安徽大学《机器学习》实验报告 1

学号：WA2214014　　专业：人工智能　　姓名：杨跃浙

实验日期：24.11.25　　教师签字：　　成绩：

[**实验名称**] 线性回归实验

[**实验目的**]

1．熟悉和掌握单变量线性回归算法

2．熟悉和掌握批处理梯度下降算法

3．熟悉和掌握多变量线性回归算法

[**实验要求**]

1．采用 Python、Matlab 等高级语言进行编程，推荐优先选用 Python 语言

2．核心模型和算法需自主编程实现，不得直接调用 Scikit-learn、PyTorch 等成熟框架的第三方实现

3．代码可读性强：变量、函数、类等命名可读性强，包含必要的注释

4．提交实验报告要求：1）报告文件：命名为“学号-姓名-Lab”； 2）提交时间：下次实验课上课前

[**实验原理**]

线性回归是一种常用的统计学习方法，用于建立自变量与因变量之间的线性关系模型。本实验的核心是通过最小化目标函数（均方误差）找到最佳的模型参数，使模型能够尽可能准确地预测目标值。

1.线性回归模型

线性回归模型的目标是找到一个线性函数，使得输入变量与输出变量之间的关系可以用以下公式表示：

$$y = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

其中：y 为目标变量；$x_1, x_2, \ldots, x_n$为输入特征；$w_0, w_1, \ldots, w_n$为待学习的模型参数。

对于单变量线性回归，公式可以简化为：

$$y = w_0 + w_1x$$

2.最小平方误差目标函数

在线性回归中，目标是最小化预测值与实际值之间的误差，常用的目标函数为均方误差(MSE)：

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(h(x_i) - y_i)^2$$

其中：$h(x_i) = w_0 + w_1x_1 + \ldots + w_nx_n$是模型的预测值；$y_i$是实际值；$m$ 是样本数量。

目标函数衡量了模型预测值与实际值的差距，越小的值表示模型拟合效果越好。

3.梯度下降法

为了最小化目标函数$J(w)$,使用梯度下降算法更新模型参数。更新公式为：

$$w_j := w_j - \alpha\frac{\partial J(w)}{\partial w_j}$$

其中：α是学习率，控制参数更新的步长；$\frac{\partial J(w)}{\partial w_j}$是目标函数对参数$w_j$的偏导数。

批量梯度下降法(Batch Gradient Descent)在每次迭代时计算所有样本的梯度，使得参数更新稳定，但计算开销较大。

4.特征归一化

多变量线性回归中，不同特征的数值范围可能相差较大，例如“面积”的数值范围可能远大于“房间数量”。未经归一化处理时，数值较大 的特征会主导目标函数，导致模型优化困难。归一化的方法包括：

零均值、单位方差：

$$x_{norm} = \frac{x - \mu}{\sigma}$$

最小—最大缩放：

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

归一化可以加速梯度下降的收敛速度，提升模型稳定性。

5．模型评估

通过损失值（如均方误差）和训练误差曲线，可以评估模型的收敛性和拟合效果。线性回归假设输入特征与目标值之间存在线性关系，对于非线性数据，其效果可能受限。

基于上述原理，本实验实现了单变量和多变量线性回归模型的训练与优化，并通过梯度下降法有效调整模型参数，验证了线性回归的基本理论。

[**实验内容**]

一、单变量线性回归

1．采用数据集“data/regress_data1.csv”进行单变量线性回归实验

2．借助 matplotlib 画出原始数据分布的散点图（x=“人口”，y=”收益”）

3．以最小平方误差为目标函数，构造模型的损失（误差）计算函数：

$$J(w)=\frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right)-y^{(i)}\right)^2$$

