山东大学 计算机科学与技术 学院

机器学习（双语） 课程实验报告

学号：	姓名：	班级：
实验题目：Experiment 1: Linear Regression		
实验学时：4	实验日期： 2022/9/14	

实验目的：

1. 实现实验指导书中线性回归的相关内容；
2. 学习使用 MATLAB 等工具进行实验；
3. 理解体会线性回归、梯度下降等基本概念。

硬件环境：

Inter（R） Core（TM） i7-8750H

RAM：16.0 GB

软件环境：

Visual Studio Code

版本: 1.67.2 (user setup)

OS: Windows_NT x64 10.0.19044

Python 3.9.7

numpy 1.20.3

matplotlib 3.4.3

实验步骤与内容：

1. 假设函数：

$$h_\theta(x) = \theta^T x = \sum_{j=0}^{n} \theta_j x_j$$

```python
def h(theta, X):
    return np.dot(X, theta)
```

可见此处实现的假设函数与公式中不同，公式中为每一个 theta 与对应位置的 x 相乘，而此处编写的函数是将 X（2 维，n×2）与 theta（1 维，2×1）做点乘，结果为 n×1 的矩阵，n 行中的第 i 行代表第 i 个数据的结果。

因此，这样可以一次计算出所有 x 与 theta 相乘的结果，能够简化计算过程，加快计算速度。

2. 损失函数：

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

```python
def J(theta, X, Y):
    return (1./2*m) * np.dot((h(theta, X)-Y).T, h(theta, X)-Y)
```

此处损失函数的实现使用向量化形式，实现更加简单，运算更快。

3. 梯度下降：$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

```python
def descendGradient(learning_rate, theta, X, Y, iterations):
    theta_record = []    # 记录theta
    J_record = []        # 记录损失
    temp_theta = theta
    for i in range(iterations):
        theta_record.append(temp_theta.tolist())
        theta = temp_theta
        # print(theta_record)
        for j in range(len(theta)):
            temp_theta[j] = theta[j] - (learning_rate/m) * np.sum((h(theta, X)-Y) * X[:, j].reshape(-1, 1))
        J_record.append(J(theta, X, Y).tolist())
    theta_record = np.array(theta_record).reshape(-1, len(theta))
    J_record = np.array(J_record).reshape(-1)
    return theta, theta_record, J_record
```

在进行梯度下降的过程中，记录 theta 及代价 J 的变化情况。

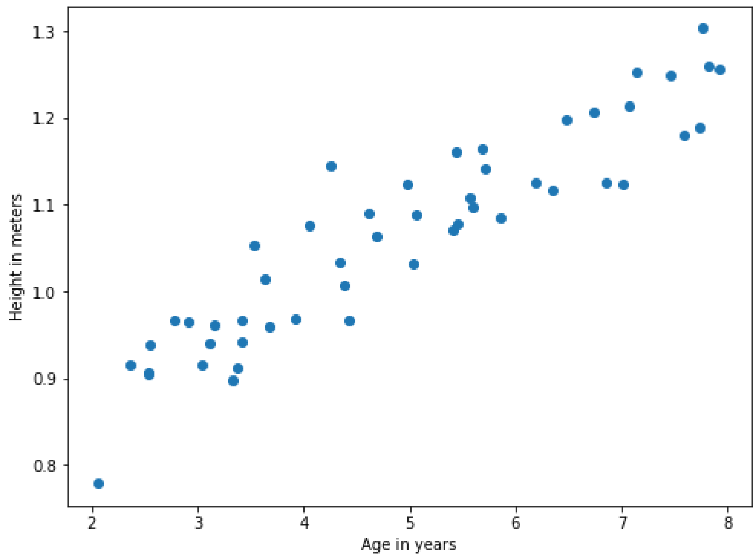
一、单变量线性回归

4. 载入数据

通过 numpy 载入数据：

```python
x = np.loadtxt('data1/ex1_1x.dat')  # 载入数据
y = np.loadtxt('data1/ex1_1y.dat')
```

