模型-计算与模拟-计算机模拟-蒙特卡洛模拟【hxy】

1. 模型名称

蒙特卡洛模拟（Monte Carlo Simulation)

2. 适用范围

对于那些由于计算过于复杂而难以得到解析解或者根本没有解析解的问题，蒙特卡罗模拟是一种有效的求出数值解的方法

3. 形式

所求解的问题同一定的概率模型相联系

4. 求解过程

4.1 求解π

4.1.1 方法

边长为1的正方形内部有一个相切的圆，它们的面积之比是$\pi/4$

现在, 在这个正方形内部, 随机产生100000个点 (即100000个坐标对(x, y))，计算它们与中心点的距离，从而判断是否落在圆的内部。如果这些点均匀分布，那么圆内的点应该占到所有点的$\pi/4$， 因此将这个比值乘以4，就是π的值

4.1.2 Matlab

- 核心代码

```
sim = monte_carlo(100000, @gen, @cond) * 4.0;
function Point = gen()
    Point = rand([1,2]);
end
function ans = cond(x)
    ans = sqrt((x(1)-0.5).^2+(x(2)-0.5).^2) < 0.5;
end
```

- 完整代码

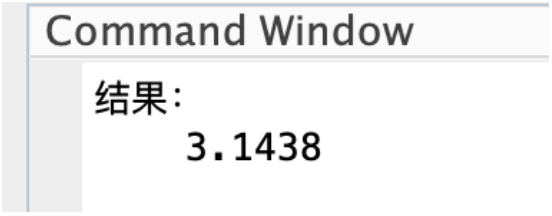
  monte_carlo.m

```
function prob = monte_carlo(rand_times, gen, cond)
    cnt = 0;
    for i=1:rand_times
        % 随意产生一个点
        x=gen();
        % 若在范围内，数量加一
        if cond(x)
            cnt = cnt + 1;
        end
    end
    % 计算落在范围内的概率
    prob = 1.0 * cnt / rand_times;
end
```

  pi_montecarlo.m

```
% 求解pi
clc, clear
sim = monte_carlo(100000, @gen, @cond) * 4.0;
disp('结果：');
disp(sim);
function Point = gen()
    Point = rand([1,2]);
end
function ans = cond(x)
    ans = sqrt((x(1)-0.5).^2+(x(2)-0.5).^2) < 0.5;
end
```

- 结果



4.1.3 Python

- 核心代码

```
def generate():
    return [random.random(), random.random()]

def judge(x):
    return math.sqrt((x[0]-0.5)*(x[0]-0.5)+(x[1]-0.5)*(x[1]-0.5)) < 0.5
```

- 完整代码

  pi_python.py

```
'''
求解pi
修改generate和judge两个函数对象即可
根据需要对输出的概率进行进一步加工
'''
import random
import math
class monte_carlo(object):
    def __init__(self, rand_times, gen, cond):
        self.rand_times = rand_times
        self.generator = gen
        self.condition = cond
    def getprob(self):
        cnt = 0
        for i in range(self.rand_times):
            x = self.generator()
            if (self.condition(x)):
                cnt = cnt + 1
        return 1.0 * cnt / self.rand_times


def generate():
    return [random.random(), random.random()]

def judge(x):
    return math.sqrt((x[0]-0.5)*(x[0]-0.5)+(x[1]-0.5)*(x[1]-0.5)) < 0.5

if __name__ == "__main__":
    sim = monte_carlo(100000, generate, judge)
```

```
    print (sim.getprob() * 4.0)
```

- 结果

```
>>>
= RESTART: /Users/xinyuanhe/Desktop/working/2021美赛/模型/【正式】模型-计算与模拟-计算机模拟-蒙特卡洛模拟【hxy】/pi_python.py
3.14744
```

4.1.4 C++

- 核心代码

```
struct Point{ double x, y; };
struct pi_gen {
    Point operator () () {
        return (Point){double(rand() % 1000) / 1000.0, double(rand() % 1000) / 1000.0};
    }
};
struct judge {
    bool operator () (const Point &x) {
        return sqrt((x.x-0.5)*(x.x-0.5)+(x.y-0.5)*(x.y-0.5)) < 0.5;
    }
};
```

- 完整代码

  monte_carlo.h

```
#ifndef MONTE_CARLO_H
#define MONTE_CARLO_H

template<class Point, int rand_times, class gen, class cond>
class Monte_carlo {
/*
 * dimension 维度
 * rand_times 模拟次数
 * gen 生成点的方式
 * cond 计数条件
 */
private:
    gen generator;
    cond condition;

public:
    double getprob() {
        int cnt = 0;
        for (int i = 1; i <= rand_times; ++i) {
            Point x = generator();
            if (condition(x)) ++cnt;
```

```
        }
        return double(cnt) / rand_times;
    }
};

#endif //MONTE_CARLO_H
```

pi.cpp

```
#include <cmath>
#include <iostream>
#include <cstdlib>
#include <iomanip>
#include "monte_carlo.h"
using namespace std;

struct Point{ double x, y; };
struct pi_gen {
    Point operator () () {
        return (Point){double(rand() % 1000) / 1000.0, double(rand() % 1000) /
1000.0};
    }
};
struct judge {
    bool operator () (const Point &x) {
        return sqrt((x.x-0.5)*(x.x-0.5)+(x.y-0.5)*(x.y-0.5)) < 0.5;
    }
};

Monte_carlo<Point, 10000000, pi_gen, judge> sim;

int main()
{
    cout << setprecision(9);
    cout << sim.getprob()*4 << endl;
    return 0;
}
```

