2018321084???HW2

May 28, 2019

```
In [1]: import numpy as np
        import pandas as pd
        import scipy.stats
```

1 Q3

1.0.1 True value

It is easy to compute integral term

$$\theta = \int_0^1 \frac{e^x - 1}{e - 1} dx = \frac{e - 2}{e - 1}$$

Thus true value of θ is $(e-2)/(e-1)$

Fix the number of iteration $n = 1000$

```
In [2]: n=1000
```

1.1 (a) Cude Monte Carlo

Let $X = \{x_1, x_2, \ldots, x_n\}$ and $x_i \sim^{iid} Unif(0,1)$ for $i = 1, \ldots, n$
Suppose $h(x) = \frac{e^x - 1}{e-1}$ and $f(x) = 1$

$$E_f[h(X)] = \int_0^1 \frac{e^x - 1}{e - 1} dx \approx \frac{1}{n} \sum_{i=1}^n h(x_i)$$
$$\hat{\theta}^{crude} = \frac{1}{n} \sum_{i=1}^n \frac{e_i^x - 1}{e - 1}$$

1.1.1 bias

$$bias_\theta(\hat{\theta}^{crude}) = E[\hat{\theta}^{crude}] - \theta$$

First calculate the expectation of $\frac{e_i^x - 1}{e-1}$. As x_i's follow $Unif(0,1)$

$$E_f\left[\frac{e_i^x - 1}{e - 1}\right] = \int_0^1 \frac{e_i^x - 1}{e - 1} \cdot 1 dx_i = \frac{e - 2}{e - 1}$$

Since, x_i's are independent

$$E[\hat{\theta}^{crude}] = \frac{1}{n} \sum_{i=1}^n E\left[\frac{e_i^x - 1}{e - 1}\right] = \frac{e - 2}{e - 1} = \theta$$

Then bias of Cude Monte Carlo Estimator is

$$bias_\theta(\hat{\theta}^{crude}) = E[\hat{\theta}^{crude}] - \theta = 0$$

1.1.2 variance

$$\begin{aligned}
Var[\hat{\theta}^{crude}] &= Var\left[\frac{1}{n}\sum_{i=1}^{n}\frac{e_i^x - 1}{e-1}\right] \\
&= \frac{1}{n^2(e-1)^2}\sum_{i=1}^{n} Var[e_i^x - 1] \\
&= \frac{1}{n^2(e-1)^2}\sum_{i=1}^{n} Var[e_i^x] \\
&= \frac{1}{n^2(e-1)^2}\sum_{i=1}^{n} [E[e^{2\cdot x_i}] - E[e_i^x]^2]
\end{aligned}$$

as $x_i \sim Unif(0,1)$ we can use mgf of Unifrom distribution

$$E_f[e^{tx}] = \frac{e^t - 1}{t}$$

Thus

$$\begin{aligned}
Var_f[\hat{\theta}^{crude}] &= \frac{1}{n^2(e-1)^2}\sum_{i=1}^{n}\left[\frac{e^2-1}{2} - (e-1)^2\right] \\
&= \frac{1}{n(e-1)^2}\left[\frac{e^2-1}{2} - (e-1)^2\right] \\
&= \frac{3-e}{2n(e-1)}
\end{aligned}$$

1.1.3 MSE

$mse = bias^2 + var$. Thus,

$$MSE(\hat{\theta}^{crude}) = 0 + \frac{3-e}{2n(e-1)} = \frac{3-e}{2n(e-1)}$$

In [3]:
```python
def CrudeMC(n=100,prt = False):
    theta = (np.exp(1)-2)/(np.exp(1)-1)
    X = np.random.uniform(0,1,n)
    theta_hat = ((np.exp(X)-1)/(np.exp(1)-1)).mean()
    mse = (3-np.exp(1))/(2*n*(np.exp(1)-1))
    if prt==True:
        print('theta estimator is : %f' %theta_hat)
        print('MSE is : %f' %mse)
    return(theta_hat,mse)
```

In [4]:
```python
crude_est,crude_mse = CrudeMC(n,1)
```

```
theta estimator is : 0.415960
MSE is : 0.000082
```

