**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： 《最速下降法原理及实现》

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**

最速下降法是一种常用的数值优化算法，用于求解无约束优化问题。该方法基于梯度下降的思想，通过不断沿着目标函数的负梯度方向迭代更新自变量，以逐步逼近目标函数的极小值点。最速下降法的基本思想是沿着当前点的梯度方向，以步长因子乘以梯度的负方向作为搜索方向，从而实现迭代更新。然而，最速下降法也存在一些问题，例如可能出现收敛速度慢、在目标函数等值线附近可能出现震荡等情况。因此，在实际应用中，需要结合具体问题特点和算法性能进行选择和调优。

**关键词**：最速下降法；梯度下降；负梯度；收敛速度；震荡

目录

[1问题描述 1](#_Toc155817813)

[2算法原理及流程 1](#_Toc155817814)

[3运行代码 1](#_Toc155817815)

[4运行截图 3](#_Toc155817816)

[5总结与分析 4](#_Toc155817817)

# 

# 1问题描述

最速下降法是指根据当前所在点，确定搜索方向，调整搜索步长，迭代变量，最终靠近目标函数最优值的一种方法，最速下降法的优点包括：

简单易实现：最速下降法是一种直观且易于实现的优化算法，仅需要计算目标函数的梯度，并沿着负梯度方向进行迭代更新。

内存消耗小：最速下降法通常不需要存储大量的历史信息或者矩阵，因此在内存消耗方面较为节省。

对大规模问题有效：最速下降法在处理大规模优化问题时具有一定的优势，因为它不需要存储大规模的矩阵或者历史信息。

适用性广泛：最速下降法适用于一般的凸优化问题，而且可以很容易地与其他优化方法结合使用，例如共轭梯度法等。

这些特点使得最速下降法在实际应用中得到了广泛的应用。本次实验要求python实现最速下降法，并通过测试样例，验证最速下降法的正确性，体会最速下降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，置

