PS4-New

April 20, 2017

0.1 Estimation Of Variance For Single Segment without Shift

Let's try to estimate σ_I analytically for one segment.

$$\sigma_I^2 = \left\langle \frac{f^2}{w^2} \right\rangle - \left\langle \frac{f}{w} \right\rangle^2$$

Let $g = \frac{f}{w}$, then,

$$\sigma_{f/w}^2 = \left\langle g^2 \right\rangle - \left\langle g \right\rangle^2$$

We can expand g using Taylor Series around $\frac{b+a}{2}$,

$$g(x) = g\left(\frac{b+a}{2}\right) + g'\left(\frac{b+a}{2}\right)\left(x - \frac{b+a}{2}\right) + \frac{1}{2}g''\left(\frac{b+a}{2}\right)\left(x - \frac{b+a}{2}\right)^2$$

We can also aprroximate first and second derivative very accurately for a small region,

$$g'\left(\frac{b+a}{2}\right) = \frac{g(b) - g(a)}{b-a}$$

$$g''\left(\frac{b+a}{2}\right) = \frac{g(b) - 2g(\frac{b+a}{2}) + g(a)}{(\frac{b-a}{2})^2}$$

To simplify equations we can define the following variables,
$p = g(\frac{b+a}{2})$
$q = \frac{g(b)-g(a)}{b-a}$
$r = \frac{g(b)-2g(\frac{b+a}{2})+g(a)}{(\frac{b-a}{2})^2}$
$u = (x - \frac{b+a}{2})$
Then,

$$g(x) = ru^2 + qu + p$$

Since we forced w to be equal to f at the boundaries, q is 0.
Our job reduces to calculate

$$\left\langle (ru^2+p)^2 \right\rangle - \left\langle ru^2 + p \right\rangle^2$$

Expanding the terms,

$$\sigma^2_{f/w} = \left\langle r^2u^4 + 2rpu^2 + p^2 \right\rangle - \left\langle ru^2 + p \right\rangle^2$$

$$= (r^2 \left\langle u^4 \right\rangle + 2rp \left\langle u^2 \right\rangle + p^2) - (r^2 \left\langle u^2 \right\rangle^2 + 2rp \left\langle u^2 \right\rangle + p^2)$$

$$= r^2 \left\langle u^4 \right\rangle - r^2 \left\langle u^2 \right\rangle^2$$

$$= r^2(\left\langle u^4 \right\rangle - \left\langle u^2 \right\rangle^2)$$

This could further be simplified as,

$$\sigma^2_{f/w} = r^2\sigma^2_{u^2}$$

Finally σ_I can be estimated as,

$$\sigma_I = \frac{(b-a)}{\sqrt{N}} r\sigma_{u^2} = \frac{(b-a)}{\sqrt{N}} \frac{g(b) - 2g(\frac{b+a}{2}) + g(a)}{(\frac{b-a}{2})^2} \frac{2(b-a)^3}{3}$$

$$\sigma_I = \frac{8(b-a)^2}{3\sqrt{N}}(g(b) - 2g(\frac{b+a}{2}) + g(a))$$

or

$$\sigma_I = \frac{2(b-a)^4}{3\sqrt{N}} g'' \left(\frac{a+b}{2}\right)$$

0.2 Estimation Of Variance with Constant Shift

As we know when a weight function is used during Monte Carlo integration, the variance could be reduced by shifting the function by a constant amount. Fortunately, we do not have to do all calculations from scratch since the only change will be the constant terms in the final equation, namely p, r, q. This is because we did all the calculations for a generic gunction g and the final equation should be valid for all nice functions provided that constants are calculated for relevant function.

