Fathan Ananta Nur
236-D8749

44135 Applied Information Theory

Assignment 2 (Revision)

In these experiments, the parameters below will be used:

```
Length (N) = 1000000

Initial value = 0.752352

Parameter c = {0.3, 0.4}

Parameter t = {0.3, 0.4}

Iteration n = 60
```

1. Generate memoryless information binary sequences by skew Bernoulli maps. Next compute the following probabilities.

$$\mathrm{P}(0), \mathrm{P}(1), \mathrm{P}(00), \mathrm{P}(01), \mathrm{P}(10), \mathrm{P}(11)$$
$$\mathrm{P}(\mathrm{S}_0|\mathrm{S}_0) = \frac{\mathrm{P}(00)}{\mathrm{P}(0)}, \mathrm{P}(\mathrm{S}_1|\mathrm{S}_0) = \frac{\mathrm{P}(01)}{\mathrm{P}(0)},$$
$$\mathrm{P}(\mathrm{S}_0|\mathrm{S}_1) = \frac{\mathrm{P}(10)}{\mathrm{P}(1)}, \mathrm{P}(\mathrm{S}_1|\mathrm{S}_1) = \frac{\mathrm{P}(11)}{\mathrm{P}(1)}.$$

Then compare them to the theoretical (designed) values. Also, confirm that the binary sequences are not a memoryless source when $c \neq t$.

These tables below are the result of experiments using combination of parameter c and t written above.

Table 1. Comparison of c = 0.3 with Different Value of t

c = t = 0.3

	Computed value	Theoretical value	Formula
P(0)	0.299	0.3	c
P(1)	0.701	0.7	$1-c$
P(00)	0.089	0.09	c^2
P(01)	0.210	0.21	$c(1-c)$
P(10)	0.210	0.21	$c(1-c)$
P(11)	0.490	0.49	$(1-c)^2$
$\mathrm{P}(\mathrm{S}_0\|\mathrm{S}_0)$	0.298	0.3	c
$\mathrm{P}(\mathrm{S}_0\|\mathrm{S}_1)$	0.300	0.3	c
$\mathrm{P}(\mathrm{S}_1\|\mathrm{S}_0)$	0.702	0.7	$1-c$
$\mathrm{P}(\mathrm{S}_1\|\mathrm{S}_1)$	0.700	0.7	$1-c$

c = 0.3, t = 0.4

	Computed value
P(0)	0.401
P(1)	0.599
P(00)	0.221
P(01)	0.180
P(10)	0.180
P(11)	0.419
$\mathrm{P}(\mathrm{S}_0\|\mathrm{S}_0)$	0.551
$\mathrm{P}(\mathrm{S}_0\|\mathrm{S}_1)$	0.300
$\mathrm{P}(\mathrm{S}_1\|\mathrm{S}_0)$	0.449
$\mathrm{P}(\mathrm{S}_1\|\mathrm{S}_1)$	0.700

Table 2. Comparison of c = 0.4 with Different Value of t

c = t = 0.4

	Computed value	Theoretical value	Formula
$P(0)$	0.399	0.4	c
$P(1)$	0.601	0.6	$1-c$
$P(00)$	0.159	0.16	c^2
$P(01)$	0.240	0.24	$c(1-c)$
$P(10)$	0.240	0.24	$c(1-c)$
$P(11)$	0.361	0.36	$(1-c)^2$
$P(S_0\|S_0)$	0.399	0.4	c
$P(S_0\|S_1)$	0.399	0.4	c
$P(S_1\|S_0)$	0.601	0.6	$1-c$
$P(S_1\|S_1)$	0.601	0.6	$1-c$

c = 0.4, t = 0.3

	Computed value
$P(0)$	0.300
$P(1)$	0.700
$P(00)$	0.120
$P(01)$	0.180
$P(10)$	0.180
$P(11)$	0.520
$P(S_0\|S_0)$	0.399
$P(S_0\|S_1)$	0.258
$P(S_1\|S_0)$	0.601
$P(S_1\|S_1)$	0.742