5. 依据数据绘制散点图：

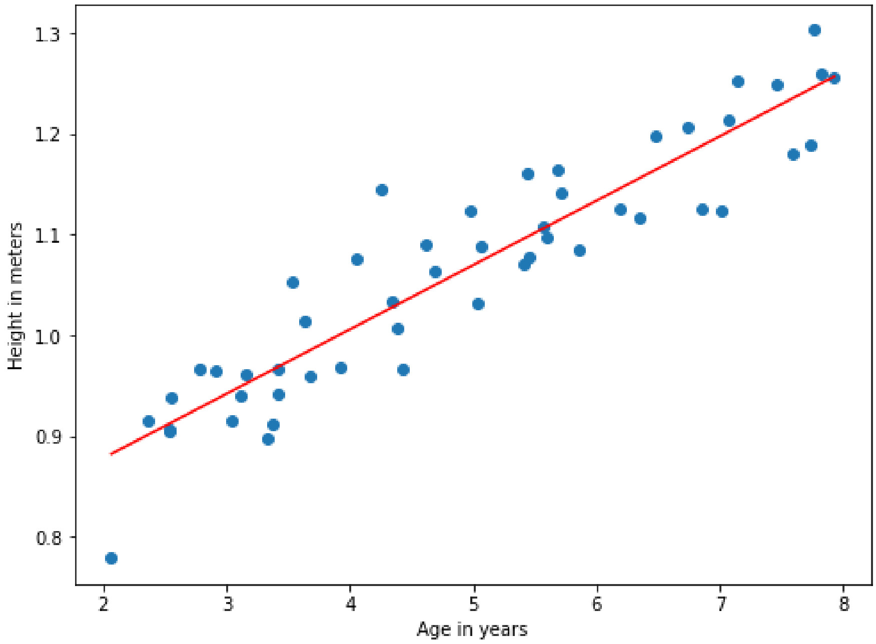


6. 设定学习率 learning_rate 为 0.07，初始 theta 为[0, 0]，迭代次数为 1500，进行训练：

```python
learning_rate = 0.07
theta = np.zeros((x.shape[1], 1))
```

```python
theta, theta_record, J_record = descendGradient(learning_rate, theta, x, y, 1500)
```

7. 依据训练后得到的 theta，在散点图上绘制预测曲线：



8. 对年龄分别为 3.5 和 7 的身高进行预测：

```
a = 3.5
b = 7
pred1 = pred(theta, a)
pred2 = pred(theta, b)
print('input=', a, ', pred=', pred1)
print('input=', b, ', pred=', pred2)
```

