CSE 250 HW4

Akhil Nallacheruvu

October 2024

4.1 Maximum likelihood estimation in belief networks

a) $\mathrm{P}(\mathrm{X}_1 = x_1) = \frac{COUNT(X_1=x_1)}{COUNT(X_1)} = \frac{COUNT(X_1=x_1)}{T}$

$P(X_2 = x_2|X_1 = x_1) = \frac{COUNT(X_1=x_1,X_2=x_2)}{COUNT(X_1=x_1)}$

$P(X_3 = x_3|X_1 = x_1, X_2 = x_2) = P(X_3 = x_3|X_2 = x_2) = \frac{COUNT(X_2=x_2,X_3=x_3)}{COUNT(X_2=x_2)}$

$\boxed{P_{ML}(X_{i+1} = x_{i+1}|X_i = x_i) = \frac{COUNT(X_i=x_i,X_{i+1}=x_{i+1})}{COUNT(X_i=x_i)}}$

b)$P(X_n = x_n) = \frac{COUNT(X_n=x_n)}{COUNT(X_n)}$

$P(X_{n-1} = x_{n-1}|X_n = x_n) = \frac{COUNT(X_{n-1}-x_{n-1},X_n=x_n)}{COUNT(X_n)}$

$\boxed{\mathrm{P}_{ML}(X_i = x_i|X_{i+1} = x_{i+1}) = \frac{COUNT(X_{i+1}=x_{i+1},X_i=x_i)}{COUNT(X_{i+1}=x_{i+1})}}$

c)$P(G_1 = g_1) = P(X_1, ..., X_n) = P(X_1)P(X_2|X_1)...P(X_n|X_1, ..., X_{n-1}) =$
$P(X_1)P(X_2|X_1)P(X_3|X_2)...P(X_n|X_{n-1}) =$

$\frac{COUNT(X_1=x_1)}{T} * \frac{COUNT(X_2=x_2,X_1=x_1)}{COUNT(X_1=x_1)} * \frac{COUNT(X_3=x_3,X_2=x_2)}{COUNT(X_2=x_2)} .... = \boxed{\frac{\Pi_{i=1}^{n-1} COUNT(X_{i+1}=x_{i+1},X_i=x_i)}{T\Pi_{i=2}^{n-1} COUNT(X_i=x_i)}}$

$P(G_2 = g_2) = P(X_1, ..., X_n) = P(X_n)P(X_{n-1}|X_n)P(X_{n-2}|X_{n-1}, X_n)...P(X_1|X_2, ..., X_n) =$

$P(X_n)P(X_{n-1}|X_n)P(X_{n-2}|X_{n-1})...P(X_1|X_2) = \frac{COUNT(X_n=x_n)}{T} * \frac{COUNT(X_{n-1}=x_{n-1},X_n=x_n)}{COUNT(X_n=x_n)} *$

$\frac{COUNT(X_{n-2}=x_{n-2},X_{n-1}=x_{n-1})}{COUNT(X_{n-1}=x_{n-1})} = \boxed{\frac{\Pi_{i=1}^{n-1} COUNT(X_{i+1}=x_{i+1},X_i=x_i)}{T\Pi_{i=2}^{n-1} COUNT(X_i=x_i)}}$

