EE 381 Project 2

William Thinh Luu
California State University of Long Beach

EE 381 Probability and Statistics, with Application on Computing

Behruz Revani, Ph.D.

10/09/2024

Part 1

Introduction

Considering the required probabilities: p0 = 0.4, e0 = 0.02, e1 = 0.015. An experiment is run N = 10,000 times to determine if the R signal was received incorrectly and its probability is calculated.

Methodology

In order to create one of the transmitted messages "S", I created a function that takes a random number m and p0 as parameters. The function checks if m <= p0, then the function returns S to be 0, else the function returns S to be 1. To create the received signal "R", I created a function with the parameters of a random number t, the signal "S", e0, and e1. This function will then return 1 or 0 depending on the signal "S" and if the random number t <= e0 or t>e0 and t <= e1 or t>e1. These functions are then called in a for loop N times and then a variable named fails increments up everytime R!=S, and then returns fails/N to return the probability.

Results

Probability of transmission error	
Ans.	**p= 0.0155**

Conclusion

The experiment successfully simulated the transmission of a binary message through a noisy communication channel, revealing the probability of transmission error. With parameters set at p0 = 0.4, e0 = 0.02, e1 = 0.015, the calculated probability of a transmission error was found to be p=0.0155. This result indicates that there is approximately a 1.55% chance that a transmitted signal will be received incorrectly.

Appendix 1

```python
import random

def createSignalS(m, p0):
    if (m<=p0):
        s = 0
    else:
        s = 1
    return s

def createSignalR(t, s, e0, e1):
    if s==0 and t<=e0:
        r=1
    elif s==0 and t>e0:
        r=0
    elif s==1 and t>=e1:
        r=1
    elif s==1 and t<=e1:
        r=0
    return r

def experiment(N):

    p0 = 0.40
    e0 = 0.02
    e1 = 0.015
    fails = 0

    for i in range(N):

        m = random.random()
        t = random.random()
        #ensure t != m
        while (m==t):
            t = random.random()

        s = createSignalS(m, p0)
        r = createSignalR(t, s, e0, e1)

        if (r!=s):
            fails+=1

    return fails / N


if __name__ == "__main__":
    N=10000
    print(f"Probability of transmission error: {experiment(N)}")
```

Part 2

Introduction

With the following required probabilities: e0 = 0.02, e1 = 0.015. Create a code that transmits a one-bit message “S” as of part 1, and calculate the conditional probability P(R=1|S=1) where the focus is only in the transmission where S=1. Repeat the experiment N=100,000 times to determine the conditional probability of P(R=1|S=1).

Methodology

By utilizing code used in part 1, I slightly adjusted it so that for every experiment, S=1. Instead of tracking for the number of failures to receive the bit, the code is adjusted to track the number of successes. By creating a variable named successes, increment successes by 1 when R is equal to S. Then return successes/N as the condition probability of P(R=1|S=1).

Results

Conditional Probability of P(R=1|S=1)	
Ans.	**p= 0.98532**

Conclusion

The experiment effectively determined the conditional probability P(R=1 | S=1) by simulating the transmission of a one-bit message through a noisy communication channel. With a fixed parameter setup of e0 = 0.02 and e1 = 0.015, the experiment was conducted 100,000 times, yielding a conditional probability of p = 0.98532. This result indicates that when a signal of 1 is transmitted, there is approximately a 98.53% chance that it will be correctly received as 1.

Appendix 2

```python
import random

def createSignalS(m, p0):
    if (m<=p0):
        s = 0
    else:
        s = 1
    return s

def createSignalR(t, s, e0, e1):
    if s==0 and t<=e0:
        r=1
    elif s==0 and t>e0:
        r=0
    elif s==1 and t>=e1:
        r=1
    elif s==1 and t<=e1:
        r=0
    return r

def experiment(N):
    e0 = 0.02
    e1 = 0.015
    successes = 0

    for i in range(N):
        # Always generate signal S = 1
        s = 1
        t = random.random()
        r = createSignalR(t, s, e0, e1)
        if r == s:
            successes += 1

    # Return the conditional probability P(R=1 | S=1)
    return successes / N


if __name__ == "__main__":
    N=100000
    print(f"Conditional Probability P(R=1 | S=1): {experiment(N)}")
```

Part 3

Introduction

The following experiment has the required probabilities: p0 = 0.40, e0 = 0.02, e1 = 0.015. Create code that transmits a one-bit message “S” and calculate the conditional probability P(S=1|R=1). For all events for which the received signal is R=1, look at the transmitted bit S. The experiment is a success if R=1 and S=1. Loop this N = 10,000 times and find the conditional probability.

