Cooperative Communication and Networks Assignment 1

Instructor: Kuang-Hao (Stanley) Liu
Student: Shao-Heng (Paul) Chen
March 1, 2022

– Numerical Questions –

Q1) SNR

A receiver measured the received SNR equal to 5 dB and the thermal noise power equal to $N_0 = 10^{-14}$ Watts. Find the received power strength in Watts?

(1)

by definition $\boxed{SNR(dB) = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right)}$

bring the info. of the problem description in above equation

then we have $5\ dB = 10 \cdot \log_{10}\left(\frac{P_{signal}}{10^{-14}}\right)\ dB$

$$\Rightarrow \log_{10}\left(\frac{P_{signal}}{10^{-14}}\right) = 0.5 \qquad \therefore P_{signal} = 10^{-13.5}\ W \ \#$$

$$\Rightarrow 10^{0.5} = \frac{P_{signal}}{10^{-14}} = 10^{-(-14)} \cdot P_{signal}$$

finally, we get the received power strength P_{signal} be equal to $10^{-13.5}$ Watts, which is approximately equal to $3.1623 \cdot 10^{-14}$ Watts. #

Q2) Shadow fading

The path loss due to distance is deterministic but it becomes non-deterministic when shadowing comes into play. Find the probability that the path loss due to both distance and log-normal shadowing with zero mean and variance 5^2 dB at distance of 500 m is below 118 dB, given the path-loss exponent $\nu = 4.35$ and the reference distance $d_0 = 1$ m.

(2) path loss $P_L(dB) = \frac{P_t}{P_r}(dB) = 10\log(\frac{P_t}{P_r})$

$= -10\cdot\log K + 10\nu\cdot\log(\frac{d}{d_0}) - 10\cdot\log X < 118\ dB$

$\xrightarrow{K=1}$ $= 0 + 43.5\cdot\log 500 - 10\log X < 118$

$\nu = 4.35$

$d = 500m$

$d_0 = 1m$ $\doteq 117.41 - 10\log X < 118$

$10\log X > 0.59$, $10\cdot\log X \triangleq Z$ is a Gaussian r.v.

$\Rightarrow P(Z > z) = Q(z)$

$P(Z > 0.59) = Q(0.59) \doteq 0.2776$ #

Q3) Distribution

Suppose X is an indicator variable that characterizes whether a transmitter has sent a bit successfully.

$$X = \begin{cases} 1, & if\ a\ bit\ is\ sent\ in\ error \\ 0, & otherwise \end{cases}$$

a) Let P_e denote the transmission error probability. Derive the mean and variance of X in terms of P_e.

(a) from the description, we can tell that X is a Bernoulli distribution $\therefore P(X=x) = p^x(1-p)^{1-x} = \begin{cases} P_e, & X=1 \text{ error occured} \\ 1-P_e, & X=0 \text{ error free} \end{cases}$

with the pmf of r.v. X we can find

$E[X] = \sum_x xP(X=x) = 0\cdot(1-P_e) + 1\cdot P_e = P_e$ #

$$var[X] \triangleq E[(X-E[X])^2] = \sum_x (x-E[X])^2 \cdot P(X=x)$$
$$= (0-P_e)^2 \cdot (1-P_e) + (1-P_e)^2 P_e$$
$$= -P_e^2(1-P_e) + (1-P_e)^2 P_e$$
$$= -P_e^2 + P_e^3 + P_e - P_e^3 = P_e - P_e^2$$
$$= P_e(1-P_e)$$
#

b) Suppose the transmitter sends n bits among which the number of erroneous bits is denoted as Y. Derive the mean and variance of Y.

