**中国矿业大学计算机学院**

**2021 级本科生课程报告**

课程名称： 最优化理论与方法

报告题目： 《拟牛顿法原理及实现》

报告时间： 2024.1.10

学生姓名： 杨学通

学 号： 08213129

专 业： 人工智能2021-1班

任课教师： 赵佳琦

**课程报告成绩评定及评分依据**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 评分依据 | 分值 | 实际得分 |
| 1 | 课程报告描述了最优化技术或算法的基本原理 | 20 |  |
| 2 | 课程报告使用python或C++程序设计语言（附源代码） | 40 |  |
| 3 | 给出至少2个测试用例，并且给出可视化的运行结果及分析 | 30 |  |
| 4 | 课程报告给出了个人学习体会 | 10 |  |
| 总分 | | |  |
| 教师评语：  教师签名：  年 月 日 | | | |

**摘 要**

拟牛顿法和最速下降法(Steepest Descent Methods)一样只要求每一步迭代时知道目标函数的梯度。通过测量梯度的变化，构造一个目标函数的模型使之足以产生超线性收敛性。这类方法大大优于最速下降法，尤其对于困难的问题。另外，因为拟牛顿法不需要二阶导数的信息，所以有时比牛顿法(Newton's Method)更为有效。如今，优化软件中包含了大量的拟牛顿算法用来解决无约束，约束，和大规模的优化问题。

拟牛顿法是解非线性方程组及最优化计算中最有效的方法之一.它是一类使每步迭代计算量少而又保持超线性收敛的牛顿型迭代法。

**关键词**：拟牛顿法；超线性收敛性；优化问题；牛顿型迭代法

目录

[1问题描述 1](#_Toc155829931)

[2算法原理及流程 1](#_Toc155829932)

[3运行代码 1](#_Toc155829933)

[4运行截图 3](#_Toc155829934)

[5总结与分析 4](#_Toc155829935)

# 1问题描述

牛顿法成功的关键在于利用了Hessian矩阵提供的曲率信息，而计算Hessian矩阵工作量大，并且有的目标函数的Hessian矩阵很难计算，甚至不好求出，这就导致仅利用目标函数一阶导数的方法，拟牛顿法就是利用目标函数值f和一阶导数g的信息，构造出目标函数的曲率近似，而不需要明显形成Hessian矩阵，同时具有收敛速度快的优点。

