

Guide of Python

内容概要： 数学建模算法

创建时间： 2022/4/7 13:41

更新时间： 2022/4/17 15:27

作者： TwinkelStar

模拟退火算法

Simulated Annealing Algorithm

1、操作系统相关环境

1) 硬件环境：

- 电脑

2) 软件环境：

- Python3.7(向下兼容 Python3)(程序设计语言)
- Numpy1.19.5(兼容大部分版本)(科学计算库)

3) 操作系统(2 选 1)：

- Windows7
- Windows10
- Windows11

2、模拟退火算法

模拟退火算法(Simulated Annealing, SA)的思想借鉴于固体的退火原理，当固体的温度很高的时候，内能比较大，固体的内部粒子处于快速无序运动，当温度慢慢降低的过程中，固体的内能减小，粒子的慢慢趋于有序，最终，当固体处于常温时，内能达到最小，此时，粒子最为稳定。模拟退火算法便是基于这样的原理设计而成。

1) 基本原理

模拟退火算法从某一较高的温度出发，这个温度称为初始温度，伴随着温度参数的不断下降，算法中的解趋于稳定，但是，可能这样的稳定解是一个局部最优解，此时，模拟退火算法中会以一定的概率跳出这样的局部最优解，以寻找目标函数的全局最优解。如上图中所示，若此时寻找到了 A 点处的解，模拟退火算法会以一定的概率跳出这个解，如跳到了 D 点重新寻找，这样在一定程度上增加了寻找到全局最优解的可能性。

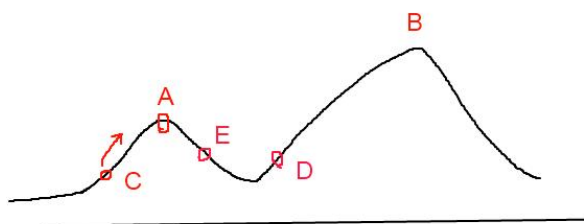


图 1.模拟退火算法寻找最优解

fig1. Simulated annealing algorithm to find the optimal solution

3、程序步骤：

①随机挑选一个单元 k ，并给它一个随机的位移，求出系统因此而产生的能量变化 ΔE ；

②若 $\Delta E_k \leq 0$ ，该位移可采纳，而变化后的系统状态可作为下次变化的起点；

若 $\Delta E_k > 0$ ，位移后的状态可采纳的概率为：

$$P_k = \frac{1}{1 + e^{-\Delta E_k / T}} \quad (1)$$

式 (1) 中 T 为温度，然后从 $(0, 1)$ 区间均匀分布的随机数中

挑选一个数 R ，若 $R < P_k$ ，则将变化后的状态作为下次的起点；否则，将变化前的状态作为下次的起点。

③转第①步继续执行，直到达到平衡状态为止。

程序框图如图 2 所示：

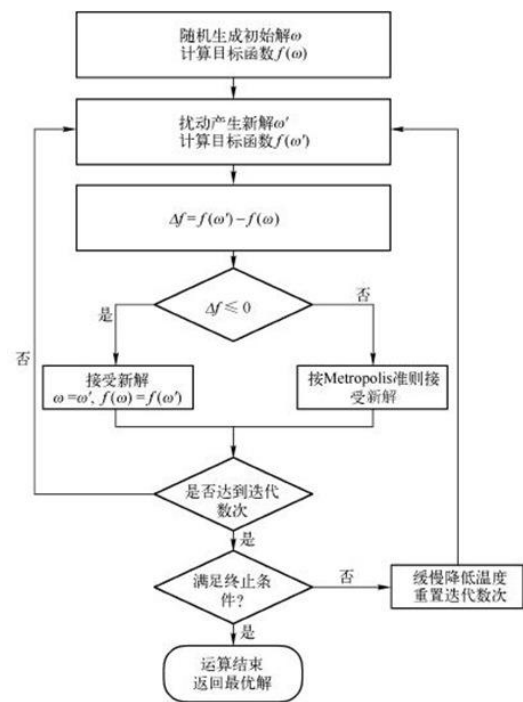


图 2.程序框图
fig2. Program flowchart

4、例题与程序设计

1) 例题

求解函数（2）的最小值， x ， y 的取值范围为 $[-5, 5]$ 。

$$f(x,y) = 4x^2 - 2.1x^4 + \frac{x^6}{3} + xy - 4y^2 + 4y^4 \tag{2}$$

2) 代码实现

编写目标函数值的函数代码，如图 3 所示：

```
def E(x, y):
    """
    目标函数 x的取值范围为 -5 < x < 5

    Parameters
    -----
    x : 实数
        x1的值,
    y : 实数
        x2的值,

    Returns
    -----
    res : 实数
        返回目标函数值.

    """
    res= 4*x**2-2.1*x**4+(x**6)/3+x*y-4*y**2+4*y**4
    return res
```

