Error Analysis

September 28, 2022

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sy
from IPython.display import Math, display

plt.rcParams['figure.figsize'] = (12,7)
```

0.1 (i) Error Propagation

0.1.1 (a) Suppose your first attempt is the simple mean $\overline{z} = (z_1 + z_2)/2$. Find $\overline{z}$ and the corresponding error $\sigma_{\overline{z}}$ using the standard propagation of error formula.

$z_1 = 6 \pm 2$
$z_2 = 4 \pm 1$
$\sigma_{\overline{z}} = \pm\sqrt{\sigma_{z_1}^2 + \sigma_{z_2}^2} = \pm\sqrt{2^2 + 1^2} = \pm\sqrt{5}$
$\overline{z} = 5 \pm \sqrt{5}$

0.1.2 (b) Consider the weighted mean $\overline{z}_w = wz_1 + (1-w)z_2$. Find the corresponding error $\sigma_{\overline{z}_w}$ using the standard propagation of error formula, as a function of w.

$\sigma_{\overline{z}_w}^2 = w^2\sigma_{z_1}^2 + (1-w)^2\sigma_{z_2}^2$
$\sigma_{\overline{z}_w} = \pm\sqrt{w^2\sigma_{z_1}^2 + (1-w)^2\sigma_{z_2}^2} = \pm w\sqrt{\sigma_{z_1}^2 + (\frac{1-w}{w})^2\sigma_{z_2}^2}$

0.1.3 (c) Find the value of w that minimizes the error $\sigma_{\overline{z}_w}$. For this example, show the weights $w_1 = w$ and $w_2 = (1-w)$ associated with z_1 and z_2 are inversely proportional to $\sigma_{z_1}^2$ and $\sigma_{z_2}^2$, respectively.

[2]:
```
w, sigma_z1, sigma_z2 = sy.symbols('w \\sigma_{z_1} \\sigma_{z_2}')
expr = (w**2 * sigma_z1**2 + (1 - w)**2 * sigma_z2**2)**0.5
dz = sy.diff(expr, w)

#need derivative for minimum value of w
display(Math('\\frac{d\\sigma_{\\overline{z}_w}}{dw} = 0 = ' +  sy.latex(dz)))
print("Now solve for w:")
display(Math('(\\sigma_{z_1}^2 + \\sigma_{z_2}^2)w - \\sigma_{z_2}^2 = 0␣
↪\\Rightarrow w = \\frac{\\sigma_{z_2}^2}{\\sigma_{z_1}^2 +␣
↪\\sigma_{z_2}^2}'))
```

```
#substitute given values for w1, w2
print("Substitute w1 and w2 into initial derivative:")
display(Math('\\sigma_{z_1}^2 w_1 - \\sigma_{z_2}^2 w_2 = 0 \\Rightarrow
 \\frac{w_1}{\\sigma_{z_2}^2} = \\frac{w_2}{\\sigma_{z_1}^2}'))
```

$$\frac{d\sigma_{\overline{z}_w}}{dw} = 0 = \frac{1.0\sigma_{z_1}^2 w + 0.5\sigma_{z_2}^2 \cdot (2w-2)}{\left(\sigma_{z_1}^2 w^2 + \sigma_{z_2}^2 (1-w)^2\right)^{0.5}}$$

Now solve for w:

$$(\sigma_{z_1}^2 + \sigma_{z_2}^2)w - \sigma_{z_2}^2 = 0 \Rightarrow w = \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}$$

Substitute w1 and w2 into initial derivative:

$$\sigma_{z_1}^2 w_1 - \sigma_{z_2}^2 w_2 = 0 \Rightarrow \frac{w_1}{\sigma_{z_2}^2} = \frac{w_2}{\sigma_{z_1}^2}$$

0.1.4 (d) What is the corresponding $\overline{z}_w$ and $\sigma_{\overline{z}_w}$?

[3]:
```
display(Math('\\overline{z}_w = \\frac{\\sigma_{z_2}^2}{\\sigma_{z_1}^2 +
 \\sigma_{z_2}^2}z_1 + \\frac{\\sigma_{z_1}^2}{\\sigma_{z_1}^2 +
 \\sigma_{z_2}^2}z_2'))