其中，$h(x)=w^TX=w_0x_0+w_1x_1+w_2x_2+...+w_nx_n$

4．实现**批量梯度下降算法**（Batch Gradient Decent）用于优化线性回归模型：

$$w_j:=w_j-a\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial w_j}J(w)$$

其中，w_j 为参数向量，m 样本量。

提示：

- 批量梯度下降算法可以参考 第 5 章神经网络的优化算法（P104）
- 线性回归模型的偏置 b 可吸收进参数向量 w，从而采用向量形式（P55）

5．采用上述批量梯度下降法，优化单变量线性回归模型。其中，迭代轮数 epoch 设定为 1000 轮，学习率设定为 0.01，参数初始化为 0。

代码输出：

- 优化结束时的损失值和模型参数
- 将模型拟合的直线和原始数据散点图画到同一张图中

二、多变量线性回归

1．采用数据集“data/regress_data2.csv”进行多变量线性回归实验，通过房子的

大小和房间数量两个变量 回归房子的价格。

2. 对数据进行特征归一化：$z_i = \frac{x_i - \mu}{\sigma}$

3. 采用上述批量梯度下降法，优化多变量线性回归模型。其中，迭代轮数 epoch 设定为 1000 轮，学习率设定为 0.01，参数初始化为 0。

代码输出：

- 优化结束时的损失值和模型参数
- 画图输出训练误差（损失）随着迭代轮数 epoch 的变化曲线

[实验代码和结果]

