ECE 595 HW5

Ruijie Song

Apr.1.2021

Exercise 1

ECE 595 HW5

1.

a) i) $g = h_1 = [\bullet, \bullet, \bullet, \bullet, \bullet, \bullet, \bullet, \bullet]$

ii) Agree on 3 out-sample pts : f_8

2 : f_7, f_6, f_4

1 : f_2, f_3, f_5

0 : f_1

b) i) $g = h_2 = [\circ, \circ, \circ, \circ, \circ, \circ, \circ, \circ]$

ii) Agree on 3 out-samples pts : f_1

2 : f_2, f_3, f_5

1 : f_7, f_6, f_4

0 : f_8

c) i) $g = [\circ, \bullet, \bullet, \circ, \bullet, \circ, \circ, \bullet]$

ii) Agree 3 out-sample pts : f_2

2 : f_1, f_6, f_4

1 : $f_3, \cancel{f_4}, f_5, f_8$

0 : f_7

d) i) $g = [\circ, \bullet, \bullet, \circ, \bullet, \bullet, \bullet, \circ]$

ii) Agree 3 pts : f_7

2 : f_3, f_5, f_8

1 : f_1, f_4, f_6

0 : f_2

Exercise 2

1. μ1 = μrand = μmin = 0.5

2.

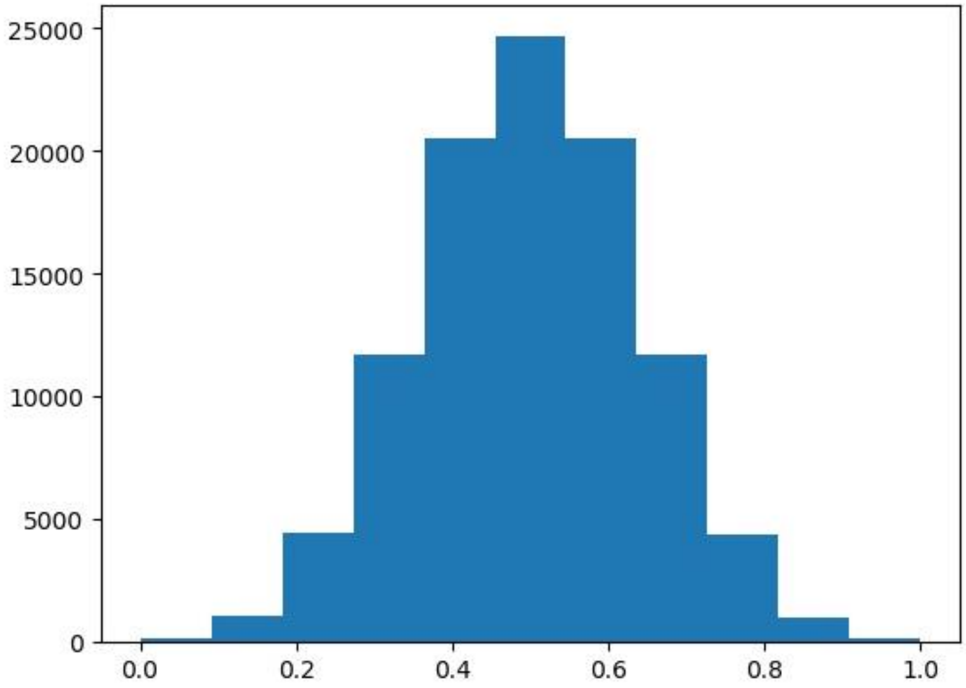


Figure 1. V1

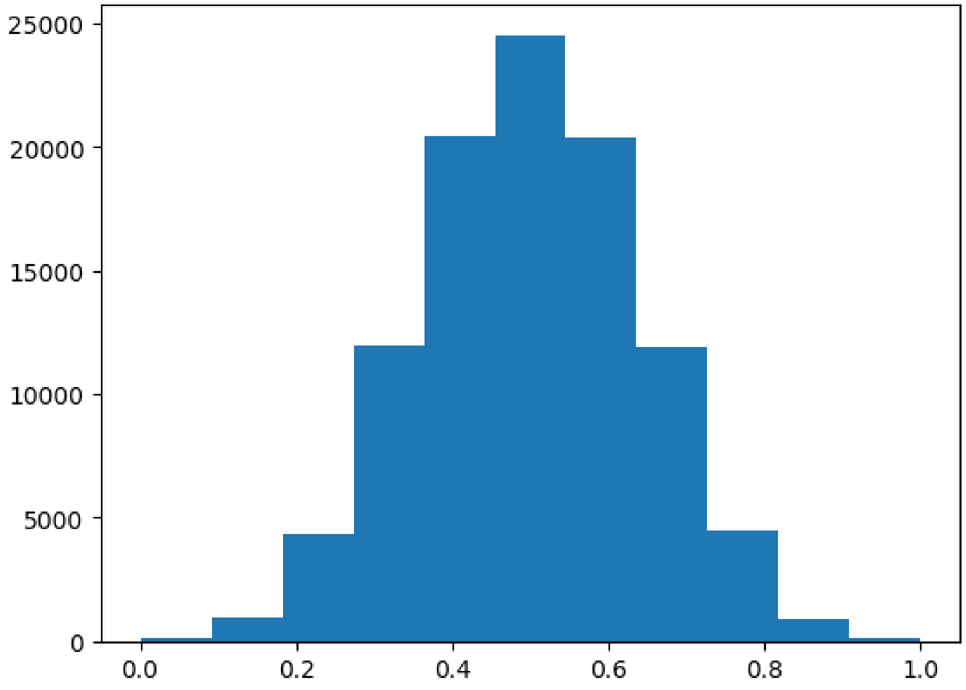


Figure 2. Vrand

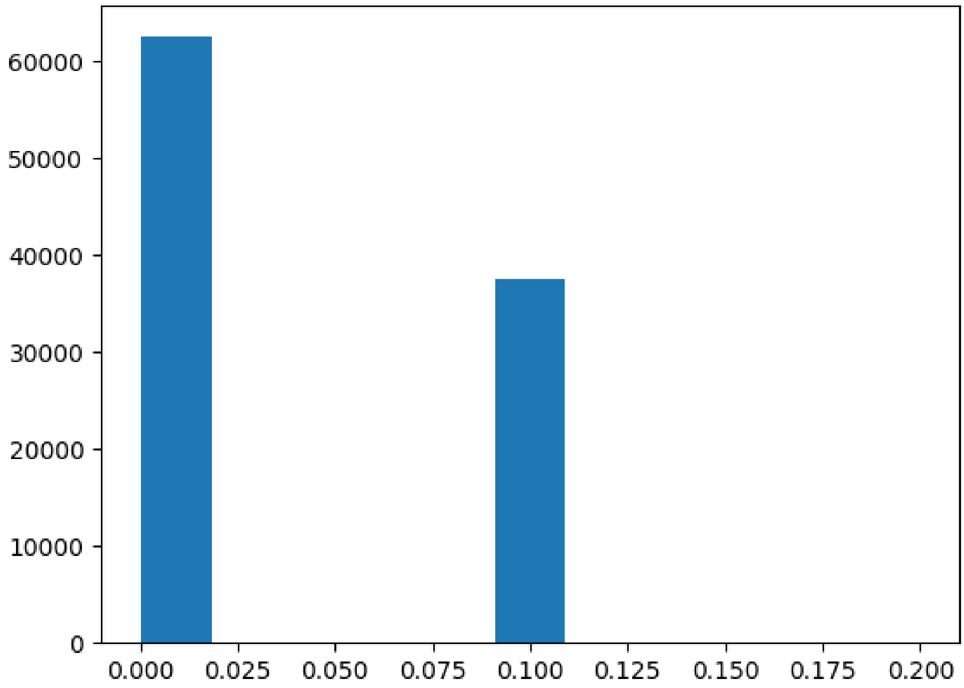
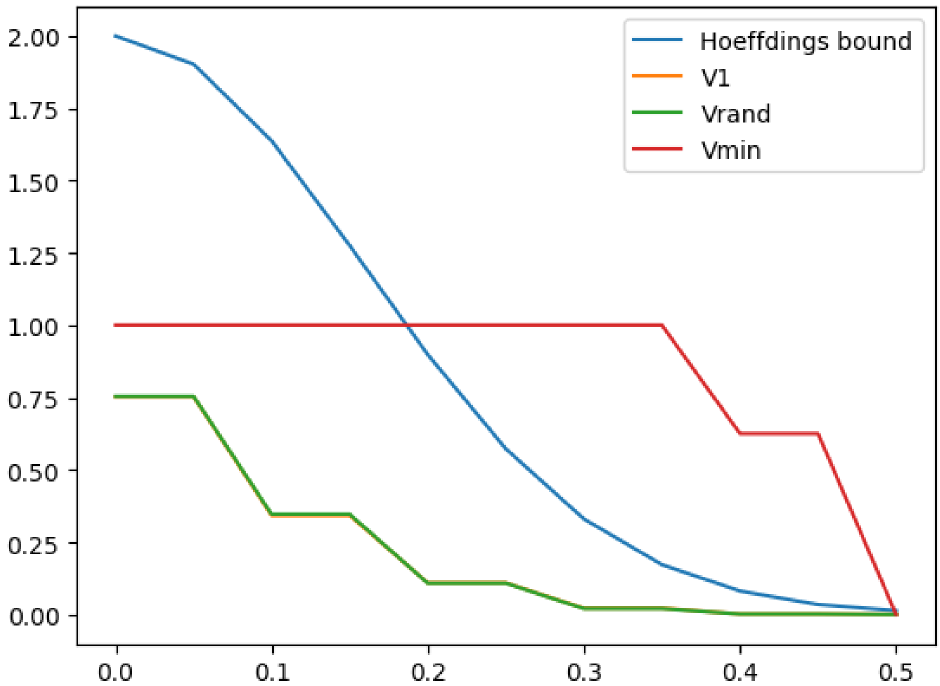


