Assignment 1 ChenYeh Lin
A69036794
CSE250A

1.1 Conditioning on background evidence

(a)

$$P(X,Y|E) = \frac{P(X,Y,E)}{P(E)}$$

$$= \frac{P(E)\cdot P(Y|E)\cdot P(X|Y,E)}{P(E)} \quad \text{(By product rule)}$$

$$= P(X|Y,E)\cdot P(Y|E) \quad \#$$

(b)

$$P(X|Y,E) = \frac{P(X,Y,E)}{P(Y,E)}$$

$$= \frac{P(E)\cdot P(X|E)\cdot P(Y|X,E)}{P(Y,E)}$$

$$= \frac{P(E)\cdot P(X|E)\cdot P(Y|X,E)}{P(E)\cdot P(Y|E)}$$

$$= \frac{P(Y|X,E)P(X|E)}{P(Y|E)} \quad \#$$

(C)

$$p(x|E) = \frac{P(x, E)}{P(E)}$$

$$P(x, E) = \sum_y P(x, Y=y, E)$$

$$P(x | Y=y, E) = \frac{P(x, y, E)}{P(y, E)}$$

$$= \sum_y P(x | Y=y, E) \cdot P(Y=y, E)$$

$$p(x|E) = \frac{\sum_y P(x | Y=y, E) \cdot P(Y=y, E)}{P(E)}$$

$$P(x, Y=y | E) = \frac{P(x, Y=y, E)}{P(E)}$$

$$P(x, Y=y, E) = P(x, Y=y | E) P(E)$$

Thus,

$$p(x|E) = \frac{\sum_y p(x, Y=y | E) \cdot p(E)}{p(E)}$$

$$p(x|E) = \sum_y p(x, Y=y | E) \quad \#$$

1-2 conditional independence

(1) implies (2) & (3)

$$P(x,Y|E) = P(x|E)\cdot P(Y|x,E)$$
$$= P(Y|E)\cdot P(x|Y,E)$$

Using the given $P(x,Y|E) = P(x|E)P(Y|E)$

$$P(Y|E)\cdot P(x|Y,E) = P(x|E)\cdot P(Y|E)$$

$\Rightarrow P(x|Y,E) = P(x|E)$
which prove statement (2)

$$P(x|E)\cdot P(Y|x,E) = P(x|E)P(Y|E)$$

$\Rightarrow P(Y|x,E) = P(Y|E)$
which prove statement (3)

(2) implies (1) & (3)

$$P(x,Y|E) = \frac{P(\cancel{E})\cdot P(Y|E)\cdot P(x|Y,E)}{P(\cancel{E})} = P(x|Y,E)P(Y|E)$$

since we are given that $P(x|Y,E) = P(x|E)$

$\Rightarrow P(x,Y|E) = P(x|E)P(Y|E)$
which proved statement (1)

$P(X|Y,E) = P(X|E)$ implies that X is conditional independent of Y given E. Therefore, Y must also be conditionally independent of X given E

$P(Y|X,E) = P(Y|E)$. Thus, statement (3) is true if statement (2) is true.

(3) implies (2) & (1)

$$P(X,Y|E) = \frac{P(\cancel{E}) \cdot P(X|E) \cdot P(Y|X,E)}{P(\cancel{E})} = P(X|E) \cdot P(Y|X,E)$$

since we are given that $P(Y|X,E) = P(Y|E)$

$P(X,Y|E) = P(X|E) \cdot P(Y|E)$

which proved statement (1)

since Y is conditionally independent of X given E. Therefore, X must also be conditionally independent of Y given E. In other words:

$$P(X|Y,E) = P(X|E)$$

Thus, statement (2) is true if statement (3) is true.

1-3 Creative writing

(a)

1.

$p(X=1)$. The probability that Person A is late to school without additional information

2.

$p(X=1 | Y=1)$: The probability that Person A is late to school given that Person A missed the school bus

3.

$p(X=1 | Y=1, Z=1)$: The probability that Person A is late to school given that Person A missed the school bus and his alarm clock is broken

$p(X=1 | Y=1) > p(X=1)$, it is reasonable that Person A is more likely to be late if he didn't get on the school bus on time. $p(X=1 | Y=1, Z=1) > p(X=1 | Y=1)$ The presence of both missing the school bus and the alarm clock is broken provide stronger cumulative evidence, further increasing the likelihood that Person A will be late.

(b)

1. $P(X=1)$: The probability of Person A go camping with friends

2. $p(X=1 | Y=1)$: The probability of Person A go camping with friends given that Annie, the girl that he admires, will show up

3. $p(X=1 | Y=1, Z=1)$: The probability of Person A go camping with friends given that Annie, the girl that he admires, will show up and it might rain at the camping day

In this scenario, $p(X=1 \mid Y=1) > p(X=1)$. Since Annie is also going camping, it increase the pobability that Person A will go camping. While it might rain that day, the probability that Person A will go camping decreases.

