# 最优化算法——随机优化算法

随机优化算法的思想是抛开求导，采用撒豆成兵的方法，即只要在给定的取值范围内，自变量取值足够多，就能找到最值的近似解。

以求解最小值为例，具体算法如下：

首先，用遍历首先确定函数极值点的大致区间范围。

循环开始：

step1：在这个区间内随机产生自变量$x$，带入函数$y=f(x)$计算出函数值。（多变量的话，只需要生产多个自变量即可）

step2（如没有约束条件，则省略）：判断约束条件。满足则跳转 step3，不满足则跳转 step1

step3：将每次循环中计算出函数值$y$存放在一个向量$Y$中，并计算向量$Y$的最小值$\min y$，并将最小值$\min y$，存放在另外一个向量$\min Y$中。（这一步是每一次循环都要做的，每一次循环向量$Y$就增加一个值，$\min Y$中存放的是每次循环中向量$Y$的最小值，因为向量$Y$的元素在增加，$\min Y$中的元素也是每循环一次就要增加一个，且每增加一个要么和前一个的值相同，要么比前一个的值小）

step4：如果是第一次循环当前循环，记录此次循环 step1 中产生的自变量$x$，即$x_0 \leftarrow x$；如果是后续循环，那么进行如下操作，如果$\min Y$中最新的一个值比前一个值小，那么记录此次循环 step1 中产生的自变量$x$，即$x_0 \leftarrow x$，计数值$s$置 0；反之，则计数值$s$增 1。

step5：如果计数值小于阈值，即$s<N$，则循环从 step1 重新开始，反之，则结束循环，输出 step4 中最后一次更新的$x_0$和$\min Y$中的最后一个元素。（最后一次更新的$x_0$和$\min Y$中的最后一个元素就是函数的最小值对应的自变量和函数的最小值）

求解函数$y=x^4-\frac{32}{3}x^3+34x^2-35x+5, x\epsilon[0,6]$上的最小值

举个例子（以下函数值带入计算都是我瞎说的，带入计算太麻烦了，具体请自行带入函数算一下）

初始化计数值 s=0，阈值 N=10

第一次循环

x 在 0~6 随机产生为 0.1，带入函数计算 y=4.9，然后将 4.9 存在 Y 中，那么 Y=[4.9]，然后计算向量 Y 的最小值 miny=4.9，那么将 4.9 存入向量 minY 中，minY=[4.9]，因为是第一次循环要将 0.1 保存下来，赋值给 $x_0$，第一次循环结束。

第一次循环结束的结果：Y=[4.9],minY=[4.9], $x_0$=0.1,s=0

第二次循环

x 在 0~6 随机产生为 5.3，带入函数计算 y=6.2，然后将 6.2 存在 Y 中，那么 Y=[4.9,6.2]，然后计算向量 Y 的最小值 miny=4.9，那么将 4.9 存入向量 minY 中，minY=[4.9,4.9]，因为是第二次循环 minY 中新增元素和第一次循环相比没有变化，所以计数值 s 增 1，第二次循环结束。

第二次循环结束的结果：Y=[4.9,6.2],minY=[4.9,4.9] , $x_0$=0.1,s=1

第三次循环

x 在 0~6 随机产生为 2.8，带入函数计算 y=3.6，然后将 3.6 存在 Y 中，那么 Y=[4.9,6.2,3.6]，然后计算向量 Y 的最小值 miny=3.6，那么将 3.6 存入向量 minY 中，minY=[4.9,4.9,3.6]，因为是第二次循环 minY 中新增元素和第一次循环相比减小，意味着本次循环 x=2.8 计算出的函数值比前两次循环更小，所以计数值 s 置 0，同时，2.8 赋值给 $x_0$，第三次循环结束。