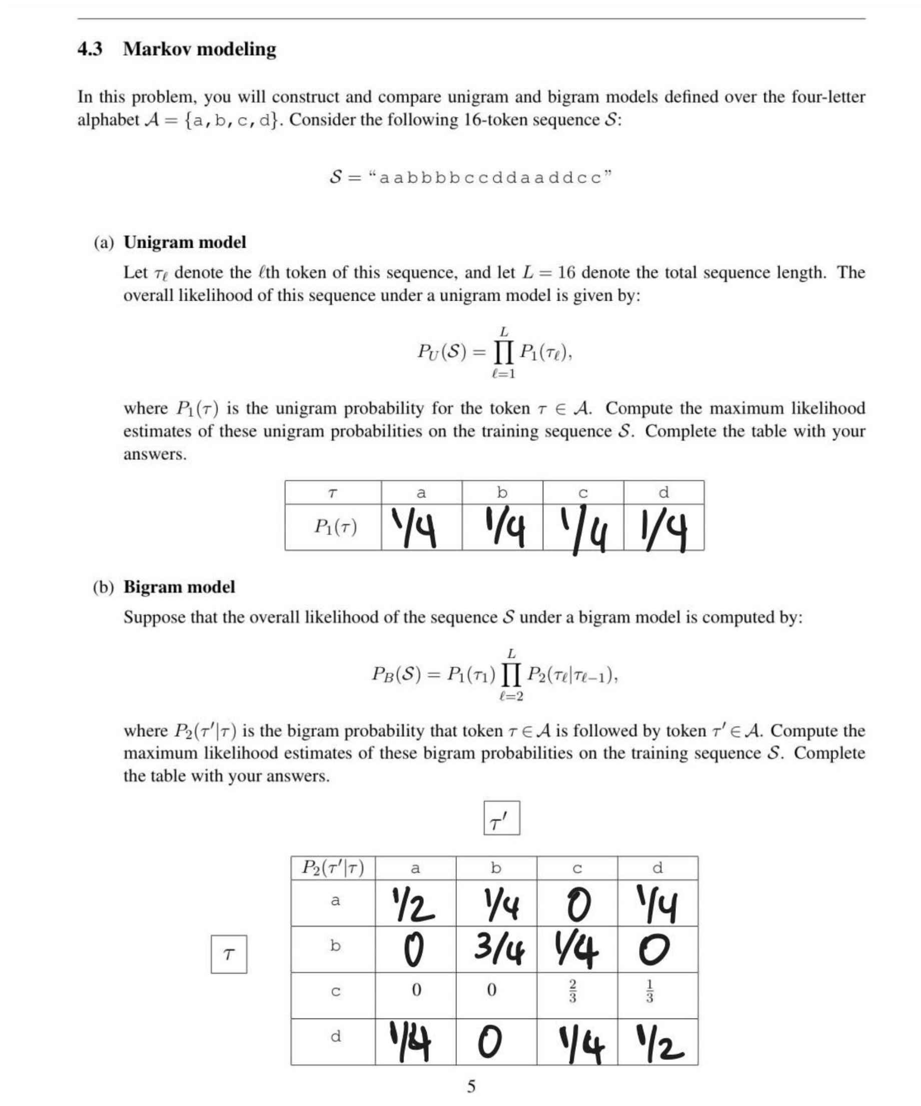
$\boxed{\mathrm{P}(\mathrm{G}_1 = g_1) = P(G_2 = g_2)}$

d)$P(G_3 = g_3) = P(X_1, ..., X_n) = P(X_1 = x_1|X_2 = x_2)P(X_2 = x_2)P(X_3 = x_3|X_2 = x_2, X_4 = x_4)P(X_4 = x_4)P(X_5 = x_5|X_4 = x_4, X_6 = x_6).....P(X_{n-2} = x_{n-2}|X_{n-3} = x_{n-3}, X_{n-1} = x_{n-1})P(X_n = x_n|X_{n-1} = x_{n-1}) = \frac{COUNT(X_1=x_1,X_2=x_2)}{COUNT(X_2=x_2)} *$
$\frac{COUNT(X_{n-1}=x_{n-1},X_n=x_n)}{COUNT(X_{n-1}=x_{n-1})} * \Pi_{i=2k,k=1}^{i=n-1,k=(n-1)/2} \frac{COUNT(X_{i-1}=x_{i-1},X_i=x_i,X_{i+1}=x_{i+1})}{COUNT(X_{i-1}=x_{i-1},X_{i+1}=x_{i+1})} *$

$\Pi_{i=2k+1,k=1}^{i=n-2,k=(n-2)/2} \frac{COUNT(X_i=x_i)}{T}$

$\boxed{\text{No. P(G}_3 = g_3\text{) won't give rise to the same joint distribution.}}$

