Computer Assignment (CA) NO.6: Mean and Variance Revisited

Tyler Berezowsky

March 15, 2015

1 Problem Statement

Use MATLAB's random number generator and generate uniformly distributed random numbers on the range [0,1]. We all agree that the mean, μ, should be 0.5, and the variance, σ^2, should be ??? (can you derive this?). The tasks to be accomplished are:

1. Generate N random numbers, denoted by the signal $x[n]$. Estimate the mean (equation 1.1) and variance (equation 1.2) using N data points. Compute the error of these estimates as: $E_{\tilde{\mu}}[N] = \tilde{\mu_X}[N] - \mu$ and $E_{\tilde{\sigma}}[N] = \tilde{\sigma}^2_X[N] - \sigma^2$.

$$\tilde{\mu}_x[N] = \frac{1}{n}\sum_{n=0}^{N-1} x[N] \tag{1.1}$$

$$\tilde{\sigma}^2[N] = \frac{1}{N}\sum_{n=0}^{N-1} (x[n] - \tilde{\mu})^2 \tag{1.2}$$

   Plot $E_{\tilde{\mu}}[N]$ and $E_{\tilde{\mu}}[N]$ for $N = [1, 1e6]$. Use a log base 10 scale for the horizontal axis (n - the number of points) and a linear scale for the error. Explain this plot.

2. Estimate the pdf of $x[n]$ using a bin size of 0.01 (100 bins for the range [0,1]). Plot the mean-squared error between the measured distribution and the actual distribution using:

$$MSE = \frac{1}{B}\sum_{b=0}^{B-1} [\tilde{p}_X(x) - p(x)]^2 \tag{1.3}$$

   where B is the number of bins, $p(x)$ is the "true" distribution (a uniform distribution in this case), and is the estimate of the distribution. Obviously, for $N < B$, some of the bins will be empty. Does that remind you of the exam problem? Estimate the pdf for $N = [1, 1e6]$, and plot $MSE[N]$ using the same linear/log scale as above. Explain your findings. Plot the pdfs for N = 101, 103, and 106 and compare/contrast them.

2 Approach and Results

A uniform distribution is described by equation 2.1 below. The variance of a uniform distribution was calculated by finding the second moment of the distribution and subtracting it from the first squared. The result is displayed below.

$$p_X(x) = \begin{cases} \frac{1}{b-a} & : \quad a \leq x \leq b \\ 0 & : \quad otherwise \end{cases} \tag{2.1}$$

$$\sigma_X^2 = E[X]^2 - E[X^2] = \frac{1}{2}(b+a) - \int_a^b x^2 p_x(x) dx = \frac{1}{12}(b-a)^2 \tag{2.2}$$

Instead of utilizing MATLAB for calculations, python coupled with numpy and matplotlib was utilized. The script `ca_06.py` generates a random uniformly distributed vector (x) of 1×10^6 samples. Equations 1.1 to 1.3 are then run on the vector. The results are stored to file. An additional script was written, `ca_06_publishing.py`, to load the data, generate the requested histograms, and the error plots. The scripts were separated to run the bulk of the calculation on the electrodata server provided by Temple.

The results are listed below. Figures 2.1 to 2.3 display the error between the estimated mean, variance and pdf, and their actual values for a uniform distribution. The abscissa is base-10 logarithmically scaled and represents the number of samples taken from x for the calculated. The error for figure 2.1 and figure 2.2 is scaled linearly. The MSE between the estimated and expected pdf is scaled logarithmically.

