Homework 2

Kevin Chao

October 21, 2019

Question 1:

a. Given the information, we have P(HasCarpal) = .1 , and P(HasCarpal and Positive) = .96 . Now, it is given that P(NoCarpal and Positive) =.44 . With these information, we can now have P(NoCarpal) = .9 , P(HasCarpal and Negative) = .04 , and P(NoCarpal and Negative) = .56 . To find the posterior probability that they have Carpal Tunnel Syndrome given only a positive test, it must be trying to find a conditional probability: P(HasCarpal — Positive).

$$\begin{aligned} P(HasCarpal|Positive) =& \frac{.10 * .96}{.10 * .96 + .90 * .44} \\ P(HasCarpal|Positive) =& .1951 \end{aligned}$$

b. (b) is similar to (a), but now asks a general equation and asks for an inequality. Thus, the equation to solve for p (where false positive rate = NoCarpal and Positive) is

$$\begin{aligned} \frac{.10 * .96}{.10 * .96 + .90 * p} >=& .75 \\ \frac{.10 * .96 + .9 * p}{.10 * .96} <=& \frac{1}{.75} \\ .096 + .9p <=& \frac{.096}{.75} \\ p <=& .0356 \end{aligned}$$

The largest possible false positive rate is 3.56%

Question 2:

a. Since the dice is a fair, 6-sided dice, then the chance of getting a specific number is $\frac{1}{6}$.

So, the chance of getting not a 6 is $\frac{5}{6}$, and thus $P(R_n)$ would be :

$$P(R_n) = (\tfrac{5}{6})^n$$

b. If S_n denotes the payoff after n dice roll, then finding the expected pay-off for n rolls would be simplest by first discovering how much a pay-off for one roll would be. Since each trial is independent, then $E[S_1|R_1]$ looks like

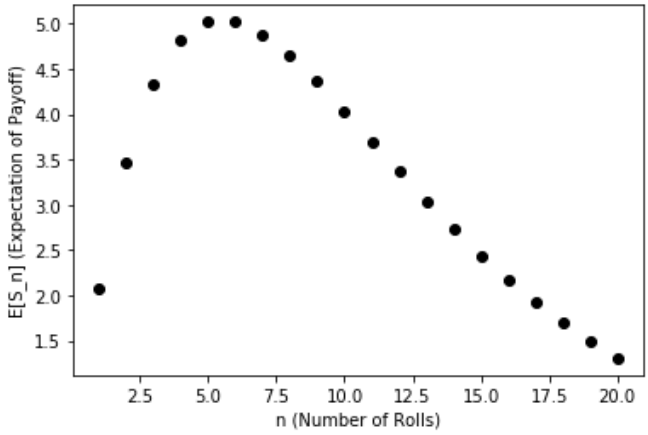
$$\begin{aligned} E[S_1|R_1] =& 1\frac{1}{6} + 2\frac{2}{6} + 3\frac{3}{6} + 4\frac{4}{6} + 5\frac{5}{6} \\ =& \frac{1}{6}(1+2+3+4+5) \\ =& 2.5 \end{aligned}$$

Thus, $E[S_n|R_n] = 2.5n$

c. Given (a) and (b), we can now use the total expectation theorem. The theorem would look like

$$E[S_n] = P(R_n) * E[S_n|R_n]$$

Plugging in from 1 to 20 for n with the above formula, the plotted values of $E[S_n]$ are below.



From this, it is given that the smallest n that maximizes the expected payoff would be when $n = 5$

Question 3:

a. Given the table, we can calculate P(C=1) as a sum of probabilities.

$$\begin{aligned} P(C=1) &= 0 + .1 + .05 + 0.25 \\ P(C=1) &= .4 \end{aligned}$$

b. In order to calculate this, first we must find when P$P(C=0, X=1, Y=0)$ and $P(X=1, Y=0)$, which are found below:

$$P(C = 0, X = 1, Y = 0) = .2$$
$$P(X = 1, Y = 0) = .2 + .05 = .25$$

Now the former must divide the latter:

$$P(C = 0|X = 1, Y = 0) = \frac{.2}{.25} = .8$$

c. $P(X = 0, Y = 0)$ can be calculated as:

$$P(X = 0, Y = 0) = .1 + 0 = .1$$

d. In order to calculate this, Bayes theory requires two probabilities: $P(C = 0, X = 0)$ and $P(X = 0)$. Those are found below:

$$P(C = 0, X = 0) = .1 + .2 = .3$$
$$P(X = 0) = .1 + .2 + 0 + .1 = .4$$

Now, divide the former with the latter.