4.3 Markov Modeling



4.3 Markov modeling

In this problem, you will construct and compare unigram and bigram models defined over the four-letter alphabet $\mathcal{A} = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$. Consider the following 16-token sequence $\mathcal{S}$:

$$\mathcal{S} = \text{“}\texttt{a a b b b b c c d d a a d d c c}\text{”}$$

(a) **Unigram model**

Let τ_ℓ denote the ℓth token of this sequence, and let $L = 16$ denote the total sequence length. The overall likelihood of this sequence under a unigram model is given by:

$$P_U(\mathcal{S}) = \prod_{\ell=1}^{L} P_1(\tau_\ell),$$

where $P_1(\tau)$ is the unigram probability for the token $\tau \in \mathcal{A}$. Compute the maximum likelihood estimates of these unigram probabilities on the training sequence $\mathcal{S}$. Complete the table with your answers.

τ	a	b	c	d
$P_1(\tau)$	1/4	1/4	1/4	1/4

(b) **Bigram model**

Suppose that the overall likelihood of the sequence $\mathcal{S}$ under a bigram model is computed by:

$$P_B(\mathcal{S}) = P_1(\tau_1) \prod_{\ell=2}^{L} P_2(\tau_\ell | \tau_{\ell-1}),$$

where $P_2(\tau' | \tau)$ is the bigram probability that token $\tau \in \mathcal{A}$ is followed by token $\tau' \in \mathcal{A}$. Compute the maximum likelihood estimates of these bigram probabilities on the training sequence $\mathcal{S}$. Complete the table with your answers.

τ'

τ

$P_2(\tau' \vert \tau)$	a	b	c	d
a	1/2	1/4	0	1/4
b	0	3/4	1/4	0
c	0	0	$\frac{2}{3}$	$\frac{1}{3}$
d	1/4	0	1/4	1/2

5

(c) **Likelihoods**

Consider again the training sequence $\mathcal{S}$, as well as three test sequences $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ of the same length, shown below. Note that $\mathcal{T}_2$ and $\mathcal{T}_3$ contain bigrams (underlined) that are not in the training sequence $\mathcal{S}$.

$$\begin{aligned}
\mathcal{S} &= \text{“a a b b b b c c d d a a d d c c”} \\
\mathcal{T}_1 &= \text{“a d a d a d a d a d a d a d a d”} \\
\mathcal{T}_2 &= \text{“a a a a d d d d c c c }\underline{\text{c b}}\text{ b b b”} \\
\mathcal{T}_3 &= \text{“}\underline{\text{b d b d b d b d b d b d b d b d}}\text{”}
\end{aligned}$$

