



计算智能导论实验报告

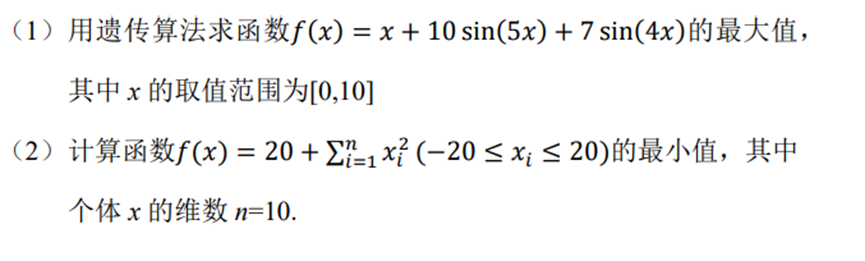
|  |  |
| --- | --- |
| 名 称： | 遗传算法的实现 |
| 学生姓名： | 刘易行 |
| 学 号： | 2020905896 |
| 专 业： | 人工智能 |
| 班 级： | 2020240401 |
| 指导教师： | 王威 |
| 时 间： | 2022.11.12 |

# 一、实验目的

掌握遗传算法的特性和应用范围，能够用遗传算法解决实际问题。

能够根据应用需求，设计合适的算子实现经典的遗传算法。

# 二、实验内容



# 三、实验原理

遗传算法是受自然进化理论启发的一系列搜索算法。通过模仿自然选择和繁殖的过程，遗传算法可以为涉及搜索，优化和学习的各种问题提供高质量的解决方案。同时，它们类似于自然进化，因此可以克服传统搜索和优化算法遇到的一些障碍，尤其是对于具有大量参数和复杂数学表示形式的问题。

遗传算法是类比自然界中的达尔文进化实现的简化版本。达尔文进化论的原理概括总结如下：

变异：种群中单个样本的特征（性状，属性）可能会有所不同，这导致了样本彼此之间有一定程度的差异。

遗传：某些特征可以遗传给其后代，导致后代与双亲样本具有一定程度的相似性。

选择：种群通常在给定的环境中争夺资源，更适应环境的个体在生存方面更具优势，因此会产生更多的后代。

换句话说，进化维持了种群中个体样本彼此不同。那些适应环境的个体更有可能生存，繁殖并将其性状传给下一代。这样，随着世代的更迭，物种变得更加适应其生存环境。而进化的重要推动因素是交叉（crossover）或重组（recombination）或杂交——结合双亲的特征产生后代。交叉有助于维持人口的多样性，并随着时间的推移将更好的特征融合在一起。此外，变异（mutations）或突变（特征的随机变异）可以通过引入偶然性的变化而在进化中发挥重要作用。

基因型（Genotype）: 在自然界中，通过基因型表征繁殖，繁殖和突变，基因型是组成染色体的一组基因的集合。

在遗传算法中，每个个体都由代表基因集合的染色体构成。例如，一条染色体可以表示为二进制串，其中每个位代表一个基因



种群（Population）：遗传算法保持大量的个体（individuals）——针对当前问题的候选解集合。由于每个个体都由染色体表示，因此这些种族的个体（individuals）可以看作是染色体集合

适应度函数：在算法的每次迭代中，使用适应度函数（也称为目标函数）对个体进行评估。目标函数是用于优化的函数或试图解决的问题。

适应度得分更高的个体代表了更好的解，其更有可能被选择繁殖并且其性状会在下一代中得到表现。随着遗传算法的进行，解的质量会提高，适应度会增加，一旦找到具有令人满意的适应度值的解，终止遗传算法。