图 3.目标函数代码
fig3. Objective function code

随机生成扰乱 x_1 , x_2 , 如图 4 所示:

```
def randValue(t, x, y):
    """
    扰乱生成x1, x2的值

    Parameters
    -----
    t : 实数
        当前的温度.
    x : 实数
        x1的值,
    y : 实数
        x2的值,

    Returns
    -----
    x_ : 实数
        x1的新值,
    y_ : 实数
        x2的新值,

    """
    while 1:
        x_ = x + t * (np.random.rand(1)[0] - np.random.rand(1)[0])
        y_ = y + t * (np.random.rand(1)[0] - np.random.rand(1)[0])
        if x_ >= -5 and x_ <= 5 and y_ >= -5 and y_ <= 5:
            break
    return x_, y_
```

图 4.目标函数代码
fig4. Objective function code

随机下降法则，防止函数陷入局部最优，如图 5 所示:

```
def select(t ,e, e_):
    """
    是否进行下降法则

    Parameters
    -----
    t : 实数
        当前温度,
    e : 实数
        当前函数值,
    e_ :实数
        新的函数值,

    Returns
    -----
    int
        是否选择 选择为1 拒绝为0.

    """
    if e_ <= e:
        return 1
    else:
        p = np.power(np.e, -(e_ - e) / t)
        if p > np.random.rand(1)[0]:
            return 1
        else:
            return 0
```

图 5.随机下降法则
fig5. Random descent rule

主函数代码如图 6 所示：

```
if __name__ == "__main__":
    t0 = 100 # 初始温度
    t = t0 # 当前温度
    alpha = 0.99 # 降温系数
    tf = 0.01 # 最终温度
    x = (np.random.rand(1,100) * 10) - 4 # 随机生成一组解 为-6 到 6 可以取到 5
    y = (np.random.rand(1,100) * 10) - 4 # 随机生成一组解 为-6 到 6 可以取到 5

    while t > tf:
        for i in range(t0):
            e = E(x[0, i], y[0, i])
            x_, y_ = randValue(t, x[0, i], y[0, i])
            e_ = E(x_, y_)
            if select(t, e, e_):
                x[0, i] = x_
                y[0, i] = y_
        e_result = E(x, y)
        e_best = np.min(e_result)
        e_inx = np.argmin(e_result)

        x_value = x[0, e_inx]
        y_value = y[0, e_inx]

        t = t * alpha
        print("e_best", e_best)
        print("x_value", x_value)
        print("y_value", y_value)
```

图 6.主函数
fig6. Main

函数符号说明如表 1 所示：

表 1.符号描述图表
table1.Symbol Description

Symbol	Description
t0	初始温度
t	当前温度
alpha	降温系数
tf	最终温度

3）代码调用

运行代码 `refire.py` 直接得出结果，如图 7 所示：

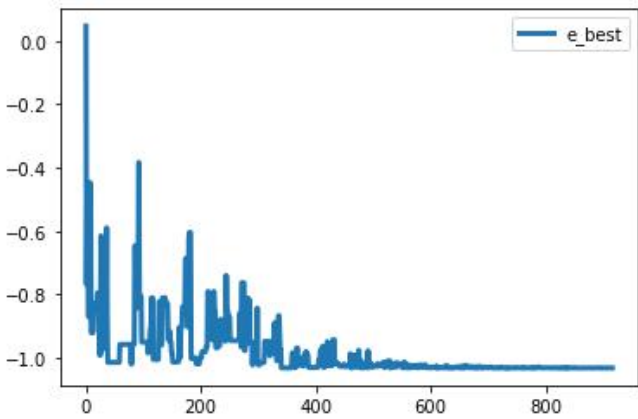
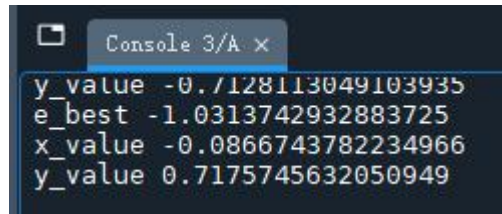


图 7.主函数
fig7. Simulated annealing results

函数返回结果如图 8 所示：



```
y_value -0.7128113049103935
e_best -1.0313742932883725
x_value -0.0866743782234966
y_value 0.7175745632050949
```

图 8.函数返回结果
fig8. Function returns result

从结果中可知，结果 800 次的迭代之后，函数值收敛。
最小值为-1.0313742932883725，x 的取值为
-0.0866743782234966，y 的取值为：0.7175745632050949。