Figure 3. Vmin

3.



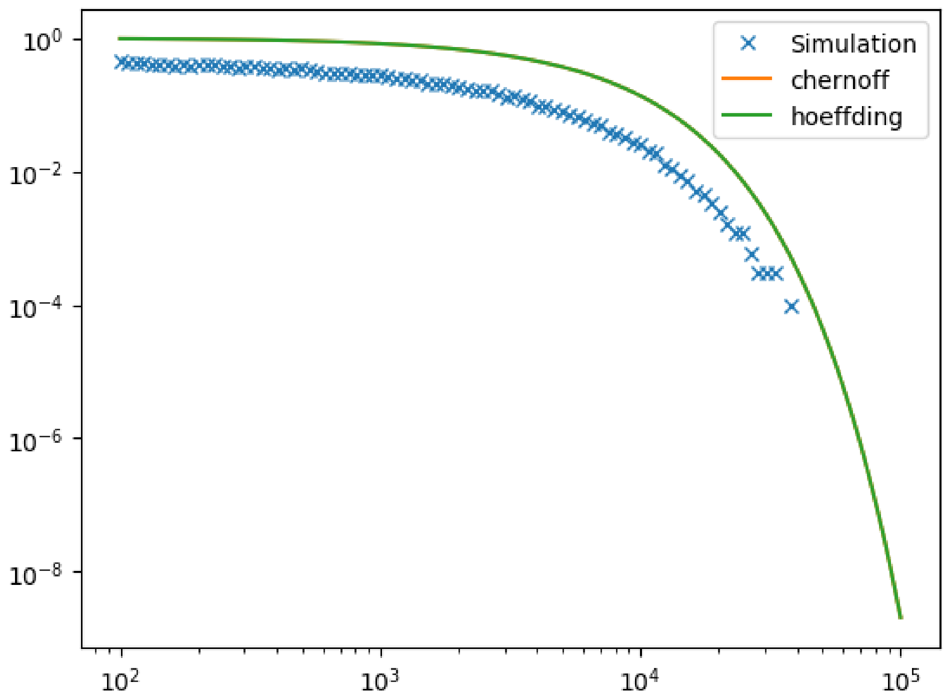
d.

The coin1 and coin_rand obey the Hoeffding's bound, and coin_min does not. That is because the coin_min is not independent of the samples. E[V1] = E[Vrand] = 0.5.

However, E[Vmin] = 0.04, which is far away from μ.

Exercise 3.

b.



a.

