

**Harbin Institute of Technology, Weihai**

**运筹学与优化方法实验报告册**

**学 号： 2201110126**

**姓 名： 宋致远**

**班 级： 2011101**

**任课教师： 晁国清**

**开课学期： 2021-2022春季学期**

哈尔滨工业大学（威海）计算机科学与技术学院

|  |  |
| --- | --- |
| 实验一 | 实验二 |
|  |  |

# 目 录

[实验一 梯度下降法 1](#_Toc26376274)

[实验二 牛顿法 7](#_Toc26376275)

# 实验一 梯度下降法

——线性回归Linear Regression

1. 实验目的

熟练掌握梯度下降法的基本思想和基本步骤，并熟练掌握一种线搜索方法

1. 实验内容

设计并实现梯度下降法求线性回归问题，并可以对给定数据集进行训练和预测

1. 实验说明

1）回归模型及梯度下降法

线性回归模型, 是要优化的参数，是维的特征向量，注意, 是一个截断值也叫偏置项。

给定个样本数据组成的数据集,线性回归的目标是用线性函数拟合，即通过最小化如下目标函数寻找参数的最优解。

梯度下降法的迭代公式如下：

参数是步长。

2）数据集data1及训练回归模型

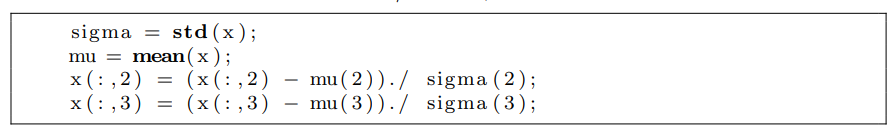
给定的数据集, 即该数据集有50个样本数据点，每个数据点只有一个特征，注意实验时还要考虑偏置项，即给每个样本数据增加一维特征取值为1，所以可理解为数据是2维特征。样本的一维特征是小孩的年龄，拟合的是小孩的身高。该任务是用线性回归模型从小孩的年龄拟合身高。

3) 实验要求

1. 用步长实现梯度下降法。初始化参数(即), 运行一次梯度下降法的迭代，记录下的值
2. 继续运行梯度下降法，直到收敛。收敛标准可以是, 可以取或。并将迭代次数作为横坐标，目标函数值作为纵坐标，画出表明收敛的图形。
3. 算法收敛后，就确定了线性回归的参数，此时就可以利用对于新的样本点进行预测，即给定小孩的年龄，可以预测他的身高。请利用你训练出来的模型，预测出年龄分别为3.5岁和7岁的两位小孩的身高。

4）多变量线性回归

data2是包含47个样本点2个特征的数据集，两维特征分别表示房子面积（单位平方英尺）和卧室数目（个），表示相应房子的价格（美元）。仍然要注意添加偏置项。由于房子面积和卧室数目取值大小差别很大，需要对这些特征分别做标准化。伪代码如下：



这次步长的范围要求在内取，可以采用学过的线性搜索方法确定步长，也可以网格试探，譬如以5倍间隔试探，步长取值0.001,0.005，0.01，… , 10。然后找出合适的步长，进行梯度下降法实验。

目标函数可用向量表示为, 这里的分别为：

,

写出向量版本的梯度下降法迭代公式，并实现。

回答如下3个问题：

1. 观察目标函数随步长的变化，步长大时如何表现，步长小时如何表现
2. 用最好的步长运行梯度下降法直到收敛(收敛准则可以参考单变量线性回归)，并给出最优的取值
3. 预测一下1650平方英尺3卧室的房子的价格。
4. 实验过程

注意：对于线性回归和多变量线性回归，可以在下面各分为两部分描述，如果实现的版本可以同时处理两者，也可以合并描述

1. 问题描述

*(问题分析及功能描述)*

给出一组儿童年龄与身高对应关系，并要求进行预测，为线性回归部分。

给出一组住房面积、户型与房价的对应关系，并要求预测房价，为多变量线性回归。

1. 算法设计

*(关键算法思路+伪代码或流程图)*

采用梯度下降法进行实现。

对于线性回归部分：说明中已经给出迭代公式，使用该迭代公式不断迭代即可。

伪代码如下：

While 大于eps

直至符合收敛标准，获取θ的值，并以此进行预测。

对于多变量线性回归部分：首先通过求解梯度，获取负梯度方向d，并使用d与θ的值，拟采用黄金分割法求解适合的步长。

伪代码如下：

While 大于eps:

其中getAlpha部分伪代码：

getAlpha(θ, d, l, r):   
 l = 0.001, r = 10   
 while (l < r):   
 compare (f(l + len \* 0.618), f(r – len \* 0.618)):   
 r = l + len \* 0.618   
 else l = r – len \* 0.618

程序实现

*(函数说明+函数之间的调用关系+关键算法的实现代码)*

线性回归代码实现：

import csv   
import numpy as np   
import matplotlib.pyplot as plt   
  
x = []   
y = []   
csv\_reader = csv.reader(open("E:/Download/QQDownload/2201110126/2201110126/data/data1\_x.csv "))   
for line in csv\_reader:   
 x.append([1.0, float(line[0])])   
csv\_reader = csv.reader(open("E:/Download/QQDownload/2201110126/2201110126/data/data1\_y.csv "))   
for line in csv\_reader:   
 y.append([float(line[0])])   
x = np.array(x)   
y = np.array(y)   
  
eps = 1e-8   
def judge(nextTheta: np.ndarray, theta: np.ndarray):   
 return np.linalg.norm(nextTheta - theta) \*\* 2 < eps   
  
alpha = 0.01   
def getNextTheta(theta: np.ndarray):   
 nextTheta = theta - alpha / len(x) \* np.dot(x.T, (np.dot(x, theta) - y))   
 return nextTheta   
  
theta = np.array([[0.0], [0.0]])   
nextTheta = getNextTheta(theta)   
  
historyTheta = []   
while not judge(nextTheta, theta):   
 historyTheta.append(theta.copy())   
 theta = nextTheta.copy()   
 nextTheta = getNextTheta(theta)   
   
def h(theta: np.ndarray, x: np.ndarray):   
 return np.dot(x, theta)   
print(theta)

print("对3.5的预测结果为：", h(theta, np.array([1.0, 3.5])))

print("对7.0的预测结果为：", h(theta, np.array([1.0, 7.0])))  
def J(theta: np.ndarray):   
 return (np.linalg.norm(h(theta, x) - y) \*\* 2) \* len(x) / 2   
   
# 绘图   
plt.rcParams['font.sans-serif'] = ['SimHei']   
plt.xlabel('线性回归次数')   
plt.ylabel('h\_θ (x)')   
x\_data = []   
y\_data = []   
for i in range(len(historyTheta)):   
 x\_data.append(i)   
 y\_data.append(J(historyTheta[i]))   
plt.plot(x\_data, y\_data)   
plt.show()

多变量线性回归代码实现：

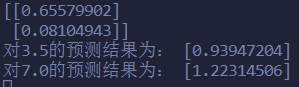
import csv   
import numpy as np   
import matplotlib.pyplot as plt   
   
  
x = []   
y = []   
csv\_reader = csv.reader(open("E:/Download/QQDownload/2201110126/2201110126/data/data2\_x.csv"))   
for line in csv\_reader:   
 x.append([1.0, float(line[0].split(" ")[1]), float(line[0].split(" ")[2])])   
csv\_reader = csv.reader(open("E:/Download/QQDownload/2201110126/2201110126/data/data2\_y.csv"))   
for line in csv\_reader:   
 y.append([float(line[0])])   
x = np.array(x)   
y = np.array(y)   
sigma = [np.std(x[:, i]) for i in range(3)]   
mu = [np.mean(x[:, i]) for i in range(3)]   
x[:, 1] = (x[:, 1] - mu[1]) / sigma[1]   
x[:, 2] = (x[:, 2] - mu[2]) / sigma[2]   
   
def J(theta: np.ndarray):   
 return (np.linalg.norm(np.dot(x, theta) - y) \*\* 2) / (len(x) \* 2)   
def getGradient(theta: np.ndarray):   
 return np.dot(x.T, (np.dot(x, theta) - y)) / len(x)   
eps = 1e-6   
def getAlpha(theta: np.ndarray, d: np.ndarray):   
 l = 0.001   
 r = 10   
 while (r - l > eps):   
 ll = l + (r - l) \* 0.382   
 rr = l + (r - l) \* 0.618   
 if (J(theta + ll \* d) < J(theta + rr \* d)):   
 r = rr   
 else:   
 l = ll   
 return l  
historyTheta = []   
theta = np.array([[0.0], [0.0], [0.0]])   
d = -getGradient(theta)   
alpha = getAlpha(theta, d)   
while np.linalg.norm(d) > eps:   
 theta = theta + alpha \* d   
 d = -getGradient(theta)   
 alpha = getAlpha(theta, d)   
 historyTheta.append(theta.copy())   
   