1.2 (b) Importance sampling

Let $g(x) = \frac{e^{-t}}{1-e^{-1}}$. We need to sample from $g(x)$ which follows truncated exponential distribution. We can use Inverse CDF method

Suppose $G(x)$ is CDF of $g(x)$ then,

$$G(x) = \int_0^x \frac{e^{-t}}{1-e^{-1}} dt = \frac{1-e^{-x}}{1-e^{-1}} \sim Unif(0,1)$$
$$G^{-1}(U) = -log\left(1 - U(1-e^{-1})\right)$$

When $U \sim Unif(0,1)$, $G^{-1}(U)$ follows truncated exponentail distribution

Like Crude Monte Carlo method suppose $h(x) = \frac{e^x-1}{e-1}$ and $f(x) = 1$ and x_i's follows $g(x)$ which is truncated exponential distribution

$$\hat{\theta}^{Importance} = E_g\left[h(x)\frac{f(x)}{g(x)}\right] \approx \frac{1}{n}\sum_{i=1}^{n} h(x_i)\frac{f(x_i)}{g(x_i)}$$
$$= \frac{1}{n}\sum_{i=1}^{n} \frac{e^{x_i}-1}{e^{-(x_i-1)}}$$

1.2.1 bias

$$bias_\theta(\hat{\theta}^{Importance}) = E[\hat{\theta}^{Importance}] - \theta$$

First calculate the expectation of $\frac{e^{x_i}-1}{e^{-(x_i-1)}}$

As x_i's are iid sampled from pdf $g(x)$

$$E_g\left[\frac{e^{x_i}-1}{e^{-(x_i-1)}}\right] = \int_0^1 \frac{e^{x_i}-1}{e^{-(x_i-1)}} g(x_i) dx_i = \int_0^1 \frac{e_i^x - 1}{e-1} dx_i = \frac{e-2}{e-1}$$

Since, x_i's are independent

$$E[\hat{\theta}^{Importance}] = \frac{1}{n}\sum_{i=1}^{n} E_g\left[\frac{e^{x_i}-1}{e^{-(x_i-1)}}\right] = \frac{e-2}{e-1} = \theta$$

Then bias of Importance sampling is

$$bias_\theta(\hat{\theta}^{Importance}) = E[\hat{\theta}^{Importance}] - \theta = 0$$

1.2.2 variance

$$Var[\hat{\theta}^{Importance}] = Var\left[\sum_{i=1}^{n} \frac{e^{x_i}-1}{e^{-(x_i-1)}}\right]$$
$$= \frac{1}{n^2e^2}\sum_{i=1}^{n} Var\left[\frac{e^{x_i}-1}{e^{-x_i}}\right]$$
$$= \frac{1}{n^2e^2}\sum_{i=1}^{n} Var[e^{2x_i} - e^{x_i}]$$
$$= \frac{1}{n^2e^2}\sum_{i=1}^{n} \left[E_g[e^{4x_i}] - E_g[e^{2x_i}]^2 + E_g[e^{2x_i}] - E_g[e^{x_i}]^2 - 2\left(E_g[e^{3x_i}] - E_g[e^{2x_i}]E_g[e^{x_i}]\right)\right]$$

First calulate the MGF of g(x)

$$M_x(t) = E_g[e^{tx}] = \int_0^1 e^{tx} \frac{e^{-x}}{1-e^{-1}} dx = \frac{e^t - e}{(t-1)(e-1)}$$

When $t > 1$ and $M_x(1) = e/(e-1)$ Thus

$$Var_g[\hat{\theta}^{Importance}] = \frac{1}{ne^2} \left[M_x(4) - M_x(2)^2 + M_x(2) - M_x(1)^2 - 2\left(M_x(3) - M_x(2)M_x(1)\right) \right]$$

