# 实验2 线性回归

## 一、实验目的

1. 理解线性回归的基本原理，包括简单线性回归（一元）和多元线性回归的数学模型。
2. 掌握线性回归模型的构建、训练、评估方法。
3. 学会使用Python及相关库（numpy、pandas、matplotlib、scikit-learn）实现线性回归模型。
4. 能够分析实验结果，解释模型参数（系数、截距）的意义及评估指标（如MSE、R²）的含义。

## 二、实验原理

### 2.1 线性回归模型定义

线性回归是一种用于预测连续型因变量与自变量之间线性关系的监督学习方法。其核心思想是通过拟合一条直线（简单线性回归）或超平面（多元线性回归），最小化预测值与真实值之间的误差。

#### 2.1.1 简单线性回归（一元线性回归）

当自变量只有1个时，称为简单线性回归。模型公式为：

y=wx+b+ϵ

* y：因变量（预测目标）；
* x：自变量（特征）；
* w：权重（系数），表示x对y的影响程度；
* b：偏置（截距），表示当x=0时y的基准值；
* ϵ：随机误差项，服从均值为0的正态分布。

#### 2.1.2 多元线性回归

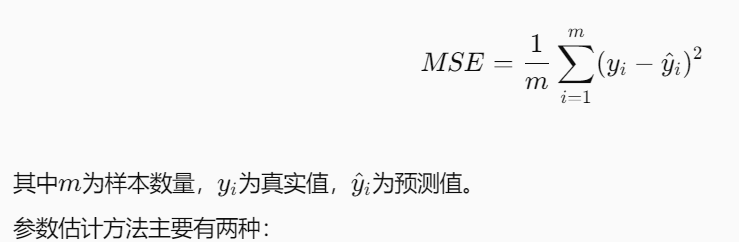
当自变量有多个时，称为多元线性回归。模型公式为：

y=w1x1+w2x2+...+wnxn+b+ϵ

* x1 ,x2 ,...,xn ：多个自变量；
* w1,w2,...,wn：对应自变量的系数；
* b：截距项。

### 2.2 模型优化目标

线性回归的目标是找到最优参数www和bbb，使得预测值y^\hat{y}y^ 与真实值yyy的误差最小。常用**损失函数**（Loss Function）为**均方误差（MSE）**：



* **最小二乘法**：通过求导直接计算使MSE最小的参数（适用于简单问题）。
* **梯度下降**：通过迭代优化参数（适用于大规模数据或复杂模型）。

### 2.3 模型评估指标

* **均方误差（MSE）**：衡量预测值与真实值的平均平方误差，值越小越好。
* **决定系数（R2）**：表示模型解释因变量变异的能力，取值范围为[0,1]，越接近1说明拟合效果越好。

## 二、实验环境

* Python 3.7+
* 所需库：numpy（数据处理）、pandas（数据读取）、matplotlib（可视化）、scikit-learn（模型构建与评估）
* 安装命令： bash

pip install numpy pandas matplotlib scikit-learn

## 三、实验步骤

### 步骤1：数据准备

本次实验使用**模拟数据**（避免外部数据依赖，便于复现），分为“简单线性回归”和“多元线性回归”两个案例。

### 步骤2：简单线性回归（一元）

#### 2.1 数据生成与可视化

生成满足y=2x+3+ϵ的样本（其中ϵ为随机噪声，模拟真实数据的波动）：

import numpy as np

import matplotlib.pyplot as plt

# 生成特征x（100个样本，范围[0,10]）

x = np.linspace(0, 10, 100).reshape(-1, 1) # reshape为二维数组（sklearn要求特征为二维）

# 生成真实标签y（加入噪声）

np.random.seed(42) # 固定随机种子，保证结果可复现

y\_true = 2 \* x + 3 + np.random.normal(0, 1, size=x.shape) # 噪声均值0，标准差1

# 可视化数据分布

plt.scatter(x, y\_true, color='blue', label='样本数据')

plt.xlabel('x (特征)')

plt.ylabel('y (标签)')

plt.title('简单线性回归数据分布')

plt.legend()

plt.show()

#### 2.2 模型构建与训练

使用scikit-learn的LinearRegression类构建模型，并拟合数据：

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

# 划分训练集（80%）和测试集（20%）

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y\_true, test\_size=0.2, random\_state=42)

# 初始化模型

model\_simple = LinearRegression()

# 训练模型（拟合数据）

model\_simple.fit(x\_train, y\_train)

# 输出训练得到的参数

print(f"简单线性回归参数：")

print(f"系数 w: {model\_simple.coef\_[0]:.4f}") # 真实w=2

print(f"截距 b: {model\_simple.intercept\_[0]:.4f}") # 真实b=3

#### 2.3 模型评估

在测试集上计算MSE和R2：

python

from sklearn.metrics import mean\_squared\_error, r2\_score

# 预测测试集

y\_pred\_simple = model\_simple.predict(x\_test)

# 计算评估指标

mse\_simple = mean\_squared\_error(y\_test, y\_pred\_simple)

r2\_simple = r2\_score(y\_test, y\_pred\_simple)

print(f"\n简单线性回归评估指标：")

print(f"测试集 MSE: {mse\_simple:.4f}")

print(f"测试集 R²: {r2\_simple:.4f}")