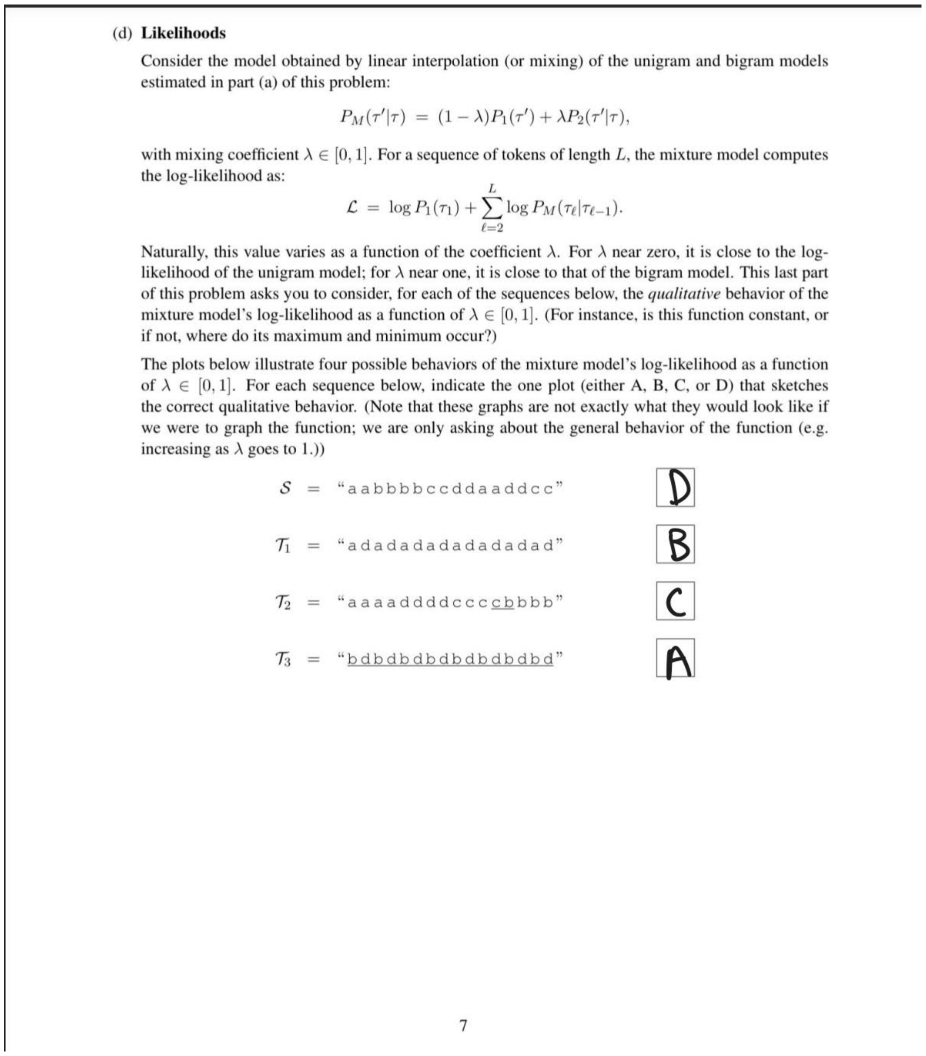
Consider the probabilities of these sequences under the unigram and bigram models from parts (a) and (b) of this problem (i.e., the models that you estimated from the training sequence $\mathcal{S}$). For each of the following, indicate whether the probability on the left is equal $(=)$, greater $(>)$, or less $(<)$ than the probability on the right.

Note: you can (and should) answer these questions without explicitly computing the numerical values of the expressions on the left and right hand sides.

$$P_U(\mathcal{S}) \ \boxed{<}\ P_U(\mathcal{T}_1)$$
$$P_U(\mathcal{S}) \ \boxed{=}\ P_U(\mathcal{T}_2)$$
$$P_U(\mathcal{S}) \ \boxed{<}\ P_U(\mathcal{T}_3)$$

$$P_B(\mathcal{T}_1) \ \boxed{<}\ P_B(\mathcal{S})$$
$$P_B(\mathcal{T}_2) \ \boxed{<}\ P_B(\mathcal{S})$$
$$P_B(\mathcal{T}_3) \ \boxed{=}\ P_B(\mathcal{T}_2)$$

$$P_U(\mathcal{S}) \ \boxed{<}\ P_B(\mathcal{S})$$
$$P_U(\mathcal{T}_1) \ \boxed{>}\ P_B(\mathcal{T}_1)$$
$$P_U(\mathcal{T}_2) \ \boxed{<}\ P_B(\mathcal{T}_2)$$
$$P_U(\mathcal{T}_3) \ \boxed{>}\ P_B(\mathcal{T}_3)$$



(d) **Likelihoods**

Consider the model obtained by linear interpolation (or mixing) of the unigram and bigram models estimated in part (a) of this problem:

$$P_M(\tau'|\tau) \;=\; (1-\lambda)P_1(\tau') + \lambda P_2(\tau'|\tau),$$

with mixing coefficient $\lambda \in [0,1]$. For a sequence of tokens of length L, the mixture model computes the log-likelihood as:

$$\mathcal{L} \;=\; \log P_1(\tau_1) + \sum_{\ell=2}^{L} \log P_M(\tau_\ell|\tau_{\ell-1}).$$

Naturally, this value varies as a function of the coefficient λ. For λ near zero, it is close to the log-likelihood of the unigram model; for λ near one, it is close to that of the bigram model. This last part of this problem asks you to consider, for each of the sequences below, the *qualitative* behavior of the mixture model's log-likelihood as a function of $\lambda \in [0,1]$. (For instance, is this function constant, or if not, where do its maximum and minimum occur?)