1.2.3 MSE

$mse = bias^2 + var$. Thus,

$$\begin{aligned} MSE(\hat{\theta}^{Importance}) &= 0 + \frac{1}{ne^2} \left[M_x(4) - M_x(2)^2 + M_x(2) - M_x(1)^2 - 2\left(M_x(3) - M_x(2)M_x(1)\right) \right] \\ &= \frac{1}{ne^2} \left[M_x(4) - M_x(2)^2 + M_x(2) - M_x(1)^2 - 2\left(M_x(3) - M_x(2)M_x(1)\right) \right] \end{aligned}$$

```
In [5]: def Samplegx(n=100):
            U = np.random.uniform(0,1,n)
            X = -np.log(1-U*(1-np.exp(-1)))
            return(X)
```

```
In [6]: def mgfgx(t):
            if t==1:
                out = np.exp(1)/(np.exp(1)-1)
            elif t>1:
                out = (np.exp(t)-np.exp(1))/((t-1)*(np.exp(1)-1))
            return(out)
```

```
In [7]: def importance(n=100,prt = False):
            X = Samplegx(n)
            theta_hat = ((np.exp(X)-1)/(np.exp(-X+1))).mean()
            mse = (mgfgx(4) - mgfgx(2)**2 + mgfgx(2) - mgfgx(1)**2 - 2*(mgfgx(3)-mgfgx(2)*mgfgx(
            if prt==True:
                print('theta estimator is : %f' %theta_hat)
                print('MSE is : %f' %mse)
            return(theta_hat,mse)
```

```
In [8]: imps_est,imps_mse = importance(n)
```

1.3 (c) Control Variates

as (a) Suppose $h(x) = \frac{e^x - 1}{e-1}$, $f(x) = 1$ and $g(x) = \frac{x}{e-1}$

$$\begin{aligned} E_f[h(X)] &= E_f[g(X)] + E_f[h(X) - g(X)] \\ \hat{\theta}^{CV} &= E_f[g(X)] + \frac{1}{n}\sum_{i=1}^{n} (h(x_i) - g(x_i)) \\ &= \frac{1}{2(e-1)} + \frac{1}{n(e-1)} \sum_{i=1}^{n} (e^{x_i} - x_i - 1) \end{aligned}$$

1.3.1 bias

$$E_f[x] = 0.5$$

From (a)

$$E_f[e^{tx}] = \frac{e^t - 1}{t}$$

Thus

$$E\left[\hat{\theta}^{CV}\right] = \frac{1}{2(e-1)} + \frac{1}{n(e-1)} \sum_{i=1}^{n} \left(E\left[e^{x_i}\right] - E\left[x_i\right] - 1\right) = \frac{e-2}{e-1} = \theta$$

Then bias of Control variates is

$$bias_\theta(\hat{\theta}^{CV}) = E[\hat{\theta}^{CV}] - \theta = 0$$

1.3.2 variance

as x_i's follows $Unif(0,1)$

$$E[x_i^2] = var[x_i] + E[x_i]^2 = \frac{1}{3}$$

$$E[e^{x_i}x_i] = \int_0^1 e^{x_i}x_i dx_i = 1$$

Variance of Control variates estimator is

$$\begin{aligned}
Var_f[\hat{\theta}^{CV}] &= \frac{1}{n^2(e-1)^2} \sum_{i=1}^{n} Var_f\left[e^{x_i} - x_i - 1\right] \\
&= \frac{1}{n^2(e-1)^2} \sum_{i=1}^{n} Var_f\left[e^{x_i} - x_i\right] \\
&= \frac{1}{n^2(e-1)^2} \sum_{i=1}^{n} Var_f\left[e^{x_i} - x_i\right] \\
&= \frac{1}{n^2(e-1)^2} \sum_{i=1}^{n} \left[E_f[e^{2x_i}] - E_f[e^{x_i}]^2 + E_f[x_i^2] - E_f[x_i]^2 - 2\left(E[e^{x_i}x_i] - E_f[e^{x_i}]E_f[x_i]\right)\right] \\
&= \frac{1}{n(e-1)^2}\left[(e-1)\left(\frac{5-e}{2}\right) - \frac{23}{12}\right]
\end{aligned}$$