(b) we can perceive event Y as repeating a Bernoulli trial for n times, which is a binomial distribution.

$$Y = X_1 + X_2 + \cdots + X_n$$
$$E(Y) = E(X_1 + X_2 + \cdots + X_n) = E(X_1) + \cdots + E(X_n) = n \cdot P_e$$
#
$$var(Y) = var(X_1 + X_2 + \cdots + X_n)$$
$$\xrightarrow{\because \text{ all } X \text{ are iid}} = var(X_1) + \cdots + var(X_n)$$
$$= n \cdot P_e(1-P_e)$$
#

c) The average rate of transmission errors can be found as $\widehat{P_e} = Y/n$ (which is also known as the sample mean or the empirical mean). Derive the mean and variance of $\widehat{P_e}$.

(c) sample mean and sample variance of $\widehat{P_e}$

$$E(\widehat{P_e}) = E(\frac{Y}{n}) = \frac{1}{n}E(Y) = \frac{1}{n} \cdot nP_e = P_e$$
#
$$var(\widehat{P_e}) = var(\frac{Y}{n}) = \frac{1}{n^2} \cdot var(Y) = \frac{1}{n^2} \cdot nP_e(1-P_e) = \frac{1}{n}P_e(1-P_e)$$
#

d) When n goes large, $\widehat{P}_e$ can be approximately normal. Derive the 95% confidence interval for $\widehat{P}_e$.

(d) 95% confidence interval ⇒ $\alpha = 0.05$, $Z_{\frac{\alpha}{2}} = 1.96$

$$\left[\bar{X} - Z_{\frac{\alpha}{2}} \cdot \frac{S}{\sqrt{n}},\ \bar{X} + Z_{\frac{\alpha}{2}} \cdot \frac{S}{\sqrt{n}}\right]$$

$$\bar{X} = E(\widehat{P}_e) = P_e$$

$$S^2 = \frac{1}{n-1} \cdot \sum_{x} [X - E(X)]^2 = \frac{n}{n-1} \cdot var(\widehat{P}_e) = \frac{1}{n-1} P_e(1-P_e)$$

