HW1

0.4.1)

a) $X \in \{1, 2, 3, \cdots, 20\}$

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

data is all sample data,

$= \log P(s_1, s_2, \cdots s_n)$

For n samples,
s_1: sample 1.

$= \log P(s_1) P(s_2) \cdots P(s_n)$

$\because$ I I D.

$= \log \prod_{i=1}^{n} P(s_i)$

$= \sum_{i=1}^{n} \log P(s_i)$

$= \sum_{i=1}^{20} \log C_{s_i} p_{s_i}$

where s_i can be any one of the outcomes.

$= \sum_{t=1}^{20} \log C_t p_t$

$= \sum_{t=1}^{20} C_t \log p_t$

$\because$ Add^n of prob of samples is equal to no of time the outcome occured in the sample set //

Q 4.1)

b) Lagrang multiplier :-

$$L(x,y,\lambda) = f(x,y) - \lambda g(x,y)$$

$$= \sum_{d=1}^{20} C_d \log p_d - \lambda\left(\left(\sum_{d=1}^{20} p_d\right) - 1\right)$$

$$\frac{\partial L(x,y,\lambda)}{\partial p_d} = \sum_{d=1}^{20} \frac{C_d}{p_d} - \lambda(1)$$

To get max :-

$$\sum_{d=1}^{20} \frac{C_d}{p_d} - \lambda = 0$$

$$\therefore p_d = \frac{C_d}{\lambda} \quad //$$

$\therefore$ Non neg

Now, $\sum_{d=1}^{20} p_d = 1$ //

$$\sum_{d=1}^{20} \frac{C_d}{\lambda} = 1$$

$$\therefore \lambda = \sum_{d=1}^{20} C_d$$

$$\therefore p_d = \frac{C_d}{\sum_{d=1}^{20} C_d} \quad //$$

1) The roll of a dice is even for,

$$P(2)+P(4)\ P(6) \cdots P(20)$$

i.e. $P_{even} = \sum_{d=1}^{D} P_{2d}$ //

Similarly,

$$P_{odd} = \sum_{d=1}^{D} P_{(2d-1)}$$ //

Now,

For,

$$P_{even} = P_{odd}$$

$$P_{even} - P_{odd} = 0$$

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

$$= P_2 - P_1 + P_4 - P_3 + \cdots P_{20} - P_{20-1}$$

$$= \sum_{d=1}^{2D} (-1)^d P_d$$

$$= \sum_{d=1}^{2D} (-1)^d P_d$$

HP //

Q.4.1)

d) Now Lagrange for multiple constraints:-

$$\mathcal{L}(x,y,\lambda_1,\lambda_2) = f(x,y) - \lambda_1 g(x,y) - \lambda_2 h(x,y)$$

$$= \sum_{d=1}^{20} C_d \log p_d - \lambda_1 \left(\left(\sum_{d=1}^{20} p_d \right) - 1 \right) - \lambda_2 \sum_{d=1}^{20} (-1)^d p_d$$

$$\therefore \frac{\partial \mathcal{L}}{\partial p_d} =$$

$$\therefore \frac{\partial \mathcal{L}}{\partial p_d} = \sum_{d=1}^{20} \frac{C_d}{p_d} - \lambda_1 - \lambda_2 \sum_{d=1}^{20} (-1)^d$$

To max :-

$$\therefore \sum_{d=1}^{20} \frac{C_d}{p_d} - \lambda_1 - \lambda_2 \sum_{d=1}^{20} (-1)^d = 0$$

For even :- $\sum_{d=1}^{20} \frac{C_d}{p_d} - \lambda_1 - \lambda_2 = 0$

$$\therefore p_{d_{even}} = \frac{C_{d_{even}}}{\lambda_1 + \lambda_2}$$

For odd :- Similarly,

$$p_{d_{odd}} = \frac{C_{d_{odd}}}{\lambda_1 - \lambda_2}$$

Now, $\lambda_1 + \lambda_2 = \frac{C_{even}}{\rho_{even}}$

$$= \frac{\sum_{d=1}^{\infty} C_{2d}}{\sum_{d=1}^{\infty} \rho^{2d}}$$

$$= \frac{\sum_{d=1}^{\infty} C_{2d}}{0.5}$$

$$= 2 \sum_{d=1}^{\infty} C_{2d} = 2 C_{even}$$

Similarly for odd,

$$\lambda_1 - \lambda_2 = 2 \sum_{d=1}^{\infty} C_{2d-1}$$

$$= 2 C_{odd}$$

$\therefore$ ~~$\rho^{2d} = \frac{C_d}{2 C_{even}}$~~

$\therefore \rho_d = \frac{C_d}{2 C_{even}}$ For $d \in$ even

& $\rho_d = \frac{C_d}{2 C_{odd}}$ For $d \in$ odd

=

Q.2)