The plots below illustrate four possible behaviors of the mixture model's log-likelihood as a function of $\lambda \in [0,1]$. For each sequence below, indicate the one plot (either A, B, C, or D) that sketches the correct qualitative behavior. (Note that these graphs are not exactly what they would look like if we were to graph the function; we are only asking about the general behavior of the function (e.g. increasing as λ goes to 1.))

$\mathcal{S}$ = “a a b b b b c c d d a a d d c c” $\boxed{D}$

$\mathcal{T}_1$ = “a d a d a d a d a d a d a d a d” $\boxed{B}$

$\mathcal{T}_2$ = “a a a a d d d d c c c $\underline{\text{c b}}$ b b b” $\boxed{C}$

$\mathcal{T}_3$ = “$\underline{\text{b d b d b d b d b d b d b d b d}}$” $\boxed{A}$

7

HW4

October 28, 2024

[1]:
```python
import pandas as pd
import math
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```python
#a)
with open('hw4_vocab.txt', 'r') as f:
    words = f.readlines()
    f.close()
with open('hw4_unigram.txt', 'r') as f:
    unigram_counts = f.readlines()
    f.close()
with open('hw4_bigram.txt', 'r') as f:
    bigram_counts = f.readlines()
    f.close()
```

[3]:
```python
int_unigram_ct = []
for i in range(len(unigram_counts)):
    int_unigram_ct.append(int(unigram_counts[i]))
```

[4]:
```python
return_list = []
for i in range(len(words)):
    if words[i][0] == 'M':
        return_list.append([words[i], int(unigram_counts[i]) / sum(int_unigram_ct)])
```

[5]:
```python
return_list
```

[5]:
```
[['MILLION\n', 0.002072759168154815],
 ['MORE\n', 0.0017088989966186725],
 ['MR.\n', 0.0014416083492816956],
 ['MOST\n', 0.0007879173033190295],
 ['MARKET\n', 0.0007803712804681068],
 ['MAY\n', 0.00072989731562895322],
 ['M.\n', 0.0007034067394618568],
 ['MANY\n', 0.0006967290595970209],
 ['MADE\n', 0.0005598610827336895],
 ['MUCH\n', 0.0005145971758110562],
 ['MAKE\n', 0.0005144626437991272],
 ['MONTH\n', 0.00044490959363187093],
 ['MONEY\n', 0.00043710673693999306],
 ['MONTHS\n', 0.0004057607781605526],
 ['MY\n', 0.0004003183467688823],
 ['MONDAY\n', 0.00038198530259784006],
 ['MAJOR\n', 0.00037089252670515475],
 ['MILITARY\n', 0.00035204581485220204],
 ['MEMBERS\n', 0.00033606096579846475],
 ['MIGHT\n', 0.00027358919153183117],
 ['MEETING\n', 0.00026573741410834 27],
 ['MUST\n', 0.00026650791563120 84],
 ['ME\n', 0.00026357267173457725],
 ['MARCH\n', 0.0002597935452176646],
 ['MAN\n', 0.0002528834918776787],
 ['MS.\n', 0.0002389900041002911],
 ['MINISTER\n', 0.00023977273580605944],
 ['MAKING\n', 0.0002117044660445237 8],
 ['MOVE\n', 0.0002099555498894477],
 ['MILES\n', 0.00020596851026319035]]
```

```
[6]: #b)
for i in range(len(bigram_counts)):
    bigram_counts[i] = bigram_counts[i].split()
    for j in range(len(bigram_counts[i])):
        bigram_counts[i][j] = int(bigram_counts[i][j])
```

```
[7]: the_cond = []
for i in range(len(bigram_counts)):
    if bigram_counts[i][0] == 4:
        the_cond.append(bigram_counts[i])
```

```
[8]: probabilities = []
total_ct = 0
for i in range(len(the_cond)):
    total_ct += the_cond[i][2]
```

```
[9]: word_list = []
for i in range(len(the_cond)):
    probabilities.append(the_cond[i][2] / total_ct)
    word_list.append(words[the_cond[i][1] - 1])
```

```
[10]: word_list = pd.DataFrame(word_list)
word_list.columns = ['Words']
word_list['Probabilities'] = probabilities
word_list = word_list.sort_values(by='Probabilities', ascending=False).reset_index()
del word_list['index']
```

```
[11]: print(word_list[0:10])
```

```
          Words  Probabilities
0       <UNK>\n       0.615020
1          U.\n       0.013372
2       FIRST\n       0.011720
3     COMPANY\n       0.011659
4         NEW\n       0.009451
5      UNITED\n       0.008672
6  GOVERNMENT\n       0.006803
7    NINETEEN\n       0.006651
```

```
8        SAME\n       0.006287
9         TWO\n       0.006161
```