3. a) $\mathbb{P}[\bar{X}_N - \mu \ge \varepsilon] = \mathbb{P}[\bar{X}_N \ge \varepsilon + \mu] \le \dfrac{\mathbb{E}[e^{s\bar{X}_N}]}{e^{s(\varepsilon+\mu)}}$

$= e^{-s(\varepsilon+\mu) + \ln(M_{\bar{X}}(s))}$

$M_{\bar{X}}(s) = (\frac{1}{2} + \frac{1}{2} e^{s})^{N}$

$\Downarrow$

$\therefore \exp[-s(\varepsilon+\mu) + \ln[(\frac{1}{2} + \frac{1}{2} e^{s})^{N}]]$

$s^* = \underset{s>0}{\arg\min}\{\ \}$ $\downarrow$ $= \underset{s>0}{\arg\min}\{-s(\varepsilon+\mu) + \ln[(\frac{1}{2} + \frac{1}{2} e^{s})^{N}]\}$

$\frac{d}{ds}(\cdot) = 0$

$\therefore -(\varepsilon+\mu) + N\dfrac{e^{s}}{1+e^{s}} = 0 \qquad \cancel{N}\dfrac{e^{s}}{1+e^{s}} = \dfrac{\varepsilon+\mu}{\cancel{N}}$

$\dfrac{1+e^{s}}{e^{s}} = \dfrac{\cancel{N}\ 1}{\varepsilon+\mu} \qquad e^{-s} + 1 = \dfrac{N}{\varepsilon+\mu} \qquad e^{-s} = \dfrac{N}{\varepsilon+\mu} - 1 = \dfrac{N-\varepsilon-\mu}{\varepsilon+\mu}$

$+s^* = -\ln(\dfrac{N-\varepsilon-\mu}{\varepsilon+\mu})$

$\therefore e^{[\ln(\frac{N-\varepsilon-\mu}{\varepsilon+\mu})](\varepsilon+\mu) + \ln[(\frac{1}{2} + \frac{1}{2}(\frac{\varepsilon+\mu}{N-\varepsilon-\mu}))^{N}]}$

$= (\dfrac{N-\varepsilon-\mu}{\varepsilon+\mu})^{\varepsilon+\mu} \cdot (\dfrac{1}{2} + \dfrac{1}{2}(\dfrac{\varepsilon+\mu}{N-\varepsilon-\mu}))^{N}$

$= (\dfrac{N-\varepsilon-\frac{1}{2}}{\varepsilon+\frac{1}{2}})^{\varepsilon+\frac{1}{2}} \cdot (\dfrac{1}{2} + \dfrac{1}{2}(\dfrac{\varepsilon+\frac{1}{2}}{N-\varepsilon-\frac{1}{2}}))^{N}$

$= (\dfrac{N-\varepsilon-\frac{1}{2}}{\varepsilon+\frac{1}{2}})^{\varepsilon+\frac{1}{2}} \cdot (\dfrac{N-\cancel{\varepsilon}-\cancel{\frac{1}{2}}+\cancel{\varepsilon}+\cancel{\frac{1}{2}}}{2(N-\varepsilon-\frac{1}{2})})^{N}$

$= (\dfrac{N-\varepsilon-\frac{1}{2}}{\varepsilon+\frac{1}{2}})^{\varepsilon+\frac{1}{2}} \cdot (\dfrac{N}{2(N-\varepsilon-\frac{1}{2})})^{N}$

$= 2 \wedge \log_2 [(\dfrac{N-\varepsilon-\frac{1}{2}}{\varepsilon+\frac{1}{2}})^{\varepsilon+\frac{1}{2}} \cdot (\dfrac{N}{2(N-\varepsilon-\frac{1}{2})})^{N}]$

$2 \wedge ((\varepsilon+\frac{1}{2})\log_2(\dfrac{N-\varepsilon-\frac{1}{2}}{\varepsilon+\frac{1}{2}}) + N\log_2\dfrac{N}{2(N-\varepsilon-\frac{1}{2})})$

$= 2 \wedge ((\varepsilon+\frac{1}{2})[\log_2(N-\varepsilon-\frac{1}{2}) - \log_2(\varepsilon+\frac{1}{2})] + N[\log_2 N - 1 - \log_2(N-\varepsilon-\frac{1}{2})])$

$N = 1$

$= 2 \wedge ((\varepsilon+\frac{1}{2})[\log_2(\frac{1}{2}-\varepsilon) - \log_2(\varepsilon+\frac{1}{2})] + [-1 - \log_2(\frac{1}{2}-\varepsilon)])$

$= 2 \wedge ((\varepsilon+\frac{1}{2})\log_2(\frac{1}{2}-\varepsilon) - (\varepsilon+\frac{1}{2})\log_2(\varepsilon+\frac{1}{2}) - 1 - \log_2(\frac{1}{2}-\varepsilon))$