(C)

$p(X=1, Y=1) \neq p(X=1)p(Y=1) \Rightarrow X, Y$ independent

$X=1$: Person A watches a movie
$Y=1$: Person A eats popcorn
$Z=1$: person A in the movie theater

$p(X=1, Y=1) \neq p(X=1)p(Y=1)$
$\Rightarrow$ Watching a movie make the person more likely to eat popcorn indicating dependence

$p(X=1, Y=1 \mid Z=1) = p(X=1 \mid Z=1)\, p(Y=1 \mid Z=1)$
$\Rightarrow$ Being in the theater does not necessarily mean that they will eat popcorn, nor does it determine the type of movie they will choose to watch. Thus, their decision to watch a movie and their decision to eat popcorn become conditionally independent

1-4 Bayes Rule

$P(D=1) = 0.01$, $P(D=0) = 0.99$

$P(T=1 \mid D=0) = 0.05 \Rightarrow P(T=0 \mid D=0) = 0.95$

$P(T=0 \mid D=1) = 0.1 \Rightarrow P(T=1 \mid D=1) = 0.90$

(a)

$$P(D=0 \mid T=0) = \frac{P(T=0 \mid D=0) \cdot P(D=0)}{P(T=0)}$$

$$\Rightarrow P(T=0) = P(T=0 \mid D=1) \cdot P(D=1) + P(T=0 \mid D=0) \cdot P(D=0)$$

$$= 0.1 \times 0.01 + 0.95 \times 0.99$$

$$= 0.9415$$

$$P(D=0 \mid T=0) = \frac{0.95 \times 0.99}{0.9415} \cong 0.9985 \quad \#$$

(b)

$$P(D=1 \mid T=1) = \frac{P(T=1 \mid D=1) \cdot P(D=1)}{P(T=1)}$$

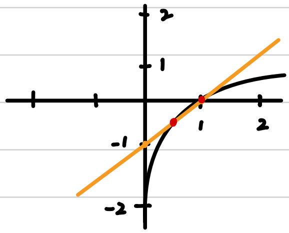
$$P(T=1) = P(T=1 \mid D=0) \cdot P(D=0) + P(T=1 \mid D=1) \cdot P(D=1)$$

$$= 0.0585$$

$$P(D=1 \mid T=1) = \frac{0.9 \times 0.01}{0.0585} \cong 0.1538 \quad \#$$

1-5 Kullback - Leibler distance

(a)



1. $x=1$ both function intersect at the point $(1,0)$
2. for $x<1$, $\log(x)<0$ and $x-1<0$, but $\log(x)$ decrease faster than $x-1$, so $\log(x)$ is less than $x-1$
3. for $x>1$, $\log(x)>0$ and $x-1>0$, but $\log(x)$ grow slower than $x-1$, leading to $\log(x)<x-1$ for all $x>1$

$$f(x) = \log(x) - (x-1)$$

$$f'(x) = \frac{1}{x} - 1 \qquad f'(x)=0 \Rightarrow x=1 \text{ (critical point)}$$

$$f''(x) = -\frac{1}{x^2}$$ ($f''(x)$ is negative for $x>0$, which mean $f(x)$ is concave down)

Since $f(x)$ is concave down and maximum of 0 at $x=1$, it indicates that $f(x) \leq 0$ when $x>0$

$$\Rightarrow f(x) = \log(x) - (x-1) \leq 0$$

$$\Rightarrow \log(x) \leq x-1 \quad \#$$

(b)

$$KL(p,q) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right) = -\sum_i p_i \log\left(\frac{q_i}{p_i}\right)$$

give $\log(x) \leq x-1$, we have

$$-\sum_i p_i \log\left(\frac{q_i}{p_i}\right) \geq -\sum_i p_i \left(\frac{q_i}{p_i} - 1\right) = -\sum_i q_i + \sum_i p_i = 0$$

$$\Rightarrow KL(p,q) \geq 0 \quad \#$$

The equality $KL(p,q)=0$ holds if and only if $\log\left(\frac{q_i}{p_i}\right)=0$, which occurs when $\frac{q_i}{p_i}=1$ for all i #

(c)

$$KL(p,q) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right) = -\sum_i p_i \log\left(\frac{q_i}{p_i}\right)$$

given $\log x = 2\log\sqrt{x}$

$$KL(p,q) = -\sum_i 2p_i \log\sqrt{\left(\frac{q_i}{p_i}\right)} \geq -\sum_i 2p_i\left(\sqrt{\frac{q_i}{p_i}}-1\right)$$

$$\Rightarrow KL(p,q) \geq -2\sum_i \sqrt{p_i q_i} - p_i$$

$$\Rightarrow KL(p,q) \geq \sum_i 2p_i - 2\sqrt{p_i q_i}$$

give $\sum_i p_i = \sum_i q_i = 1$

$$\Rightarrow \sum_i p_i + q_i - 2\sqrt{p_i q_i} = \sum_i (\sqrt{p_i} - \sqrt{q_i})^2 \ \#$$

