CSE 250A HW1

Akhil Nallacheruvu

October 7 2024

1.1 Conditioning on background evidence

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

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

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

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

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

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

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

c) $\Sigma_y P(X,Y=y|E) = \Sigma_y \frac{P(X,Y=y,E)}{P(E)} = \frac{\Sigma_y P(X,Y=y,E)}{P(E)} = \frac{P(X,E)}{P(E)} = \boxed{P(X|E)}$

1.2 Conditional independence

There are 3 statements:

(1) $P(X,Y|E) = P(X|E)P(Y|E)$
(2)$P(X|Y,E) = P(X|E)$
(3) $P(Y|X,E) = P(Y|E)$

Assuming Statement (1) $P(X,Y|E) = P(X|E)P(Y|E)$ is true:

$P(X|E) = \frac{P(X,Y|E)}{P(Y|E)} = \frac{\frac{P(X,Y,E)}{P(E)}}{\frac{P(Y,E)}{P(E)}} = \frac{P(X,Y,E)}{P(Y,E)} = \boxed{P(X|Y,E)} \rightarrow \boxed{\text{Statement (2) holds true}}$

$P(Y|E) = \frac{P(X,Y|E)}{P(X|E)} = \frac{\frac{P(X,Y,E)}{P(E)}}{\frac{P(X,E)}{P(E)}} = \frac{P(X,Y,E)}{P(X,E)} = \boxed{P(Y|X,E)} \rightarrow \boxed{\text{Statement (3) holds true}}$

$\boxed{\text{Statement (1) implies Statement (2) and Statement (3).}}$

Now let's assume instead we only know Statement (2) $P(X|Y,E) = P(X|E)$ to be true.
$P(X|E)P(Y|E) = P(X|Y,E)P(Y|E) = \frac{P(X,Y,E)}{P(Y,E)} \frac{P(Y,E)}{P(E)} = \frac{P(X,Y,E)}{P(E)} = \boxed{P(X,Y|E)} \rightarrow$ $\boxed{\text{Statement (1) holds true.}}$

Since we have proven that Statement (1) is true:
$P(Y|E) = \frac{P(X,Y|E)}{P(X|E)} = \frac{P(X,Y,E)}{P(X,E)} = \boxed{P(Y|X,E)} \rightarrow \boxed{\text{Statement (3) holds true.}}$

$\boxed{\text{Statement (2) implies Statement(1) and Statement(3).}}$

Now let's assume instead we only know Statement (3) $P(Y|X,E) = P(Y|E)$ to be true.
$P(X|E)P(Y|E) = P(X|E)P(Y|X,E) = \frac{P(X,E)}{P(E)} \frac{P(Y,X,E)}{P(X,E)} = \frac{P(Y,X,E)}{P(E)} = \boxed{P(X,Y|E)} \rightarrow$ $\boxed{\text{Statement (1) holds true.}}$

Since we have proven Statement (1) true:
$P(X|E) = \frac{P(X,Y|E)}{P(Y|E)} = \frac{\frac{P(X,Y,E)}{P(E)}}{\frac{P(Y,E)}{P(E)}} = \frac{P(X,Y,E)}{P(Y,E)} = \boxed{P(X|Y,E)} \rightarrow \boxed{\text{Statement (2) holds true.}}$

$\boxed{\text{Statement (3) implies Statement (1) and Statement (2).}}$

1.3 Creative writing

a) $P(X=1) < P(X=1|Y=1) < P(X=1|Y=1,Z=1)$

X = A road accident happens
Y = There is heavy traffic on the road
Z = There is heavy rainfall

b) $P(X=1|Y=1) > P(X=1)$
$P(X=1|Y=1,Z=1) < P(X=1|Y=1)$

X = A road accident happens
Y = There is heavy traffic on the road
Z = There is a police car present

c) P(X=1, Y=1) $\neq P(X=1)P(Y=1)$
$P(X=1,Y=1|Z=1) = P(X=1|Z=1)P(Y=1|Z=1)$

X = Student studied for exam
Y = Student received a good score
Z = Student already knew the subject