display(Math('\\sigma_{\\overline{z}_w} =
 \\pm\\sqrt{\\left(\\frac{\\sigma_{z_2}^2}{\\sigma_{z_1}^2 +
 \\sigma_{z_2}^2}\\right)^2 \\sigma_{z_1}^2 +
 \\left(\\frac{\\sigma_{z_1}^2}{\\sigma_{z_1}^2 + \\sigma_{z_2}^2}\\right)^2
 \\sigma_{z_2}^2 } = \
             \\pm \\sqrt{\\frac{\\sigma_{z_1}^2\\sigma_{z_2}^2(\\sigma_{z_1}^2
 + \\sigma_{z_2}^2)}{(\\sigma_{z_1}^2 + \\sigma_{z_2}^2)^2}} = \
             \\pm \\frac{\\sigma_{z_1}\\sigma_{z_2}}{\\sqrt{\\sigma_{z_1}^2 +
 \\sigma_{z_2}^2}}'))
```

$$\overline{z}_w = \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} z_1 + \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} z_2$$

$$\sigma_{\overline{z}_w} = \pm\sqrt{\left(\frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right)^2 \sigma_{z_1}^2 + \left(\frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right)^2 \sigma_{z_2}^2} = \pm\sqrt{\frac{\sigma_{z_1}^2\sigma_{z_2}^2(\sigma_{z_1}^2 + \sigma_{z_2}^2)}{(\sigma_{z_1}^2 + \sigma_{z_2}^2)^2}} = \pm\frac{\sigma_{z_1}\sigma_{z_2}}{\sqrt{\sigma_{z_1}^2 + \sigma_{z_2}^2}}$$

0.2 (ii) Importance Sampling:

0.2.1 (a) Develop an expression and program to sample from the exponential distribution, $\pi(x) = e^{-x}, x \geq 0$

[4]:
```
def sampleFromExpDist(npts, isPlot = False):
    x = np.random.exponential(1, size=npts) #beta = 1
    if isPlot:
        plt.figure(figsize=(7, 4))
        plt.hist(x, 20, density=True)
        plt.xlabel('x')
    return x
```

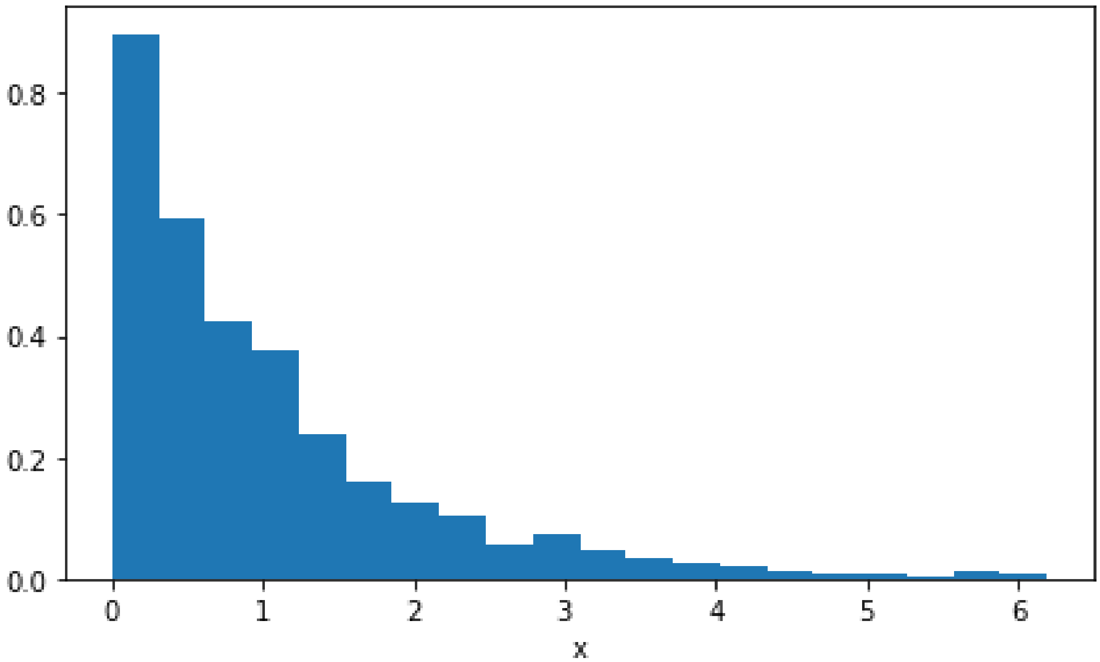
0.2.2 (b) Use importance sampling with 10^3 points to sample from $\pi(x)$, and estimate the integral.

[5]:
```
points = 10**3
sampledPts = sampleFromExpDist(points, isPlot = True)

def importanceSampling1(npts, x):
    #x = sampleFromExpDist(npts)
    f = x**0.5 * np.exp(-x)
    p = np.exp(-x)

    #integral = expected value of f/pi
    intg = np.mean(f / p)