  
print(theta)   
print(np.dot(np.array([1.0, (1650.0 - mu[1]) / sigma[1], (3.0 - mu[2]) / sigma[2]]), theta))   
  
plt.rcParams['font.sans-serif'] = ['SimHei']   
plt.xlabel('线性回归次数')   
plt.ylabel('h\_θ (x)')   
x\_data = []   
y\_data = []   
for i in range(len(historyTheta)):   
 x\_data.append(i)   
 y\_data.append(J(historyTheta[i]))   
plt.plot(x\_data, y\_data)   
plt.show()

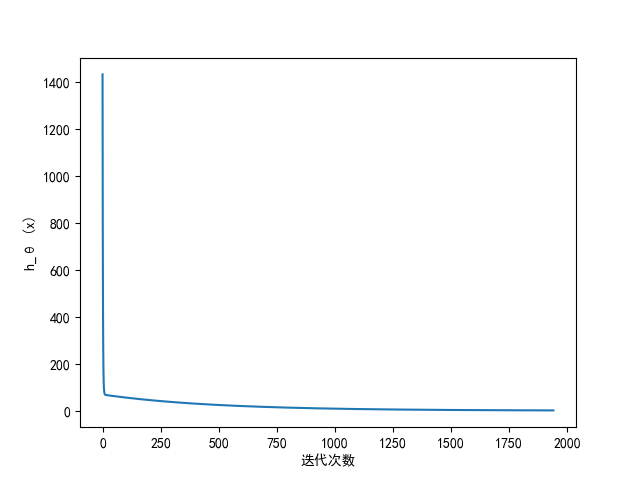
1. 实验结果

*(运行截图+结果分析描述+遇到的问题和解决办法等)*

线性回归结果截图：

收敛时为[0.6557, 0.0810]

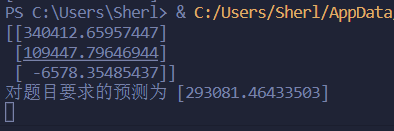




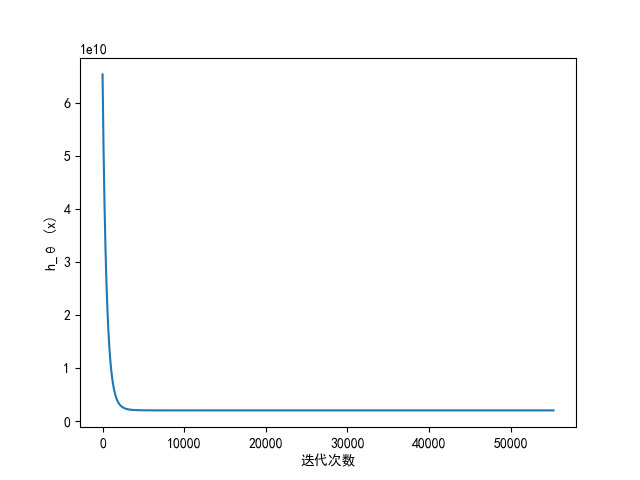
多变量线性回归结果截图：

[1650, 3]的预测结果为 293081.4643

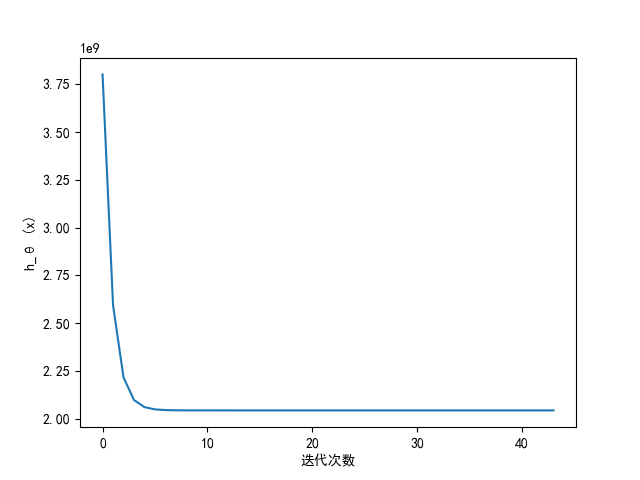
收敛时为[340412.6595, 109447.7964, -6578.3548]