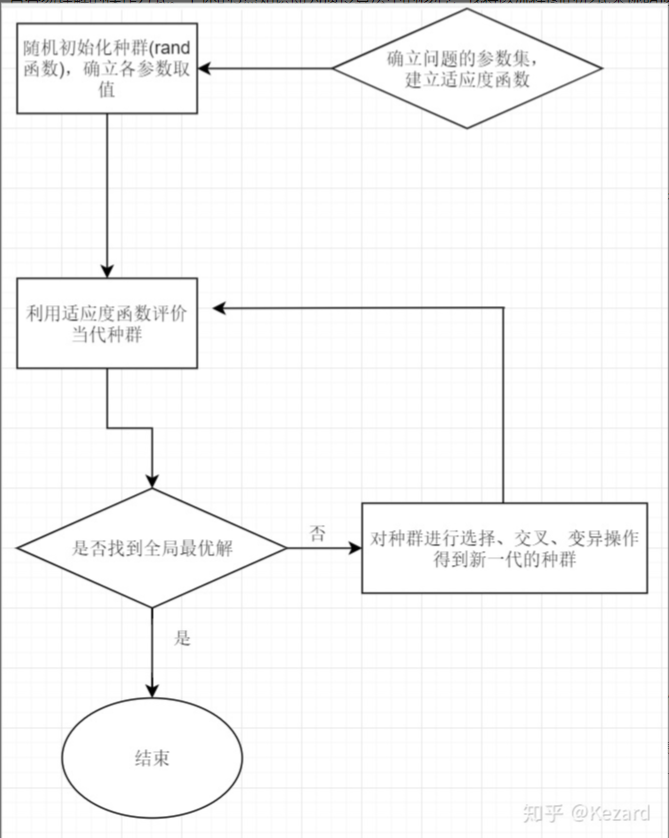
选择（Selection）：在计算出种群中每个个体的适应度后，使用选择过程来确定种群中的哪个个体将用于繁殖并产生下一代，具有较高值的个体更有可能被选中，并将其遗传物质传递给下一代。

仍然有机会选择低适应度值的个体，但概率较低。这样，就不会完全摒弃其遗传物质。

交叉（Crossover）：为了创建一对新个体，通常将从当前代中选择的双亲样本的部分染色体互换（交叉），以创建代表后代的两个新染色体。此操作称为交叉或重组

变异（Mutation）：突变操作的目的是定期随机更新种群，将新模式引入染色体，并鼓励在解空间的未知区域中进行搜索。

突变可能表现为基因的随机变化。变异是通过随机改变一个或多个染色体值来实现的。例如，翻转二进制串中的一位



# 四、实验步骤

## 题目一

* 题目一实现方法采用Python，利用Python中的Numpy科学计算包和Matplotlib绘图模块完成，从0开始实现遗传算法的各个环节
* 题目一的主要内容为使用遗传算法计算一个函数的最大值，题目中给定了目标函数的函数表达式和自变量的取值范围
* 目标函数表达式：f(X) = X+10sin(5X)+7sin(4X)
* X的取值范围为[0,10]
* 根据遗传算法的基本原理，使用Python语言编写选择、交叉、变异等一系列遗传算法子模块
* 根据老师在群中发送的遗传算法参数进行调参
* 种群数目：50 染色体二进制编码长度：20
* 最大进化代数：100 交叉概率：0.8 变异概率：0.1

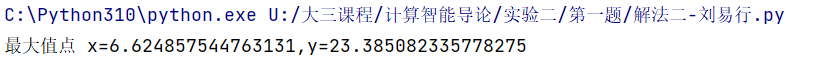
## 题目二

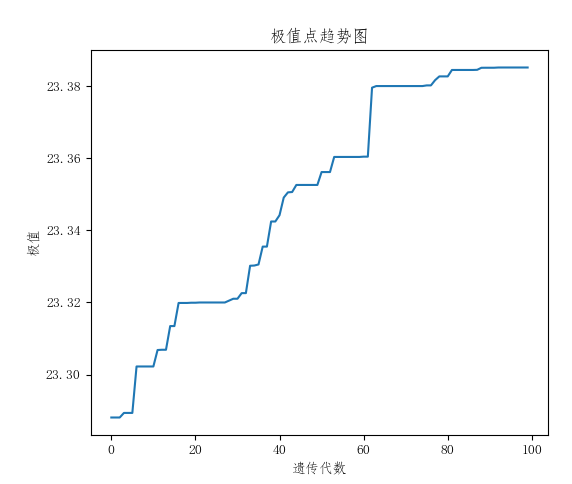
* 此前在题目一中已经通过纯Python方法实现了从0开始手撕实现遗传算法的过程
* 题目二中为求解多维度函数的最小值问题
* 从最大值问题到最小值问题，需要调整适应函数的撰写方式以及选择函数
* 题目二，通过查阅资料，选择使用目前机器学习中较为流行的Scikit-opt库中的GA方法实现，通过面向对象的方式调整参数即可
* scikit-opt为一个群体智能算法库，其中封装了诸如差分进化算法、遗传算法、粒子群算法、模拟退火算法、蚁群算法、免疫优化算法、鱼群算法等一系列算法
* 根据Scikit-opt库官方文档给出的实现步骤
* STEP1：定义我们的问题：多维函数 f(x)=Σ(xi\*xi)+20 x的维度为10 x的取值范围为[-20,20]
* STEP2：运行遗传算法
  + from sko.GA import GA
* STEP3:使用matplotlib库查看结果