- 结果

```
/Users/xinyuanhe/Desktop/working/2021美赛/模型/【正式】模型-计算与模拟-计算机模拟-蒙特卡洛模拟【hxy】/pi_cpp/cmake-build-debug/pi_cpp
3.1416056

Process finished with exit code 0
```

4.2 求解积分

4.2.1 方法

在空间里取足够多随机的的点, 统计点在积分范围内的概率, 再乘以空间的大小, 就可以算出积分的值

4.2.2 Matlab

- 核心代码

```
sim = monte_carlo(100000, @gen, @cond);
function Point = gen()
    Point = rand([1,2]);
end
function ans = cond(x)
    ans = x(2) < x(1)^2;
end
```

- 完整代码

  monte_carlo.m

```
function prob = monte_carlo(rand_times, gen, cond)
    cnt = 0;
    for i=1:rand_times
        x=gen();
        if cond(x)
            cnt = cnt + 1;
        end
    end
    prob = 1.0 * cnt / rand_times;
end
```

  integral_montecarlo.m

```
% 求解积分
clc, clear
sim = monte_carlo(100000, @gen, @cond);
disp('结果: ');
disp(sim);
function Point = gen()
    Point = rand([1,2]);
end
function ans = cond(x)
    ans = x(2) < x(1)^2;
end
```

- 结果

4.2.3 Python

- 核心代码

```
def generate():
    return [random.random(), random.random()]

def judge(x):
    return x[1] < x[0]*x[0]
```

- 完整代码

  integral_python.py

```
'''
求解积分
修改generate和judge两个函数对象即可
根据需要对输出的概率进行进一步加工
'''
import random
import math
class monte_carlo(object):
    def __init__(self, rand_times, gen, cond):
        self.rand_times = rand_times
        self.generator = gen
        self.condition = cond
    def getprob(self):
        cnt = 0
        for i in range(self.rand_times):
            x = self.generator()
            if (self.condition(x)):
                cnt = cnt + 1
        return 1.0 * cnt / self.rand_times

def generate():
    return [random.random(), random.random()]

def judge(x):
    return x[1] < x[0]*x[0]

if __name__=="__main__":
    sim = monte_carlo(100000, generate, judge)
    print (sim.getprob())
```

- 结果

```
>>>
= RESTART: /Users/xinyuanhe/Desktop/working/2021美赛/模型/【正式】模型-计算与模拟-计算机模拟-蒙特卡洛模拟【hxy】/integral_python.py
0.33342
```

4.2.4 C++

- 核心代码

```
struct Point{ double x, y; };
struct pi_gen {
    Point operator () () {
        return (Point){double(rand() % 1000) / 1000.0, double(rand() % 1000) / 1000.0};
    } // 积分区间[0,1]
};

struct judge {
    bool operator () (const Point &x) {
        return x.y < x.x*x.x;
    } // 对y = x^2积分
};
```

- 完整代码

  monte_carlo.h

```
#ifndef MONTE_CARLO_H
#define MONTE_CARLO_H

template<class Point, int rand_times, class gen, class cond>
class Monte_carlo {
/*
 * dimension 维度
 * rand_times 模拟次数
 * gen 生成点的方式
 * cond 计数条件
 */
private:
    gen generator;
    cond condition;

public:
    double getprob() {
        int cnt = 0;
        for (int i = 1; i <= rand_times; ++i) {
            Point x = generator();
            if (condition(x)) ++cnt;
        }
```

```
        return double(cnt) / rand_times;
    }
};

#endif //MONTE_CARLO_H
```

integral.cpp

```
#include <iostream>
#include <cstdlib>
#include <iomanip>
#include "monte_carlo.h"
using namespace std;

struct Point{ double x, y; };
struct pi_gen {
    Point operator () () {
        return (Point){double(rand() % 1000) / 1000.0, double(rand() % 1000) /
1000.0};
    } // 积分区间[0,1]
};

struct judge {
    bool operator () (const Point &x) {
        return x.y < x.x*x.x;
    } // 对y = x^2积分
};

Monte_carlo<Point, 10000000, pi_gen, judge> sim;

int main()
{
    cout << setprecision(9);
    cout << sim.getprob() << endl;
    return 0;
}
```

- 结果

```
/Users/xinyuanhe/Desktop/working/2021美赛/模型/【正式】模型-计算与模拟-计算机模拟-蒙特卡洛模拟【hxy】/integrap_cpp/cmake-build-debug/in
0.3332601

Process finished with exit code 0
```

5. 参考资料

1. 蒙特卡罗方法入门-5个案例
2. 数模官网