Kevin Long

Professor Chassiakos

EE 381

27 February 2019

Project 2 – Conditional Probabilities

1. **Probability of Erroneous Transmission**
   a. **Introduction**
   We are transmitting a one-bit message *S* and look at the received signal *R*. If *R* = *S*, the particular experiment is a success. If not, it is a failure. This is conducted 100,000 times. At the end, the probability of failure is calculated. p0 = 0.6 (prob. that "0" appears in the signal), e0 = 0.05 (prob. of transmission error for symbol 0), e1 = 0.03 (prob. of transmission error for symbol 1)

   b. **Methodology**
   The function "nSidedDie(p)" simulates the chance of bit transmission.
   The values, p0, e0, e1 are passed into the function "problem1(p0, e0, e1)." This is where the probability is calculated. For each experiment, S is generated and depending on its value, the correlating if/else statements are called upon to generate the received bit R. If they are not equal to each other, then a counter for the number of errors is incremented. The result is printed with the total number of errors divided by the total number of experiments.

   c. **Result(s) and Conclusion(s)**

Probability of Transmission error	
Ans.	***p* = 0.04178**

   d. **Source Code**

```
import numpy as np

"""
1)
"""
def nSidedDie(p):
    n = len(p) #Number of elements in 'p'

    pSum = 0
    for k in range(n):
        pSum += p[k]

    if pSum > 1.0: #Checking for correct probability total
        print("Cumulative sum of p cannot be greater than 1.0")
```

```
    else:
        N = 1 #Number of rolls

        sampleSpace = np.zeros((N,1)) #Return a new array of given shape and type, filled with zeros.
        cs = np.cumsum(p) #Return the cumulative sum of the elements along a given axis.
        cp = np.append(0, cs) #Append values to the end of an array.

        for i in range(0, N): #For every iteration in N
            r = np.random.rand()
            for j in range(0, n): #count how frequent that number 'n' is rolled
                if r > cp[j] and r <= cp[j+1]:
                    d = j + 1
            sampleSpace[i] = d
    return d

def problem1(p0, e0, e1):
    N = 100000 #Num of trials

    error = 0
    for i in range(0, N):
        S = nSidedDie([p0, 1 - p0]) #Message genration
        S -= 1

        if S == 1: #Transmitting message simulation
            R = nSidedDie([e1, 1 - e1])
            R -= 1

        elif S == 0:
            R = nSidedDie([1 - e0, e0])
            R -= 1

        if R != S: #Checking if bits are correct
            error += 1

    print("Probability of transmission error:", error / N)

p0, e0, e1 = 0.6, 0.05, 0.03
problem1(p0, e0, e1)
```

2. **Conditional Probability: P(R = 1|S = 1)**
    a. **Introduction**
    We create and transmit a one-bit message *S*. We will focus on transmissions where *S* = 1. For each time the transmitted signal is such, we look at the received bit *R* and see if it is also equal to 1. Is so, the experiment is a success. The experiment is run 1000,000 times. Calculate for the conditional probability *P(R=1,S=1)*, the prob. that *S* will be received correctly. p0 = 0.6 (prob. that “0”

appears in the signal), e0 = 0.05 (prob. of transmission error for symbol 0), e1 = 0.03 (prob. of transmission error for symbol 1)

b. **Methodology**
The function "nSidedDie(p)" simulates the chance of bit transmission.
The values, p0, e0, e1 are passed into the function "problem2(p0, e0, e1)." This is where the probability is calculated. Counters are set for the "sent(S)" and "received(R)" bits. For each experiment, S is generated. Depending on the value generated, appropriate if/else statements are executed. R is generated in either case(s). If S is equal to 1, it is counted. R is only counted if it is equal to S. At the end, the total bits received are divided by the total bits sent to calculate for the probability of the event.

c. **Result(s) and Conclusion(s)**

Conditional Probability P(R=1|S=1)	
Ans.	***p* = 0. 0.9681676437570164**

d. **Source Code**

```
"""
2)
"""
def problem2(p0, e0, e1):
    N = 100000 #Num of trials

    sent = 0
    received = 0
    r = 0
    for i in range(0, N):
        s = nSidedDie([p0, 1 - p0])
        s -= 1

        if s == 1:
            sent += 1
            r = nSidedDie([e1, 1 - e1])
            r -= 1

            if r == 1:
                received += 1
        else:
            r = nSidedDie([1 - e0, e0])
            r -= 1

    print()
    print("Focus on S=1, P(R=1|S=1):", received/sent)
```

```
problem2(p0, e0, e1)
```