The results from those tables illustrate a key insight about the behavior of the system under experiment. When the parameter c and the threshold t are set to equal values, the calculated results closely match the theoretical predictions, indicating that the system behaves as expected under these conditions. This alignment suggests that the model is accurately capturing the underlying theoretical framework. However, **when c and t are different, the calculated values diverge from the theoretical expectations**. This discrepancy reveals that the binary string is **not memoryless**, meaning that the current state is influenced by previous states. This dependency on historical states contrasts with a memoryless process, where each state occurs independently of past states. Therefore, the difference in values between c and t highlights the presence of memory in the system, affecting its behavior and outcomes.

2. Generate a memoryless information binary sequence by skew Bernoulli maps and save it as a text file. Next compress the file with a file compression software. By performing this for several values of c, consider the relation between the entropy and the compression ratio.

$$\text{compression ratio} = \frac{\text{uncompressed size}}{\text{compressed size}} \qquad (1)$$

In the experiment below, the parameter c will be incremented from 0.05 approaching 0.5, as 0.5 cannot be used because of bit-shift error. The increment of parameter c can be defined on the following list.

```
Parameter c = {0.05, 0.1 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.499, 0.501}
```

For each parameter c, a binary sequence will be generated and saved as a text file (.txt). The text file is then compressed by ZIP, which is available as standard on the Windows 11. For the entropy function, we can define $H(1-c) = -(1-c)log_2(1-c) - clog_2c$. Compression ratio can be calculated by using equation (1). Table 3 provides information on the effects caused by increasing the parameter c, as well as entropy value.

Table 3. Effect of Entropy on Compression Ratio

c	$H(1-c)$	Uncompressed size (byte)	Compressed size (byte)	Compression ratio
0.05	0.286396957	1000000	68860	14.52221
0.1	0.468995594	1000000	101600	9.842519685
0.15	0.609840305	1000000	125006	7.999616018
0.2	0.721928095	1000000	141453	7.069485978
0.25	0.811278124	1000000	152683	6.549517628
0.3	0.881290899	1000000	160819	6.218170739
0.35	0.934068055	1000000	164990	6.060973392
0.4	0.970950594	1000000	166922	5.990822061
0.45	0.992774454	1000000	167087	5.984906067
0.499	0.999997115	1000000	166915	5.991073301
0.501	0.999997115	1000000	166970	5.989099838

From Table 3, a graph of the relationship between the entropy value and the compression ratio value can be generated.