1.4 Bayes Rule

a) D=0 $\rightarrow$not doping
D=1$\rightarrow$doping
$T=0$ $\rightarrow$negative test
$T=1$ $\rightarrow$positive test

$$P(D=0|T=0) = \frac{P(T=0|D=0)P(D=0)}{P(T=0|D=0)P(D=0)+P(T=0|D=1)P(D=1)}$$

$P(D=1) = 0.01 \rightarrow P(D=0) = 0.99$
$P(T=1|D=0) = 0.05 \rightarrow P(T=0|D=0) = 0.95$
$P(T=0|D=1) = 0.1 \rightarrow P(T=1|D=1) = 0.9$

$$P(D=0|T=0) = \frac{0.95(0.99)}{0.95(0.99)+0.1(0.01)} = \boxed{0.999}$$

b)$P(D=1|T=1) = \frac{P(T=1|D=1)P(D=1)}{P(T=1|D=1)P(D=1)+P(T=1|D=0)P(D=0)} = \frac{0.9(0.01)}{0.9(0.01)+0.05(0.99)} = \boxed{0.154}$

1.5 Kullback-Leibler distance

a) 1) Graph:

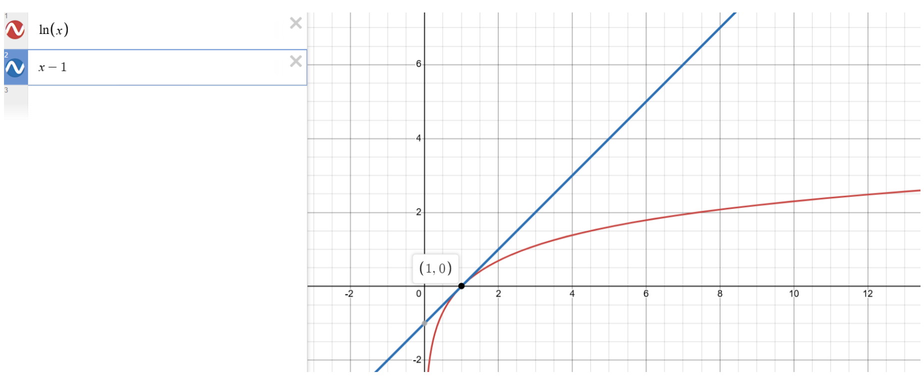

Figure 1: Graph of log(x) and x-1

The graphs of log(x) and x-1 show that the value of log(x) is always less than x-1 except for at x=1 where both log(x) and x-1 are equal to 0. This can be confirmed by the graph of log(x)-(x-1), which shows the difference between log(x) and x-1 to always be negative except at x=1 where the difference is 0 since they're equal.

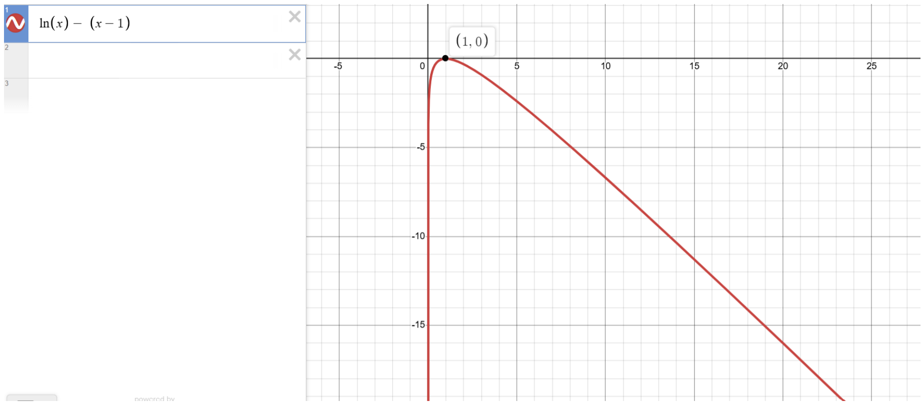

Figure 2: Graph of log(x)-(x-1)

