4.1

(a) Let T denotes the # of trails, t denotes a specific trail:

$$L = \log(P(\text{data}))$$

$$= \log\left(\prod_{t=1}^{T} P(X_d^{(t)})\right) \quad (\text{IID assumption})$$

$$= \sum_{d=1}^{2D}\sum_{t=1}^{T} \log\left(P_{xd}^{(t)} \mid p_{ad}^{(t)}\right) = \sum_{d=1}^{2D}\sum_{t=1}^{T} \log\left(P_{xd}^{(t)}\right)$$

Let $C_d$ denotes the count of $P(X=d)$

$$\sum_{d=1}^{2D}\sum_{t=1}^{T} \log\left(P_{xd}^{(t)}\right) = \sum_{d=1}^{2D} \cdot C_d \log(P(X=d))$$

$$= \sum_{d=1}^{2D} C_d \log p_d$$

(b) $$L = C_d \log p_d - \lambda\left(\sum_{d=1}^{2D} p_d - 1\right)$$

$$\frac{\partial L}{\partial p_d} = C_d \cdot \frac{1}{p_d} - \lambda$$

When $\frac{\partial L}{\partial p_d} = 0$ (optimal), $p_d = \frac{C_d}{\lambda}$ ①

Given the constraint $\sum_{d}^{2D} p_d = 1$, $\sum_{d}^{2D} \frac{C_d}{\lambda} = 1$

$\lambda = \sum_{d}^{2D} C_d$ ② Give ①, ②, $p_d = \frac{C_d}{\sum_{d}^{2D} C_d}$

Given $C_d$ (as count) $\geq 0$, $p_d$ is non negative

(c) $\sum_{d=1}^{2D} (-1)^d p_d = 0$

$\Rightarrow -\sum_{d=1}^{D} p_{2d-1} + \sum_{d=1}^{D} p_{2d} = 0$

$\Rightarrow \sum_{d=1}^{D} p_{2d} = \sum_{d=1}^{D} p_{2d-1}$

$\Rightarrow P(X \in \{2, 4, 6 \cdots 2D\}) = P(X \in \{1, 3, 5, 2D-1\})$

(d) From part b, we know :

Given constraint $\sum_{d=1}^{2D} (-1)^d p_d = 0$, we got

$L = C_d \log p_d - \lambda_1 \left(\sum_{d=1}^{2D} p_d - 1\right) - \lambda_2 \left(\sum_{d=1}^{2D} (-1)^d p_d\right)$

$\frac{\partial L}{\partial p_d} = \frac{C_d}{p_d} - \lambda_1 - \lambda_2 (-1)^d$

$\frac{\partial L}{\partial p_d} = 0 \Rightarrow \boxed{p_d = \frac{C_d}{\lambda_1 + \lambda_2 (-1)^d}}$ (1)

$\sum_{d=1}^{2D} (-1)^d p_d = 0$, substituting in (1)

$\sum_{d=1}^{2D} (-1)^d \frac{C_d}{\lambda_1 + \lambda_2 (-1)^d} = 0$

$-\sum_{d=1}^{D} \frac{C_d}{\lambda_1 - \lambda_2} + \sum_{d=1}^{D} \frac{C_d}{\lambda_1 + \lambda_2} = 0 \Rightarrow \boxed{\frac{C_{even}}{\lambda_1 + \lambda_2} = \frac{C_{odd}}{\lambda_1 - \lambda_2}}$ (2)

Bring Back $\sum_{d=1}^{2D} p_d = 1$, $\frac{C_{even}}{\lambda_1 + \lambda_2} + \frac{C_{odd}}{\lambda_1 - \lambda_2} = 1$

$\begin{cases} \lambda_1 - \lambda_2 = 2C_{odd} \\ \lambda_1 + \lambda_2 = 2C_{even} \end{cases} \Rightarrow \lambda_1 = 2C_{odd} + 2C_{even}$