$$P(C = 0|X = 0) = .3/.4 = .75$$

e. In order to declare independence of the two variables X and Y, this statement must hold true: $P(X = 0, Y = 0) = P(X = 0) * P(Y = 0)$. So first, we must find P(X=0), P(Y=0), and P(X=0, Y=0).

$$\begin{aligned} P(X = 0) &= .4 \\ P(Y = 0) &= .1 + .2 + 0 + .05 \\ &= .35 \\ P(X = 0, Y = 0) &= .1 \\ P(X = 0)P(Y = 0) &= .35 * .4 \\ &= .14 \\ P(X = 0, Y = 0) &\neq P(X = 0) * P(Y = 0) \end{aligned}$$

Therefore, X and Y are not independent.

f. To be conditionally independent, X and Y given C must have this relationship: $P(X = 0, Y = 0|C = 0) = P(X = 0, Y = 0)$. The calculations below will help find this relationship.

$$\begin{aligned} P(X = 0, Y = 0, C = 0) &= .1 \\ P(C = 0) &= .1 + .2 + .2 + .1 = .6 \\ P(X = 0, Y = 0|C = 0) &= \frac{.1}{.6} \\ &= .1667 \\ P(X = 0, Y = 0) &= .1 \\ P(X = 0, Y = 0|C = 0) &\neq P(X = 0, Y = 0) \end{aligned}$$

Therefore, X and Y are not conditionally independent given C.

Question 4:

a. Baye's rule works as followed:

$$P(A|B) = \frac{P(B|A)*P(A)}{P(B)}$$

Applying that to the previous equation, $P(Y_i = S|X_i) > P(Y_i = H|X_i)$, we know get:

$$\frac{P(X_i|Y_i=S)P(Y_i=S)}{P(X_i)} > \frac{P(X_i|Y_i=H)P(Y_i=H)}{P(X_i)}$$

The denominator can be ignored since $P(X_i)$ is shared on both side and is $P(X_i) > 0$.

$$P(X_i|Y_i = S)P(Y_i = S) > P(X_i|Y_i = H)P(Y_i = H)$$

And since $P(Y_i = S) = P(Y_i = H) = 0.5$, then you can cancel 0.5 on each side since both sides have 0.5, thus leaving us with:

$$P(X_i|Y_i = S) > P(X_i|Y_i = H)$$

b.
```
vocabInds = np.arange(W)  # Part (g): full vocabularly of all W words
# Separate "ham" and "spam" classes, subsample selected vocabulary words
trainHam  = trainFeat[trainLabels == 0][:, vocabInds]
trainSpam = trainFeat[trainLabels == 1][:, vocabInds]
# Number of training examples of each class
numHam = len(trainHam)
numSpam = len(trainSpam)
# Count number of times each word occurs in each class
countsHam = np.sum(trainHam, axis=0)
# P(X_ij=1 | Y_i=H) can be computed from countsHam and numHam
countsSpam = np.sum(trainSpam, axis=0)
# P(X_ij=1 | Y_i=S) can be computed from countsSpam and numSpam
for i in range(0,W):
       wordExists = countsHam[i]/numHam
       wordDoesNot = 1 - wordExists
       print("The word is " + str(vocab[i]) + " and the probability
       word exists given Ham is " + str(wordExists))
       print("The probabilty that the word doesn't exist
       given Ham is " + str(wordDoesNot))
for i in range(0,W):
       wordExists = countsSpam[i]/numSpam
       wordDoesNot = 1 - wordExists
       print("The word is " + str(vocab[i]) + " and the probability word
       exists given Spam is " + str(wordExists))
       print("The probabilty that the word doesn't exist given
       Spam is " + str(wordDoesNot))
```

c. Using the code from (b), the following are the numerical values of the conditional probabilities:

$$P(X_{ij} = 1|Y_i = S) = .164$$
$$P(X_{ij} = 1|Y_i = H) = .030$$

From these probabilities, the word "money" is predicted to be spam, and the absence predicts it to be ham. I found the accuracy of individual through this code:

```
actualSpam = 0
actualHam = 0
for i in range(0,len(testFeat)):
      if testFeat[i][179] == 1 and testLabels[i] == 1:
            actualSpam += 1
      elif testFeat[i][179] == 0 and testLabels[i] == 0:
            actualHam += 1
print((actualHam + actualSpam)/len(testFeat))
```

The accuracy is 55.73% .

d. Using the code from (b), the following are the numerical values of the conditional probabilities:

$$P(X_{ij} = 1|Y_i = S) = .069$$
$$P(X_{ij} = 1|Y_i = H) = .320$$

From these probabilities, the word "thanks" is predicted to be ham and the absence is predicted to be spam. I found the accuracy of individual through a similar code from (c):

```
actualSpam = 0
actualHam = 0
for i in range(0,len(testFeat)):
      if testFeat[i][859] == 0 and testLabels[i] == 1:
            actualSpam += 1
      elif testFeat[i][859] == 1 and testLabels[i] == 0:
            actualHam += 1
print((actualHam + actualSpam)/len(testFeat))
```