Methodology

Utilizing the functions that create the signal S and receiver R, I created the experiment to loop N times. I initialize variables rIs1 to track the number of times R = 1 and successes to track the number of successes. In each loop, I created S and R the same way I did in part 1 and checked using the if statement to see if R=1. If this is the case, then increment rIs1 by 1 and if S =1 after R=1 is true, then successes += 1.

Results

Conditional Probability of P(S=1|R=1)	
Ans.	**p= 0.9873**

Conclusion

The experiment successfully calculated the conditional probability P(S=1 | R=1) by simulating the transmission of a one-bit message through a noisy communication channel. Utilizing a parameter setup of p0 = 0.40, e0 = 0.02, and e1 = 0.015, the experiment was conducted 10,000 times. The resulting conditional probability was p = 0.9873, indicating that when the received signal is 1, there is approximately a 98.73% chance that the transmitted bit was also 1.

Appendix 3

```python
import random

def createSignalS(m, p0):
    if (m<=p0):
        s = 0
    else:
        s = 1
    return s

def createSignalR(t, s, e0, e1):
    if s==0 and t<=e0:
        r=1
    elif s==0 and t>e0:
        r=0
    elif s==1 and t>=e1:
        r=1
    elif s==1 and t<=e1:
        r=0
    return r

def experiment(N):

    p0 = 0.40
    e0 = 0.02
    e1 = 0.015
    successes = 0
    rIs1 = 0

    for i in range(N):

        m = random.random()
        t = random.random()
        #ensure t != m
        while (m==t):
            t = random.random()

        s = createSignalS(m, p0)
        r = createSignalR(t, s, e0, e1)

        if (r==1):
            rIs1+=1
            if (s==1):
                successes+=1

    return successes/rIs1


if __name__ == "__main__":
    N=10000
    print(f"Conditional Probability P(R=1 | S=1): {experiment(N)}")
```

Part 4

Introduction

The following experiment has the required probabilities: p0 = 0.40, e0 = 0.02, e1 = 0.015. In this experiment, the same bit “S” is now transmitted 3 times instead of just once. The receiver bits “R” are now received in 3, shown as (R1, R2, R3) which can be equal to one of the following eight triplets {(001), (001), (010), (100), (011), (101), (110), (111)}. By using the voting and majority rule, determine what the bit “S” originally was transmitted as when receiving the triplets.

Methodology

By creating a function called majorityVote that takes r1, r2, and r3 as parameters, it returns 1 if the sum of r1, r2, and r3 is greater than or equal to 2, following the voting and majority rule. If not then it returns 0. In the experiment, I created a loop that loops N = 10,000 times, and each iteration of the loop creates an S signal as well as t1, t2, and t3 which are random variables of either 1 or 0. Create R1 ,R2, and R3 using the S signal and the t1, t2, t3 variables as well as the required probabilities. Then by calling the majorityVote function using the 3 receiver signals to determine the decoded bit D. Finally check if bit D is not equal to S, and increment fails by 1, find the probability of error with enhanced transmission by returning fails/N.

Results

Probability of error with enhanced transmission	
Ans.	**p= 0.0019**

Conclusion

The experiment demonstrated that transmitting the same bit three times and applying the majority voting rule significantly reduces the probability of error compared to single-bit transmission. By simulating the process over 10,000 iterations, we observed that the probability of decoding the transmitted bit incorrectly was much lower. This outcome emphasizes the effectiveness of redundant transmission and voting in improving reliability in communication systems, especially in noisy environments. The method ensures that even with some transmission errors, the original message is likely to be recovered correctly.

Appendix 4

```python
import random

def createSignalS(m, p0):
    if (m<=p0):
        s = 0
    else:
        s = 1
    return s

def createSignalR(t, s, e0, e1):
    if s==0 and t<=e0:
        r=1
    elif s==0 and t>e0:
        r=0
    elif s==1 and t>=e1:
        r=1
    elif s==1 and t<=e1:
        r=0
    return r

def majorityVote(r1, r2, r3):
    if (r1+r2+r3) >=2:
        return 1
    else:
        return 0

def experiment(N):
    p0 = 0.40
    e0 = 0.02
    e1 = 0.015
    fails = 0

    for i in range(N):
        m = random.random()
        s = createSignalS(m, p0)

        # transmit the same bit S three times into R1, R2, R3
        t1, t2, t3 = random.random(), random.random(), random.random()
        r1 = createSignalR(t1, s, e0, e1)
        r2 = createSignalR(t2, s, e0, e1)
        r3 = createSignalR(t3, s, e0, e1)

        # calculating if D = 1 or D = 0 via majority rule
        d = majorityVote(r1, r2, r3)

        if d != s:
            fails+=1

    return fails/N



if __name__ == "__main__":
    N=10000
    print(f"Probability of error with enhanced transmission: {experiment(N)}")
```