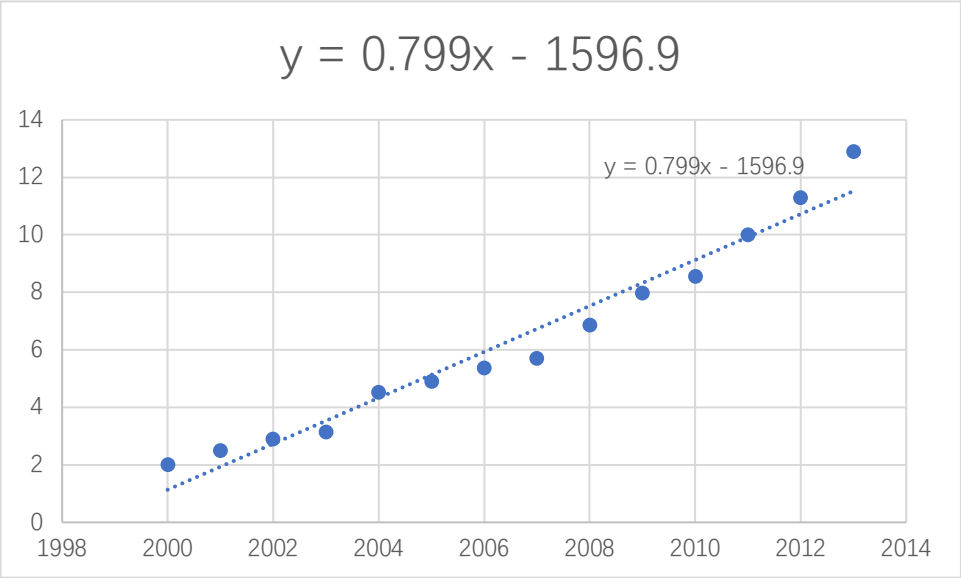
9181040G0818 黄海浪 机器学习第二次作业

1.使用最小二乘法得到



预测 2014 年房价为 12.286

2.使用梯度下降法求解

归一化（快速收敛）：

$$x = \frac{\mathrm{x\text{-}min(x)}}{\mathrm{max(x)\text{-}min(x)}}$$

Cost 函数（方差）：

$$\mathrm{cost} = \frac{1}{2m}\sum_{i=1}^{m}(\mathrm{h(x_i)\text{-}y_i})^2$$

梯度推导出有：

$$\frac{\partial \mathrm{J}}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 \mathrm{x_i\text{-}y_i})$$

$$\frac{\partial \mathrm{J}}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m}[(\theta_0 + \theta_1 \mathrm{x_i\text{-}y_i})\mathrm{x_i}]$$

迭代步骤（$\alpha = 0.01$）：

$$\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0}$$

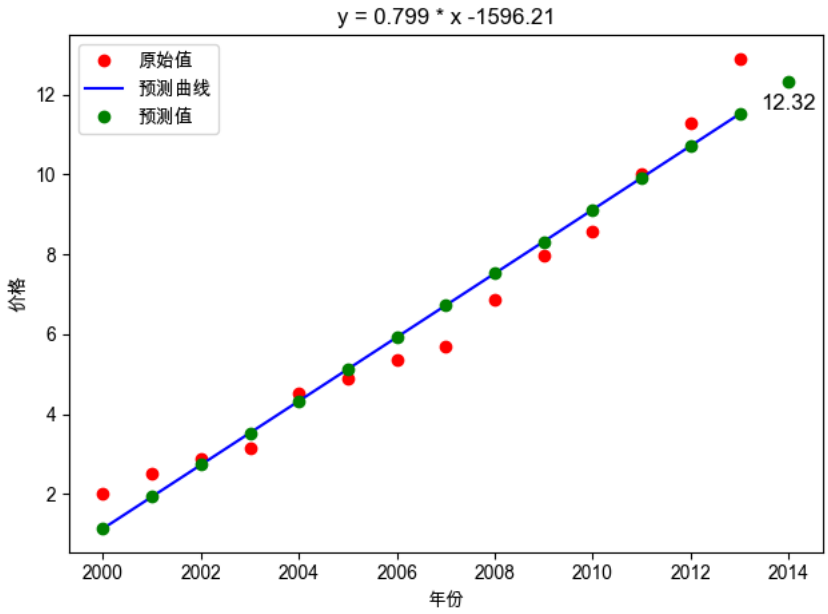
$$\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1}$$