得到如下结果：

```
input= 3.5 , pred= [0.97374443]
input= 7 , pred= [1.19733227]
```

二、可视化梯度下降过程：

9. 生成数据

生成网格数据：

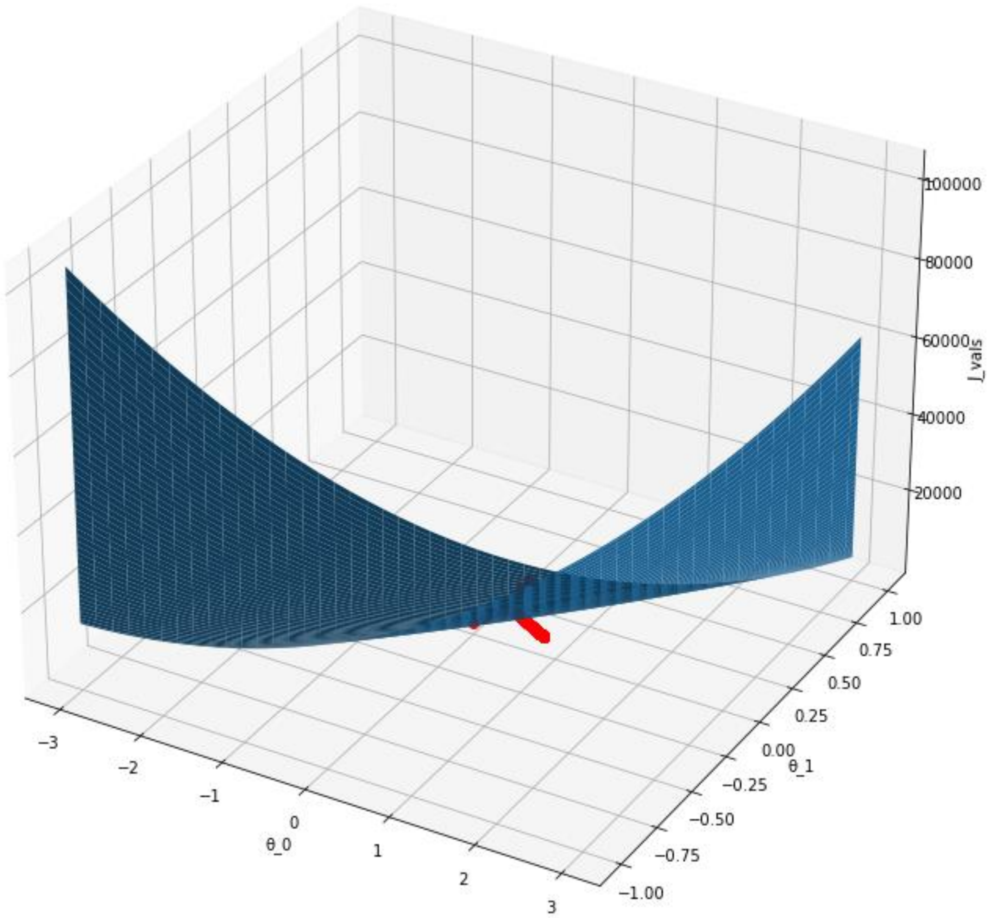
```
step_num = 100
J_vals = np.zeros((step_num, step_num))
theta_vals_0 = np.linspace(-3, 3, step_num)
theta_vals_1 = np.linspace(-1, 1, step_num)
```

通过之前训练得到的 theta 计算网格数据的代价：

```
for i in range(step_num):
    for j in range(step_num):
        J_vals[i][j] = J(np.array((theta_vals_0[i], theta_vals_1[j])).reshape(-1, 1), x, y)
```

10. 可视化代价平面：

```
plt.figure(figsize=(15, 12))
ax3d = plt.axes(projection='3d')
ax3d.plot_surface(theta_vals_0, theta_vals_1, J_vals)
ax3d.set_xlabel('θ_0')
ax3d.set_ylabel('θ_1')
ax3d.set_zlabel('J_vals')
plt.plot(theta_record[:, 0], theta_record[:, 1], J_record, 'ro')
plt.show()
```



其中，红线为训练过程中的梯度下降过程。

三、多变量线性回归

11. 载入数据：

```
x = np.loadtxt('data1/ex1_2x.dat')  # 载入数据
y = np.loadtxt('data1/ex1_2y.dat')
```