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

$$= \log \prod_{t=1}^{T} P\left(x_1^{(t)}, x_2^{(t)}, \dots x_n^{(t)}\right) \quad \because \text{IID}$$

$$= \log \prod_{t=1}^{T} \prod_{i=1}^{n} P\left(x_i^{(t)} \mid pa_i^{(t)}\right) \quad \because \text{Bnet Rule}$$

$$= \sum_{i=1}^{n} \sum_{t=1}^{T} \log P\left(x_i^{(t)} \mid pa_i^{(t)}\right)$$

$$= \sum_{i=1}^{n} \sum_{x} \sum_{\Pi} count(x_i = x, pa_i = \Pi) \log P(x_i = x \mid pa_i = \Pi)$$

Now, for nodes with parents,

$$P_{ML}(x_i = x \mid pa_i = \Pi) = \frac{count(x_i = x, pa_i = \Pi)}{count(pa_i = \Pi)}$$

i.e. for this G1 :-

$$P_{ML}(X_{n+1} = x' \mid X_n = x) = \frac{count(X_{n+1} = x', X_n = x)}{count(X_n = x)}$$

$$= count_i(x, x') / count_i(x) \quad //$$

& for X_1 :-

$$P_{ML}(X_1) = \frac{count_i(x)}{T} \quad //$$

b)

For ⊛ nodes with parents :-

$$P_{ML}(X_i = x \mid X_{i+1} = x') = \frac{count_i(x, x')}{count_i(x')}$$

For leaf :-

$$P_{ML}(X_n = x') = \frac{count_i(x')}{T}$$

//

c) Now joint distribution for G1 :-

$$P_{G_1} = P(X_1, X_2 \cdots X_n)$$

$$= P(X_1)\, P(X_2 \mid X_1)\, P(X_3 \mid X_2, X_1) \cdots P(X_n \mid X_{n-1}, X_{n-2} \cdots)$$

∵ of d(i) ≥ 2 ∵ Bayes rule

$$= P(X_1)\, P(X_2 \mid X_1)\, P(X_3 \mid X_2) \cdots P(X_n \mid X_{n-1})$$

Now from (a) :-

$$= \frac{count(X_1 = x_1)}{T} \times \frac{count(X_2 = x_2, X_1 = x_1)}{count(X_1 = x_1)} \times \frac{count(X_3 = x_3, X_2 = x_2)}{count(X_2 = x_2)} \cdots$$

$$= \prod_{i=2}^{n} \frac{count(X_i = x_i, X_{i-1} = x_{i-1})}{T \times \prod_{i=2}^{n} count(X_i = x_i)} \quad \text{— (1)}$$

//ly of G 2,

Initial :-

$$PG2 = P(X_n) \; P(X_{n-1} | X_n) \cdots P(X_1 | X_2)$$

from (1) :-

$$= \frac{count(X_n = x_n)}{T} \times \frac{count_2(X_{n-1}, X_n)}{count(X_n = x_n)} \times \frac{count_2(X_{n-2}, X_{n-1})}{count(X_{n-1})} \times \cdots \times \frac{count(X_2 = x_1, X_1 = x_2)}{count(X_2 = x_2)}$$

$$= \frac{\prod_{i=2}^{n} count(X_{i-1} = x_i, X_i = x_i)}{T \times \prod_{i=2}^{n} count(X_i = x_i)}$$

$$= PG1$$

$$\therefore PG1 = PG2$$

HP

//

d) In G3 there are 2 parents i.e.,

$$P_{ML}(X_{n-2} \mid X_{n-1}, X_{n-3}) = \frac{\text{count}_i(X_{n-2}, X_{n-1}, X_{n-3})}{\text{count}_i(X_{n-1}, X_{n-3})}$$

$$= \frac{\text{count}(X_{n-2}=x_{n-2}, X_{n-1}=x_{n-1}, X_{n-3}=x_{n-3})}{\text{count}(X_{n-1}=x_{n-1}, X_{n-3}=x_{n-3})}$$

Using this, PG3 ≠ PG1 ≠ PG2 :

∴ PG3 won't give the same joint distribution.