[12]:
```
#c)
sentence = ['THE','STOCK','MARKET','FELL','BY','ONE','HUNDRED','POINTS','LAST','WEEK']
prob = 1
for i in range(len(sentence)):
    idx = words.index(sentence[i]+'\n')
    prob *= int(unigram_counts[idx]) / sum(int_unigram_ct)
lu = math.log(prob)
```

[13]:
```
lu
```

[13]:
```
-64.50944034364878
```

[14]:
```
sentence_b = ['<s>',␣
↪'THE','STOCK','MARKET','FELL','BY','ONE','HUNDRED','POINTS','LAST','WEEK']
#pb = p(the|s)p(stock|the)p(market|stock)....p(week|last)
word_indices = []
for i in range(len(sentence_b)):
    word_indices.append(words.index(sentence_b[i]+'\n')+1)
```

[15]:
```
prob_b = 1
for i in range(1, len(sentence_b)):
    total_count = 0
    idx_w1 = word_indices[i-1]
    idx_w2 = word_indices[i]
    for j in range(len(bigram_counts)):
        if(bigram_counts[j][0] == idx_w1):
            total_count += bigram_counts[j][2]
            if bigram_counts[j][1] == idx_w2:
                prob_num = bigram_counts[j][2]
    prob_b *= prob_num/total_count
lb = math.log(prob_b)
```

[16]:
```
lb
```

[16]:
```
-40.91813213378977
```

[17]:
```
#d)
sentence_d_u = ['THE','SIXTEEN','OFFICIALS','SOLD','FIRE','INSURANCE']
prob = 1
prob_arr = []
for i in range(len(sentence_d_u)):
    idx = words.index(sentence_d_u[i]+'\n')
    prob *= int(unigram_counts[idx]) / sum(int_unigram_ct)
    prob_arr.append(int(unigram_counts[idx]) / sum(int_unigram_ct))
lu = math.log(prob)
```

[18]:
```
lu
```

[18]:
```
-44.291934473132606
```

[19]:
```
sentence_d_b = ['<s>','THE','SIXTEEN','OFFICIALS','SOLD','FIRE','INSURANCE']
word_indices = []
prob_b_arr = []
for i in range(len(sentence_d_b)):
    word_indices.append(words.index(sentence_d_b[i]+'\n')+1)
prob_b = 1
for i in range(1, len(sentence_d_b)):
    total_count = 0
    prob_num = float('-inf')
    idx_w1 = word_indices[i-1]
    idx_w2 = word_indices[i]
    for j in range(len(bigram_counts)):
        if(bigram_counts[j][0] == idx_w1):
            total_count += bigram_counts[j][2]
            if bigram_counts[j][1] == idx_w2:
                prob_num = bigram_counts[j][2]
    prob_b *= prob_num/total_count
    prob_b_arr.append(prob_num / total_count)
lb = math.log(prob_b)
```