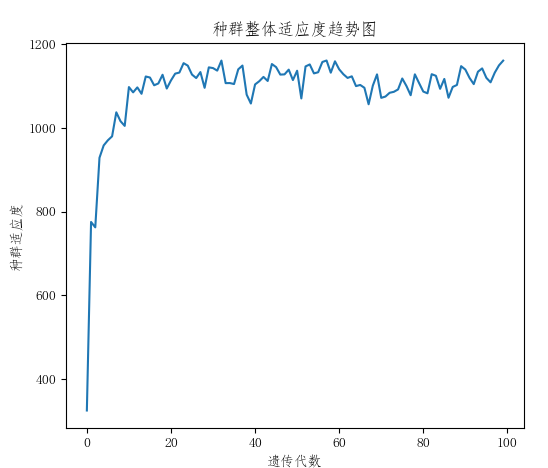
# 五、实验结果与分析

## 题目一

运行算法源代码文件，最大值点和最大值如下：

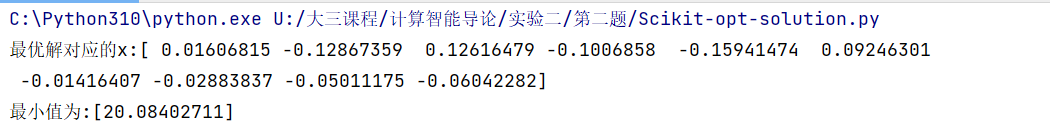


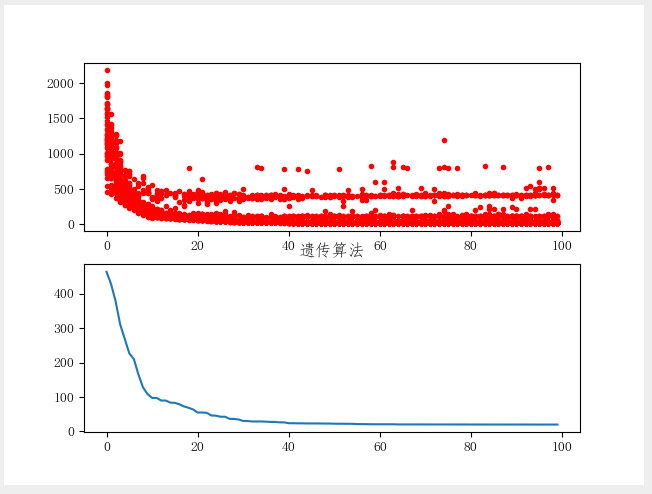




## 题目二

运行代码源文件，最小值点和对应最小值如下图：





# 六、实验程序代码

## 题目一- 题目一.py

*# 计算智能导论实验二 第二题  
# 2020240401-2020905896-刘易行  
  
# 遗传算法求解函数最大值  
# f(x) = x + 10sin(5x)+7sin(4x)  
# 0 <= x <= 10*import random  
import numpy as np  
import matplotlib.pyplot as plt  
  
*#定义遗传算法的参数*popSize=50 *#种群规模*epoches = 100 *# 遗传代数*DNAlength = 20 *#染色体编码长度*pjc = 0.8 *# 交叉概率*pby = 0.1 *# 变异概率*targetFunc = lambda x:x+10\*np.sin*(*5\*x*)* + 7\*np.sin*(*4\*x*)*min\_x = 0 *# 定义域下限*max\_x = 10 *# 定义域上限  
  