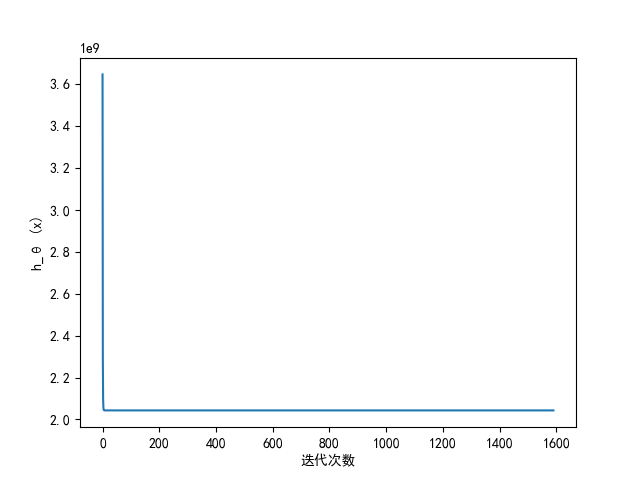
当步长固定为0.00114514时截图：



为1时



当步长用黄金分割求的时候



步长大的时候，收敛速度快但有可能误差较大

步长小的时候，收敛速度慢但有可能误差较小

1. 实验总结

*(实验体会、学习收获、过程总结 等)*

经过此次实验，我们学会了简单的用python进行数学算法的编写，使用，和画图，利用已经有的简易模型，我们已经能通过各种回归算法进行简单预测，加深了我们对线性回归的理解

# 实验一 牛顿法

——逻辑回归Logistic Regression

1. 实验目的

熟练掌握牛顿法的基本思想和基本步骤

1. 实验内容

设计并实现牛顿法求逻辑回归问题，并可以对给定数据集进行训练和预测

1. 实验说明

1)逻辑回归及牛顿法

逻辑回归是一种简单的分类器。Hypothesis function 为

给定数据集, 代表第个样本的特征，代表第个样本的标签{1,0}。可定义似然函数如下：

为方便计算，通常对似然函数取对数，得到对数似然函数如下：

可以通过最大化对数似然寻找参数的最优解，通过添加负号将最大化问题转化为最小化问题：

该目标函数的梯度可计算为

Hessian矩阵可以计算为

牛顿法更新公式：

这里的表示都是向量形式。对于数据同样需要增加一个,从而使得, 和都是标量。

取为初始值，注意这里是向量。收敛标准可以是*,* 可以取或。

2）数据集

data2是学校学生是否录取任务的数据集。共有80名学生，40名被录取，40名没被录取，它们对应的y标签分别为1和0。每名学生包含两位特征和分别表示第一次考试成绩和第二次考试成绩。该任务为用逻辑回归分类器训练，然后使训练好的逻辑回归分类器可以对新的学生预测是否会被录取。

3）实验要求

1. 达到收敛时的取值是什么？

2. 以迭代次数为横坐标，L取值为纵坐标，画图。

3. 预测第一次考试20分和第二次考试80分学生能否被录取。

4. 通过牛顿法实验，对比梯度下降法，有什么收获，学习到什么。

1. 实验过程
2. 问题描述

*(问题分析及功能描述)*

给出一组学生分数与是否录取关系的数据集，使用牛顿法逻辑回归训练模型，并根据给出学生得分来预测是否可以被录取。

1. 算法设计

*(关键算法思路+伪代码或流程图)*

采用牛顿法求解逻辑回归问题，具体步骤已经在实验说明中给出，通过numpy库来求解梯度、Hessian矩阵等所需要的变量值，通过牛顿法进行迭代至达到收敛条件，并根据求得的θ进行预测。

while |nextTheta - theta| \*\* 2 >= eps:   
 theta = nextTheta   
 nextTheta = theta - np.dot(np.linalg.inv(getHessian(theta)), getGradient(theta))