    #error is sigma_(f/pi) / sqrt(npts)
    stdI = np.std(f / p)/np.sqrt(npts)
    return intg, stdI

estInt, estErr = importanceSampling1(points, sampledPts)
display(Math('I \\approx ' + sy.latex(estInt) + '\\text{, with estimated error␣
↪} \\sigma_{I} = ' + sy.latex(estErr)))

'''
Other way I found to do problem
#define our given functions
f = lambda x: x**0.5
p = lambda x: np.exp(-x)
g = lambda x: f(x) * p(x)

#estimate the integral by taking the mean of f(Xj), where Xj drawn from pi(x)
fbar = np.mean(f(sampledPts))
display(Math('\\overline{f} = ' + str(fbar)))

err = np.std(g(sampledPts) / p(sampledPts)) / np.sqrt(points)
'''
```

```
#actual value of integral
x = sy.symbols('x')
I1 = sy.integrate(x**0.5 * sy.exp(-x), (x, 0, sy.oo))
display(Math('I = ' + str(I1)))
```

$I \approx 0.898672326291434$, with estimated error $\sigma_I = 0.0152163938835387$

$I = 0.886226925452758$



The formula used to sample from the exponential distribution was from numpy's exponential distribution function in its *random* library with $\beta = 1$. Then, I took the mean of the sum of values $\frac{f(X_i)}{\pi(X_i)}$ in order to estimate the integral, I.

0.3 (iii) Transformation Method

0.3.1 (a) Normalize $\pi(x)$ so that $\int_0^{\pi} \pi(x)dx = 1$

[6]:
```
A = sy.symbols('A')
a = sy.symbols('a', positive = True)
expDist = sy.exp(-a*x)
#integrate with normalization constant
expInt = sy.integrate(A * expDist, (x, 0, sy.pi))
#display evaluated definite integral
display(Math(sy.latex(sy.collect(sy.collect(expInt, a), A))))
```

```python
#solve equation
normEq = sy.Eq(expInt, 1)
normConst = sy.solve(normEq, A, check = False)[0]
display(Math('A = ' + sy.latex(normConst))) #normalization constant
```

$$\frac{A\left(1 - e^{-\pi a}\right)}{a}$$

$$A = \frac{ae^{\pi a}}{e^{\pi a} - 1}$$

0.3.2 (b) Use the transformation method to sample from $\pi(x)$.

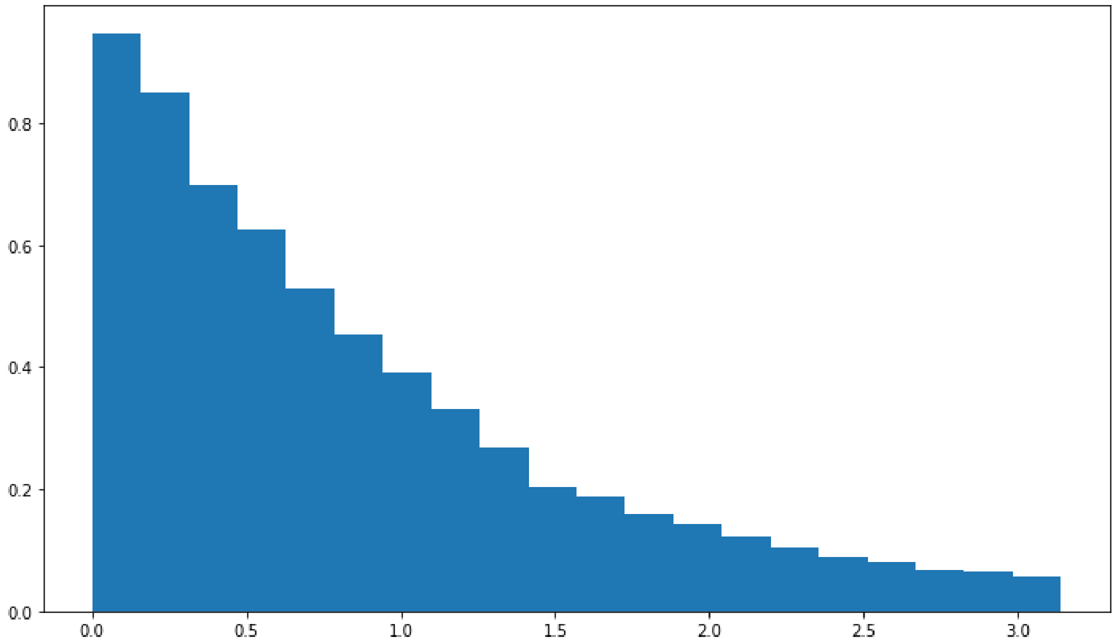
[7]:
```python
#get the CDF of pi(x)
F = sy.integrate(normConst * expDist, (x, 0, x))
display(Math("F(x) = \\int_{0}^x \\frac{a}{1-e^{-\\pi a}} e^{-ax'}dx' = " + sy.
    ↪latex(sy.together(sy.powsimp(F)))))

