PROJECT 2 – CONDITIONAL PROBABILITIES

EE 381 – Probability and Stats Computing

Abstract

This project explores the concept of conditional probabilities. The scenario for this is communication through a noisy channel

Brian Nguyen

015188925

1. Probability of an Erroneous Transmission

Introduction

This project deals with message transmissions through a noisy communication channel. A digital message "M" is sent through the channel; its signal "S" consists of zeroes and ones.

- Symbol "0" appears in the signal with probability $p_0 = 0.6$
- Symbol "1" appears in the signal with probability $1 - p_0 = 0.4$

Due to the noise in the channel, the bits may change, resulting in erroneous transmissions.

- A transmitted bit 0 may be received as 1 with probability $e_0 = 0.05$
- A transmitted bit 1 may be received as 0 with probability $e_1 = 0.03$

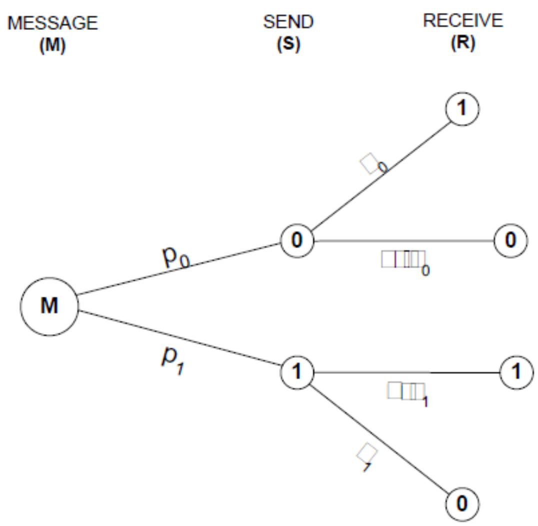

Figure 1 - Probabilities for symbol transmission error

For this problem, we are to transmit a one-bit message, compare it to its signal, and count the number of times the experiment fails. The experiment is repeated for *N* = 100,000 times.

Methodology

In order to generate bits, I used my `nSidedDie()` function for both the signal and the received message. I created two lists – `sList` and `rList` – to store signal and received bits respectively.

- The bits for the signal are generated with S = `nSidedDie([p0, 1 - p0])`, where p0 = 0.6. The function by itself will only return either "1" or "2". To fix this, I used S = S – 1.
- For the received bit, I used the same method to generate either "0" or "1."
  - If `S = 1`, `R = nSidedDie([e1, 1 - e1])`, where `e1 = 0.03; R = R - 1`
  - If `S = 0`, `R = nSidedDie([1 - e0, e0])`, where `e0 = 0.05; R = R - 1`

I compared each list of stored bits and counted the number of times the signal bit does not match with the received bit.

Results and Conclusions

Probability of transmission error	
Ans.	***p* = 0.04258**

The chances of receiving an incorrect bit are relatively low. Considering the probabilities given to us, the chances of a transmitted bit 0 being received as 1 is p = 0.05, while the chances of a received bit 1 being received as 0 is p = 0.03. The average between the two is 0.04, which is about what I received in my experiment.

Appendix

```
import numpy as np


p0 = 0.6
e0 = 0.05
e1 = 0.03


def errorTransmission(N):
    sList = []
    rList = []
    for x in range(N):
        S = nSidedDie([p0, 1 - p0])
        S = S - 1
        sList.append(S)
        if S == 1:
            R = nSidedDie([e1, 1 - e1])
            R = R - 1
            rList.append(R)
        elif S == 0:
            R = nSidedDie([1 - e0, e0])
            R = R - 1
            rList.append(R)
    # print('S = ', sList)
    # print('R = ', rList)
    fail_count = 0
    for k in range(0, len(sList)):
        if sList[k] != rList[k]:
            fail_count += 1
    print('-------\nRegular transmission\nprobability of failure', fail_count/N)

def nSidedDie(p):
    n = len(p)
    cs = np.cumsum(p)
    cp = np.append(0, cs)
    r = np.random.rand()
    for k in range(0, n):
        if r > cp[k] and r <= cp[k + 1]:
            d = k + 1
    return d

def main():
     errorTransmission(100000)
main()
```

2. Conditional Probability: P(R = 1 | S = 1)

Introduction

For this experiment, we are to create and transmit a one-bit message S as before and calculate the conditional probability $P(R = 1 \mid S = 1)$. For all bits in which $S = 1$, if $R = 1$, then the experiment is considered a success. The experiment is repeated for $N = 100{,}000$ times.