一、单变量线性回归

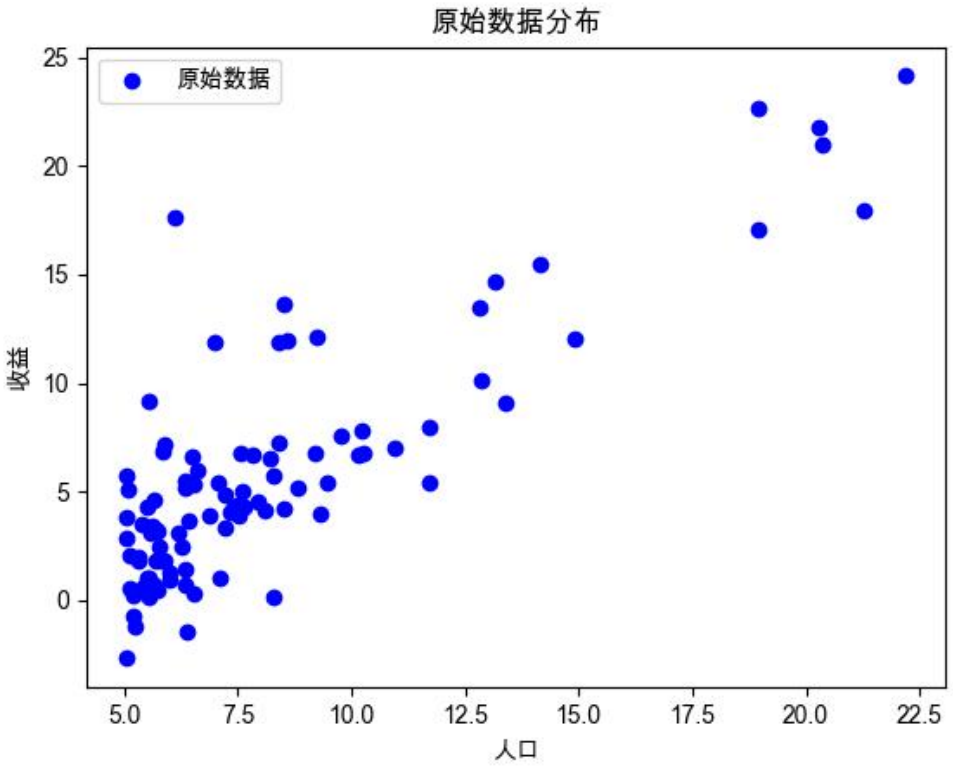
1. 实验代码

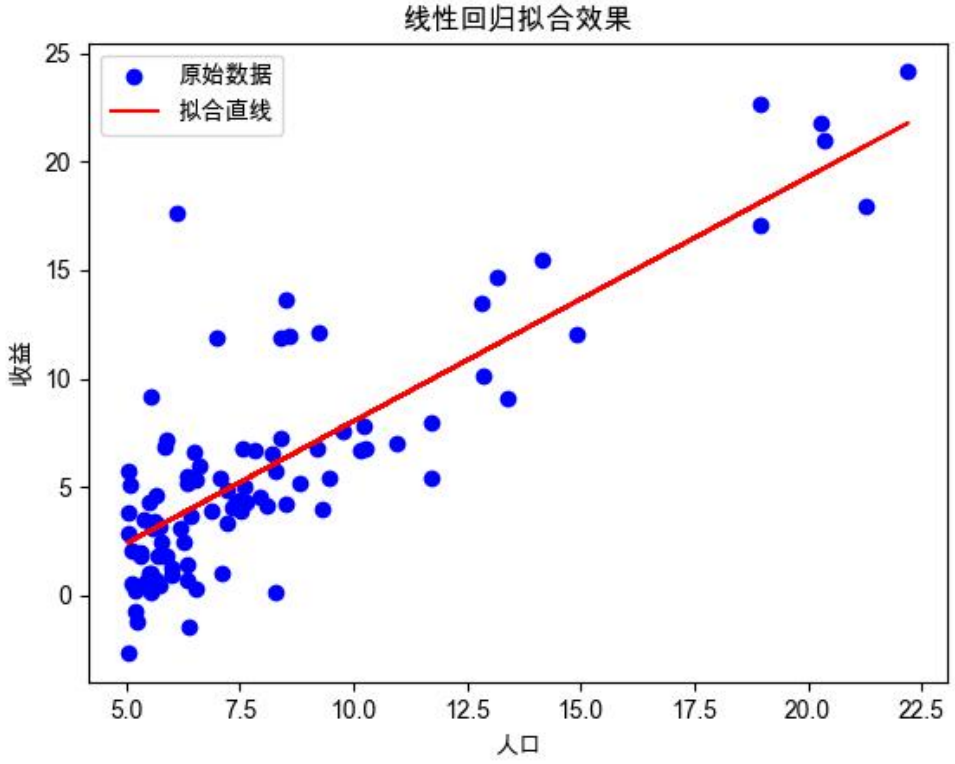
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['font.sans-serif'] = ['Arial Unicode MS']
rcParams['axes.unicode_minus'] = False
# 1. 加载数据
data = pd.read_csv('data/regress_data1.csv')

# 假设数据集中列名为 "人口" 和 "收益"
X = data["人口"].values
y = data["收益"].values

# 数据可视化
plt.scatter(X, y, color='blue', label='原始数据')
plt.xlabel("人口")
plt.ylabel("收益")
plt.title("原始数据分布")
plt.legend()
```

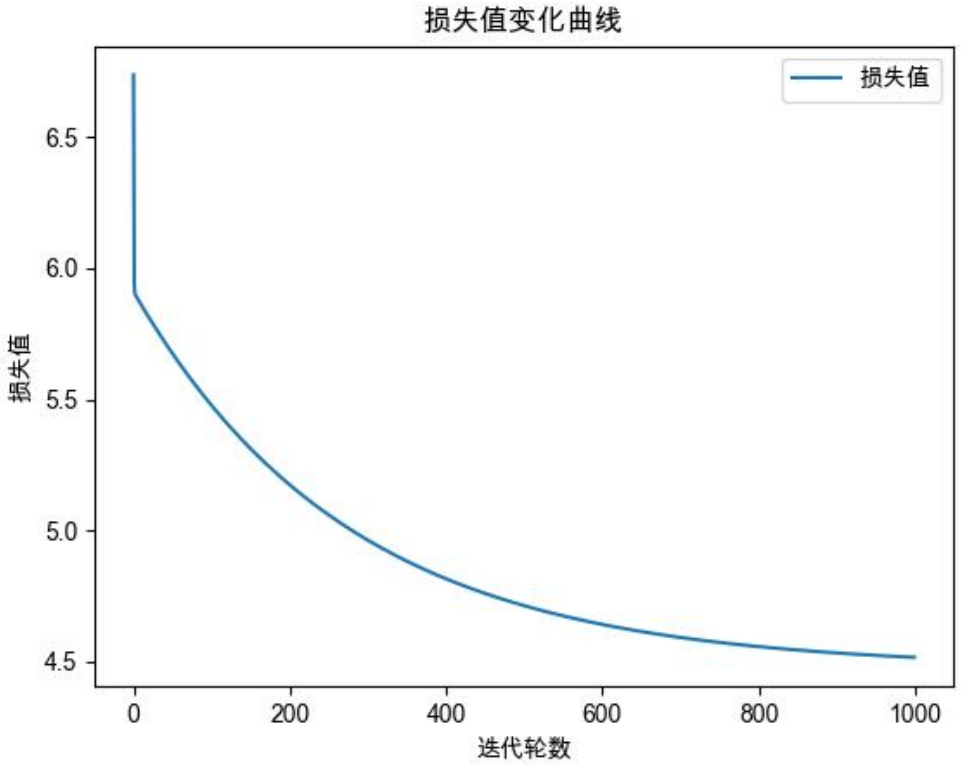
```
plt.show()