# matplotlib绘图参数*plt.rcParams*[*'font.sans-serif'*]* = *[*'FangSong'*] # 设置中文字体*plt.rcParams*[*'axes.unicode\_minus'*]* = False *# 支持特殊符号*def getChromosome*(*size,length*)*:  
 *# 生成 size个长度为length 的染色体 返回一个二维nparray* temp = *[]* for i in range*(*size*)*:  
 *# 生成长度为length的随机二进制列表，并存放到temp列表中* temp.append*([*random.randint*(*0,1*)* for \_ in range*(*length*)])* return temp  
  
def getAccuracy*(*minV,maxV,length*)*:  
 *# 计算搜索精度* return *(*maxV-minV*)* / *(*2\*\*length - 1*) # 套用精度计算公式*def ChromosomeDecode*(*DNAList,minV,accuracy*)*:  
 *# 染色体解码函数* decimal = int*(*''.join*([*str*(*i*)* for i in DNAList*])*,2*) # 二进制列表转为十进制整型* return minV + accuracy \* decimal  
  
def getFitness*(*x*)*:  
 return targetFunc*(*x*) # 计算适应度 适应度函数即为目标函数 此处计算最大值问题 不用额外操作*def select*(*chromosome\_list, fitness\_list*)*:  
 *"""  
 选择(轮盘赌算法)  
 :param chromosome\_list: 二维列表的种群  
 :param fitness\_list: 适应度列表  
 :return: 选择之后的种群列表  
 """* population\_fitness = np.array*(*fitness\_list*)*.sum*() # 种群适应度* fit\_ratio = *[*i / population\_fitness for i in fitness\_list*] # 每个个体占种群适应度的比例* fit\_ratio\_add = *[*0*] # 个体累计概率* for i in fit\_ratio:  
 fit\_ratio\_add.append*(*fit\_ratio\_add*[*len*(*fit\_ratio\_add*)* - 1*]* + i*) # 计算每个个体的累计概率，并存放到fit\_ratio\_add中* fit\_ratio\_add = fit\_ratio\_add*[*1:*] # 去掉首位的0* rand\_list = *[*random.uniform*(*0, 1*)* for \_ in chromosome\_list*] # 生成和种群规模相等的随机值列表，用于轮盘赌选择个体* rand\_list.sort*()* fit\_index = 0  
 new\_index = 0  
 new\_population = chromosome\_list.copy*()  
  
 # 开始个体选择* while new\_index < len*(*chromosome\_list*)*:  
 if rand\_list*[*new\_index*]* < fit\_ratio\_add*[*fit\_index*]*:  
 new\_population*[*new\_index*]* = chromosome\_list*[*fit\_index*]* new\_index = new\_index + 1  
 else:  
 fit\_index = fit\_index + 1  
 *# 结束个体选择* return new\_population  
  
  
def exchange*(*chromosome\_list, pc*)*:  
 *"""  
 交叉  
 :param chromosome\_list: 二维列表的种群  
 :param pc: 交叉概率  
 """* for i in range*(*0, len*(*chromosome\_list*)* - 1, 2*)*:  
 if random.uniform*(*0, 1*)* < pc:  
 c\_point = random.randint*(*0, len*(*chromosome\_list*[*0*])) # 随机生成交叉点* '''对第i位和i+1位进行交叉 start'''  
 exchanged\_list1 = *[]* exchanged\_list2 = *[]* exchanged\_list1.extend*(*chromosome\_list*[*i*][*0:c\_point*])* exchanged\_list1.extend*(*chromosome\_list*[*i + 1*][*c\_point:len*(*chromosome\_list*[*i*])])* exchanged\_list2.extend*(*chromosome\_list*[*i + 1*][*0:c\_point*])* exchanged\_list2.extend*(*chromosome\_list*[*i*][*c\_point:len*(*chromosome\_list*[*i*])])* '''对第i位和i+1位进行交叉 end'''  
  
 '''将新交叉后的染色体替换原染色体 start'''  
 chromosome\_list*[*i*]* = exchanged\_list1  
 chromosome\_list*[*i + 1*]* = exchanged\_list2  
 '''将新交叉后的染色体替换原染色体 end'''  
  
def mutation*(*chromosome\_list, pm*)*:  
 *# 变异* for i in range*(*len*(*chromosome\_list*))*:  
 if random.uniform*(*0, 1*)* < pm:  
 m\_point = random.randint*(*0, len*(*chromosome\_list*[*0*])* - 1*) # 随机生成变异点* chromosome\_list*[*i*][*m\_point*]* = chromosome\_list*[*i*][*m\_point*]* ^ 1  
  