Methodology

Just like the first problem, I generated a list of S bits and R bits using the `nSidedDie()` function and stored them in a list. In order to find the probability $P(R = 1 \mid S = 1)$, I iterated through both lists. If a bit at the current index in `sList` is 1, I look at the bit in `rList`, and if it is 1 as well, I counted that as a success.

Results and Conclusion

Conditional probability $P(R = 1 \mid S = 1)$	
Ans.	***p = 0.97067***

Appendix

```
import numpy as np

p0 = 0.6
e0 = 0.05
e1 = 0.03

def conditionalProbability(N):
    print('-------\nP(R = 1 | S = 1)\nCalculating...')
    sList = []
    rList = []
    for x in range(0, N):
        S = nSidedDie([p0, 1 - p0])
        S = S - 1
        sList.append(S)
        if S == 1:
            # epsilon 1 = 0.03
            R = nSidedDie([e1, 1 - e1])
            R = R - 1
            rList.append(R)
        elif S == 0:
            # epsilon 0 = 0.05
            R = nSidedDie([1 - e0, e0])
            R = R - 1
            rList.append(R)
    print('S =', sList)
    print('R =', rList)
    success = 0
    for k in range(len(sList)):
        if sList[k] == 1:
            if rList[k] == 1:
                success += 1
    one_count = sList.count(1)
    print("p =", success / one_count)

def nSidedDie(p):
    n = len(p)
    cs = np.cumsum(p)
    cp = np.append(0, cs)
    r = np.random.rand()
    for k in range(0, n):
        if r > cp[k] and r <= cp[k + 1]:
            d = k + 1
    return d

def main():
    N = 100000
    conditionalProbability(N)
main()
```

3. Conditional Probability: P(S = 1 | R = 1)

Introduction

For this experiment, we are to create and transmit a one-bit message S as before and calculate the conditional probability $P(S = 1 \mid R = 1)$. For all bits in which $R = 1$, if $S = 1$, then the experiment is considered a success. The experiment is repeated for $N = 100{,}000$ times.

Methodology

Just like the first problem, I generated a list of S bits and R bits using the `nSidedDie()` function and stored them in a list. In order to find the probability $P(S = 1 \mid R = 1)$, I iterated through both lists. If a bit at the current index in `rList` is 1, I look at the bit in `sList`, and if it is 1 as well, I counted that as a success.

Results and Conclusion

Conditional probability $P(R = 1 \mid S = 1)$	
Ans.	***p = 0.9284***

Appendix

```
import numpy as np

p0 = 0.6
e0 = 0.05
e1 = 0.03


def conditionalProbability2(N):
    print('-------\nP(S = 1 | R = 1)\nCalculating...')
    sList = []
    rList = []
    for x in range(0, N):
        S = nSidedDie([p0, 1 - p0])
        S = S - 1
        sList.append(S)
        if S == 1:
            # epsilon 1 = 0.03
            R = nSidedDie([e1, 1 - e1])
            R = R - 1
            rList.append(R)
        elif S == 0:
            # epsilon 0 = 0.05
            R = nSidedDie([1 - e0, e0])
            R = R - 1
            rList.append(R)
    print('S =', sList)
    print('R =', rList)
    success = 0
    for k in range(len(sList)):
        if rList[k] == 1:
            if sList[k] == 1:
                success += 1
    one_count = rList.count(1)
    print("p =", success / one_count)


def nSidedDie(p):
    n = len(p)
    cs = np.cumsum(p)
    cp = np.append(0, cs)
    r = np.random.rand()
    for k in range(0, n):
        if r > cp[k] and r <= cp[k + 1]:
            d = k + 1
    return d


def main():
    N = 100000
    conditionalProbability2(N)
main()
```

4. Enhanced Transmission Method

Introduction

In this experiment, the transmission is enhanced by sending the same bit "S" three times (S S S). Since this is the case, the received message will be (R R R).