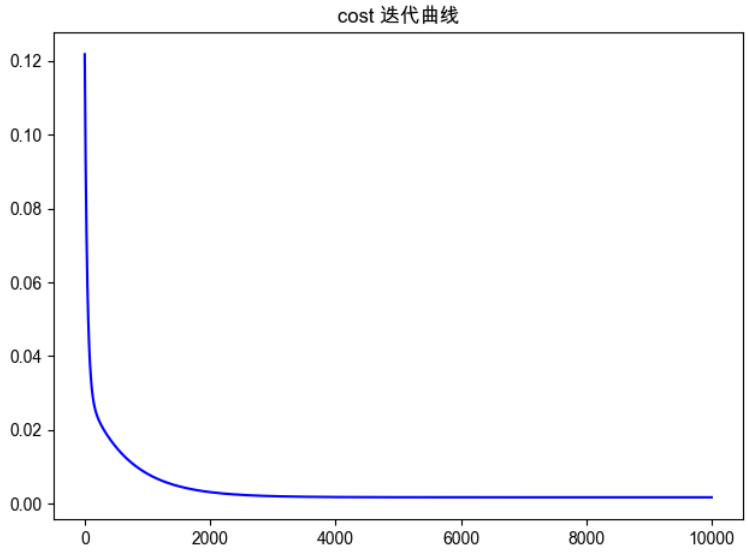
计算得到归一化处理后的

$\theta_{01} = -0.07914576590047893$ $\theta_{11} = 0.9525442068162185$

反归一化后得到$\theta_0 = -1596.2060510478204$ $\theta_1 = 0.7986716810997524$

$$Y = \theta_1 X + \theta_0$$





预测 2014 年房价为 12.318714687080956

梯度下降法代码（其中数据 csv 为 x 一行，y 一行）：

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# @time    : 2020/10/17 14:14
# @author  : lerogo
# @fileName: main.py

import csv

import numpy as np
import matplotlib.pyplot as plt

# 解决 plt 画图中文字体字体
plt.rcParams['font.sans-serif'] = ['Arial Unicode MS']  # macos
# plt.rcParams['font.sans-serif'] = ['KaiTi'] # windows
plt.rcParams['axes.unicode_minus'] = False  # 解决保存图像是负号'-'显示为方块的问
题


# 从 csv 读取数据
def readData():
    csv_reader = csv.reader(open("data.csv"))
    rows = []
    for row in csv_reader:
        rows.append(row)
    x = rows[0]
    y = rows[1]
    for i in range(0, len(x)):
        x[i] = int(str(x[i]).strip())
    for i in range(0, len(y)):
        y[i] = float(str(y[i]).strip())
    return np.array(x), np.array(y)


# 归一化
def normalized(para):
    para = (para - para.min()) / (para.max() - para.min())
    return para


# 归一化 特殊
def normalized_special(para, special):
    special = (special - para.min()) / (para.max() - para.min())
    return special
```

```python
# 反归一化
def reverse_normalized(para, special):
    special = special * (para.max() - para.min()) + para.min()
    return special


# sita[0] sita[1] 代价函数
def get_cost(x, y, sita):
    cost = ((sita[0] + sita[1] * x - y) ** 2).sum()
    return cost / 2 / len(x)


# x,y 步长 sita[0]sita[1] 接受的 cost 最小多少  最多迭代次数
def get_gradient(x, y, alpha, sita, accept_cost, max_times):
    m = len(x)  # 多少个量
    dev = [0, 0]  # 梯度
    times = 0  # 迭代次数
    cost = get_cost(x, y, sita)  # 计算第一次 cost
    cost_list = []  # 储存迭代出的 cost
    # 开始迭代
    while cost > accept_cost and times < max_times:
        dev[0] = ((sita[0] + sita[1] * x - y).sum()) / m  # 梯度 sita0
        dev[1] = (((sita[0] + sita[1] * x - y) * x).sum()) / m  # 梯度 sita1
        # 重新计算 sita
        sita[0] -= alpha * dev[0]
        sita[1] -= alpha * dev[1]
        cost = get_cost(x, y, sita)  # 重新计算 cost
        cost_list.append(cost)  # 加入 cost_list 方便画图
        times += 1
    return sita, cost_list


if __name__ == '__main__':
    data_x, data_y = readData()
    x = normalized(data_x)
    y = normalized(data_y)
    sita, cost_list = get_gradient(
        x=x,
        y=y,
        alpha=0.01,
        sita=[0, 0],
        accept_cost=1e-5,
        max_times=1e4
```

```
    )
    print(sita)

    # 计算预测值 老算法
    # predict_y = sita[0] + sita[1] * x
    # predict_y = reverse_normalized(data_y, predict_y)
    #  计算 反归一化后的 sita
    sita2 = [0, 0]
    sita2[1] = sita[1] / (data_x.max() - data_x.min()) * (data_y.max() -
data_y.min())
    sita2[0] = (sita[0] - data_x.min() * sita[1] / (data_x.max() -
data_x.min())) * (
            data_y.max() - data_y.min()) + data_y.min()
    print(sita2)

    predict_2014 = sita2[0] + sita2[1] * 2014
    predict_y = sita2[0] + sita2[1] * data_x
    # 作出 cost 的迭代曲线
    plt.title("cost 迭代曲线")
    plt.plot([x for x in range(int(1e4))], cost_list, c='blue', label='cost 迭
代曲线')
    plt.show()
    # 作出原始值
    plt.title("y = " + str(round(sita2[1], 3)) + " * x " +
str(round(sita2[0], 2)))
    plt.plot(data_x, data_y, 'o', c='red', label='原始值')
    plt.plot(data_x, predict_y, c='blue', label='预测曲线')
    plt.plot(data_x, predict_y, 'o', c='green', label='预测值')
    plt.plot(2014, predict_2014, 'o', c='green')
    plt.text(2014, predict_2014 - 0.8, "%.2f" % predict_2014, ha='center',
va='bottom', fontsize=12)
    plt.plot()
    plt.xlabel('年份')
    plt.ylabel('价格')
    plt.legend()  # 显示图例
    plt.show()
```