1.3.3 MSE

$mse = bias^2 + var$. Thus,

$$\begin{aligned}
MSE(\hat{\theta}^{CV}) &= 0 + \frac{1}{n(e-1)^2}\left[(e-1)\left(\frac{5-e}{2}\right) - \frac{23}{12}\right] \\
&= \frac{1}{n(e-1)^2}\left[(e-1)\left(\frac{5-e}{2}\right) - \frac{23}{12}\right]
\end{aligned}$$

```
In [9]: def ControlVariates(n=100, prt = False):
            theta = (np.exp(1)-2)/(np.exp(1)-1)
            X = np.random.uniform(0,1,n)
            theta_hat = 1/(2*(np.exp(1)-1)) + (np.exp(X) - X -1).mean()/(np.exp(1)-1)
            mse = ((np.exp(1)-1)*((5-np.exp(1))/2) - 23/12)/(n*(np.exp(1)-1)**2)
```

```
        if prt==True:
            print('theta estimator is : %f' %theta_hat)
            print('MSE is : %f' %mse)
        return(theta_hat,mse)
```

```
In [10]: CV_est , CV_mse = ControlVariates(n,1)
```

```
theta estimator is : 0.423796
MSE is : 0.000015
```

1.4 Antithetic Variates

As $x \sim Unif(0,1)$, $1 - x \sim Unif(0,1)$. Thus

$$\begin{aligned}\hat{\theta}^{Antithetic} &= \frac{1}{2n}\sum_{i=1}^{n}[h(x_i) - h(1-x_i)] \\ &= \frac{1}{2n}\sum_{i=1}^{n}\frac{e^{x_i} + e^{1-x_i} - 2}{e-1}\end{aligned}$$

1.4.1 bias

As $x \sim Unif(0,1)$, $1 - x \sim Unif(0,1)$. Thus

$$E[e^{x-i}] = E[e^{1-x_i}] = e - 1$$

Therefore

$$\begin{aligned}E[\hat{\theta}^{Antithetic}] &= E\left[\frac{1}{2n}\sum_{i=1}^{n}\frac{e^{x_i} + e^{1-x_i} - 2}{e-1}\right] \\ &= \frac{1}{2n}\sum_{i=1}^{n}\frac{E[e^{x_i}] + E[e^{1-x_i}] - 2}{e-1} \\ &= \frac{e-2}{e-1} = \theta\end{aligned}$$

Then bias of Antithetic variates is

$$bias_{\theta}(\hat{\theta}^{Antithetic}) = E[\hat{\theta}^{Antithetic}] - \theta = 0$$

1.4.2 variance

as x_i's follows $Unif(0,1)$ and $1-x_i$'s follows $Unif(0,1)$ We can use same expectation and mgf
Variance of Antithetic variates estimator is

$$
\begin{aligned}
Var_f[\hat{\theta}^{Antithetic}] &= \frac{1}{4n^2(e-1)^2}\sum_{i=1}^{n} Var_f\left[e^{x_i} - e^{1-x_i} - 2\right] \\
&= \frac{1}{4n^2(e-1)^2}\sum_{i=1}^{n} Var_f\left[e^{x_i} - e^{1-x_i}\right] \\
&= \frac{1}{4n^2(e-1)^2}\sum_{i=1}^{n} 2\left[E_f[e^{2x_i}] - E_f[e^{x_i}]^2 - \left(E[e] - E_f[e^{x_i}]^2\right)\right] \\
&= \frac{1}{4n(e-1)^2}\left[e^2 - 1 - 4(e-1)^2 + 2e\right] \\
&= \frac{1}{4n(e-1)^2}\left[-3e^2 + 10e - 5\right]
\end{aligned}
$$

1.4.3 MSE

$mse = bias^2 + var$. Thus,

$$
\begin{aligned}
MSE(\hat{\theta}^{Antithetic}) &= 0 + \frac{1}{4n(e-1)^2}\left[-3e^2 + 10e - 5\right] \\
&= \frac{1}{4n(e-1)^2}\left[-3e^2 + 10e - 5\right]
\end{aligned}
$$

```
In [11]: def Antithetic(n=100,prt = False):
             theta = (np.exp(1)-2)/(np.exp(1)-1)
             X = np.random.uniform(0,1,n)
             theta_hat = ((np.exp(X) + np.exp(1-X) -2)/(np.exp(1)-1)).mean()/2
             mse = (-3 * np.exp(1)**2 + 10 *np.exp(1) -5)/(4*n*((np.exp(1)-1)**2))
             if prt==True:
                 print('theta estimator is : %f' %theta_hat)
                 print('MSE is : %f' %mse)
             return(theta_hat,mse)
```

```
In [12]: ant_est, ant_mse = Antithetic(n,1)
```

```
theta estimator is : 0.416557
MSE is : 0.000001
```

