模拟退火

单变量函数优化

from random import \*  
from math import \*  
import numpy as np  
import matplotlib.pyplot as plt  
f = lambda x: x \*\* 2 #目标函数  
  
#模拟退火算法有总共5个要素  
t=20000#初始温度，这是一个较高的值，用于驱动算法开始时的随机搜索  
end\_t=10\*\*(-7)#结束温度，当算法的温度降至这个值以下时，它会停止并输出当前找到的最优解  
alpha=0.98#衰减系数，用于控制温度下降的速度  
mirkiv\_len=1000#马尔可夫链长度，这定义了马尔可夫过程迭代的次数，即在每次迭代中，我们都会根据当前解产生一个新的可能解  
s=3#初始解，这是算法开始搜索的最初始点  
  
#给定求函数最优值考虑到的区间，定义了搜索空间的范围  
a=0  
b=9  
  
#外层循环：退火,直到温度小于等于结束温度  
x=[]  
y=[]  
z=0  
while t>end\_t:  
 #内层循环：马尔可夫扰动  
 ins=0 #记录迭代次数  
 while ins<=mirkiv\_len:  
 if a<s<b:  
 snew=s+0.01\*(2\*random()-1)  
 elif s<=a:  
 snew=s+0.01\*random()  
 elif s>=b:  
 snew=s-0.01\*random()  
 if f(snew)<f(s):  
 s=snew  
 else:  
 p=random()  
 if p<e\*\*((f(s)-f(snew))/t):  
 s=snew  
 else:  
 pass  
 ins+=1  
 z+=1  
 x.append(z)  
 y.append(f(s))  
 t=t\*alpha  
  
#这里是原来的函数图像，读者可以去掉#号自己查看  
x2=np.linspace(0,9,10000)  
y2=f(x2)  
#plt.figure()  
#plt.plot(x2,y2)  
#plt.show()  
  
y1=[min(y2) for i in range(0,len(y))]  
print(f(s))#输出了最小值  
  
plt.figure()  
plt.plot(x,y,'b-',label='SA')#模拟退火算法的搜索过程用蓝线表示  
plt.plot(x,y1,'r-',label='min(f)')#最小值用红线表示  
plt.legend()  
plt.show()

多变量函数优化

demo\_func = lambda x: x[0] \*\* 2 + (x[1] - 0.05) \*\* 2 + x[2] \*\* 2#目标函数  
  
from sko.SA import SA  
sa = SA(func=demo\_func, x0=[1, 1, 1], T\_max=1, T\_min=1e-9, L=300, max\_stay\_counter=150,lb=0,ub=9)  
'''  
 func: 要优化的函数。  
 x0: 初始解。  
 T\_max: 初始温度。  
 T\_min: 结束温度。  
 L: 每个温度下的迭代次数。  
 max\_stay\_counter: 当一个解在连续max\_stay\_counter次迭代中没有改进时，算法将降低温度。  
 lb: 每个维度的下界。  
 ub: 每个维度的上界。  
'''  
  
#最优解best\_x  
# 最优解对应的函数值best\_y  
best\_x, best\_y = sa.run()  
print('best\_x:', best\_x, 'best\_y', best\_y)  
  
import matplotlib.pyplot as plt  
import pandas as pd  
  
#绘制sa.best\_y\_history的历史记录，也就是函数值随着算法的迭代逐渐变小的过程  
# y轴是函数值，x轴是算法的迭代次数  
plt.plot(pd.DataFrame(sa.best\_y\_history).cummin(axis=0))  
plt.show()