The accuracy is 63.15% . From a human perspective, most formal emails that are not spam can typically end with 'thanks'. With that, there would be more emails than emails that contain 'money' (atleast, thats what I would think if hes simply a faculty member at UCI). From that, I would imagine there would be more instances where words with thanks would be more likely to be ham and those without it to be more likely to be considered spam, atleast in comparison to (c).

e. Using the code from (b), the following are the numerical values of the conditional probabilities:

$$P(X_{ij} = 1|Y_i = S) = .002518$$
$$P(X_{ij} = 1|Y_i = H) = .002523$$

From these probabilities, the word 'possibilities' is predicted to be spam and the absence of the word to be. I found the accuracy of individual through a similar code from (c):

```
actualSpam = 0
actualHam = 0
for i in range(0,len(testFeat)):
      if testFeat[i][2211] == 1 and testLabels[i] == 1:
             actualSpam += 1
      elif testFeat[i][2211] == 0 and testLabels[i] == 0:
             actualHam += 1
print((actualHam + actualSpam)/len(testFeat))
```

The accuracy is 49.10% . To me, 'possibilities' as a word to appear in emails would be exceedingly rare in comparison to words like 'thanks' or 'money'. With that, the accuracy that 'possibilities' would have would probably come heavily from the absence of the word (it would be really close to 50% since $P(Y_i = S) = P(Y_i = H) = 0.5$)

f. This code looks at more of each possible case (i.e. Money = 1 Thanks = 1 Possibilities = 0) and makes predictions based on it. The code below is rather brute-forced and does each individual case, so it will be shortened but still understandable.

```
def compare(args):
   if len(args) > 0:
        ham = 1
        spam = 1
        for item in args:
            ham = ham * (countsHam[item]/numHam)
            spam = spam *( countsSpam[item]/numSpam)
        if ham > spam:
            return 'ham'
        elif ham < spam:
            return 'spam'
    else:
        ham = (1-(countsHam[179]/numHam)) * (1-(countsHam[859]/numHam))
        * (1-(countsHam[2211]/numHam))
        spam = (1-(countsSpam[179]/numSpam)) * (1-(countsSpam[859]/numSpam))
        * (1-(countsSpam[2211]/numSpam))
```

```
        if ham > spam:
            return 'ham'
        elif ham < spam:
            return 'spam'

def checkWords(email):
    indices = []
    if testFeat[email][179] == 1:
        indices.append(179)
    if testFeat[email][859] == 1:
        indices.append(859)
    if testFeat[email][2211] == 1:
        indices.append(2211)
    return indices

actualHam = 0
actualSpam = 0
for i in range(0,len(testFeat)):
    check = checkWords(i)
    compared = compare(check)
    if compared == 'ham' and testLabels[i] == 0:
        actualHam += 1
    elif compared == 'spam' and testLabels[i] == 1:
        actualSpam += 1
print((actualHam + actualSpam)/len(testFeat))
```

The accuracy for ['money','thanks','possibilities'] is 63.61% .

g. The editted code for the classifier, in accordance to log-probabilities and of all words, now look like this:

```
def compare(args):
   if len(args) > 0:
        ham = 1
        spam = 1
        for item in args:
            ham = ham * (countsHam[item]/numHam)
            spam = spam *( countsSpam[item]/numSpam)
        ham = np.log(ham)
        spam = np.log(spam)
        if ham > spam:
            return 'ham'
        elif ham < spam:
            return 'spam'
   else:
        ham = 1
```

```
        spam = 1
        for i in range(0,len(countsHam)):
            ham = ham * (1-(countsHam[i]/numHam))
        for i in range(0,len(countsSpam)):
            spam = spam * (1-(countsSpam[i]/numSpam))
        ham = np.log(ham)
        spam = np.log(spam)
        if ham > spam:
            return 'ham'
        elif ham < spam:
            return 'spam'
correctPredictions = 0
for i in range(0,len(testFeat)):
    wordsInEmail = []
    for ind in range(0,W):
        if testFeat[i][ind] == 1:
            wordsInEmail.append(ind)
    compared = compare(wordsInEmail)
    if compared == 'ham' and testLabels[i] == 0:
        correctPredictions += 1
    elif compared == 'spam' and testLabels[i] == 1:
        correctPredictions += 1
print(correctPredictions/len(testFeat))
```

The accuracy for this is 87.06%