# 2. 定义损失函数
def compute_loss(w, X, y):
m = len(y)
X_b = np.c_[np.ones((m, 1)), X] # 添加偏置项
y_pred = X_b.dot(w)
loss = (1 / (2 * m)) * np.sum((y_pred - y) ** 2)
return loss

# 3. 批量梯度下降法
def batch_gradient_descent(X, y, learning_rate, epochs):
m = len(y)
X_b = np.c_[np.ones((m, 1)), X] # 添加偏置项
w = np.zeros(X_b.shape[1]) # 参数初始化为 0
loss_history = []

for epoch in range(epochs):
y_pred = X_b.dot(w)
gradient = (1 / m) * X_b.T.dot(y_pred - y)
w -= learning_rate * gradient
loss = compute_loss(w, X, y)
loss_history.append(loss)
if epoch % 100 == 0:
print(f"Epoch {epoch}, Loss: {loss:.4f}")

return w, loss_history

# 4. 使用梯度下降法优化
learning_rate = 0.01
epochs = 1000

w_opt, loss_history = batch_gradient_descent(X, y,
learning_rate, epochs)

print(f"优化结束时的损失值: {loss_history[-1]:.4f}")
print(f"模型参数: ")

# 5. 绘制拟合直线与原始数据
```

```
plt.scatter(X, y, color='blue', label='原始数据')
plt.plot(X, w_opt[0] + w_opt[1] * X, color='red', label='
拟合直线')
plt.xlabel("人口")
plt.ylabel("收益")
plt.title("线性回归拟合效果")
plt.legend()
plt.show()

# 绘制损失值变化
plt.plot(range(epochs), loss_history, label="损失值")
plt.xlabel("迭代轮数")
plt.ylabel("损失值")
plt.title("损失值变化曲线")
plt.legend()
plt.show()
```

2. 实验结果







Output：

Epoch 0，Loss：6.7372

Epoch 100，Loss：5.4764

Epoch 200，Loss：5.1736