#### 2.4 结果可视化

绘制测试集数据点与回归线：

plt.scatter(x\_test, y\_test, color='blue', label='测试数据')

plt.plot(x\_test, y\_pred\_simple, color='red', linewidth=2, label='回归线')

plt.xlabel('x (特征)')

plt.ylabel('y (标签)')

plt.title('简单线性回归拟合结果')

plt.legend()

plt.show()

### 步骤3：多元线性回归（扩展）

#### 3.1 数据生成

生成3个特征的多元线性回归数据，真实模型为

：

python

# 生成3个特征（100个样本）

np.random.seed(42)

x\_multi = np.random.randn(100, 3) # 3个特征，服从标准正态分布

# 真实参数

w\_true = [1.5, 0.8, -2.3]

b\_true = 5

# 生成标签（加入噪声）

y\_multi\_true = x\_multi @ w\_true + b\_true + np.random.normal(0, 0.5, size=100) # @表示矩阵乘法

#### 3.2 模型构建、训练与评估

python

# 划分训练集和测试集

x\_train\_multi, x\_test\_multi, y\_train\_multi, y\_test\_multi = train\_test\_split(

x\_multi, y\_multi\_true, test\_size=0.2, random\_state=42

)

# 初始化并训练多元线性回归模型

model\_multi = LinearRegression()

model\_multi.fit(x\_train\_multi, y\_train\_multi)

# 输出参数

print(f"\n多元线性回归参数：")

print(f"系数 w: {model\_multi.coef\_}（真实值：{w\_true}）")

print(f"截距 b: {model\_multi.intercept\_:.4f}（真实值：{b\_true}）")

# 评估

y\_pred\_multi = model\_multi.predict(x\_test\_multi)

mse\_multi = mean\_squared\_error(y\_test\_multi, y\_pred\_multi)

r2\_multi = r2\_score(y\_test\_multi, y\_pred\_multi)

print(f"\n多元线性回归评估指标：")

print(f"测试集 MSE: {mse\_multi:.4f}")

print(f"测试集 R²: {r2\_multi:.4f}")

## 四、代码实现完整版本

# 线性回归实验完整代码

# ---------------------- 简单线性回归 ----------------------

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

# 1. 生成简单线性回归数据

x = np.linspace(0, 10, 100).reshape(-1, 1) # 特征x（二维数组）

np.random.seed(42)

y\_true = 2 \* x + 3 + np.random.normal(0, 1, size=x.shape) # 真实模型：y=2x+3+噪声

# 2. 数据可视化

plt.scatter(x, y\_true, color='blue', label='样本数据')

plt.xlabel('x (特征)')

plt.ylabel('y (标签)')

plt.title('简单线性回归数据分布')

plt.legend()

plt.show()

# 3. 划分数据集

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y\_true, test\_size=0.2, random\_state=42)

# 4. 模型训练

model\_simple = LinearRegression()

model\_simple.fit(x\_train, y\_train)

# 5. 输出参数

print("===== 简单线性回归结果 =====")

print(f"系数 w: {model\_simple.coef\_[0]:.4f}（真实值：2）")

print(f"截距 b: {model\_simple.intercept\_[0]:.4f}（真实值：3）")

# 6. 模型评估

y\_pred\_simple = model\_simple.predict(x\_test).reshape(-1, 1) # 预测测试集

mse\_simple = mean\_squared\_error(y\_test, y\_pred\_simple)

r2\_simple = r2\_score(y\_test, y\_pred\_simple)

print(f"测试集 MSE: {mse\_simple:.4f}")

print(f"测试集 R²: {r2\_simple:.4f}")

# 7. 结果可视化

plt.scatter(x\_test, y\_test, color='blue', label='测试数据')

plt.plot(x\_test, y\_pred\_simple, color='red', linewidth=2, label='回归线')

plt.xlabel('x (特征)')

plt.ylabel('y (标签)')

plt.title('简单线性回归拟合结果')

plt.legend()

plt.show()

# ---------------------- 多元线性回归 ----------------------

# 1. 生成多元线性回归数据

np.random.seed(42)

x\_multi = np.random.randn(100, 3) # 3个特征（100样本）

w\_true = [1.5, 0.8, -2.3] # 真实系数

b\_true = 5 # 真实截距

y\_multi\_true = x\_multi @ w\_true + b\_true + np.random.normal(0, 0.5, size=100) # 真实模型+噪声

# 2. 划分数据集

x\_train\_multi, x\_test\_multi, y\_train\_multi, y\_test\_multi = train\_test\_split(

x\_multi, y\_multi\_true, test\_size=0.2, random\_state=42

)

# 3. 模型训练

model\_multi = LinearRegression()

model\_multi.fit(x\_train\_multi, y\_train\_multi)

# 4. 输出参数与评估

print("\n===== 多元线性回归结果 =====")

print(f"系数 w: {np.round(model\_multi.coef\_, 4)}（真实值：{w\_true}）")

print(f"截距 b: {model\_multi.intercept\_:.4f}（真实值：{b\_true}）")

y\_pred\_multi = model\_multi.predict(x\_test\_multi)

mse\_multi = mean\_squared\_error(y\_test\_multi, y\_pred\_multi)

r2\_multi = r2\_score(y\_test\_multi, y\_pred\_multi)

print(f"测试集 MSE: {mse\_multi:.4f}")

print(f"测试集 R²: {r2\_multi:.4f}")

## 五、实验要求

1.写出实验完整过程，数据可视化并截图；

1. 分析拟合出来的参数，并与设计值进行对比，说明误差的原因；
2. 提出改进措施并验证。