1. 程序实现

*(函数说明+函数之间的调用关系+关键算法的实现代码)*

import csv   
import numpy as np   
import matplotlib.pyplot as plt   
   
x = []   
y = []   
csv\_reader = csv.reader(open("E:/Download/QQDownload/2201110126/2201110126/data/data2x.csv"))   
for line in csv\_reader:   
 x.append([1.0, float(line[0].split(" ")[1]), float(line[0].split(" ")[2])])   
csv\_reader = csv.reader(open("E:/Download/QQDownload/2201110126/2201110126/data/data2y.csv"))   
for line in csv\_reader:   
 y.append([float(line[0])])   
x = np.array(x)   
y = np.array(y)   
def h(x: np.ndarray, theta: np.ndarray):   
 return 1 / (1 + np.exp(-np.dot(x, theta)))   
def getGradient(theta: np.ndarray):

return np.dot(x.T, (h(x, theta) - y)) / len(x)   
def getHessian(theta: np.ndarray):   
  
 return np.dot(np.dot(x.T, np.diag((h(x, theta) \* (1 - h(x, theta))).reshape(-1))), x) / len(x)   
   
def L(theta: np.ndarray):   
 return (np.linalg.norm(np.dot(y.T, np.log(h(x, theta)))) + np.linalg.norm(np.dot((1 - y).T, np.log(1 - h(x, theta))))) / len(x)   
   
def getNextTheta(theta: np.ndarray):   
 return theta - np.dot(np.linalg.inv(getHessian(theta)), getGradient(theta))   
  
eps = 1e-6   
historyTheta = []   
theta = np.array([[0.0], [0.0], [0.0]])   
nextTheta = getNextTheta(theta)   
while np.linalg.norm(nextTheta - theta) \*\* 2 >= eps:   
 theta = nextTheta   
 nextTheta = getNextTheta(theta)   
 historyTheta.append(theta.copy())   
  
print(theta)

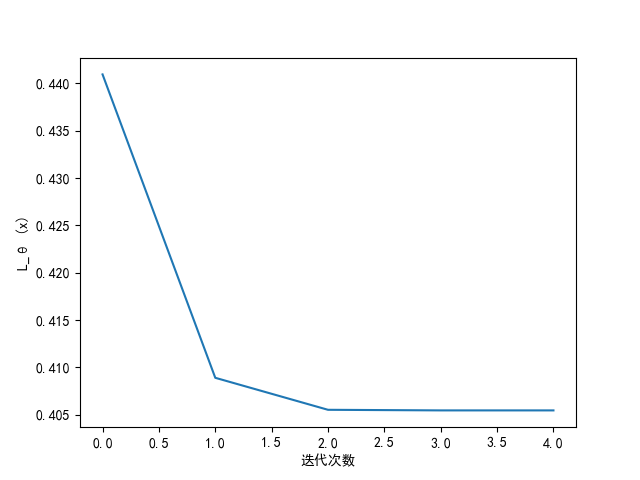
print("预测能否录取",h(np.array([1.0, 20.0, 80.0]), theta))

print("能录取" if h(np.array([1.0, 20.0, 80.0]), theta) > 0.5 else "不行")  
plt.rcParams['font.sans-serif'] = ['SimHei']   
plt.xlabel('牛顿法次数')   
plt.ylabel('L\_θ (x)')   
x\_data = []   
y\_data = []   
for i in range(len(historyTheta)):   
 x\_data.append(i)   
 y\_data.append(L(historyTheta[i]))   
plt.plot(x\_data, y\_data)   
plt.show()

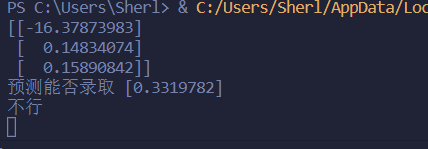
1. 实验结果

*(运行截图+结果分析描述+遇到的问题和解决办法等)*

（1）收敛时θ值为[-16.378, 0.1483, 0.1589]

（2）

（3）



有一说一，牛顿法在计算量上比起之前的方法更大，但是速度也更快，适合对速度要求比较高，对算力限制不太大的地方

1. 实验总结

*(实验体会、学习收获、过程总结 等)*

经过此次实验，我们学会了简单的用python进行数学算法的编写，使用，和画图，利用已经有的简易模型，我们已经能通过各种算法进行简单预测，加深了我们对牛顿法的理解