$= 2 \wedge (-1 - (\varepsilon+\frac{1}{2})\log_2(\varepsilon+\frac{1}{2}) - (\frac{1}{2}-\varepsilon)\log_2(\frac{1}{2}-\varepsilon))N$

$\therefore \beta =$ ~~~~ $\qquad \mu = 0.5$

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Apr  1 01:26:25 2021

@author: 11327
"""

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy.matlib

# Exercise 2
# b
'''
coin_num = 1000
flip_times = 10

run = 100000
V1 = np.zeros(run)
Vrand = np.zeros(run)
Vmin = np.zeros(run)
for i in range(run):
    COIN = np.random.randint(0,2,(flip_times,coin_num))
    # count the number of heads of each coin
    head = np.zeros(coin_num)
    for kk in range(flip_times):
        for ii in range(coin_num):
            if COIN[kk,ii] == 1:
                head[ii] = head[ii] + 1
    # calculating V1
    V1[i] = head[0] / flip_times
    # calculating Vrand
    rand = np.random.randint(0,1000)
    Vrand[i] = head[rand] / flip_times
    # calculating Vmin
    Vmin[i] = min(head) / flip_times

# plot histogram
plt.figure()
plt.hist(V1,bins=11)

plt.figure()
plt.hist(Vrand,bins=11)

plt.figure()
plt.hist(Vmin,bins=11)
'''
'''
# c
N = 10
epsilon = np.linspace(0,0.5,11)
Hb = 2 * np.exp(-2 * epsilon**2 * N) # Hoeffding's bound
# plot Hoeffding's bound
plt.figure()
l1, = plt.plot(epsilon,Hb)
# plot P(|V1-miu1| > epsilon)
miu = 0.5
P1 = np.zeros(len(epsilon))
for j in range(len(epsilon)):
    count = 0
```

```
    for i in range(run):
        if np.abs(V1[i]-miu) > epsilon[j]:
            count = count + 1
    P1[j] = count / run
l2, = plt.plot(epsilon,P1)
# plot P(|Vrand-miu| > epsilon)
Prand = np.zeros(len(epsilon))
for j in range(len(epsilon)):
    count = 0
    for i in range(run):
        if np.abs(Vrand[i]-miu) > epsilon[j]:
            count = count + 1
    Prand[j] = count / run
l3, = plt.plot(epsilon,Prand)
# plot P(|Vmin-miu| > epsilon)
Pmin = np.zeros(len(epsilon))
for j in range(len(epsilon)):
    count = 0
    for i in range(run):
        if np.abs(Vmin[i]-miu) > epsilon[j]:
            count = count + 1
    Pmin[j] = count / run
l4, = plt.plot(epsilon,Pmin)
plt.legend(handles=[l1,l2,l3,l4],labels = ['Hoeffdings bound','V1','Vrand','Vmin'],loc='upper r
plt.show()
'''
# mean1 = np.mean(V1)
# meanrand = np.mean(Vrand)
# meanmin = np.mean(Vmin)

# 3
# b
# Sum of Bernoulli = Binomial
p = 0.5
epsilon = 0.01
Nset = np.round(np.logspace(2,5,100)).astype(int)
x = np.zeros((10000,Nset.size))
prob_simulate  = np.zeros(100)
prob_chernoff = np.zeros(100)
prob_hoeffding = np.zeros(100)
beta = 1+(0.5+epsilon)*np.log2(0.5+epsilon)+(0.5-epsilon)*np.log2(0.5-epsilon)
for i in range(Nset.size):
  N = Nset[i]
  x[:,i] = stats.binom.rvs(N, p, size=10000)/N
  prob_simulate[i]  = np.mean((x[:,i]-p>=epsilon).astype(float))
  # prob_chebyshev[i] = p*(1-p)/(N* (epsilon**2) )
  prob_chernoff[i]  = 2**(-beta*N)
  prob_hoeffding[i] = np.exp(-2*N*epsilon**2)

plt.figure()
l1, = plt.loglog(Nset, prob_simulate,'x')
l2, = plt.loglog(Nset, prob_chernoff)
l3, = plt.loglog(Nset, prob_hoeffding)
plt.legend(handles=[l1,l2,l3],labels=['Simulation','chernoff','hoeffding'])
```