2) Differentiation: Graphing $\frac{d}{dx}$(log(x)-(x-1)) results in f'(x) = $\frac{1}{x}$-1 . f'(x) reaches a critical point at x=1, which means that f(x) either has a minimum, maximum, or inflection point at that point. Based on the behavior of the graph of log(x)-(x-1), we see that there is a maximum at x=1 where f(1) = 0. If the maximum of the difference is 0, then both log(x) and x-1 are equal at that point and $log(x) < x - 1$ at all other points.

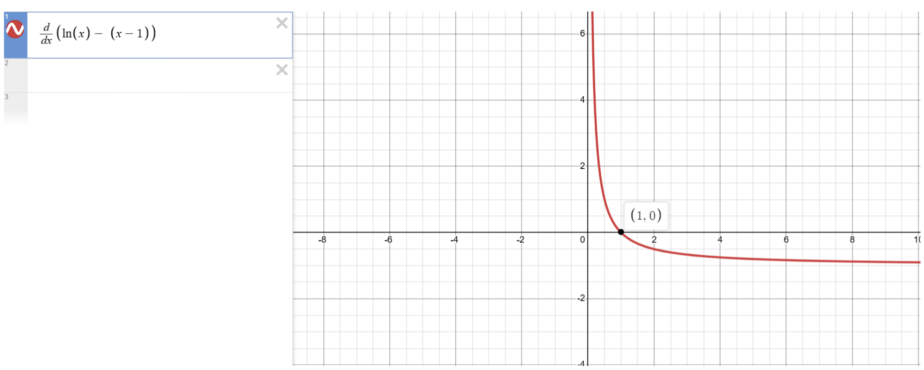

Figure 3: Graph of $\frac{d}{dx}$(log(x)-(x-1))

b) KL(p,q) = $\Sigma_i p_i log(\frac{p_i}{q_i})$ = $(\Sigma_i p_i log(p_i) - \Sigma_i p_i log(q_i))$ = $-(\Sigma_i p_i log(q_i) - \Sigma_i p_i log p_i)$ = $-\Sigma_i p_i log(\frac{q_i}{p_i})$

Since we know that $log(x) \leq x - 1$, if we set x = $\frac{q_i}{p_i}$, we get $log(\frac{q_i}{p_i}) \leq \frac{q_i}{p_i}$-1
This means:

$\Sigma_i p_i log(\frac{q_i}{p_i}) \leq \Sigma_i p_i(\frac{q_i}{p_i} - 1)$
$-\Sigma_i p_i log(\frac{q_i}{p_i}) \geq -\Sigma_i p_i(\frac{q_i}{p_i} - 1) \rightarrow -\Sigma_i p_i log(\frac{q_i}{p_i}) \geq -\Sigma_i q_i + \Sigma_i p_i$

According to the axioms of probability, $\Sigma_i q_i = 1, \Sigma_i p_i = 1$
$-\Sigma_i p_i log(\frac{q_i}{p_i}) \geq 0$
Since $KL(p,q) = -\Sigma_i p_i log(\frac{q_i}{p_i}), KL(p,q) \geq 0$.

