****蒙特卡洛算法实验****

这么看来蒙特卡洛方法的理论支撑其实是概率论或统计学中的大数定律。基本原理简单描述是先大量模拟，然后计算一个事件发生的次数，再通过这个发生次数除以总模拟次数，得到想要的结果。下面我们以三个经典的小实验来学习下蒙特卡洛算法思想。

****1.计算圆周率pi（π）值****

实验原理：在正方形内部有一个相切的圆，圆面积/正方形面积之比是(PixRxR)/(2Rx2R)= Pi/4。在这个正方形内随机产生n个点，假设点落在圆内的概率为P，那么P=圆面积/正方形面积，则P= Pi/4。如何计算点落在圆内的概率P？可以计算点与中心点的距离，判断是否落在圆的内部，若这些点均匀分布，用M表示落到圆内投点数 ， N表示总的投点数，则圆周率Pi=4P=4xM/N。

****实验步骤：****

（1）将圆心设在原点（0，0），以R为半径形成圆，则圆面积为PixRxR

（2）将该圆外接正方形, 坐标为（-R，-R）（R，-R）（R， R）（-R，R），则该正方形面积为R\*R

（3）随即取点（X，Y），使得-R <=X<=R并且-R <=Y<=R，即点在正方形内

（4）通过公式 XxX+YxY<= RxR判断点是否在圆周内（直角三角形边长公式）。

（5）设所有点（也就是实验次数）的个数为N，落在圆内的点（满足步骤4的点）的个数为M，则P=M/N，于是Pi=4xM/N。

（6）运行结果为3.143052

def cal\_pai\_mc(n=1000000):

r = 1.0

a, b = (0.0, 0.0)

x\_neg, x\_pos = a - r, a + r

y\_neg, y\_pos = b - r, b + r

m = 0

for i in range(0, n+1):

x = random.uniform(x\_neg, x\_pos)

y = random.uniform(y\_neg, y\_pos)

if x\*\*2 + y\*\*2 <= 1.0:

m += 1

return (m / float(n)) \* 4

****2.计算函数定积分值****

实验原理：若要求函数f(x)从a到b的定积分，我们可以用一个比较容易算得面积的矩型包围在函数的积分区间上（假设其面积为Area），定积分值其实就是求曲线下方的面积。随机地向这个矩形框里面投点，统计落在函数f(x)下方的点数量占所有点数量的比例为P，那么就可以据此估算出函数f(x)从a到b的定积分为Area×P。此处我们将a和b设为0和1，函数f(x)=x2。

运行结果为0.333749

def cal\_integral\_mc(n = 1000000):

x\_min, x\_max = 0.0, 1.0

y\_min, y\_max = 0.0, 1.0

m = 0

for i in range(0, n+1):

x = random.uniform(x\_min, x\_max)

y = random.uniform(y\_min, y\_max)

# x\*x > y 表示该点位于曲线的下面。

if x\*x > y:

m += 1

#所求的积分值即为曲线下方的面积与正方形面积的比

return m / float(n)

****3.计算函数极值，可避免陷入局部极值****

实验原理：极值是“极大值” 和 “极小值”的统称。如果一个函数在某点的一个邻域内处处都有确定的值，函数在该点的值大于或等于在该点附近任何其他点的函数值，则称函数在该点的值为函数的“极大值”。如果函数在该点的值小于或等于在该点附近任何其他点的函数值，则称函数在该点 的值为函数的“极小值”。此处在区间[-2,2]上随机生成一个数，求出其对应的y，找出其中最大值认为是函数在[-2,2]上的极大值。

运行结果发现极大值185.1204262706596, 极大值点为1.5144491499169481

def cal\_extremum\_mc(n = 1000000):

y\_max = 0.0

x\_min, x\_max = -2.0, 2.0

y = lambda x:200\*np.sin(x)\*np.exp(-0.05\*x)#匿名函数

for i in range(0, n+1):

x0 = random.uniform(x\_min, x\_max)

if y(x0) > y\_max:

y\_max = y(x0)

x\_max = x0

return y\_max, x\_max