12. 计算 x 的标准差与均值：

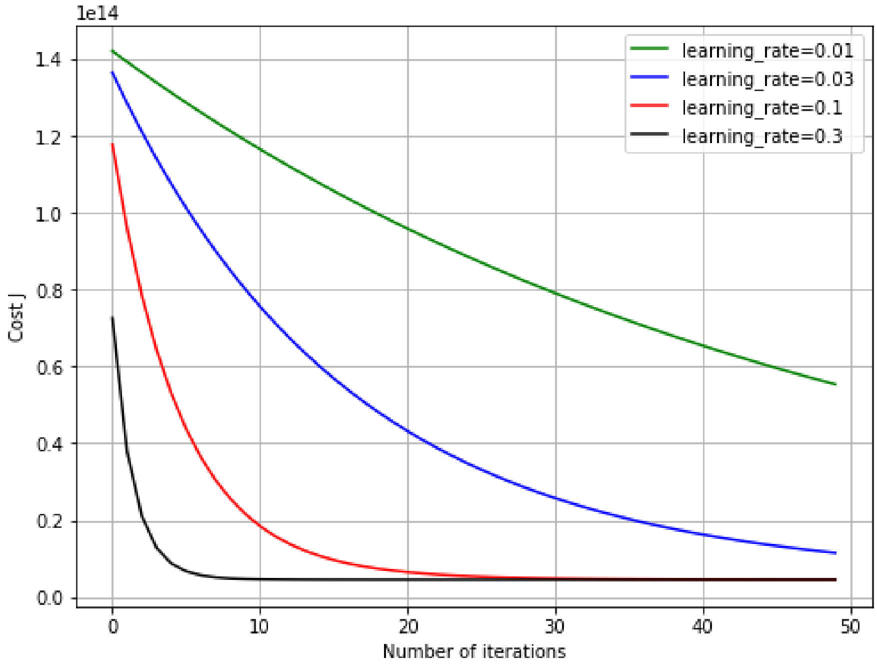
```
sigma = x.std(axis=0)
mu = x.mean(axis=0)
```

```
def normalize(x):
    temp = x.reshape(-1, 2)
    temp[:, 0] = (temp[:, 0] - mu[0]) / sigma[0]
    temp[:, 1] = (temp[:, 1] - mu[1]) / sigma[1]
    return temp
```

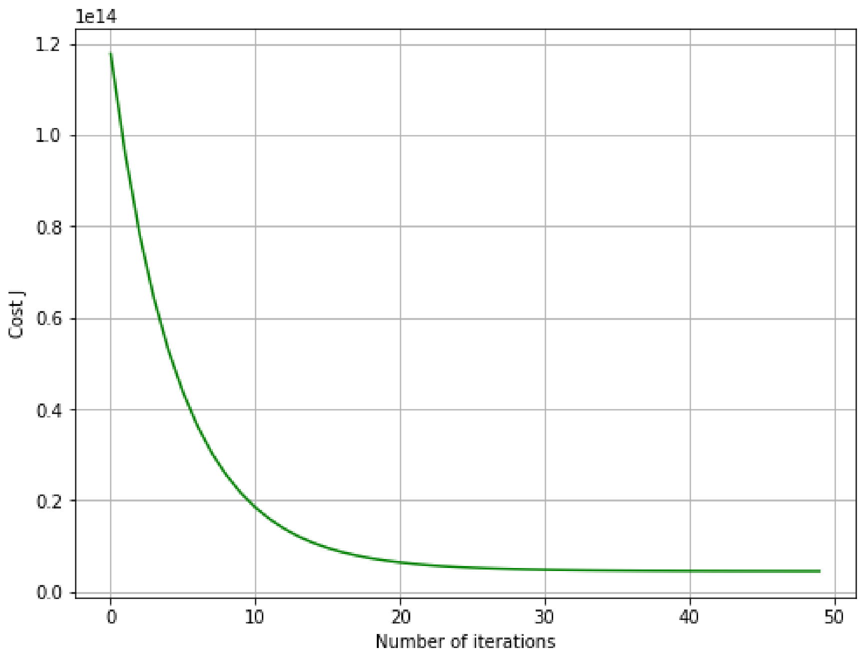
13. 设定迭代次数为 50，初始 theta 为[0,0,0]，学习率 learning_rate 分别为 0.01，0.03，0.1，0.3，记录以上训练过程中的代价变化情况：

```
iteration = 50
_, _, J_record_1 = descendGradient(0.01, np.zeros((x.shape[1], 1)), x, y, iteration)
_, _, J_record_2 = descendGradient(0.03, np.zeros((x.shape[1], 1)), x, y, iteration)
_, _, J_record_3 = descendGradient(0.1, np.zeros((x.shape[1], 1)), x, y, iteration)
_, _, J_record_4 = descendGradient(0.3, np.zeros((x.shape[1], 1)), x, y, iteration)
```