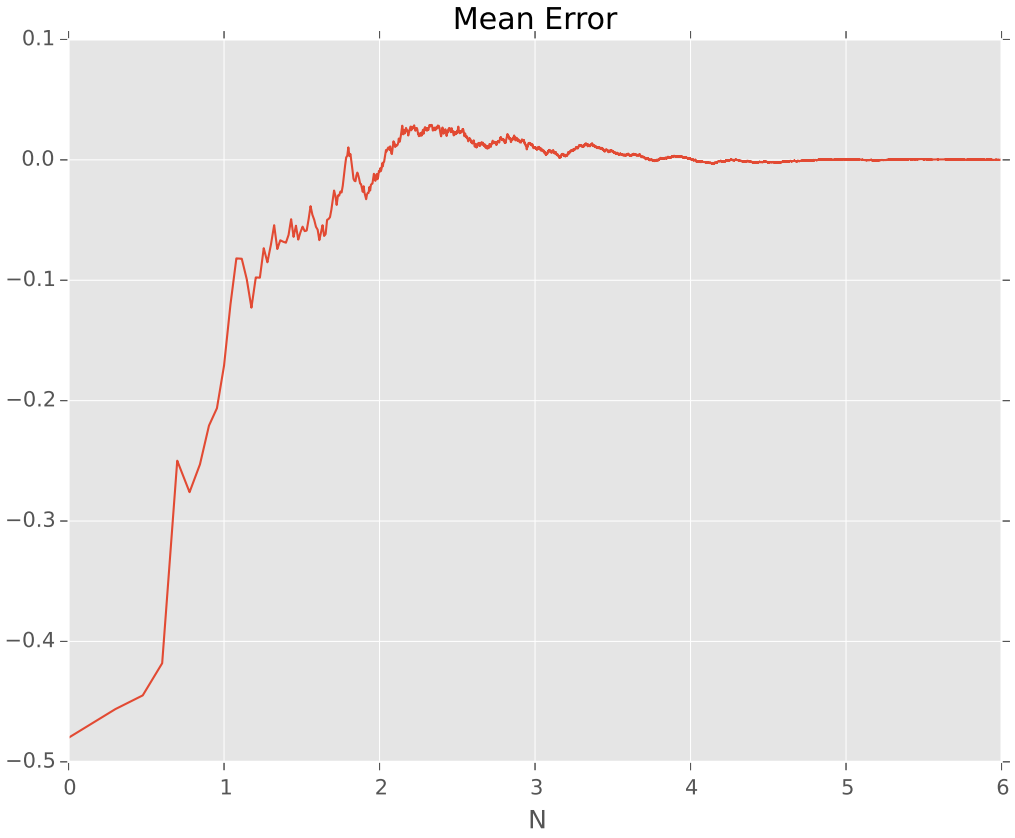

Figure 2.1: $Err[\tilde{\mu}_X](N)$, $N = [1, 1e6]$

Figure 2.2: $Err[\tilde{\sigma}^2_X](N)$, $N = [1, 1e6]$

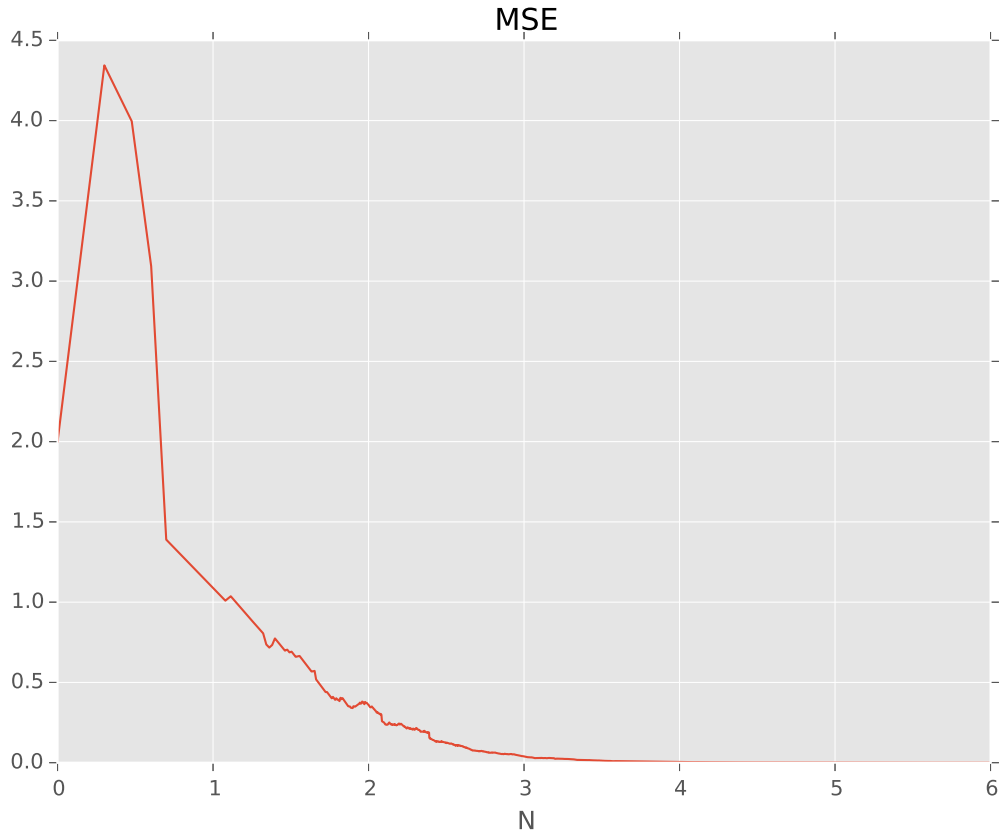

Figure 2.3: $Err[MSE](N)$, $N = [1, 1e6]$

Figure 2.4 below illustrates the histogram or pdf of the data when 10, 1,000 and 1,000,000 samples are taken from x. The abscissa is represents the possible values of the vector, and the y-axis represents the probability of each value.