$\lambda_2 = 2C_{even} - 2C_{odd}$

$$P_d = \frac{C_d}{2C_{odd} + 2C_{even} + (-1)^d (2C_{even} - 2C_{odd})}$$

$$P_d(d \in \{1, 3, 5 \cdots 2D-1\}) = \frac{C_d}{4C_{odd}}$$

$$P_d(d \in \{2, 4, 6, \cdots 2D\}) = \frac{C_d}{4C_{even}}$$

4.2

(a) $P_{ML}(X_1 = x) = \frac{Count_1(x)}{T}$ (root case)

$$P_{ML}(X_i = x \mid Pa = \pi) = P_{ML}(X_i = x \mid X_{i-1} = \pi)$$

$$= \frac{P_{ML}(x_{i-1} \mid x_i) \cdot P_{ML}(x_i)}{P_{ML}(x_{i-1})}$$

$$= \frac{\frac{Count_{i-1}(x, x')}{Count_i(x)} \cdot \frac{Count_i(x)}{T}}{\frac{Count_{i-1}(x)}{T}}$$

$$P_{ML}(X_i \mid X_{i-1}) = \frac{Count_{i-1}(x, x')}{Count_{i-1}(x)} \quad i \in \{2, 3 \cdots n\}$$

$$= \frac{Count_i(x, x')}{Count_i(x)} \quad i \in \{1, 2, 3 \cdots n\} \text{ (node with parent case)}$$

(b) $P_{ML}(X_n = x) = \frac{Count_n(x)}{T}$ (root case)

$P_{ML}(X_i = x \mid X_{i+1} = \pi) = \frac{Count_i(x, x')}{Count_{i+1}(x)}$ $i \in \{1, 2 \cdots n-1\}$

(node w parent case)

(c) For $G_1$

$$P(X_1, X_2 \cdots X_n) = \prod_{i=1}^{n} P(X_i^T \mid X_{i-1}^T, X_{i-2}^T, X_1^T)$$

$$= \prod_{i=1}^{n-1} P(X_{i+1} \mid X_i) P(X_1)$$

$$= \prod_{i=1}^{n-1} \frac{Count_i(x, x')}{Count_i(x)} \cdot \frac{Count_1 x}{T}$$

For $G_2$

$$P(X_1, X_2 \cdots X_n) = \prod_{i=1}^{n-1} P(X_i^T \mid X_{i+1}^T, X_{i+2}^T \cdots X_n^T)$$

$$= \prod_{i=1}^{n-1} P(X_i \mid X_{i+1}) P(X_n)$$

$$= \prod_{i=1}^{n-1} \frac{Count_i(x, x')}{Count_{i+1}(x)} \cdot \frac{Count_n(x)}{T}$$

$$= \prod_{i=1}^{n-1} \frac{Count_i(x, x')}{Count_{i+1}(x)} \cdot \frac{Count_n(x)}{T} \cdot \frac{Count_1(x)}{Count_1(x)}$$

$$= \prod_{i=1}^{n-1} \frac{Count_i(x, x')}{Count_i(x)} \cdot \frac{Count_1 x}{T}$$

(d) No, below is reasoning:

With $G_1$, $G_2$, pick a random node $X_a$

$X_{a-2} \rightarrow X_{a-1} \rightarrow (X_a) \rightarrow X_{a+1} \rightarrow X_{a+2}$

$X_{a-2} \leftarrow X_{a-1} \leftarrow (X_a) \leftarrow X_{a+1} \leftarrow X_{a+2}$

$X_a$ in evidence set will make $X_{a-i}$ and $X_{a+i}$ conditional independent

And such condition allows:

$$\prod_{i=1}^{n} P(X_i^T \mid X_{i-1}^T, X_{i-2}^T, X_1^T)$$

$$= \prod_{i=1}^{n-1} P(X_{i+1} \mid X_i) P(X_1)$$

$G_1$

or

$$\prod_{i=1}^{n-1} P(X_i^T \mid X_{i+1}^T, X_{i+2}^T \dots X_n^T)$$

$$\prod_{i=1}^{n-1} P(X_i \mid X_{i+1}) P(X_n)$$

$G_2$

But $G_3$

$X_{a-2} \leftarrow X_{a-1} \rightarrow (X_a) \leftarrow X_{a+1} \rightarrow X_{a+2}$