本次实验要求python实现拟牛顿法，并通过测试样例，验证拟牛顿法的正确性，体会拟牛顿法进行多维搜索的过程。

# 2算法原理及流程

Step1: 给定初始点，允许误差

Step2:置，计算出在处的梯度，置

Step3:令

Step4:从出发，沿进行一维搜索，求，使，令

Step5:若，停止

Step6:若，则令,转Step2，

Step7：令，，

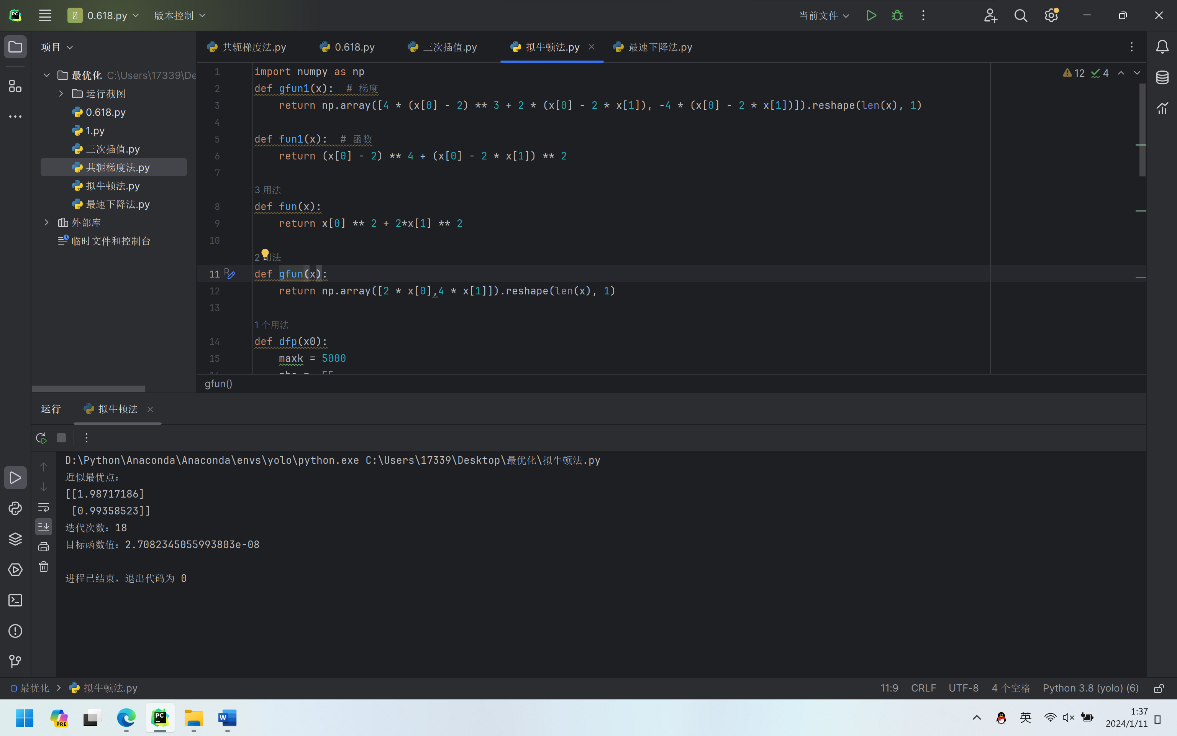
，置，转Step3

# 3运行代码

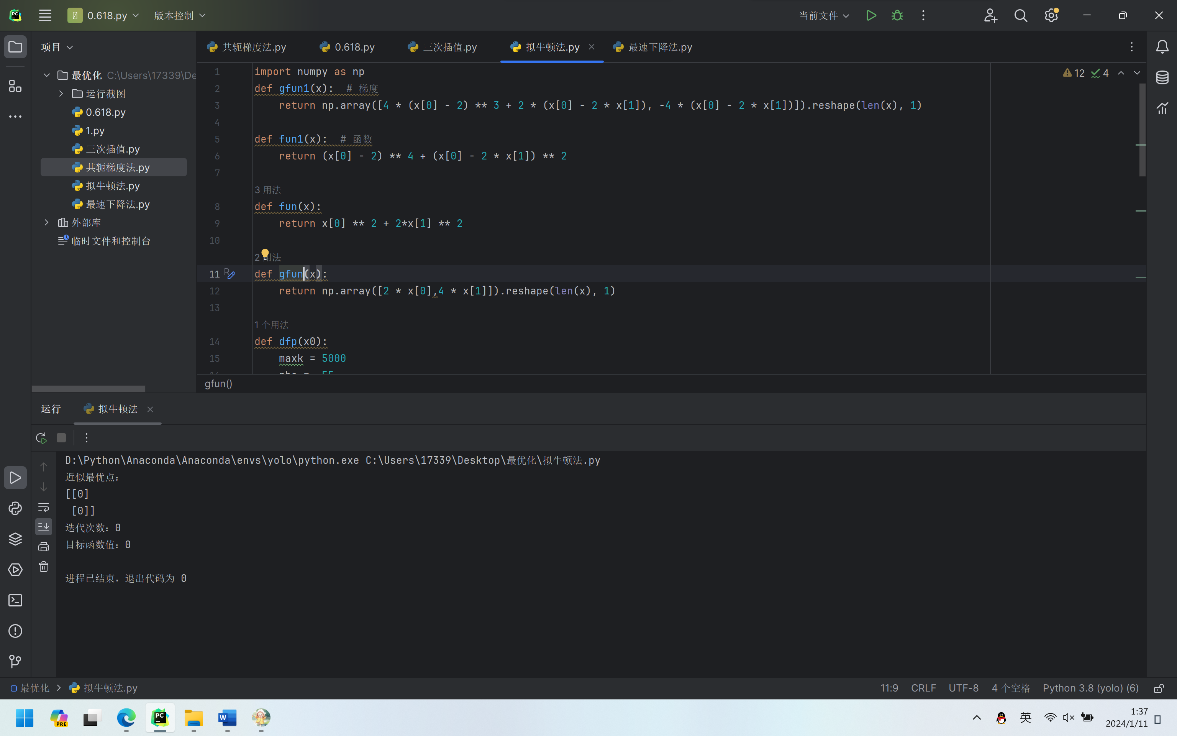
import numpy as np  
def gfun1(x): *# 梯度* return np.array([4 \* (x[0] - 2) \*\* 3 + 2 \* (x[0] - 2 \* x[1]), -4 \* (x[0] - 2 \* x[1])]).reshape(len(x), 1)  
  
def fun1(x): *# 函数* return (x[0] - 2) \*\* 4 + (x[0] - 2 \* x[1]) \*\* 2  
  
def fun(x):  
 return x[0] \*\* 2 + 2\*x[1] \*\* 2  
  
def gfun(x):  
 return np.array([2 \* x[0],4 \* x[1]]).reshape(len(x), 1)  
  
def dfp(x0):  
 maxk = 5000  
 rho = .55  
 sigma = .4  
 epsilon = 1e-5  
 k = 0  
 n = len(x0)  
 Hk = np.eye(n)  
 while k < maxk:  
 gk = gfun(x0)  
 if np.linalg.norm(gk) < epsilon:  
 break  
 dk = -Hk @ gk  
 m = 0  
 mk = 0  
 while m < 20: *# 使用Armijo搜索（非精确线搜索）* if fun(x0 + rho \*\* m \* dk) < fun(x0) + sigma \* rho \*\* m \* gk.T @ dk:  
 mk = m  
 break  
 m += 1  
 x = x0 + rho \*\* mk \* dk  
 sk = x - x0  
 yk = gfun(x) - gk  
 if sk.T @ yk > 0:  
 Hk = Hk - (Hk @ yk @ yk.T @ Hk) / (yk.T @ Hk @ yk) + (sk @ sk.T) / (sk.T @ yk)  
 k += 1  
 x0 = x  
 return x0, fun(x0), k  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 x0 = np.array([[0], [0]])  
 x0, val, k = dfp(x0)  
 print(f'近似最优点：\n{x0}\n迭代次数：{k}\n目标函数值：{val.item()}')

# 4运行截图

实例1：，初始点



实例2：，初始点



# 5总结与分析

通过本次实验，我用python实现了拟牛顿法。通过使用不同的函数测试，取得很好的效果。拟牛顿法拟牛顿法是一种用于求解非线性优化问题的迭代算法，它通过逼近目标函数的二阶导数矩阵来更新搜索方向，从而加快收敛速度。拟牛顿法利用了目标函数的二阶导数信息，能够更快地找到最优解。同时，拟牛顿法不需要计算二阶导数，与牛顿法相比，拟牛顿法不需要显式计算目标函数的二阶导数，减少了计算量和存储开销。

拟牛顿法也存在着存储开销大、计算复杂度高、对初始点敏感等缺点。