For a uniform distribution with 100 bins, the probability of each bin should be 0.01. As the number of samples increases, the histograms approach the actual distribution. This is reflected visually in the three plots below, but also through error of the metrics calculated above. As N increases, the error between the metrics decrease because the sample space, with more data, is more reflective of the uniform distribution.

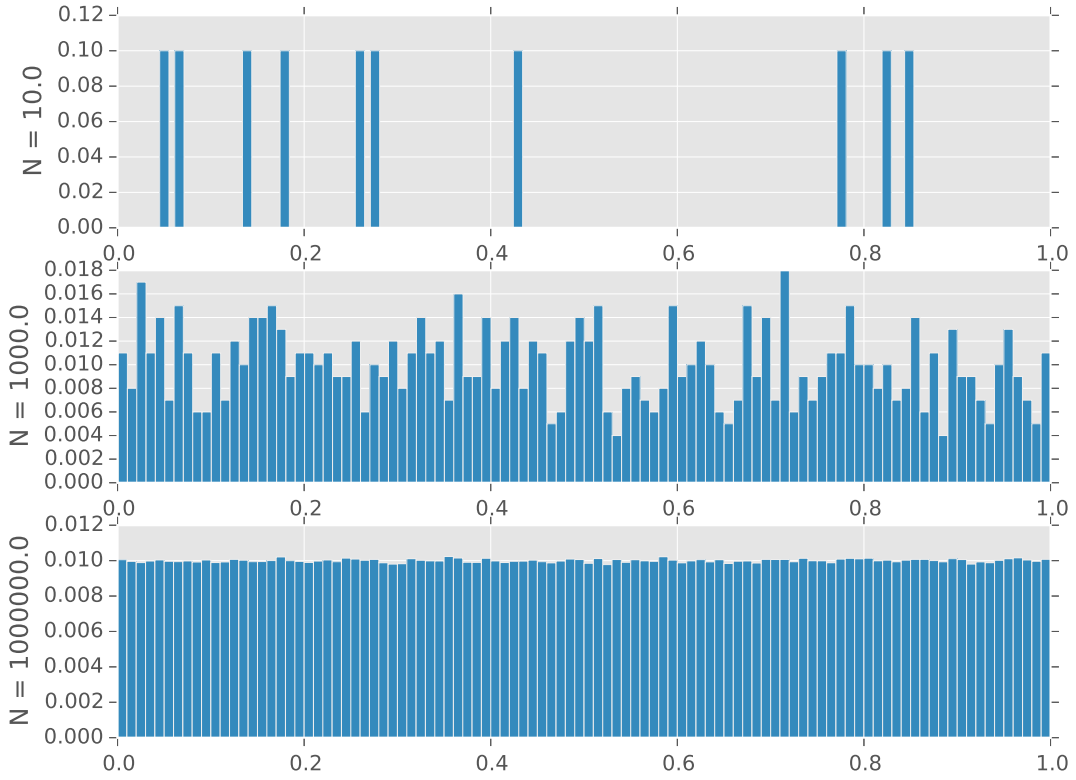

Figure 2.4: Histogram of random uniform distributed vector, $p_x(N)$, of N elements.