- The possible received bits are (R1 R2 R3) = { (000), (001), (010), (011), (100), (101), (110), (111) }
- By using the voting and majority rule, we decide what bit is originally transmitted
    - For example: if (R1 R2 R3) = (110), then the majority bit is 1.
    - Another example: if (R1 R2 R3) = (010), then the majority bit is 0.
- If the sent bits are the same as the received majority bit, then the experiment is considered a success; otherwise, it is a failure
    - For example: if S = (000) and R = (001), then the experiment is considered a success because the decoded bit is D = 0; S = D
    - Another example: if S = (111) and R = (001), then the experiment is considered a failure because the decoded bit is D = 0, S ≠ D

The experiment requires to count the number of times the transmitted bit "S" was received and decoded incorrectly for *N* = 100,000 times

Methodology

I used the same `nSidedDie()` function to create the bits and `sList` and `rList` to store sent and received bits respectively. Since this experiment sends and receives three bits, a created another list, `sublist`, to store three bits inside of a sub-list.

For the first part of the experiment, I generated bits for S.

- If `S = 1`, then `sublist = [1, 1, 1]` and is appended to `sList`.
    - The bits for (R1 R2 R3) are generated with `R = nSidedDie([e1, 1 - e1])` where `e1 = 0.03; R = R - 1.`
    - The sub-list is appended to `rList`
- If S = 0, then `sublist = [0, 0, 0]` and is appended to `sList`.
    - The bits for (R1 R2 R3) are generated with `R = nSidedDie([1 - e0, e0])` where `e0 = 0.05; R = R - 1.`
    - The sub-list is appended to `rList`

For the second part, I iterated through both lists to find any erroneous transmissions. In order to do so, I counted the number of 0's and 1's in each sub-list of `rList`.

- If the current sub-list in `sList` is [0, 0, 0] and the amount of 1's is greater than the amount of 0's in `rList`, then the experiment is a failure
- If the current sub-list in `sList` is [1, 1, 1] and the amount of 0's is greater than the amount of 1's in `rList`, then the experiment is a failure

Results and Conclusion

Probability of transmission error	
Ans.	***p = 0.00546***

With the enhanced transmission, the probability of transmission errors is reduced further.

Appendix

```
import numpy as np

p0 = 0.6
e0 = 0.05
e1 = 0.03

def enhancedTransmission(N):

    sList = []
    rList = []
    for x in range(N):
        S = nSidedDie([p0, 1 - p0])
        S = S - 1
        if S == 1:
            sublist = [1, 1, 1]
            sList.append(sublist)
            sublist = []
            for i in range(3):
                R = nSidedDie([e1, 1 - e1])
                R -= 1
                sublist.append(R)
            rList.append(sublist)

        elif S == 0:
            sublist = [0, 0, 0]
            sList.append(sublist)
            sublist = []
            for i in range(3):
                R = nSidedDie([1 - e0, e0])
                R -= 1
                sublist.append(R)
            rList.append(sublist)
    # print(sList)
    # print(rList)
    fail = 0
    for k in range(len(sList)):
        r_count_zero = rList[k].count(0)
        r_count_one = rList[k].count(1)
        if sList[k] == [0, 0, 0]:
            if r_count_one > r_count_zero:
                fail += 1
        elif sList[k] == [1, 1, 1]:
            if r_count_zero > r_count_one:
                fail += 1
    print('probability of failure = ', fail/N)


def nSidedDie(p):
    n = len(p)
    cs = np.cumsum(p)
    cp = np.append(0, cs)
    r = np.random.rand()
    for k in range(0, n):
        if r > cp[k] and r <= cp[k + 1]:
            d = k + 1
    return d

def main():
    enhancedTransmission(100000)
main()
```