$X_a$ in evidence set will make $X_{a-i}$ and $X_{a+i}$ conditional dependent

$$P(X_{a-2} \mid X_{a-1}, X_a, X_{a+1}, X_{a+2}) \neq P(X_{a-2} \mid X_{a-1})$$

So $G_3$ not close to the same distribution as $G_1$/$G_2$

# 4.3. Data handling

```
import collections

count_uni = []
vocab_uni = []
vocab_index_count_lookup = {}
vocab_count_bi = collections.defaultdict(dict) # key: wi, value: {wi+1:count wi }]

f = open('hw4files/hw4_unigram.txt','r')
for line in f:
    count_uni.append(int(line.strip()))

f.close()


f = open('hw4files/hw4_vocab.txt','r')

for line in f:
    vocab_uni.append(line.strip())
f.close()


for i, item in enumerate(zip(vocab_uni, count_uni)):
    key = item[0]
    value = item[1]
    vocab_index_count_lookup[key] = (i, value)


f = open('hw4files/hw4_bigram.txt','r')
for line in f:
    lst = line.strip().split('\t')
    w_i_ind = int(lst[0])
    w_j_ind = int(lst[1])
    w_j_follow_w_i_count = int(lst[2])


    w_i = vocab_uni[w_i_ind-1]
    w_j = vocab_uni[w_j_ind-1]
    vocab_count_bi[w_i][w_j] = w_j_follow_w_i_count
f.close()
```

```
#4.3.a
from tabulate import tabulate
T = sum(count_uni)
rows =  [[vocab_index_count_lookup[key][0]+1, key, vocab_index_count_lookup[key][1]/T ] \
         for key in vocab_index_count_lookup if key[0].upper() == 'M']
rows.insert(0, ['Index in bigram', 'Word', 'P_u_W'])
print(tabulate(rows))
```

| Index in bigram | Word | P_u_W |
|---|---|---|
| 54 | MILLION | 0.002072759168154815 |
| 69 | MORE | 0.0017088989966186725 |
| 77 | MR. | 0.0014416083492816956 |
| 121 | MOST | 0.0007879173033190295 |
| 122 | MARKET | 0.0007803712804681068 |
| 126 | MAY | 0.0007298973156289532 |
| 130 | M. | 0.0007034067394618568 |
| 131 | MANY | 0.0006967290595970209 |
| 159 | MADE | 0.0005598610827336895 |
| 178 | MUCH | 0.0005145971758110562 |
| 180 | MAKE | 0.0005144626437991272 |
| 203 | MONTH | 0.00044490959363187093 |
| 209 | MONEY | 0.00043710673693999306 |
| 227 | MONTHS | 0.0004057607781605526 |
| 230 | MY | 0.0004003183467688823 |
| 247 | MONDAY | 0.00038198530259784006 |
| 256 | MAJOR | 0.00037089252670515475 |
| 275 | MILITARY | 0.00035204581485220204 |
| 287 | MEMBERS | 0.00033606096579846475 |
| 356 | MIGHT | 0.00027358919153183117 |
| 366 | MEETING | 0.0002657374141083427 |
| 370 | MUST | 0.0002665079156312084 |
| 374 | ME | 0.00026357267173457725 |
| 375 | MARCH | 0.0002597935452176646 |
| 385 | MAN | 0.0002528834918776787 |
| 403 | MS. | 0.0002389900041002911 |
| 404 | MINISTER | 0.00023977273580605944 |
| 460 | MAKING | 0.00021170446604452378 |
| 473 | MOVE | 0.0002099555498894477 |
| 479 | MILES | 0.00020596851026319035 |

```
#4.3.b

countW_given_w = [(key,  vocab_count_bi['THE'][key]) for key in vocab_count_bi['THE']]
sum_count = sum([item[1] for item in countW_given_w])
rows = sorted([[vocab_index_count_lookup[w][0]+1, w, 'THE', count, sum_count, count/sum_count] \
               for w, count in countW_given_w], reverse=True, key= lambda x: x[-1])[:10]
rows.insert(0, ['w Index in bigram', "w'", 'w', "count_w'|w", "count_w", "P_w'|w"])
print(tabulate(rows))
```

| w Index in bigram | w' | w | count_w'\|w | count_w | P_w'\|w |
|---|---|---|---|---|---|
| 1 | <UNK> | THE | 2371132 | 3855375 | 0.6150198100055118 |
| 70 | U. | THE | 51556 | 3855375 | 0.013372499432610317 |
| 79 | FIRST | THE | 45186 | 3855375 | 0.011720260675031612 |
| 73 | COMPANY | THE | 44949 | 3855375 | 0.011658788055636611 |
| 61 | NEW | THE | 36439 | 3855375 | 0.009451480076516552 |
| 184 | UNITED | THE | 33435 | 3855375 | 0.008672308141231398 |
| 103 | GOVERNMENT | THE | 26230 | 3855375 | 0.006803488635995202 |
| 39 | NINETEEN | THE | 25641 | 3855375 | 0.006650714911000876 |
| 308 | SAME | THE | 24239 | 3855375 | 0.006287066757449016 |
| 23 | TWO | THE | 23752 | 3855375 | 0.006160749602827221 |