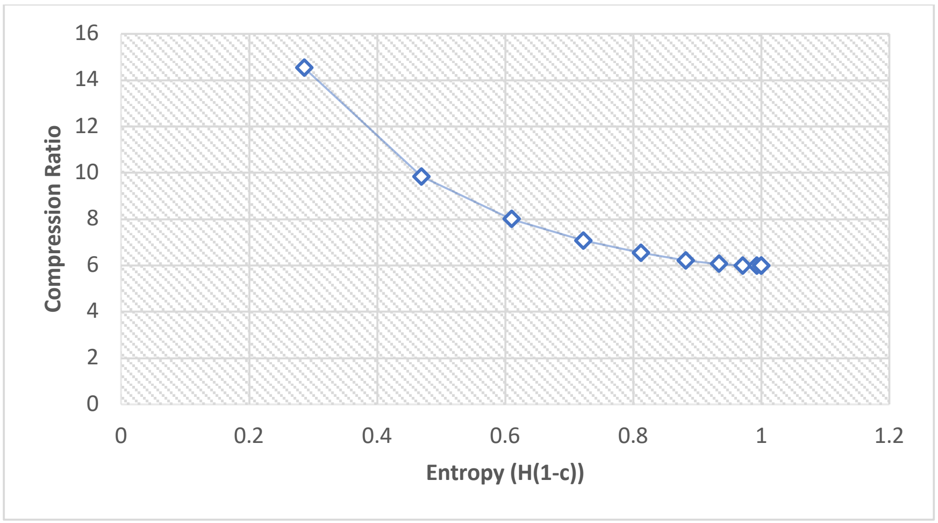

Figure 1. Relationship Between Entropy and Compression Ratio

Figure 1 illustrates that the value of entropy gradually increases as the value of parameter c increases from 0 to approaching on value 0.5. The results show that the compression ratio gradually decreases as the entropy increases. In addition, when the entropy value approaches 1, the compression ratio consistently converges to 6. These observations suggest that **higher entropy indicates a greater variability in the bit sequence**. Consequently, sequences with greater variability result in larger compression ratios compared to those with lower entropy.

Appendix

1. Bernoulli Map Function

$$\tau(x,c) = \begin{cases} \dfrac{x}{c} & 0 \leq x < c \\ \dfrac{x-c}{1-c} & c \leq x \leq 1 \end{cases}$$

2. Threshold Function

$$\theta_t(x) = \begin{cases} 0 & (x < t) \\ 1 & (x \geq t) \end{cases}$$

3. Source code

```python
import numpy as np
import matplotlib.pyplot as plt
import os

# Variables
c_params = [0.3, 0.4] # parameter c
t_params = [0.3, 0.4] # parameter t
ivs = [0.752352] # initial values
l = 1000000 # length (N)
n = 60 # iteration

def skew_bernouli_map(x, c): # Bernoulli function
    if x < c:
        return (x/c)
    else:
        return (x-c)/(1-c)

def treshold_function(x, t): # threshold (make it binary (1 or 0))
    if x < t:
        return 0
    else:
        return 1

def main():
    for c in c_params:
        for x in ivs:
            for t in t_params:
                c1 = c00 = c01 = c10 = c11 = 0 # initialization of counters
                for i in range (l):
                    b1 = treshold_function(x, t)
                    b2 = treshold_function(skew_bernouli_map(x, c), t)
                    c1 += b1 # number of 1
                    c11 += b1 * b2
                    c10 += b1 * (1 - b2)
                    c01 += (1 - b1) * b2
                    c00 += (1 - b1) * (1 - b2)
                    x = skew_bernouli_map(x, c) # next mapping

                # calculate P
                p1 = c1 / l
                p0 = 1 - p1
                p00 = c00 / l
                p01 = c01 / l
                p10 = c10 / l
                p11 = c11 / l
                p0_0 = p00 / p0 #P(S0|S0)
                p0_1 = p10 / p1 #P(S0|S1)
                p1_0 = p01 / p0 #P(S1|S0)
                p1_1 = p11 / p1 #P(S1|S1)

                # display
                print('parameter c:', c, '\nthreshold t:', t)
                print(f'P(0): {p0:.3f}')
                print(f'P(1): {p1:.3f}')
                print(f'P(00): {p00:.3f}')
                print(f'P(01): {p01:.3f}')
                print(f'P(10): {p10:.3f}')
                print(f'P(11): {p11:.3f}')
                print(f'P(0|0): {p0_0:.3f}')
                print(f'P(0|1): {p0_1:.3f}')
                print(f'P(1|0): {p1_0:.3f}')
                print(f'P(1|1): {p1_1:.3f}')

def main2():
    c_list = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.499, 0.501]
    x = ivs[0]

    for c in c_list:
        b_seq = ""
        for i in range (l):
            b1 = treshold_function(x, c)
            b_seq += str(b1)
            x = skew_bernouli_map(x, c)

        print(f"c:{c}", b_seq[:10])
        print("length", len(b_seq))
        # save
        os.makedirs('assignment2/{}'.format(c), exist_ok=True)
        with open(f'assignment2/{c}/2_{c}.txt', 'w') as f:
            f.write(b_seq)
```

```python
if __name__ == "__main__":
    print("No. 1\n\n")
    main()
    print("No. 2\n\n")
    main2()
```