3. **Conditional Probability: P(S = 1|R = 1)**
   a. **Introduction**
   We create and transmit a one-bit message *S*. We will focus on transmissions where *R* = 1. If *S* = 1, then the experiment is a success defined as *P(S=1|R=1)*. The experiment is repeated 100,000 times. p0 = 0.6 (prob. that "0" appears in the signal), e0 = 0.05 (prob. of transmission error for symbol 0), e1 = 0.03 (prob. of transmission error for symbol 1)

   b. **Methodology**
   The function "nSidedDie(p)" simulates the chance of bit transmission.
   The values, p0, e0, e1 are passed into the function "problem3(p0, e0, e1)." This is where the probability is calculated. Counters are set for the "sent(S)" and "received(R)" bits. For each experiment, S is generated. Depending on the value generated, appropriate if/else statements are executed. In either case(s), the value for R will also be generated. If R is equal to 1, then the counter is incremented. If S is also 1 when R is 1, then S will be incremented as well. The probability is printed as the total number of sent bits divided by the total number or received bits.

   c. **Result(s) and Conclusion(s)**

Conditional Probability P(S=1|R=1)	
Ans.	***p* = 0.9315773104619728**

   d. **Source Code**

```
"""
3)
"""
def problem3(p0, e0, e1):
    N = 100000 #Num of trials

    sent = 0
    received = 0
    r = 0
    for i in range(0, N):
        s = nSidedDie([p0, 1 - p0])
        s -= 1

        if s == 1:
            r = nSidedDie([e1, 1 - e1])
            r -= 1
        else:
            r = nSidedDie([1 - e0, e0])
```

```
            r -= 1
        if r == 1:
            received += 1
            if s == 1:
                sent += 1
    print()
    print("Focus on R=1, P(S=1|R=1):", sent/received)

problem3(p0, e0, e1)
```

4. **Enhanced Transmission Method**

   a. **Introduction**

   We create and transmit a one-bit message *S.* To improve the reliability of the message, *S* is transmitted three times. The received bits *R* are not always the same as *S* due to errors. The three received bits will be equal to one of the following eight triplets: (R1 R2 R3) = {(000), (001), (010), (100), (011), (101), (110), (111)}. We must decide what was the original *S* bit by using "voting and the majority rule." If a majority of the bits are 0, then D = 0. If a majority of the bits are 1, then D = 1. If *S* = 1 is transmitted three times, then the experiment is considered a success. The experiment is repeated 100,000 times to calculate the probability of error with enhanced transmission. p0 = 0.6 (prob. that "0" appears in the signal), e0 = 0.05 (prob. of transmission error for symbol 0), e1 = 0.03 (prob. of transmission error for symbol 1)

   b. **Methodology**

   The function "nSidedDie(p)" simulates the chance of bit transmission.
   The values, p0, e0, e1 are passed into the function "problem4(p0, e0, e1)." This is where the probability is calculated. A counter is created along with two lists for R and S bits. For each experiment, S is generated. The sList is then populated with three S bits. If there is a 1 inside the list, then rList is populated with randomly generated bits. The contents of rList are then summed and if greater than or equal 2, the decode variable will be assigned either a 0 or 1. If the decode variable is not equal to S, then an error has occurred and will be incremented. The probability is printed via the total number of errors divided by the number of experiments.

   c. **Result(s) and Conclusion(s)**

   The voting method used in this problem provides significant improvement as compared to the method of Problem 1. The improvement is approximately 10 times more accurate.

Probability with enhanced transmission	
Ans.	***p* = 0.00538**

   d. **Source Code**

```
"""
4)
"""
def problem4(p0, e0, e1):
    N = 100000 #Num of trials

    count = 0
    decode = -1
    rSum = 0
    sList = []
    rList = []
    for i in range(0, N):
        s = nSidedDie([p0, 1 - p0])
        s -= 1

        sList = [s, s, s]
        if 1 in sList:
            rList = [nSidedDie([e1, 1 - e1]) - 1, nSidedDie([e1, 1 - e1]) - 1, nSidedDie([e1, 1 - e1]) - 1]
        if 0 in sList:
            rList = [nSidedDie([1 - e0, e0]) - 1, nSidedDie([1 - e0, e0]) - 1, nSidedDie([1 - e0, e0]) - 1]
        rSum = np.sum(rList)
        if rSum >= 2:
                decode = 1
        else:
                decode = 0
        if decode != s:
                count+=1

    print()
    print("Probability of error with enhanced transmission: ", count/N)

problem4(p0, e0, e1)
```