(d)

$$KL = \sum_i p_i \log\left(\frac{p_i}{q_i}\right)$$

$P \in \{0,1\}$ $\quad P(P=0) = 0.3 \quad P(P=1) = 0.7$

$Q \in \{0,1\}$ $\quad P(Q=0) = 0.6 \quad P(Q=1) = 0.4$

$$KL(P,Q) = 0.3 \times \log\left(\frac{0.3}{0.6}\right) + 0.7 \times \log\left(\frac{0.7}{0.4}\right) \cong 0.0798$$

$$KL(Q,P) = 0.6 \times \log\left(\frac{0.6}{0.3}\right) + 0.4 \times \log\left(\frac{0.4}{0.7}\right) \cong 0.0834$$

$KL(P,Q) \neq KL(Q,P)$ #

1-6 Mutual information

(a)

$$I(X,Y) = \sum_x \sum_y P(x,y) \log\left[\frac{P(x,y)}{P(x)P(y)}\right]$$

$$= -\sum_x \sum_y P(x,y) \log\left[\frac{P(x)P(y)}{P(x,y)}\right]$$

$$\Rightarrow I(X,Y) \geq -\left(\sum_x \sum_y P(x,y) \cdot \left(\frac{P(x)P(y)}{P(x,y)} - 1\right)\right)$$

$$\Rightarrow I(X,Y) \geq -\left(\sum_x \sum_y P(x)P(y) - P(x,y)\right)$$

$$\Rightarrow I(X,Y) \geq -\left(\sum_x P(x) \sum_y P(y) - \sum_x \sum_y P(x,y)\right)$$

$$\Rightarrow I(X,Y) \geq -\left(\sum_x P(x) - \sum_x P(x)\right)$$ marginalization rule

$$\Rightarrow I(X,Y) \geq 0 \ \#$$

(b)

(1)

if $I(X,Y) = 0 \Rightarrow X, Y$ are independent

prove :

$$I(X,Y) = 0 \quad \Rightarrow \quad \frac{P(x,y)}{P(x)P(y)} = 1$$

$\Rightarrow P(x,y) = P(x) \cdot P(y)$ which means X, Y are independent

(2)

if X, Y are independent $\Rightarrow I(X,Y) = 0$

prove :

Since X, Y are independent $p(x,y) = p(x) \cdot p(y)$

Therefore, $I(X,Y) = \sum_x \sum_y P(x,y) \log(1) = 0 \ \#$

Cse250A Hw1 Hangman

October 6, 2024

[1]:
```python
import numpy as np

#open the hw1_word_counts_05-1.txt
with open("D:\Hangman\hw1_word_counts_05-1.txt",'r') as file:
    content=file.readlines()
```

[2]:
```python
# Initialize empty lists
nums=[]
words=[]
prior_probability={}
# Extract words and numbers from content
for item in content:
    word,num=item.split()
    if len(word)==5:
        words.append(word)
        nums.append(int(num))

#Convert lists to NumPy arrays
words=np.array(words)
nums=np.array(nums)

#Sanity check
top_fifteen=words[np.argsort(nums)[-15:][::-1]] # Sort and take the last 15,␣
 ↪then reverse for descending order
print(top_fifteen)
lest_fourteen=words[np.argsort(nums)[:14]] # Sort and take the first 14
print(lest_fourteen)
```

```
['THREE' 'SEVEN' 'EIGHT' 'WOULD' 'ABOUT' 'THEIR' 'WHICH' 'AFTER' 'FIRST'
 'FIFTY' 'OTHER' 'FORTY' 'YEARS' 'THERE' 'SIXTY']
['MAPCO' 'BOSAK' 'CAIXA' 'OTTIS' 'TROUP' 'CLEFT' 'FOAMY' 'CCAIR' 'SERNA'
 'YALOM' 'TOCOR' 'NIAID' 'PAXON' 'FABRI']
```