$$\Rightarrow \left[P_e \pm 1.96 \cdot \frac{P_e(1-P_e)}{(n-1)\sqrt{n}}\right]$$

*

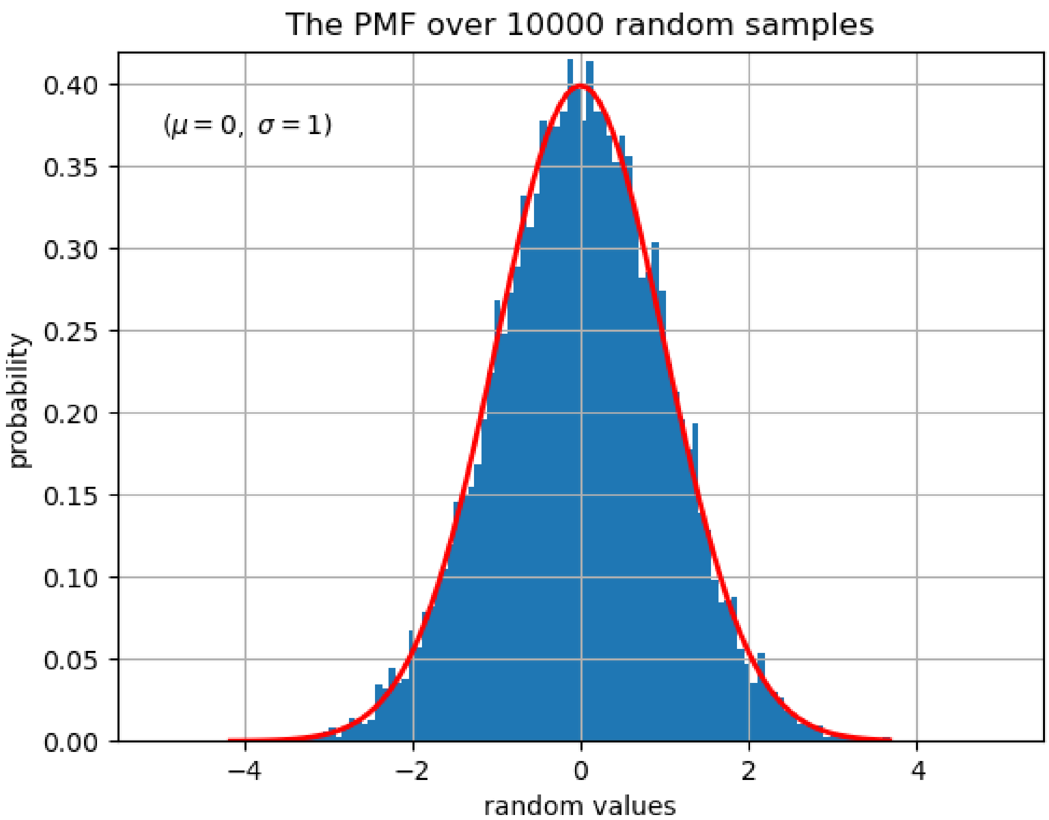
– Simulation –

Q4) Random Generator

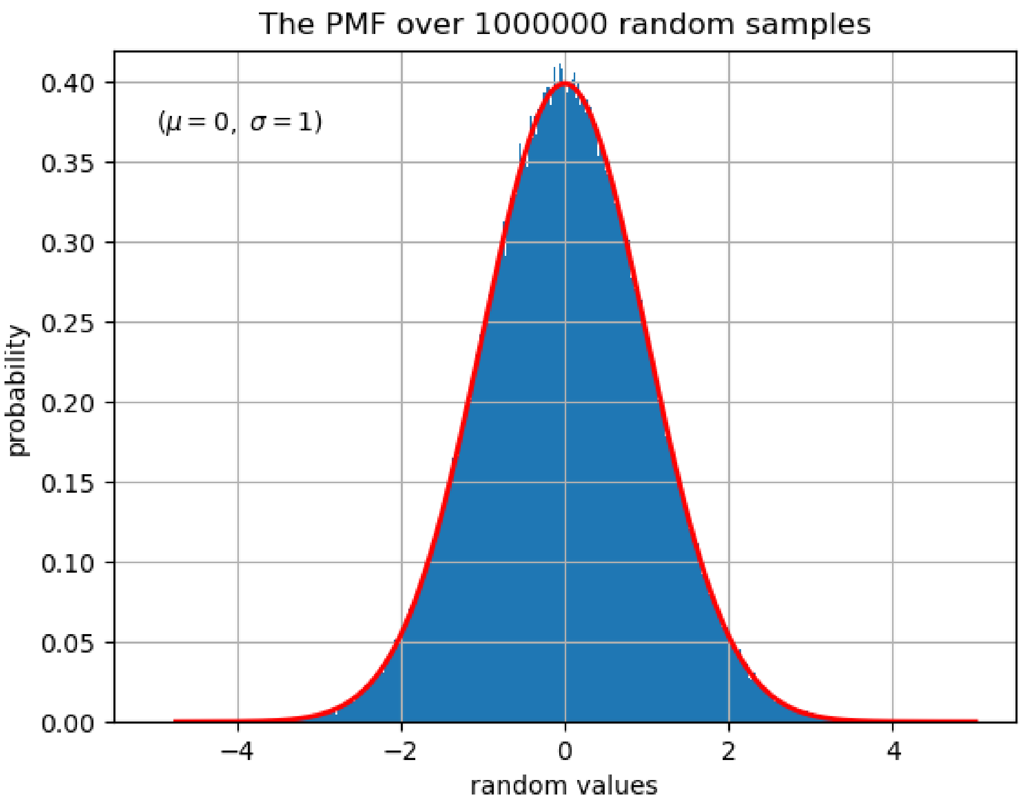
Write a computer program to generate the normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$.

a) Verify the accuracy of your random generator by plotting the probability mass function (PMF) of empirical results and comparing it with the theoretical one. In the figure, clearly indicate the simulation and theoretical curves.