```
Epoch 300, Loss: 4.9626
Epoch 400, Loss: 4.8155
Epoch 500, Loss: 4.7130
Epoch 600, Loss: 4.6415
Epoch 700, Loss: 4.5916
Epoch 800, Loss: 4.5569
Epoch 900, Loss: 4.5327
优化结束时的损失值：4.5160
模型参数：[-3.24140214  1.1272942 ]
```

二、多变量线性回归

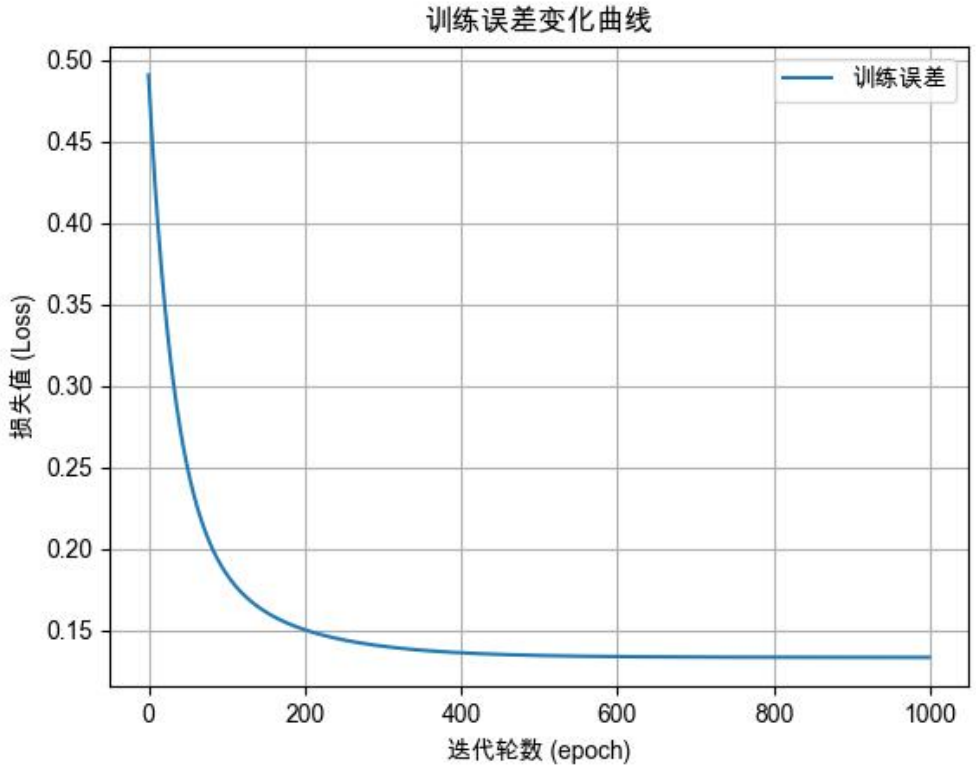
1. 实验代码

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['font.sans-serif'] = ['Arial Unicode MS']
rcParams['axes.unicode_minus'] = False

# 1. 加载数据
data = pd.read_csv('data/regress_data2.csv')
X = data[['面积', '房间数']].values # 特征
y = data['价格'].values # 目标值

# 2. 特征归一化
def feature_normalize(X):
mu = np.mean(X, axis=0) # 每列均值
sigma = np.std(X, axis=0) # 每列标准差
X_norm = (X - mu) / sigma
return X_norm, mu, sigma

X_norm, X_mu, X_sigma = feature_normalize(X)
y_norm, y_mu, y_sigma=feature_normalize(y)

# 添加偏置项
m = len(y_norm)
X_b = np.c_[np.ones((m, 1)), X_norm] # 偏置项
```

```
# 3. 定义损失函数
def compute_loss(w, X, y):
m = len(y)
y_pred = X.dot(w)
loss = (1 / (2 * m)) * np.sum((y_pred - y) ** 2)
return loss

# 4. 批量梯度下降法
def batch_gradient_descent(X, y, learning_rate, epochs):
m = len(y)
w = np.zeros(X.shape[1]) # 初始化参数
loss_history = []

for epoch in range(epochs):
y_pred = X.dot(w)
gradient = (1 / m) * X.T.dot(y_pred - y)
w -= learning_rate * gradient
loss = compute_loss(w, X, y)
loss_history.append(loss)

if epoch % 100 == 0:
print(f"Epoch {epoch}, Loss: {loss:.4f}")

return w, loss_history

# 5. 训练多变量线性回归模型
learning_rate = 0.01
epochs = 1000

w_opt, loss_history = batch_gradient_descent(X_b, y_norm,
learning_rate, epochs)

# 输出最终损失值和模型参数
print(f"优化结束时的损失值: {loss_history[-1]:.4f}")
print(f"模型参数: {w_opt}")

# 6. 绘制训练误差变化曲线
plt.plot(range(epochs), loss_history, label='训练误差')
plt.xlabel("迭代轮数 (epoch)")
```

```
plt.ylabel("损失值 (Loss)")
plt.title("训练误差变化曲线")
plt.legend()
plt.grid(True)
plt.show()
```

2. 实验结果



Output：

Epoch 0，Loss：0.4908
Epoch 100，Loss：0.1845
Epoch 200，Loss：0.1505
Epoch 300，Loss：0.1404
Epoch 400，Loss：0.1363
Epoch 500，Loss：0.1347
Epoch 600，Loss：0.1340
Epoch 700，Loss：0.1337
Epoch 800，Loss：0.1336
Epoch 900，Loss：0.1336
优化结束时的损失值：0.1335
模型参数：[-7.47097419e-17 8.79065699e-01 -4.74786542e-02]

[小结或讨论]

通过本次实验，我深入学习了线性回归的基本原理与实现方法，掌握了批量梯度下降算法在模型优化中的应用。实验分为单变量线性回归和多变量线性回归两个部分。在单变量线性回归中，通过拟合“人口”与“收益”之间的关系，成功训练出模型，并观察到损失值随着迭代次数的增加逐渐收敛，最终的拟合效果较好，验证了线性回归的有效性。

在多变量线性回归实验中，通过对房屋面积和房间数量两个特征预测房价，引入了特征归一化，解决了不同特征量级差异的问题，并对目标值进行了归一化以提升训练的稳定性。实验结果显示，模型参数合理，训练误差曲线平滑下降，证明了模型的收敛性和有效性。此外，房屋面积对价格的影响显著高于房间数量，这与实际情况较为吻合。

虽然实验取得了较好的结果，但线性回归模型的表现可能受限于特征线性假设。在未来，可以通过引入非线性特征或尝试更复杂的回归模型进一步提高预测性能。此外，为验证模型的泛化能力，还需对数据集进行训练集和测试集划分，测试其在新数据上的表现。这次实验让我加深了对线性回归的理解，同时为后续的模型扩展打下了基础。