[20]:
```
lb
```

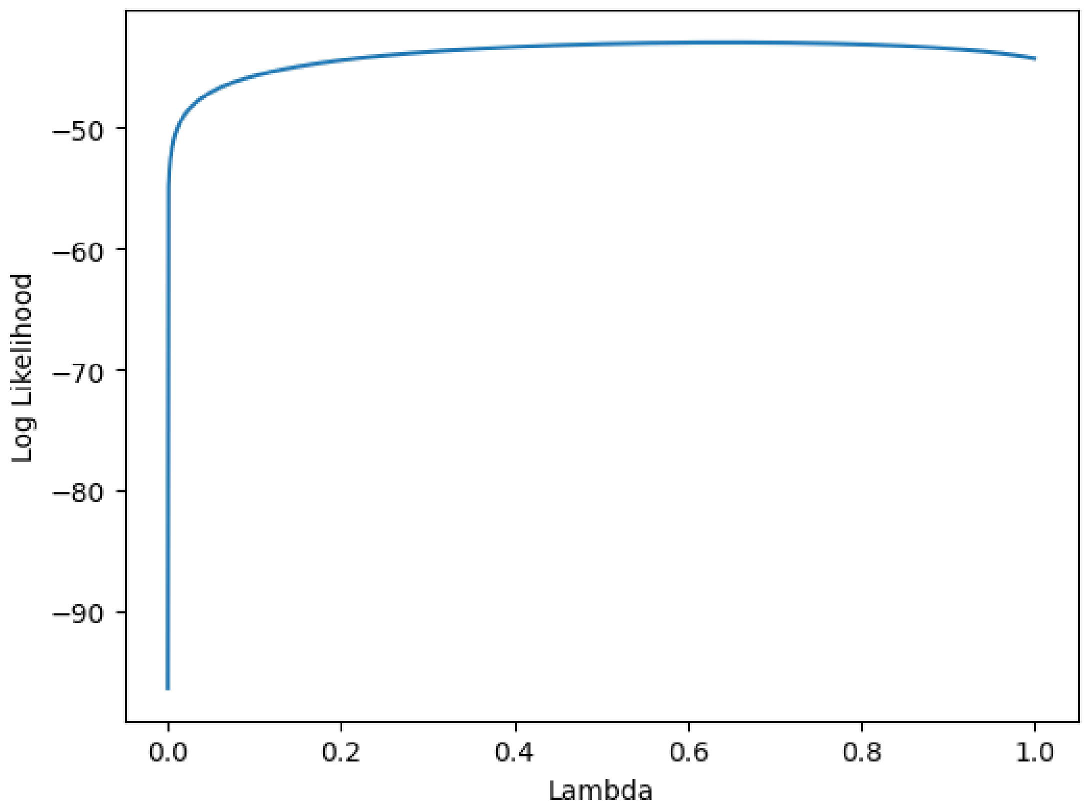
[20]:
```
inf
```

[21]:
```
#e)
def mix_model(lamb):
    pm = 1
    for i in range(len(prob_arr)):
        if prob_b_arr[i] != float('-inf'):
            temp_pm = (lamb*prob_arr[i])+((1-lamb)*prob_b_arr[i])
        else:
            temp_pm = lamb*prob_arr[i]
        pm *= temp_pm
    return math.log(pm)
```

[22]:
```
lambda_list = np.linspace(0.000000000001, 1, 1000)
log_list = []
for lamb in lambda_list:
    log_list.append(mix_model(lamb))
```

[23]:
```
plt.plot(lambda_list, log_list)
plt.xlabel('Lambda')
plt.ylabel('Log Likelihood')
```

[23]:
```
Text(0, 0.5, 'Log Likelihood')
```



```
[24]: log_list = np.array(log_list)
      log_list = np.argsort(log_list)
      best_lambda = lambda_list[log_list[-1]]
```

```
[25]: best_lambda
```

```
[25]: 0.647647647648
```

```
[26]: mix_model(best_lambda)
```

```
[26]: -42.964137594350596
```