1.5 Comparison

MSE of (a) ~ (c) methods can be used for comparison of efficiency when sample size n is fixed

```
In [13]: n = 1000
         comp = pd.DataFrame([((np.exp(1)-2)/(np.exp(1)-1),0),CrudeMC(n),importance(n),ControlVa
         comp.index = ['True Theta','Crude Monte Carlo','Importance sampling',' Control variates
         comp.columns = ['Point Estimator','MSE']
```

```python
base = comp['MSE'][comp['MSE'].index == 'Crude Monte Carlo']
comp['Efficiency'] = comp['MSE']/float(base)
display(comp)
```

```
                     Point Estimator       MSE  Efficiency
True Theta                  0.418023  0.000000    0.000000
Crude Monte Carlo           0.422746  0.000082    1.000000
Importance sampling         0.401052  0.000187    2.284921
 Control variates           0.425319  0.000015    0.180349
Antithetic variates         0.417701  0.000001    0.016165
```

As all methods are unbiased estimator we can conclude the efficiency by MSE. Antithetic variates method is most efficiency. Second is Control variates method. Third is Crude Monte Carlo mehod. Importance sampling shows worst efficiency

2 Q4

Define the varibles

$$\begin{aligned} h(X) &= I(X > c) \\ f(X) &= \phi(X) \\ g(X) &= \phi(X - b) \end{aligned}$$

where $\phi(\cdot)$ denotes a standard normal density function

$$\begin{aligned} p = E_f[h(Z)] &= E_g\left[h(Z)\frac{f(Z)}{g(Z)}\right] \\ &\approx \frac{1}{n}\sum_{i=1}^{n} h(z_i)\frac{f(z_i)}{g(z_i)} = \tilde{p}(b) \end{aligned}$$

2.1 (a) find b^*

$$\begin{aligned} Var[\tilde{p}(b)] &\propto var_g\left(h(z)\frac{f(z)}{g(z))}\right) \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\left\{E_g\left[\left(h(z_i)\frac{f(z_i)}{g(z_i)}\right)^2\right] - E_g\left[h(z_i)\frac{f(z_i)}{g(z_i)}\right]^2\right\} \end{aligned}$$

When $S(\cdot)$ be the survival function for a standard normal distribution

$$\begin{aligned} E_g\left[\left(h(z_i)\frac{f(z_i)}{g(z_i)}\right)^2\right] &= e^{b^2}\int_{-\infty}^{\infty} I(z_i > c)\phi(z_i)dz_i = e^{b^2}S(c+b) \\ E_g\left[h(z_i)\frac{f(z_i)}{g(z_i)}\right] &= E_f[h(z_i)] = S(c) \end{aligned}$$

Thus,

$$\begin{aligned} Var_g[\tilde{p}(b)] &= \frac{1}{n^2}\sum_{i=1}^{n}\left[e^{b^2}S(c+b) - S^2(c)\right] \\ &= \frac{1}{n}\left[e^{b^2}S(c+b) - S^2(c)\right] \end{aligned}$$

$Var_g[\tilde{p}(b)]$ minimized when $\partial Var_g[\tilde{p}(b)]/\partial b$

$$\frac{\partial Var_g[\tilde{p}(b)]}{\partial b} =^{let} 0$$
$$2b \cdot e^{b^2} S(b+c) - e^{b^2}\phi(b+c) = 0$$

Therefore condition for b^* that minimize $Var_g[\tilde{p}(b)]$ is,

$$2b \cdot e^{b^2} S(b+c) - e^{b^2}\phi(b+c) = 0 \rightarrow 2b^* \cdot S(b^*+c) = \phi(b^*+c)$$

2.2 (b) Find boundary condition

From (a)

$$\frac{S(b^*+c)}{\phi(b^*+c)} = \frac{1}{2b^*}$$

Use Mills' ratio

$$\frac{1}{b^*+c}\left(1 - \frac{1}{(b^*+c)^2}\right) \geq \frac{1}{2b^*} \geq \frac{1}{b^*+c}$$
$$\rightarrow \frac{2(b^*+c)^2 - 2}{(b^*+c)^3} \geq \frac{1}{b^*} \geq \frac{2}{b^*+c}$$
$$\rightarrow \frac{b^*+c}{2} \geq b^* \geq \frac{(b^*+c)^3}{2(b^*+c)^2 - 2}$$