画出图像：



14. 可见，learning_rate 为 0.1 的曲线最符合实验指导书中的情况，即：



15. 对面积为 1650，卧室数量为 3 的房子价格进行预测：

首先对输入数据进行标准化，之后得到预测结果：

```
print('final theta:', theta[0], theta[1], theta[2])
print('input=', 1650, 3, ', pred=', pred(theta, normalize(np.array([1650., 3.]))))
```

```
final theta: [338658.2492493] [103857.9363055] [-1143.58125322]
input= 1650 3 , pred= [292591.61055057]
```

Answer the following questions:

1. Observe the changes in the cost function happens as the learning rate changes. What happens when the learning rate in too small? Too large?
   当学习率过小时，模型收敛速度缓慢，消耗更多时间；
   当学习率过大时，模型不能收敛，难以得到最优解。
2. Using the best learning rate that you found, run gradient descent until convergence to find
   (a) The final values of theta
   由上述第 15 步，可以看到最终的 theta 为：

   [338658.2492493] [103857.9363055] [-1143.58125322]

   (b) The predicted price of a house with 1650 square feet and 3 bedrooms. Don't forget to scale your features when you make this prediction!

   预测结果： pred= [292591.61055057]

结论分析与体会：

1. 在实验前，需要充分理解使用 matlab、python 等工具，才能更好地进行实验，实现实验中的各个步骤。
2. 在实验中，需要理解掌握课上所学知识，结合实验指导书，才能更好地完成实验；
3. 理解线性回归、梯度下降，是学习机器学习的基础，需要认真做好实验，为将来的学习打下基础。

