计算机模拟 Homework6

汪奕晨 3180105843

数学科学学院数学与应用数学专业

1 问题描述

使用 Monte-Carol 积分计算

$$\int_0^1 \frac{1}{2}\sin(\frac{1}{x^2}) + \frac{1}{2}$$

并且在 $\epsilon = 0.01, \delta = 0.01$ 的情况下使用 chebyshev 方法和 norm-distribution 方法计算所需的抽样次数。

2 设计思路

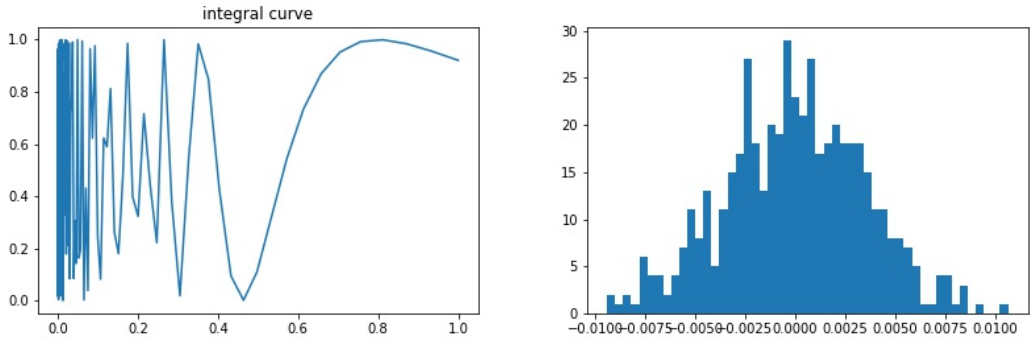
通过 equation (1,2) 可以分别用 chebyshev 方法和 norm-distribution 方法估计在给定 ϵ, δ 下所需的抽样次数。其中 Φ^{-1} 为标准正态分布逆函数，使用 python 实现时，可以调用 scipy.stats.norm.ppf 计算。

$$n_c(\epsilon, \delta) = \lceil \frac{1}{4\delta\epsilon^2} \rceil \tag{1}$$

$$n_n(\epsilon, \delta) = \left\lceil \left[\frac{\Phi^{-1}(1 - \frac{\delta}{2})}{2\varepsilon} \right]^2 \right\rceil . \tag{2}$$

3 模拟结果与分析

chebyshev 方法和 norm-distribution 方法估计的采样次数分别为 $250000, 16384$, 我们采用 $n = 17000$ 进行测试



(a) Integral Curve (b) batch test resule when $N = 17000, B = 500$

Figure 1: Performance of Simulation

如 Figure (1(b)), 只有少量的点落在误差 0.01 以外，考虑到测试了测试了 500 次，且要求的误差不超过 0.01 的概率为 99%，这些超过 0.01 的样本是可以接受的。

将 500 次计算的平均值作为计算结果，结果为 0.6431029411764705

附录

代码

```
    # DEFINE
    EPS = 0.01
    DELTA = 0.01

    def func(x):
        return (1/2*np.sin(1/x/x) + 1/2)

    class MC_int():
        def __init__(self, func):
            self.func = func


        def plot_line(self, savepath = './tex/figure/int_curve.jpg'):
            x = np.logspace(-3,0,100)
            y = self.func(x)
            plt.figure()
            plt.plot(x, y)
            plt.title('integral curve')
            plt.savefig(savepath)
            plt.show()


        def integral(self, n):
            x = np.random.rand(n)
            y = np.random.rand(n)
            s = y < self.func(x)
            return sum(s)/n


        def batch_test(self, n, b):
            x = np.random.rand(b, n)
            y = np.random.rand(b, n)
            s = y < self.func(x)
            res = s.sum(axis = 1)/n
            # 下面的代码为上面的功能，但上面的代码向量化，速度更快
            # res = np.array([self.integral(n) for i in range(b)])
            m = res.mean()
```

```
            plt.figure()
            plt.hist(res-m, bins = 50)
            plt.savefig('./tex/figure/hist_n{}_b{}.jpg'.format(n,b))
            plt.show()
            return m

        @staticmethod
        def caln_chebyshev(eps, delta):
            return np.floor(1/4/delta/eps/eps)

        @staticmethod
        def caln_norm(eps, delta):
            return np.floor(norm.ppf(1-delta/2,loc=0,scale=1)/2/eps)**2


    MC = MC_int(func)

    n1 = MC.caln_chebyshev(EPS, DELTA)
    n2 = MC.caln_norm(EPS, DELTA)
    print(n1, n2)

    print(MC.batch_test(17000, 500))
```