[3]:
```python
#prior probability
total=np.sum(nums)
for i in range(len(nums)):
    prob=nums[i]/total
    prior_probability[words[i]]=prob
```

```
[4]: # def P(L|w) = 1 if l is the ith letter of w P(L|w) = 0 otherwise
def marginal(word,next_char,position):
    flag=False
    for i in position:      #Check if the character exist in the word in␣
↪specific position
        if word[i-1]==next_char:
            flag=True
            return 1        # Return 1 indicating that the character was found.
    else:
        return 0            # Return 0 indicating that the character was not␣
↪found.
```

```
[5]: #calculate the denominator of Bayes Rule
def denominator(true_charac, true_positions, false_charac):
    false_positions=list(set([1,2,3,4,5])-set(true_positions)) # Calculate the␣
↪positions that have not been guessed
    denominator=0
    for w in words:
        flag1=True                                  # Assume that all true␣
↪characters are in the correct positions
        flag2=False                                 # Assume that false␣
↪positions do not contain incorrect characters
        for i,charac in enumerate(true_charac):
            if w[true_positions[i]-1]!=charac:
                flag1=False                         # Set flag1 to False if a␣
↪true character is in the wrong position
        # Check the false positions to ensure they do not contain guessed wrong␣
↪or already guessed characters
        for i in false_positions:
            if (w[i - 1] in false_charac) or (w[i - 1] in true_charac):
                flag2 = True                        # Set flag2 to True if a␣
↪false position has a wrong or repeated character
        if flag1 ==True and flag2 == False:
            denominator += prior_probability[w]
    return denominator
```

```
[6]: #Bayes's Rule Application
def bayes(word,true_charac, true_positions, false_charac,denominator):
    false_positions=list(set([1,2,3,4,5])-set(true_positions))
    flag1=True                                  # Assume that all true␣
↪characters are in the correct positions
    flag2=False                                 # Assume that false positions␣
↪do not contain incorrect characters
    for i,charac in enumerate(true_charac):
        if word[true_positions[i]-1]!=charac:
```

```
            flag1=False                         # Set flag1 to False if a true␣
↪character is in the wrong position
        # Check the false positions to ensure they do not contain guessed wrong␣
↪or already guessed characters
    for i in false_positions:
        if (word[i - 1] in false_charac) or (word[i - 1] in true_charac):
            flag2 = True                        # Set flag2 to True if a␣
↪false position has a wrong or repeated character
    if flag1 ==True and flag2 == False:
        numerator=prior_probability[word]
    else:
        numerator=0
    return numerator / denominator
```

[7]:
```
#compute predicitive probability
def pred_prob(next_charac,true_charac, true_positions, false_charac):
    # Initialize the probability to zero
    prob=0
    # Calculate the denominator using the provided true characters and␣
↪positions, and the false characters
    Denominator= denominator(true_charac, true_positions, false_charac)
    for word in words:
        Marginal = marginal(word,␣
↪next_charac,list(set([1,2,3,4,5])-set(true_positions)))
        # Check if the marginal probability is not zero
        if Marginal != 0:
            # Calculate the Bayes probability
            Bayes = bayes(word,true_charac, true_positions, false_charac,␣
↪Denominator)
            # Update the total probability by adding the product of Marginal␣
↪and Bayes probabilities
            prob += Marginal*Bayes
    return prob
```

[8]:
```
#TEST CASE:
correct_guess = [[], [], ["A", "S"], ["A", "S"], ["O"], [], ["D", "I"], ["D",␣
↪"I"], ["U"]]
correct_pos = [[],[], [1, 5], [1, 5], [3], [], [1, 4], [1, 4], [2]]
incorrect_guess = [[], ["E", "A"], [], ["I"], ["A", "E", "M", "N", "T"], ["E",␣
↪"O"], [], ["A"], ["A", "E", "I", "O", "S"]]
alphabet=[]
# Loop through ASCII values for uppercase letters 'A' (65) to 'Z' (90)
for i in range(65,91):
    alphabet.append(chr(i))
#Iterate through each round of guesses
for i in range(len(correct_guess)):
```

```
    true_charac, true_positions, false_charac = correct_guess[i],␣
↪correct_pos[i], incorrect_guess[i]
    max_prob = 0
    next_guess = ""

# Iterate through alphabet, excluding already guessed characters
    for char in [item for item in alphabet if item not in true_charac and item␣
↪not in false_charac]:
        prob = pred_prob(char, true_charac, true_positions, false_charac)
        if prob > max_prob:                         # If the current probability␣
↪is greater than the maximum probability
            max_prob = prob                         # Update the maximum␣
↪probability
            next_guess = char                       # Set the next guess to the␣
↪current character
    print("The next best guess is", next_guess, "with probability", max_prob)
```

```
The next best guess is E with probability 0.5394172389647948
The next best guess is O with probability 0.5340315651557679
The next best guess is E with probability 0.7715371621621622
The next best guess is E with probability 0.7127008416220354
The next best guess is R with probability 0.7453866259829711
The next best guess is I with probability 0.6365554141009618
The next best guess is A with probability 0.8206845238095241
The next best guess is E with probability 0.7520746887966806
The next best guess is Y with probability 0.6269651101630528
```

[ ]:

```

```