附录：程序源代码

```python
# %%
import numpy as np
import matplotlib.pyplot as plt

# %% [markdown]
# 3. 2D Linear Regression

# %%
x = np.loadtxt('data1/ex1_1x.dat')  # 载入数据
y = np.loadtxt('data1/ex1_1y.dat')

# %%
# 依据数据画出散点图
plt.figure(figsize=(8, 6))
plt.plot(x, y, 'o')
plt.ylabel('Height in meters')
plt.xlabel('Age in years')
plt.show()
```

```python
# %%
# 整理数据
m = y.shape[0]
x = np.hstack((np.ones((m, 1)), x.reshape(-1, 1)))
y = y.reshape(-1, 1)

# %% [markdown]
# 假设函数:
# $$
# h_{\theta}(x)=\theta^Tx=\sum_{j=0}^{n}\theta_jx_j
# $$

# %%
def h(theta, X):
    return np.dot(X, theta)

# %% [markdown]
# 损失函数:
# $$
# J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(h_{\theta}(x^{(i)})-y^{(i)})^2
# $$

# %%
def J(theta, X, Y):
    return (1./2*m) * np.dot((h(theta, X)-Y).T, h(theta, X)-Y)

# %% [markdown]
# 梯度下降:
# $$
# \theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-
y^{(i)})x_j^{(i)}
# $$

# %%
def descendGradient(learning_rate, theta, X, Y, iterations):
    theta_record = []   # 记录theta
    J_record = []       # 记录损失
    temp_theta = theta
    for i in range(iterations):
        theta_record.append(temp_theta.tolist())
        theta = temp_theta
        # print(theta_record)
        for j in range(len(theta)):
```

```python
            temp_theta[j] = theta[j] - (learning_rate/m) *
np.sum((h(theta, X)-Y) * X[:, j].reshape(-1, 1))
        J_record.append(J(theta, X, Y).tolist())
    theta_record = np.array(theta_record).reshape(-1, len(theta))
    J_record = np.array(J_record).reshape(-1)
    return theta, theta_record, J_record

# %%
learning_rate = 0.07
theta = np.zeros((x.shape[1], 1))

# %%
theta, theta_record, J_record = descendGradient(learning_rate, theta, x,
y, 1500)

# %%
# 依据数据画出散点图
plt.figure(figsize=(8, 6))
plt.plot(x[:, 1], y, 'o')
plt.plot(x[:, 1], h(theta, x), 'r')
plt.ylabel('Height in meters')
plt.xlabel('Age in years')
plt.show()

# %%
def pred(theta, x):
    x = np.array(x).reshape(-1)
    x = np.hstack(([1], x)).reshape(-1, 1)
    y = np.dot(theta.T, x)
    return y.reshape(-1)

# %%
# 预测身高
a = 3.5
b = 7
pred1 = pred(theta, a)
pred2 = pred(theta, b)
print('input=', a, ', pred=', pred1)
print('input=', b, ', pred=', pred2)

# %% [markdown]
# 4. Understanding $J(\theta)$

# %%
```

```python
step_num = 100
J_vals = np.zeros((step_num, step_num))
theta_vals_0 = np.linspace(-3, 3, step_num)
theta_vals_1 = np.linspace(-1, 1, step_num)

# %%
for i in range(step_num):
    for j in range(step_num):
        J_vals[i][j] = J(np.array((theta_vals_0[i],
theta_vals_1[j])).reshape(-1, 1), x, y)

# %%
plt.figure(figsize=(15, 12))
ax3d = plt.axes(projection='3d')
ax3d.plot_surface(theta_vals_0, theta_vals_1, J_vals)
ax3d.set_xlabel('θ_0')
ax3d.set_ylabel('θ_1')
ax3d.set_zlabel('J_vals')
plt.plot(theta_record[:, 0], theta_record[:, 1], J_record, 'ro')
plt.show()

# %% [markdown]
# 5. Multivariate Linear Regression

# %%
x = np.loadtxt('data1/ex1_2x.dat')  # 载入数据
y = np.loadtxt('data1/ex1_2y.dat')

# %%
sigma = x.std(axis=0)
mu = x.mean(axis=0)

# %%
def normalize(x):
    temp = x.reshape(-1, 2)
    temp[:, 0] = (temp[:, 0] - mu[0]) / sigma[0]
    temp[:, 1] = (temp[:, 1] - mu[1]) / sigma[1]
    return temp

# %%
x = normalize(x)

# %%
# 整理数据
```

```python
m = y.shape[0]
x = np.hstack((np.ones((m, 1)), x.reshape(-1, 2)))
y = y.reshape(-1, 1)

# %%
iteration = 50
_, _, J_record_1 = descendGradient(0.01, np.zeros((x.shape[1], 1)), x, y, iteration)
_, _, J_record_2 = descendGradient(0.03, np.zeros((x.shape[1], 1)), x, y, iteration)
_, _, J_record_3 = descendGradient(0.1, np.zeros((x.shape[1], 1)), x, y, iteration)
_, _, J_record_4 = descendGradient(0.3, np.zeros((x.shape[1], 1)), x, y, iteration)

# %%
plt.figure(figsize=(8, 6))
plt.plot(range(iteration), J_record_1, 'g-', label='learning_rate=0.01')
plt.plot(range(iteration), J_record_2, 'b-', label='learning_rate=0.03')
plt.plot(range(iteration), J_record_3, 'r-', label='learning_rate=0.1')
plt.plot(range(iteration), J_record_4, 'k-', label='learning_rate=0.3')
plt.ylabel('Cost J')
plt.xlabel('Number of iterations')
plt.grid()
plt.legend()
plt.show()

# %%
learning_rate = 0.1
theta = np.zeros((x.shape[1], 1))

# %%
theta, theta_record, J_record = descendGradient(learning_rate, theta, x, y, iteration)

# %%
plt.figure(figsize=(8, 6))
plt.plot(range(iteration), J_record, 'g-')
plt.ylabel('Cost J')
plt.xlabel('Number of iterations')
plt.grid()
plt.show()

# %%
```

```
print('final theta:', theta[0], theta[1], theta[2])
print('input=', 1650, 3, ', pred=', pred(theta,
normalize(np.array([1650., 3.]))))
```