If $p_i = q_i, KL(p,q) = 0$ since $KL(p,q) = \Sigma_i p_i log(\frac{p_i}{q_i}) = \Sigma_i p_i log(1) = 0$

c)$KL(p,q) = -\Sigma_i p_i log(\frac{q_i}{p_i}) = -\Sigma_i 2p_i log(\frac{\sqrt{q_i}}{\sqrt{p_i}})$

Since $log(x) \leq x - 1$, if we set $x = \frac{\sqrt{q_i}}{\sqrt{p_i}}$, $\Sigma_i 2p_i log(\frac{\sqrt{q_i}}{\sqrt{p_i}}) \leq \Sigma_i 2p_i(\frac{\sqrt{q_i}}{\sqrt{p_i}} - 1)$, which means $-\Sigma_i 2p_i log(\frac{\sqrt{q_i}}{\sqrt{p_i}}) \geq -\Sigma_i 2p_i(\frac{\sqrt{q_i}}{\sqrt{p_i}} - 1) = -\Sigma_i \frac{2p_i\sqrt{q_i}}{\sqrt{p_i}} - 2p_i = \Sigma_i 2p_i - \frac{2p_i\sqrt{q_i}}{\sqrt{p_i}} = \Sigma_i p_i + q_i - 2\sqrt{\frac{p_i^2 q_i}{p_i}} = \Sigma_i p_i + q_i - 2\sqrt{p_i q_i} = \Sigma_i(\sqrt{p_i} - \sqrt{q_i})^2$

$\boxed{\text{KL(p,q)} \geq \Sigma_i(\sqrt{p_i} - \sqrt{q_i})^2}$

d) Given $p_i = \{\frac{1}{4}, \frac{3}{4}\}$, $q_i = \{\frac{1}{3}, \frac{2}{3}\}$

$KL(p,q) = \frac{1}{4}log(\frac{\frac{1}{4}}{\frac{1}{3}}) + \frac{3}{4}log(\frac{\frac{3}{4}}{\frac{2}{3}}) = \frac{1}{4}log(\frac{3}{4}) + \frac{3}{4}log(\frac{9}{8}) = 0.0455$

$KL(q,p) = \frac{1}{3}log(\frac{\frac{1}{3}}{\frac{1}{4}}) + \frac{2}{3}log(\frac{\frac{2}{3}}{\frac{3}{4}}) = \frac{1}{3}log(\frac{4}{3}) + \frac{2}{3}log(\frac{8}{9}) = 0.00754$

$\boxed{\text{KL(p,q)} \neq KL(q,p)}$

1.6 Mutual information

a) I(X,Y) $= \Sigma_x\Sigma_y P(x,y)log[\frac{P(x,y)}{P(x)P(y)}] = -\Sigma_x\Sigma_y P(x,y)log[\frac{P(x)P(y)}{P(x,y)}]$

Since it is known that $log(x) \leq x - 1$, if we set $x = \frac{P(x)P(y)}{P(x,y)}$, we get

$\Sigma_x\Sigma_y P(x,y)log(\frac{P(x)P(y)}{P(x,y)}) \leq \Sigma_x\Sigma_y P(x,y)(\frac{P(x)P(y)}{P(x,y)} - 1)$
$\downarrow$
$-\Sigma_x\Sigma_y P(x,y)log(\frac{P(x)P(y)}{P(x,y)}) \geq -\Sigma_x\Sigma_y P(x,y)(\frac{P(x)P(y)}{P(x,y)} - 1) = -\Sigma_x\Sigma_y P(x)P(y) - P(x,y) = \Sigma_x\Sigma_y P(x,y) - P(x)P(y) = \Sigma_x\Sigma_y P(x,y) - \Sigma_x\Sigma_y P(x)P(y) = 1 - \Sigma_x P(x)\Sigma_y P(y) = 0$

$\boxed{\text{I(X,Y)} \geq 0}$

b) If X and Y are independent random variables, $P(x,y) = P(x)P(y)$. This means $I(X,Y) = \Sigma_x\Sigma_y P(x,y)log[\frac{P(x,y)}{P(x)P(y)}] = \Sigma_x\Sigma_y P(x,y)log(1) = 0$. Otherwise, $I(X,Y) > 0$.