This time $g = \frac{f+C}{n(w+C)}$ where n is normalization constant for the weight function and constants should be calculated for this function. Notice that as C increases,g approaches to 1 which results a zero variance. However due to precision problems after a certain point a computer will find $g = 1$ and the value of the integral will be far from true value.

```
In [15]: import numpy as np
         from numpy.polynomial import Polynomial as P
         #import plotly
         #import plotly.plotly as py
         #import plotly.figure_factory as ff
         import matplotlib.pyplot as plt
         #Integrand function
         def f(x,H):
             return (x-5)*np.exp(-(x/2-3))+H
```

```python
#Calculates the coefficients of linear weight function.
def findw(f,H,lower,upper,normalize):
    #Find the linear function.
    slope=(f(upper,H)-f(lower,H))/(upper-lower)
    a=slope
    b=-slope*upper+f(upper,H)
    #Normalization.
    A=(a/2)*(upper**2)+b*upper-(a/2)*(lower**2)-b*lower
    if normalize:
        a/=A
        b/=A
    return [a,b]
#Performs integration.
def integrate(f,lower,upper,N,C):
    H=C
    w=findw(f,H,lower,upper,True)
    #Generate uniform random inputs.
    inputs=np.random.rand(N)
    a=w[0]/2
    b=w[1]
    c=-(a*lower**2+b*lower)

    SUM=0
    SUM2=0

    inverse_inputs=[]
    for i in inputs:
        p=[(-b-np.sqrt(b**2-4*a*(c-i)))/(2*a),(-b+np.sqrt(b**2-4*a*(c-i)))
        if p[0]>=lower and p[0]<=upper:
            inverse_inputs.append(p[0])
        else :
            inverse_inputs.append(p[1])

    inverse_inputs=np.array(inverse_inputs)
    #Calculate f(inverse(x))/w(inverse(x)).
    outputsF=f(inverse_inputs,H)
    outputsW=w[0]*(inverse_inputs)+w[1]
    outputs=outputsF/outputsW
    SUM=outputs.sum()
    SUM2=(outputs*outputs).sum()
    var=SUM2/N-(SUM/N)**2
    var=var/N
    #Store generated points for variance calculation.
    Vsum=outputs.sum()
    return Vsum/N-H*(upper-lower),(upper-lower)**2*var
```

```python
def theoretical_sigma(f,lower,upper,N,C):

    w=findw(f,C,lower,upper,True)

    flower=f(lower,C)
    fmiddle=f((lower+upper)/2,C)
    fupper=f(upper,C)

    wlower=w[0]*lower+w[1]
    wmiddle=w[0]*(lower+upper)/2+w[1]
    wupper=w[0]*upper+w[1]

    gupper=fupper/wupper
    gmiddle=fmiddle/wmiddle
    glower=flower/wlower

    b=(gupper-glower)/(upper-lower)
    c=2*(gupper-2*gmiddle+glower)/(upper-lower)**2
    meanu=0
    meanu2=2/3*((upper-lower)/2)**3
    meanu2/=(upper-lower)
    meanu3=0
    meanu4=2/5*((upper-lower)/2)**5
    meanu4/=(upper-lower)

    #var=b**2*(meanu2-meanu**2)+2*b*c*(meanu3-meanu*meanu2)+c**2*(meanu4-m
    var=c**2*(meanu4-meanu2**2)
    var/=N

    return (upper-lower)*np.sqrt(var)


low=4.6
up=5.2
#Divide the region into 10 pieces.
l=[low,up]
#Real value of the integral
I_real=-.12002
#N values
N=[100,1000,10000,100000,1000000]

#Integration results.
results=[]
#Standart deviation values
sigmas=[]
for k in N:
    I=0
```

```
    sigma=0
    S=1000
    for i in range (0,len(l)-1):
        temp,temp2=integrate(f,l[i],l[i+1],k,S)
        I+=temp
        sigma+=temp2
    results.append(I)
    sigmas.append(np.sqrt(sigma))
    print(k,I,I-I_real,np.sqrt(sigma),theoretical_sigma(f,low,up,k,S)/np.s
```

```
100 -0.118440678024 0.00157932197606 0.000839993344335 1.02531262852 1.88016010687
1000 -0.119660237901 0.000359762098743 0.000272079608543 1.00100479394 1.32226777754
10000 -0.11990697722 0.000113022780329 8.63064208408e-05 0.997904646526 1.309552397
100000 -0.120012318097 7.68190323062e-06 2.72518982396e-05 0.999390905149 0.2818850
1000000 -0.120023218789 -3.21878867406e-06 8.63311580949e-06 0.997618707805 -0.3728
```