Noise Generation

Jean-Francois Arbour

September 14, 2022

Contents

1 Gaussian Noise generator

Two simple ways of generating samples from a Gaussian distribution:

1.0.1 Box and Muller method

[Box1958ANO] Let u and v be independant samples chosen from a uniform distribution on $(0, 1) \subset \mathbb{R}$ and (R, Θ) corresponding polar coordinates. Let

$$z_0 = R\cos(\Theta) = \sqrt{-2\log u}\cos(2\pi v) \tag{1}$$

and

$$z_1 = R\sin(\Theta) = \sqrt{-2\log u}\sin(2\pi v). \tag{2}$$

Then z_0 and z_1 are independant random variables both distributed a standard normal distribution of mean 0 and variance 1. Generate samples from a standard normal distribution with mean μ and variance σ^2 by considering the random variables $Z_{0,1} = \mu + \sigma * z_{0,1}$.

1.0.2 Marsaglia method

Let us first establish the geometric picture for this result: Consider the open square Sq = (-1, 1) $\times(-1, 1)$

of side length 2 in the plane, along with its inscribed open disk D = { (x, y) | $x^2 + y^2 < 1$}.

For u and v two independant random variables with values in the plane, we will denote in what follows by

$$\rho = \rho(u, v) = u^2 + v^2$$

Consider any two uniformly distributed and independant variables U, V with values in $\mathbb{S}_{\text{II}}$ but constrained by the condition that $0 < \rho(U, V) < 1$ with probability 1. Then with

$$z_0 = \frac{U}{\sqrt{\rho}} * \sqrt{\frac{-2\,\log \rho}{\rho}} \tag{3a}$$

and

$$z_1 = \frac{V}{\sqrt{\rho}} * \sqrt{\frac{-2\,\log \rho}{\rho}} \tag{3b}$$

Notice that

$$\frac{U}{\sqrt{\rho}} = \cos(\theta)$$

and similarly,

$$\frac{V}{\sqrt{\rho}} = \sin(\theta).$$

So by writing $R = \sqrt{\frac{-2\log \rho}{\rho}}$, we can write

$$\begin{cases} z_0 &= R\cos(\theta), \\ z_1 &= R\sin(\theta). \end{cases} \tag{4}$$

[[**MR172441**]] With the notation as above, z_0 and z_1 form two independant variables distributed along a standard normal distribution with mean 0 and standard deviation 1. By scaling them to

$$\begin{cases} Z_0 &= \mu + \sigma z_0, \\ Z_1 &= \mu + \sigma z_1, \end{cases} \tag{5}$$

we obtain two independant variables distributed along a normal distribution with mean μ and variance σ^2.

1.1 Test

1.1.1 Box and Muller method

```
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np
```

As a test, we'll generate samples from a normal distribution using numpy's blackbox:

```
mu = 0.0
sigma = 1.0
number_of_samples = 200
normal_samples = np.random.default_rng().normal(mu, sigma, number_of_samples)
```

Sanity checks:

```
mean_near_mu = abs( mu - np.mean(normal_samples) )
std_near_sigma = abs(sigma - np.std(normal_samples, ddof=1))

assert(mean_near_mu < 0.2)
assert(std_near_sigma < 0.2)
```

Plot the above

```
fig=plt.figure(figsize=(4,2))
count, bins, ignored = plt.hist(normal_samples, 30, ec='orange', fc='steelblue', densit
plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) *
               np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
         linewidth=2, color='violet', alpha=0.5)
fig.tight_layout()
fname="images/numpyExpectationsBarChart.png"
plt.show(fname)
#fname
```

Compare with data generated from our own gaussian sampling method:

```
import math
import random

def getGaussianPair():
```

```
    u = random.random()
    v = random.random()
    while u == 0:
        u = random.random()
    sqrt_minustwo_logu = math.sqrt(-2 * math.log(u))
    two_pi_v = 2 * math.pi * v
    z_0 = sqrt_minustwo_logu * math.cos(two_pi_v)
    z_1 = sqrt_minustwo_logu * math.sin(two_pi_v)
    return [z_0, z_1]

def sample():
    if (self.has_spare):
      self.sample.append(self.spare)
      self.has_spare = False
    else:
      self.spare, result = getGaussianPair()
      self.has_spare = True
      self.sample.append(result)

def getSample(sample_size):
    result = []
    for _ in range(sample_size):
        result.append(getGaussianPair()[0])
        print(result)

sample = getSample(number_of_samples)

fig=plt.figure(figsize=(4,2))
count, bins, ignored = plt.hist(sample, 30, ec='orange', fc='steelblue', density=True,
plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) *
               np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
         linewidth=2, color='violet', alpha=0.5)
fig.tight_layout()
fname="images/expectationsBarChart.png"
plt.show(fname)
fname
```