HW1

October 8, 2024

[1]:
```python
import pandas as pd
import string
```

[2]:
```python
lines = []
with open('hw1_word_counts_05-1.txt', 'r') as file:
    for line in file:
        lines.append(line.strip())
```

[3]:
```python
word = []
number = []
for i in range(len(lines)):
    word.append(lines[i].split()[0])
    number.append(int(lines[i].split()[1]))
```

[4]:
```python
df = list(zip(word, number))
```

[5]:
```python
df = pd.DataFrame({'Word':word, 'Frequency':number})
```

[6]:
```python
df
```

[6]:
```
       Word  Frequency
0     AARON        413
1     ABABA        199
2     ABACK         64
3     ABATE         69
4     ABBAS        290
...     ...        ...
6530  ZVIAD         30
6531  ZWEIG         44
6532  ZWICK         34
6533  ZYCIE         14
6534  ZYMAN         16

[6535 rows x 2 columns]
```

[7]:
```python
df_increasing = df.sort_values(by='Frequency', ascending=False).reset_index()
```

[8]:
```python
df_increasing
```

[8]:
```
      index   Word  Frequency
0      5821  THREE     273077
1      5102  SEVEN     178842
2      1684  EIGHT     165764
3      6403  WOULD     159875
4        18  ABOUT     157448
...     ...    ...        ...
6530   3554  MAPCO          6
6531    895  CAIXA          6
6532   4160  OTTIS          6
6533   5985  TROUP          6
6534    712  BOSAK          6

[6535 rows x 3 columns]
```

[9]:
```
for i in range(15):
    print(str(i+1)+")", df_increasing['Word'][i], df_increasing['Frequency'][i]/
 ↪sum(df['Frequency']))
```

```
1) THREE 0.035627148686531277
2) SEVEN 0.023332724928853858
3) EIGHT 0.021626496097709325
4) WOULD 0.02085818430793947
5) ABOUT 0.020541544349751077
6) THEIR 0.018974130893766185
7) WHICH 0.018545160072784138
8) AFTER 0.01436452108630337
9) FIRST 0.014345603577470525
10) FIFTY 0.013942725872119989
11) OTHER 0.013836135494765265
12) FORTY 0.012387837111638222
13) YEARS 0.011598389898206841
14) THERE 0.01128553344178502
15) SIXTY 0.009535207245223231
```

[10]:
```
df_decreasing = df.sort_values(by='Frequency', ascending=True).reset_index()
```

[11]:
```
df_decreasing
```

[11]:
```
      index   Word  Frequency
0      3554  MAPCO          6
1       712  BOSAK          6
2       895  CAIXA          6
3      4160  OTTIS          6
4      5985  TROUP          6
...     ...    ...        ...
6530     18  ABOUT     157448
6531   6403  WOULD     159875
```

```
6532   1684  EIGHT     165764
6533   5102  SEVEN     178842
6534   5821  THREE     273077

[6535 rows x 3 columns]
```

[12]:
```python
for i in range(14):
    print(str(i+1)+")", df_decreasing['Word'][i], df_decreasing['Frequency'][i]/
↪sum(df_decreasing['Frequency']))
```

```
1) MAPCO 7.827934689453437e-07
2) BOSAK 7.827934689453437e-07
3) CAIXA 7.827934689453437e-07
4) OTTIS 7.827934689453437e-07
5) TROUP 7.827934689453437e-07
6) CLEFT 9.13259047102901e-07
7) FOAMY 9.13259047102901e-07
8) CCAIR 9.13259047102901e-07
9) SERNA 9.13259047102901e-07
10) YALOM 9.13259047102901e-07
11) TOCOR 9.13259047102901e-07
12) NIAID 9.13259047102901e-07
13) PAXON 9.13259047102901e-07
14) FABRI 9.13259047102901e-07
```

The above was for part a. Below this will be part b.