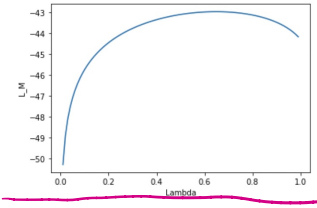
```
#4.3 c
count_w = {}
for parent in vocab_count_bi:
    count_w[parent] = sum([vocab_count_bi[parent][child] for child in vocab_count_bi[parent]])

import math
sentence = "The stock market fell by one hundred points last week"
sentence = sentence.upper()
s_lst = sentence.split(' ')

pu = 1
for key in s_lst:
    pu *= vocab_index_count_lookup[key][1]/T
Lu = math.log(pu)

s_lst.insert(0, '<s>')
pb = 1
for i in range(1, len(s_lst)):
    prev = s_lst[i-1]
    curr = s_lst[i]
    pb *= (vocab_count_bi[prev][curr]/count_w[prev])
Lb = math.log(pb)

print(Lu)
print(Lb)
```

```
-64.50944034364878
-40.91813213378977
```

```
#4.3 d
sentence = "The sixteen officials sold fire insurance"
sentence = sentence.upper()
s_lst = sentence.split(' ')

pu = 1
for key in s_lst:
    pu *= vocab_index_count_lookup[key][1]/T
Lu = math.log(pu)

s_lst.insert(0, '<s>')
pb = 1
for i in range(1, len(s_lst)):
    prev = s_lst[i-1]
    curr = s_lst[i]
    try:
        pb *= (vocab_count_bi[prev][curr]/count_w[prev])
    except:
        print(f'The pair of adjacent words not observed in the data is {prev} followed by {curr}')
        pb = 0
try:
    Lb = math.log(pb)
except:
    Lb = 'UNDEFINED'

print(Lu)
print(Lb)
```

```
The pair of adjacent words not observed in the data is SIXTEEN followed by OFFICIALS
The pair of adjacent words not observed in the data is SOLD followed by FIRE
-44.291934473132606
UNDEFINED
```

In [103]:
```
#4.3 e
sentence = "The sixteen officials sold fire insurance"
sentence = sentence.upper()
s_lst = sentence.split(' ')
pu = []
for key in s_lst:
    pu.append(vocab_index_count_lookup[key][1]/T)

s_lst.insert(0, '<s>')
pb = []
for i in range(1, len(s_lst)):
    prev = s_lst[i-1]
    curr = s_lst[i]
    try:
        pb.append(vocab_count_bi[prev][curr]/count_w[prev])
    except:
        pb.append(0)
l = 0
l_step_val = 0.01
pm = []
lm = []
l_lst = []
while l <= 1:
    l_lst.append(l)
    m = math.prod([(l*u) + ((1-l)*b) for u, b in zip(pu, pb)])
    pm.append(m)
    lm.append(math.log(m)) if m!=0 else lm.append(None)
    l += l_step_val

import matplotlib.pyplot as plt
plt.plot(l_lst, lm)
plt.xlabel('Lambda')
plt.ylabel('L_M')
plt.show()
print(f'The lambda value maximize P_M is {l_lst[pm.index(max(pm))]}')
```




```
The lambda value maximize P_M is 0.6500000000000004
```

```
#4.4a
import numpy as np
data_00 = []
data_01 = []

f = open('hw4files/nasdaq00.txt','r')
for line in f:
    data_00.append(float(line.strip()))
f.close()


f = open('hw4files/nasdaq01.txt','r')
for line in f:
    data_01.append(float(line.strip()))
f.close()

x = np.array([[data_00[i], data_00[i+1],data_00[i+2]] for i in range(0, len(data_00)-3)])
y = np.array(data_00[3:])

from numpy.linalg import inv
X_dot_X_T = np.dot(x.T,x)
X_T_dot_Y = np.dot(x.T,y)
alphas = np.dot(inv(X_dot_X_T),X_T_dot_Y )
print(alphas)
```

```
[0.03189569 0.01560133 0.95067337]
```

```
#4.4b
x_test = np.array([[data_01[i], data_01[i+1],data_01[i+2]] for i in range(0, len(data_01)-3)])
y_test = np.array(data_01[3:])


rmse_00 = np.sqrt(np.square(np.subtract(y,np.dot(x,alphas))).mean())
print(rmse_00)

rmse_01 = np.sqrt(np.square(np.subtract(y_test,np.dot(x_test,alphas))).mean())
print(rmse_01)
```

```
117.9084436177829
54.63604967520658
```