Thus boundary conditons for b^* is

$$c \leq b^* \text{ and} \frac{(b^*-c)(b^*+c)^2}{2} \leq b^*$$

2.3 (c) bisection method

Let $b^* = c + \epsilon$, where $\epsilon > 0$ and $c > 0$ Then,

$$0 \leq \frac{(b^*-c)(b^*+c)^2}{2} = \frac{\epsilon(2c+\epsilon)^2}{2} \leq c + \epsilon$$
$$\rightarrow 0 \leq \epsilon(4c^2 + 4c\epsilon + \epsilon^2) \leq 2c + 2\epsilon$$
$$\rightarrow 0 \leq 4\epsilon c^2 + 4\epsilon^2 c + \epsilon^3 \leq 2c + 2\epsilon$$
$$\rightarrow 0 \leq 4c^2 + 4\epsilon c + \epsilon^2 \leq 2\frac{c}{\epsilon} + 2$$
$$\rightarrow 0 \leq \frac{c}{\epsilon} + 1 - 2\epsilon c$$

When $0 \leq \frac{c}{\epsilon} - 2\epsilon c$, $\frac{c}{\epsilon} + 1 - 2\epsilon c$ always larger than 0

$$0 \leq \frac{c}{\epsilon} - 2\epsilon c$$
$$\rightarrow 0 \leq \frac{1}{\epsilon} - 2\epsilon$$
$$\rightarrow 0 \leq \epsilon \leq \frac{1}{\sqrt{2}}$$

So, we can setting the upper bound of bisection as $c + \frac{1}{\sqrt{2}}$

```
In [14]: def bisection_test(b,c):
             return(scipy.stats.norm.sf(b+c)/scipy.stats.norm.pdf(b+c) - 1/(2*b))
```

```
In [15]: def bisection(c,criteria = 10**(-7)):
             lower= c
             upper= c+1/np.sqrt(2)
             while True:
                 bstar = (lower+upper)/2
                 if abs(bisection_test(bstar,c))<criteria:
                     return(bstar)
                     break
                 elif bisection_test(lower,c)*bisection_test(bstar,c)<0:
                     upper = bstar
                 else:
                     lower = bstar
```

```
In [16]: for i in [2,3,4]:
             print('When c = %d, b* : %f' %(i,bisection(i)))
```

```
When c = 2, b* : 2.215929
When c = 3, b* : 3.154852
When c = 4, b* : 4.119678
```

When $c = 2, 3$ and 4
$b^* = 2.215931, 3.154846$ and 4.119675

2.4 (d) Calculate the efficiency

We know that $ =(0) $.
From (a)

$$
\begin{aligned}
Var_g[\tilde{p}(b)] &= \frac{1}{n^2}\sum_{i=1}^{n}\left[e^{b^2}S(c+b) - S^2(c)\right] \\
&= \frac{1}{n}\left[e^{b^2}S(c+b) - S^2(c)\right]
\end{aligned}
$$

Thus,

$$
Efficiency = \frac{Var[\hat{p}]}{Var[\tilde{p}(b^*)]} = \frac{S(c) - S^2(c)}{e^{b^{*2}}S(c+b^*) - S^2(c)}
$$

```
In [17]: def varatio(b,c):
             out = (scipy.stats.norm.sf(c) -
                     scipy.stats.norm.sf(c)**2)/ \
                   (np.exp(b**2)*scipy.stats.norm.sf(c+b)
                     - scipy.stats.norm.sf(c)**2)
             return(out)
```

```
In [18]: for i in [2,3,4]:
             print('Efficiency when c = %d is : %f' %(i,varatio(bisection(i),i)))
```

```
Efficiency when c = 2 is : 19.001601
Efficiency when c = 3 is : 221.916501
Efficiency when c = 4 is : 7061.451247
```

When $c = 2, 3$ and 4
$Efficiency = 19.001601, 221.916501$ and 7061.451247