u = sy.symbols('u')
expr = u - F
display(Math(sy.latex(sy.solve(expr, x))))
```

$$F(x) = \int_{0}^{x} \frac{a}{1 - e^{-\pi a}} e^{-ax'} dx' = \frac{e^{\pi a} - e^{a(\pi - x)}}{e^{\pi a} - 1}$$

$$\left[\pi + \frac{\log\left(\frac{1}{-ue^{\pi a} + u + e^{\pi a}}\right)}{a} \right]$$

[8]:
```python
def drawLinearDist(npts, a):
    u = np.random.rand(npts)
    return np.pi - np.log(u + (np.exp(np.pi * a) * (1 - u))) / a
```

[9]:
```python
#test formula for a = 1
x = drawLinearDist(10000, a = 1)
_ = plt.hist(x, 20, density = True)
```



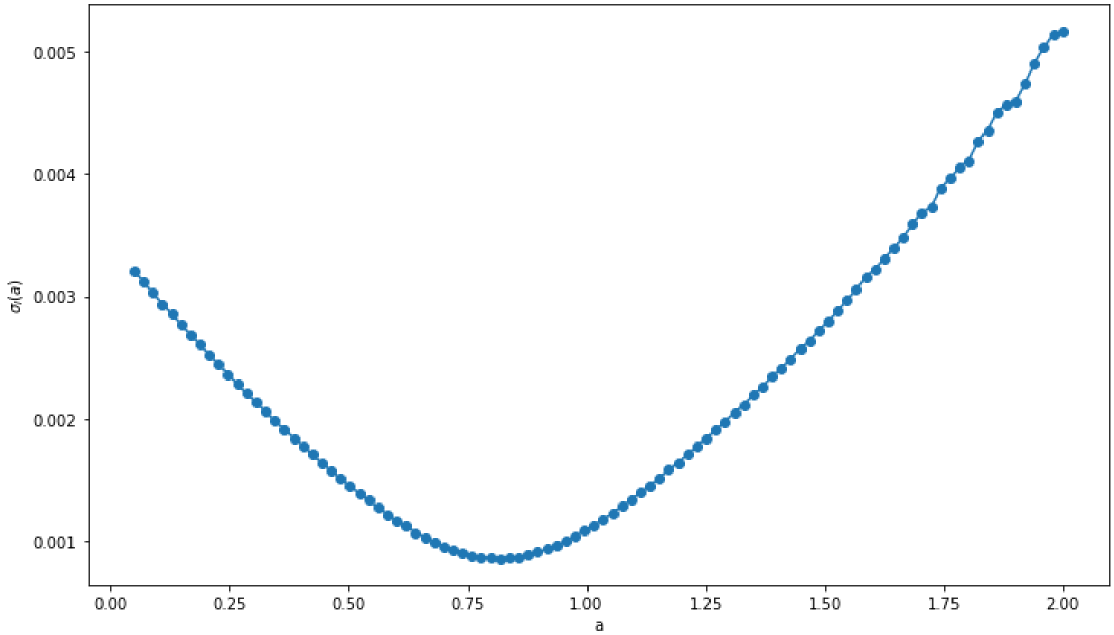
0.3.3 (c) With $n = 10^5$, vary a between 0.05 and 2.0 and make a plot of $\sigma_I(a)$. From the plot, estimate the value of a which minimizes $\sigma_I(a)$.

[10]:
```
def importanceSampling2(npts, a):
    """draw points from linear distribution"""
    x  = drawLinearDist(npts, a)
    f  = 1./(x**2 + np.cos(x)**2)
    p  = (a / (1 - np.exp(-np.pi * a))) * np.exp(-a * x)    #pi(x)
    intg = np.mean(f/p)
    stdI = np.std((f/p))/np.sqrt(npts) # error

    return intg, stdI
```

[11]:
```
mcPts = 10**5
a = np.linspace(0.05, 2.0, 100)
N = len(a)
Ig = np.zeros(N); sIg = np.zeros(N)

for i, v in enumerate(a):
    Ig[i], sIg[i] = importanceSampling2(mcPts, v)
```

[12]:
```
plt.plot(a, sIg, 'o-')

plt.xlabel('a')
plt.ylabel(r'$\sigma_I(a)$')
```

```
plt.show()
```



[13]:
```
print("The error is minimized at a =", np.round(a[np.argmin(sIg)], 2))
```

```
The error is minimized at a = 0.82
```