  
def getBest*(*fitness\_list*)*:  
 *# 求当前种群中的最优个体* return fitness\_list.index*(*max*(*fitness\_list*))*def eliminate*(*fitness\_list*)*:  
 *# 淘汰个体（去除负值） 因为是求最大值问题* fit\_value = *[]* for i in range*(*len*(*fitness\_list*))*:  
 fit\_value.append*(*fitness\_list*[*i*]* if fitness\_list*[*i*]* >= 0 else 0.0*) # 将小于0的适应度置为0* return fit\_value  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 *# 主函数运行* genResults = *[] #暂存每一代的最优解* Fitness = *[] #存放每一代中的最高适应度* pop = getChromosome*(*popSize,DNAlength*) # 种群初始化* for \_ in range*(*epoches*)*:  
 accuracy = getAccuracy*(*min\_x,max\_x,DNAlength*) # 计算搜索精度* decodeList = *[*ChromosomeDecode*(*individual,min\_x,accuracy*)* for individual in pop*] # 解码之后的列表* fitList = *[*getFitness*(*decode\_i*)* for decode\_i in decodeList*] # 计算每个个体的适应度* fitList = eliminate*(*fitList*) # 淘汰 去除负值* genResults.append*([*decodeList*[*getBest*(*fitList*)]*,  
 fitList*[*getBest*(*fitList*)]]) # 保存每一代最优解，即适应度最高的个体* Fitness.append*(*np.array*(*fitList*)*.sum*()) # 保存每一代中的最高适应度和种群适应度* pop = select*(*pop.copy*()*, fitList*)* exchange*(*pop, pjc*) # 交叉* mutation*(*pop, pby*) # 变异* genResults.sort*(*key = lambda x:x*[*1*])* print*(*'最大值点 x={},y={}'.format*(*genResults*[*-1*][*0*]*, genResults*[*-1*][*1*]))* X = *[*generation\_i for generation\_i in range*(*epoches*)]* Y1 = *[*genResults*[*generation\_i*][*1*]* for generation\_i in range*(*epoches*)]* Y2 = *[*Fitness*[*generation\_i*]* for generation\_i in range*(*epoches*)]* fig1 = plt.figure*(*'figure', figsize=*(*13, 5*))*.add\_subplot*(*121*)* fig1.plot*(*X, Y1*)* fig2 = plt.figure*(*'figure', figsize=*(*13, 5*))*.add\_subplot*(*122*)* fig2.plot*(*X, Y2*)* fig1.set\_title*(*'极值点趋势图'*)* fig1.set\_xlabel*(*"遗传代数"*)* fig1.set\_ylabel*(*"极值"*)* fig2.set\_title*(*'种群整体适应度趋势图'*)* fig2.set\_xlabel*(*"遗传代数"*)* fig2.set\_ylabel*(*"种群适应度"*)* plt.show*()*

## 题目二-ScikitSolutionGA.py

*# 计算智能导论实验二 第二题  
# 2020240401-2020905896-刘易行*import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sko.GA import GA  
  
*# matplotlib绘图参数*plt.rcParams*[*'font.sans-serif'*]* = *[*'FangSong'*] # 设置中文字体*plt.rcParams*[*'axes.unicode\_minus'*]* = False *# 支持特殊符号*def f*(*x*)*:  
 *# 目标函数 求最小值问题* return 20 + np.sum*(*np.power*(*x, 2*))  
  
# 调用Scikit-OPT中的GA算法 实例化*ga = GA*(*func=f, n\_dim=10, size\_pop=50, max\_iter=100, lb=*[*-20*]*\*10, ub=*[*20*]*\*10,precision=1e-9,prob\_mut = 0.01*)*best\_x, best\_y = ga.run*()*print*(*'最优解对应的x:{}'.format*(*best\_x*))*print*(*"最小值为:{}".format*(*best\_y*))  
  
# 绘制图线 查看结果*Y\_history = pd.DataFrame*(*ga.all\_history\_Y*)*fig, ax = plt.subplots*(*2, 1*)*ax*[*0*]*.plot*(*Y\_history.index, Y\_history.values, '.', color='red'*)*Y\_history.min*(*axis=1*)*.cummin*()*.plot*(*kind='line'*)*plt.title*(*'遗传算法'*)*plt.show*()*