- case1. 10000 samples, empirical results plotted as the blue bars, and theoretical result is the red curve



- case2. 1000000 samples, empirical results plotted as histogram, theoretical result is the red curve



b) Describe the programming software you use (name and version) and what commands/functions are used to generate random samples. Also, show the number of samples you generated to obtain the statistics.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 28 08:23:56 2022
@author: Paul
@file:   normal.py
@dependencies:
    conda env:  pytest
    python:     3.7.11
    numpy:      1.21.2
    matplotlib: 3.5.0
"""
import numpy as np
import matplotlib.pyplot as plt
import os
```

```
# hyper-parameters
SAMPLE_SIZE = 10000 # total number of smaples, which is the output shape
BIN_SIZE = 100      # the number of equal-width bins in the range

"""
# numpy.random.normal(loc=0.0, scale=1.0, size=None)
# params:
#   loc:     float(s)          -Mean, centre
#   scale:   float(s)          -Standard deviation, spread
#   size:    int, (ints..)     -Output shape
# returns:
#   samples: ndarray or scalar -Drawn samples
"""
# mean = 0 and standard deviation = 1, which means variance also equal to 1
mu, sigma = 0.0, 1.0
# store all the samples of a normal distribution to samples
samples = np.random.normal(loc=mu, scale=sigma, size=SAMPLE_SIZE)
print(type(samples)) # <class 'numpy.ndarray'>

# verify the mean and the variance (standard deviation)
print('mean difference:', abs(mu - np.mean(samples)))
# delta degrees of freedom, but I can't really tell the difference lol
# set ddof=0 return population standard deviation, biased estimation
# set ddof=1 return sample standard deviation, unbiased estimation
print('standard deviation difference:', abs(sigma - np.std(samples, ddof=1)))

"""
# matplotlib.pyplot.hist(x, bins=None, density=False, stacked=False, ...)
# params:
#   x:       (n,) array(s)     -Input values, can take single array or sequence of arrays
#   bins:    int, seq. or str  -If int, it defines the number of equal-width bins in range
#   density: bool              -If True, area under the histogram integrates to 1
#   stacked: bool              -If True, multiple data are stacked on top of each other
# returns:
#   counts:  array(s)          -The values of the histogram bins
#   bins:    array             -The edges of the bins
#   patches: BarContainer      -Just ignored, when there's only one input dataset
"""
```

```
plt.figure(1)
# plot the pmf of the samples
count, bins, ignored = plt.hist(x=samples, bins=BIN_SIZE, density=True, stacked=True)
# plot the actual normal distribution
plt.plot(bins, 1/(np.sqrt(2*np.pi) * sigma) * np.exp(-(bins - mu)**2 / (2*(sigma**2))),
         linewidth=2, color='r')

plt.xlabel('random values')
plt.ylabel('probability')
# format string f'...{SAMPLE_SIZE}...' will substitute variable in runtime
plt.title(f'The PMF over {SAMPLE_SIZE} random samples')
plt.text(-5, 0.37, r'$(\mu = 0,\ \sigma = 1)$') # text(x-coord, y-coord, '{text display}')
# be very careful on setting axis, cause it will dramatically affect how we perceive data
plt.axis([-5.5, 5.5, 0, 0.42])
plt.grid(True)

# first check whether the specified path is an existing file
path = './normal_distribution/normal.png'
# if no file exist, then save current file
if os.path.isfile(path) is False:
    plt.savefig('./normal_distribution/normal.png')

plt.show()
```

references

[1] confidence interval (https://en.wikipedia.org/wiki/Confidence_interval)

[2] sample variance (https://en.wikipedia.org/wiki/Variance#Sample_variance)

[3] WolframAlpha Q function calculator (https://www.wolframalpha.com/widgets/view.jsp?id=95784c6b00784691c55ea0a420fd7ee0)