前 50 条记录**中的前两列(花萼长度、花萼宽度)做为二维数据集,进行线性拟合。做出回归曲线，并给出回归公式。

电子 2 班将鸢尾花数据集前 50 条记录中的后两列(花瓣长度、花瓣宽度)做为二维数据集,进行线性拟合。做出回归曲线，并给出回归公式。

电子 3 班将鸢尾花数据集后 50 条记录中的前两列(花萼长度、花萼宽度)做为二维数据集,进行线性拟合。做出回归曲线，并给出回归公式。

电子 4 班将鸢尾花数据集后 50 条记录中的后两列(花瓣长度、花瓣宽度)做为二维数据集,进行线性拟合。做出回归曲线，并给出回归公式。