[13]:
```python
prior_arr = []
for i in range(len(number)):
    total_freq = sum(number)
    prior_arr.append(number[i] / total_freq)
```

[14]:
```python
def filter_words(c, correct, incorrect):
    word_validity = []
    present_positions = [pos for _, pos in correct]

    for i in range(len(word)):
        w = word[i]
        valid = True

        if any(absent in w for absent in incorrect):
            word_validity.append(0)
            continue

        for char, pos in correct:
            if w[pos] != char or w[:pos].count(char) > 0 or w[pos+1:].
↪count(char) > 0:
                valid = False
```

```
                break

        if valid and (c is None or (c in w and i not in present_positions)):
            word_validity.append(1)
        else:
            word_validity.append(0)

    return word_validity
```

[15]:
```
def calculate_probability(l, correct, incorrect):
    word_validity = filter_words(None, correct, incorrect)
    letter_validity = filter_words(l, correct, incorrect)

    total_valid_prob = sum([word_validity[i] * prior_arr[i] for i in␣
 ↪range(len(word))])

    if total_valid_prob == 0:
        return 0

    word_probabilities = [word_validity[i] * prior_arr[i] / total_valid_prob for␣
 ↪i in range(len(word))]
    letter_prob = sum([letter_validity[i] * word_probabilities[i] for i in␣
 ↪range(len(word))])

    return letter_prob
```

[16]:
```
def find_next_letter(correct, incorrect):
    max_prob = 0
    next_letter = None
    present_chars = [char for char, _ in correct]
    letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    for char in letters:
        if char not in present_chars:
            prob = calculate_probability(char, correct, incorrect)

            if prob > max_prob:
                max_prob = prob
                next_letter = char

    return next_letter
```

[17]:
```
correct = []
incorrect = []
print(find_next_letter(correct, incorrect))
print(calculate_probability(find_next_letter(correct, incorrect), correct,␣
 ↪incorrect))
```

E

```
0.5394172389647974
```

[18]:
```python
correct = []
incorrect = ['E','A']
print(find_next_letter(correct, incorrect))
print(calculate_probability(find_next_letter(correct, incorrect), correct,␣
 ↪incorrect))
```

```
O
0.5340315651557657
```

[19]:
```python
correct = [('A',0), ('S',4)]
incorrect = []
print(find_next_letter(correct, incorrect))
print(calculate_probability(find_next_letter(correct, incorrect), correct,␣
 ↪incorrect))
```

```
E
0.7715371621621622
```

[20]:
```python
correct = [('A',0), ('S',4)]
incorrect = ['I']
print(find_next_letter(correct, incorrect))
print(calculate_probability(find_next_letter(correct, incorrect), correct,␣
 ↪incorrect))
```

```
E
0.7127008416220353
```

[21]:
```python
correct = [('O',2)]
incorrect = ['A','E','M','N','T']
print(find_next_letter(correct, incorrect))
print(calculate_probability(find_next_letter(correct, incorrect), correct,␣
 ↪incorrect))
```

```
R
0.7453866259829712
```

[22]:
```python
correct = []
incorrect = ['E','O']
print(find_next_letter(correct, incorrect))
print(calculate_probability(find_next_letter(correct, incorrect), correct,␣
 ↪incorrect))
```

```
I
0.6365554141009611
```

[23]:
```python
correct = [('D',0),('I',3)]
incorrect = []
print(find_next_letter(correct, incorrect))
```

```
print(calculate_probability(find_next_letter(correct, incorrect), correct,␣
↪incorrect))
```

```
A
0.8206845238095238
```

[24]:
```
correct = [('D',0),('I',3)]
incorrect = ['A']
print(find_next_letter(correct, incorrect))
print(calculate_probability(find_next_letter(correct, incorrect), correct,␣
↪incorrect))
```

```
E
0.7520746887966805
```

[25]:
```
correct = [('U',1)]
incorrect = ['A','E','I','O','S']
print(find_next_letter(correct, incorrect))
print(calculate_probability(find_next_letter(correct, incorrect), correct,␣
↪incorrect))
```

```
Y
0.626965110163053
```