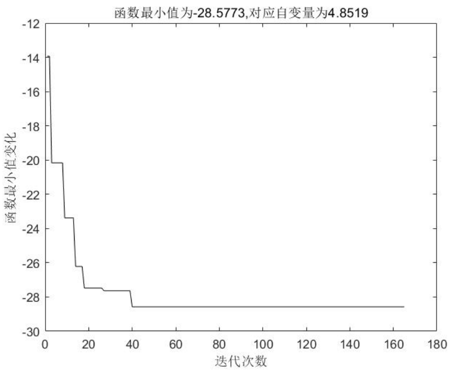
第三次循环结束的结果：Y=[4.9,6.2,3.6],minY=[4.9,4.9,3.6] , $x_0$=2.8,s=0

第四次循环

x 在 0~6 随机产生为 3.4，带入函数计算 y=5.7，然后将 5.7 存在 Y 中，那么 Y=[4.9,6.2,3.6]，然后计算向量 Y 的最小值 miny=3.6，那么将 3.6 存入向量 minY 中，minY=[4.9,4.9,3.6,3.6]，因为是第四次循环 minY 中新增元素和第三次循环相比没有变化，所以计数值 s 增 1，第四次循环结束。

第四次循环结束的结果：Y=[4.9,6.2,3.6,5.7],minY=[4.9,4.9,3.6,3.6] , $x_0$=2.8,s=1

如果这个函数的最小值就是 3.6，那继续循环下去，minY 中增加的值就是 3.6 不会变，就像第四次循环一样，同时，计数值 s 在不断增加，当 s 超过阈值 N=10 的时候，默认找到最小值结束循环。minY 中每次循环增加的元素要么不变，要么减小。最后将 minY 画出来就是一个阶梯收敛图，**这个图很重要！比赛时如果用到随机优化算法，一定要画出阶梯收敛图！**



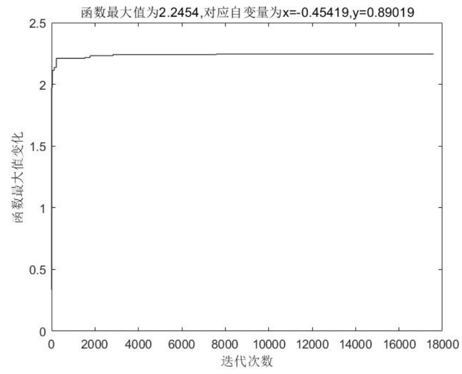

```
s=0;N=100;i=1;%初始化计数值、阈值和循环变量
while(s<N)
    x=6*rand;%rand是产生0-1上的随机数
    y=f(x);%带入函数计算
    Y(i)=y;%将当前循环计算出函数值存入Y向量中
    miny=min(Y);%计算当前循环Y向量最小值
    minY(i)=miny;%将当前循环Y向量最小值存入向量minY中
    if(i==1)
        x0=x;%第一次循环保存自变量
    else
        if(minY(i)<minY(i-1))%minY最新一次更新的值比前一次循环小
            %则认为找到一个更小的值，保存此次循环产生的自变量
            %同时计数值置0
            s=0;x0=x;
        else
            s=s+1;%反之，计数值增1
        end
    end
    i=i+1;%循环次数增1
end
```

当阈值 N 增大时，计算结果的精度会增加，这里的函数我编成 function 文件了，对于函数表达式未知的优化问题也可以用该算法求解，例如 2020A 第三问。

多变量带约束的情况

计算二元函数 $z=x^2+2y^2-x, x^2+y^2\leqslant 1, y\geqslant 0$ 的最大值（理论计算最大值点 $\left(-\frac{1}{2},\frac{\sqrt{3}}{2}\right)$）

直接 $x$ 和 $y$ $[-1,1]$ 和 $[0,1]$ 范围内搜索，在加 if($x^2+y^2\leqslant 1$)条件判断约束


函数最大值为2.2454,对应自变量为x=-0.45419,y=0.89019
函数最大值变化
迭代次数
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
0
0.5
1
1.5
2
2.5