**《最优化技术》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | | **2019级、计算机科学与技术、计科01班** | | | **姓名** | **谢双骏** |
| **实验题目** | 一维搜索算法的应用 | | | | | |
| **实验时间** | **2021年5月7日** | | **实验地点** | **DS3401** | | |
| **实验成绩** |  | | **实验性质** | **□验证性 ■设计性 □综合性** | | |
| 教师评价：  □算法/实验过程正确；□源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | |
| 一、实验目的  理解掌握一维搜索算法中的黄金分割法，牛顿法，并用于实际问题的求解。 | | | | | | |
| 二、实验项目内容   1. 给定一个函数f(x)=8e1-x+7log(x)，利用黄金分割法把区间压缩到长度只有0.23，需给出所有中间结果。 2. 给定一个函数f(x)=60-10x1-4x2+x12+x22-x1x2，利用牛顿法求解该函数的最小值，需给出中间结果。   注意：所有程序请用python语言实现。只提交本电子文档，注意本文件末尾的文件命名要求；源程序一节请用代码备注的方式说明你的算法和思路；实验结果一节需要提供测试结果截图并给出结果分析。 | | | | | | |
| 1. 实验过程或算法（源程序） 2. 黄金分割法 3. 方法简介   黄金分割法(Golden　Section　Method)又称为0.618法，是用于在单峰函数区间上求极小的一种方法。通过取试探点和进行函数值比较，使包含极小点的搜索区间不断减少，当区间长度缩短到一定程度时，就得到函数极小点的近似值。   1. 代码   #引入数学库  import math  #展示目标函数  print("目标函数为："+"f(x)=8\*exp(1-x)+7\*log(x)"+"\n")  #记录迭代次数  count=0  #输入初始的区间[a,b]和区间精度c  a=int(input("请输入起始区间的第一个数字（整数）："))  b=int(input("请输入起始区间的第二个数字（整数）："))  c=float(input("请输入区间精度(浮点数)："))  #声明初始的迭代点a1和a2，初始的函数值f1和f2  a1=b-0.618\*(b-a)  a2=a+0.618\*(b-a)  f1=8\*math.exp(1-a1)+7\*math.log(a1)  f2=8\*math.exp(1-a2)+7\*math.log(a2)  #开始迭代计算，当区间长度b-a小于c，一直迭代  while(b-a>c):  #若f1 > f2 ，极小点必在[α1 ,b] 内，消去区间[a，α1] ，令a＝α1，产生新区间[a ,b]，相应函数值更新  if f1>f2:  a=a1  a1=a2  f1=f2  a2=a+0.618\*(b-a)  f2=8\*math.exp(1-a2)+7\*math.log(a2)  #若f1 ≤ f2 ，极小点必在[a，α2] 内，消去区间[α2 ，b] ，令b＝α2，产生新区间[a ,b] ，相应函数值更新  else:  b=a2  a2=a1  f2=f1  a1=b-0.618\*(b-a)  f1=8\*math.exp(1-a1)+7\*math.log(a1)  #迭代次数加一  count+=1  #输出每一次的迭代区间和目标函数值，全都保留三位小数  print("第"+str(count)+"次迭代的区间结果是："+"["+str('%.3f'% a)+","+str('%.3f'% b)+"]")  temp=8\*math.exp(1-((a+b)/2))+7\*math.log(((a+b)/2))  print("第"+str(count)+"次迭代的函数值是："+str('%.3f'%temp)   1. 牛顿法 2. 牛顿法简介   牛顿法是用一个二次曲面去拟合当前所处位置的局部曲面；梯度下降法是用一个平面去拟合当前的局部曲面，是二阶收敛，多变量牛顿迭代公式：  (Xn)  其中，H是Hessian矩阵，Jf是雅可比矩阵  895V)R5H}N0AT2035B22NI7  2、代码  #引入numy库  import numpy as np  #定义雅可比矩阵，x为x1,...,xn的列向量,下方黑塞矩阵亦是  def jacobian(x):  return np.array([2\*x[0]-x[1]-10,2\*x[1]-x[0]-4])  #定义黑塞矩阵  def hessian(x):  return np.array([[2,-1],[-1,2]])  #定义牛顿法，x0为初始的x1,...,xn的列向量  def newton(x0):  print("初始点为："+str(x0))  #计数，因为牛顿法存在当初始点远离极小点时，牛顿法产生的点列可能不收敛或者收敛到鞍点，或者Hesse矩阵不可逆，无法计算的问题  count=1  #迭代到一千次还没有结果的话，说明无法计算，停止  count\_max=1000  #x为迭代的解，初始为x0  x=x0  #只有当迭代次数小于上限同时雅可比矩阵（梯度向量）不为为零矩阵时才迭代  while count<=count\_max and np.all(jacobian(x)==0)==0:  #np.linalg.inv()函数为求矩阵的逆函数，np.dot(X,Y)为求矩阵乘积的函数  p=np.dot(np.linalg.inv(hessian(x)),jacobian(x))  x=x-p  #输出每一次的迭代结果和目标函数值  print("第"+str(count)+"次迭代结果为： "+"["+str(int(x[0]))+" "+str(int(x[1]))+"]")  print("第"+str(count)+"次迭代函数值为： "+str(60-10\*x[0]-4\*x[1]+x[0]\*x[0]+x[1]\*x[1]-x[0]\*x[1])+"\n")  #计数次数加一  count+=1  #设定初始值  x0=np.array([2,2])  #调用迭代函数  newton(x0) | | | | | | |
| 1. 实验结果及分析和（或）源程序调试过程 2. 黄金分割 3. 结果   XV1AP]J6RPY(F9ISEQ8KYLB  }U25HBB_DQER]3S@RHO~WEI   1. 结果分析   根据第三方软件绘图，得出目标函数的大致图像如下：  wps  最终的结果应该在[1,2]之间  初始区间从[0,130]经过8次收缩右边区间得到了[0,2.768],已经十分逼近目标区间，接下来的5次迭代，区间继续收缩，最终停滞在了[1.557,1.711]这个范围，目标函数的最小值为7.681，搜索效果还是很不错的   1. 牛顿法 2. 结果   {M_$659~OPXLD~{D~OS%][G   1. 结果分析   根据第三方作图软件作图如下：  untitled  可以发现最小值在[8,6]附近，接下来开始执行代码迭代  初始点[2,2],其梯度矩阵为[-8,-2],不满足条件继续迭代，迭代一次后，解为[8,6],其梯度矩阵为[0,0],满足条件，w为最终解，停止迭代，目标函数的最小值为8 | | | | | | |