Step2:计算搜索方向

Step3:若，则停止计算，得点；否则：

从出发，沿进行一维搜索，求得，使得

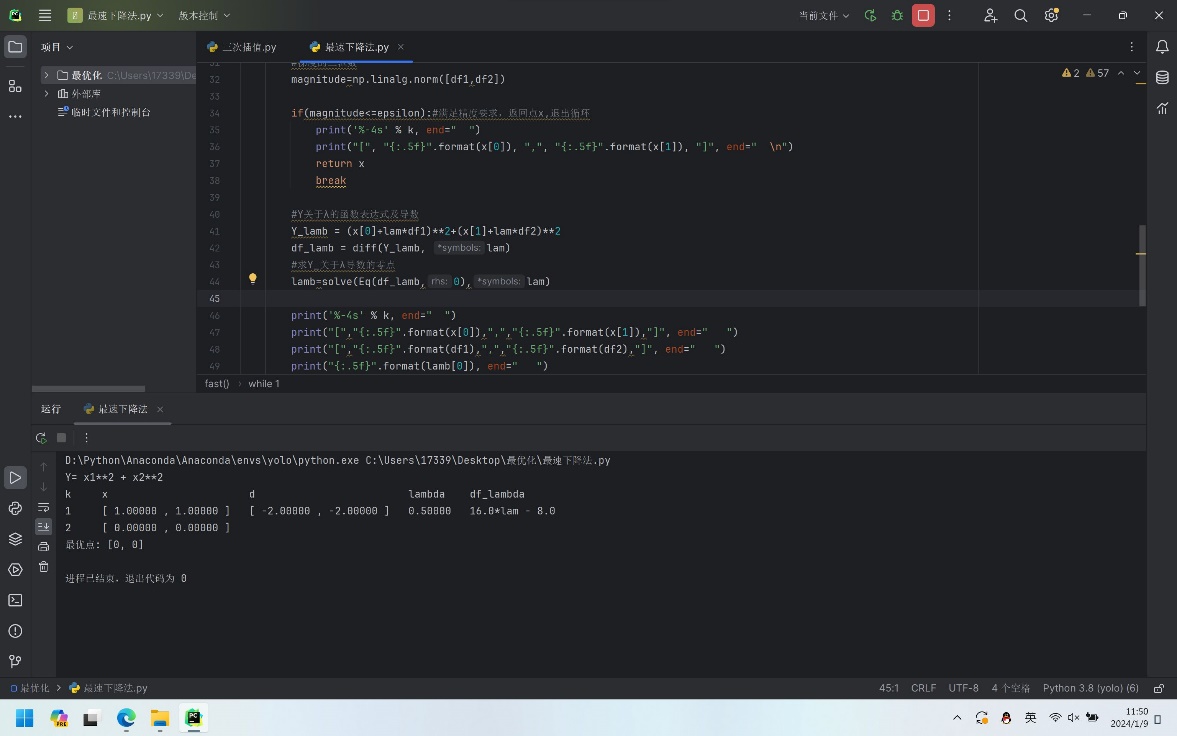
Step4:令，转Step2

# 3运行代码

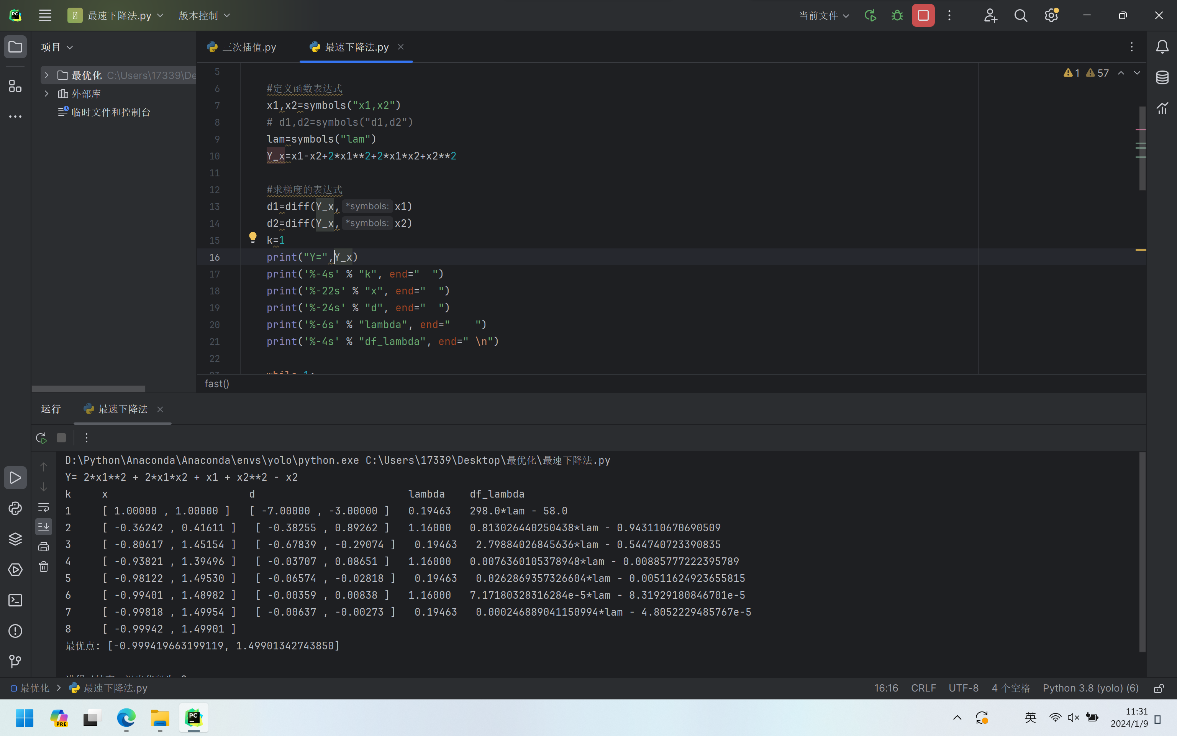
import math  
  
import numpy as np  
from sympy import \*  
  
def fast(x,epsilon):  
  
 *#定义函数表达式* x1,x2=symbols("x1,x2")  
 *# d1,d2=symbols("d1,d2")* lam=symbols("lam")  
 Y\_x=x1\*\*2+x2\*\*2  
  
 *#求梯度的表达式* d1=diff(Y\_x,x1)  
 d2=diff(Y\_x,x2)  
 k=1  
 print("Y=",Y\_x)  
 print('%-4s' % "k", end=" ")  
 print('%-22s' % "x", end=" ")  
 print('%-24s' % "d", end=" ")  
 print('%-6s' % "lambda", end=" ")  
 print('%-4s' % "df\_lambda", end=" \n")  
  
 while 1:  
  
 *#求x[0]点的负梯度* df1=float(-d1.evalf(subs ={'x1':x[0],'x2':x[1]}))  
 df2=float(-d2.evalf(subs ={'x1':x[0],'x2':x[1]}))  
  
 *#梯度的二范数* magnitude=np.linalg.norm([df1,df2])  
  
 if(magnitude<=epsilon):*#满足精度要求，返回点x,退出循环* print('%-4s' % k, end=" ")  
 print("[", "{:.5f}".format(x[0]), ",", "{:.5f}".format(x[1]), "]", end=" \n")  
 return x  
 break  
  
 *#Y关于λ的函数表达式及导数* Y\_lamb = (x[0]+lam\*df1)\*\*2+(x[1]+lam\*df2)\*\*2  
 df\_lamb = diff(Y\_lamb, lam)  
 *#求Y\_关于λ导数的零点* lamb=solve(Eq(df\_lamb,0),lam)  
  
 print('%-4s' % k, end=" ")  
 print("[","{:.5f}".format(x[0]),",","{:.5f}".format(x[1]),"]", end=" ")  
 print("[","{:.5f}".format(df1),",","{:.5f}".format(df2),"]", end=" ")  
 print("{:.5f}".format(lamb[0]), end=" ")  
 print(df\_lamb,end="\n")  
  
 *#迭代点x* x[0]+=lamb[0]\*df1  
 x[1]+=lamb[0]\*df2  
 k+=1  
  
  
print("最优点:",fast([1,1],0.001))

# 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实现了最速下降法。通过使用不同的函数测试，取得很好的效果。虽然最速下降法内存消耗小，适用性广泛，但仍存在着不足：其收敛速度慢，对目标函数的形状敏感，如果目标函数具有狭长的形状，可能会导致收敛速度变慢。同时，在等值线附近，最速下降法可能会出现震荡现象，导致收敛困难。最优化的算法各有利弊，应当根据实际情况选择合适的算法求解问题。