3 Python Code

```
# ca_06.py
import numpy as np
import matplotlib.pyplot as plt
from numba import jit


plt.style.use('ggplot')

# GENERATE RANDOM VECTOR [0, 1]
N = 1e3
MAX = 1.0
MEAN = MAX/2
VAR = (1./12)*(1 - 0)**2
BINS = 100
PDF = 1./(MAX*N)


x = np.random.random_integers(0, MAX*N, N)/N


# DEFINE FUNCTIONS
```

```
@jit
def xMean(x, N):
    '''
    x = vector of random values.
    N = index to which mean is calculated to.
    '''
    m = np.mean(x[:N])
    return m


@jit
def xVar(x, N):
    '''
    x = vector of random values.
    N = index to which variance is calculated to.
    '''
    v = np.var(x[:N])
    return v


@jit
def xPDF(x, N):
    pdf, binEdges = np.histogram(x[:N], bins=BINS, density=True)
    return pdf, binEdges

MeanError = np.zeros(N)
VarError = np.zeros(N)
MSE = np.zeros(N)


index = range(1, len(x)+1)
for i in index:
    MeanError[i-1] = xMean(x, i) - MEAN
    VarError[i-1] = xVar(x, i) - VAR
    MSE[i-1] = (1./BINS)*sum((xPDF(x, i)[0] - PDF)**2)
    print(i)

# SAVE Data
FILE1 = 'mean_error'
FILE2 = 'variance_error'
FILE3 = 'mse'
np.save(FILE1, MeanError)
np.save(FILE2, VarError)
np.save(FILE3, MSE)


# PLOTS

plt.figure(1)    # mean error
Lindex = np.log(index)/np.log(10)
plt.plot(Lindex, MeanError)
```

```
plt.title('Mean Error')
plt.xlabel('N')

plt.figure(2)
plt.plot(Lindex, VarError)
plt.title('Variance Error')
plt.xlabel('N')

plt.figure(3)
plt.plot(Lindex, np.log(MSE)/np.log(10))
plt.title('MSE')
plt.xlabel('N')
```

```
'''
ca_06_publishing.py
Loads data from ca_06.py to plot MSE, variance error
and mean error.
'''
import numpy as np
import matplotlib.pyplot as plt
from numba import jit

plt.style.use('ggplot')
plt.close('all')

# GENERATE RANDOM VECTOR [0, 1]
N = 1e6
MAX = 1.0
MEAN = MAX/2
VAR = (1./12)*(1 - 0)**2
BINS = 100
PDF = 1./(MAX*N)


x = np.random.random_integers(0, MAX*N, N)/N


'''
DEFINE FUNCTIONS
'''
@jit
def xMean(x, N):
    '''
    x = vector of random values.
    N = index to which mean is calculated to.
    '''
    m = np.mean(x[:N])
    return m


@jit
```

```
def xVar(x, N):
    '''
    x = vector of random values.
    N = index to which variance is calculated to.
    '''
    v = np.var(x[:N])
    return v


@jit
def xPDF(x, N):
    pdf, binEdges = np.histogram(x[:N], bins=BINS, density=False)
    return pdf, binEdges


'''
Plot PDFs
'''
index = [1e1, 1e3, 1e6]
subplots = [1, 2, 3]
for i, j in zip(index, subplots):
    pdf, binedges = xPDF(x, i)
    plt.figure(4)
    plt.subplot(3, 1, j)
    plt.bar(binedges[:-1], pdf/i, width=0.01)
    plt.xlim(0, 1)
    plt.ylabel(r'N = '+str(i))


'''
Load Data from ca_06.py
'''
MeanError = np.load('mean_error.npy')
VarError = np.load('variance_error.npy')
MSE = np.load('mse.npy')

index = linspace(1, N, N)

plt.figure(1)    # mean error
Lindex = np.log(index)/np.log(10)
plt.plot(Lindex, MeanError)
plt.title('Mean Error')
plt.xlabel('N')

plt.figure(2)
plt.plot(Lindex, VarError)
plt.title('Variance Error')
plt.xlabel('N')

plt.figure(3)
plt.plot(Lindex, np.log(MSE)/np.log(10))
```

```
plt.title('MSE')
plt.xlabel('N')
```

4 CONCLUSIONS

As a sample space increases with valid data, the space better reflects its governing distribution. This is clearly illustrated through the histograms and the three error plots. Therefore, if the sample space is not sufficiently large any metrics calculated to represent the space would not reflect the governing distribution.