Q4.3)

```
import numpy as np
from collections import defaultdict
import math
from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

Import Dataset

Importing vocab

```
vocab ={}
with open("hw4_vocab.txt") as file:
  f = file.readlines()
  # print(f)
  for i,l in enumerate(f):
    vocab[i+1]=l
    # break
for key in vocab:
  vocab[key] = vocab[key][:-1]

# vocab[1]
```

Importing Unigram

```
unigram = defaultdict(int)
with open("hw4_unigram.txt") as file:
  f = file.readlines()
  # print(f)
  for i,l in enumerate(f):
    unigram[vocab[i+1]]=int(l[:-1])
    # break


# unigram["THE"]
```

Importing Bigrams

```
bigram = defaultdict(list)
```

```
with open("hw4_bigram.txt") as file:
  f = file.readlines()
  # print(f)
  for i,l in enumerate(f):
    #PREPROCESS
    l = l.split("\t")
    l[-1] = l[-1][:-1]
    l = list(map(int,l))
    # print(l)
    # print(vocab[l[0]])
    bigram[vocab[l[0]]].append((vocab[l[1]],l[2]))
    # print(bigram)
    # unigram[vocab[i]]=int(l[:-1])
    # break


# bigram["THE"][:10]
```

Q4.3 A)

```
def prob(key):
  return unigram[key]/sum(unigram.values())
print("Words\t\tprobability")
print("_"*50)
for key in unigram:
  if key[0] == "M":
    print(key,"\t\t",prob(key))

```

```
Words           probability
__________________________________________________
MILLION                 0.002072759168154815
MORE            0.0017088989966186725
MR.             0.0014416083492816956
MOST            0.000787917303319029 5
MARKET          0.0007803712804681068
MAY             0.0007298973156289532
M.              0.0007034067394618568
MANY            0.0006967290595970209
MADE            0.0005598610827336895
MUCH            0.0005145971758110562
MAKE            0.0005144626437991272
MONTH           0.00044490959363187093
MONEY           0.00043710673693999306
MONTHS          0.0004057607781605526
MY              0.0004003183467688823
MONDAY          0.00038198530259784006
MAJOR           0.0003708925267051547 5
MILITARY                0.00035204581485220204
MEMBERS                 0.00033606096579846475
MIGHT           0.00027358919153183117
MEETING                 0.0002657374141083427
MUST            0.0002665079156312084
ME              0.0002635726717345772 5
```

```
MARCH           0.0002597935452176646
MAN             0.0002528834918776787
MS.             0.0002389900041002911
MINISTER                0.00023977273580605944
MAKING          0.00021170446604452378
MOVE            0.0002099555498894477
MILES           0.00020596851026319035
```

Q4.3 B)

```
def bigramProb(word):
  theFollower = defaultdict(int)

  # FOR THE
  sumi = 0
  followers = bigram[word]
  for i in followers:
    sumi += i[1]

  for i in followers:
    theFollower[i[0]] = i[1]/sumi

  theFollowers = (sorted(theFollower.items(), key=lambda item: item[1],reverse=True))
  return theFollowers

print('For word "The": ')
print("Words\t\tprobability")
print("_"*50)
for i in bigramProb("THE")[:10]:
  print(i[0],"\t\t",i[1])
```

```
For word "The":
Words           probability
__________________________________________________
<UNK>           0.6150198100055118
U.              0.013372499432610317
FIRST           0.011720260675031612
COMPANY                 0.011658788055636611
NEW             0.009451480076516552
UNITED          0.008672308141231398
GOVERNMENT              0.006803488635995202
NINETEEN                0.006650714911000876
SAME            0.006287066757449016
TWO             0.006160749602827221
```

Q 4.3) C)

Unigram Lu

```
ip = "THE STOCK MARKET FELL BY ONE HUNDRED POINTS LAST WEEK"
ip = ip.split()
```

```
# ip[0]
```

```
sumi = 0
for i in range(len(ip)):
  sumi += math.log(unigram[ip[i]]/sum(unigram.values()))
print("Lu = ",sumi)
```

```
Lu =  -64.50944034364878
```

Bigram Lb

```
ip = "<s> THE STOCK MARKET FELL BY ONE HUNDRED POINTS LAST WEEK"
ip = ip.split()
```

```
sumi = 0
for i in range(0,len(ip)-1):
  word1,word2 = ip[i],ip[i+1]
  # print(word1,word2,end=" ")
  probs = dict(bigramProb(word1))
  # print("\t",probs[word2])
  sumi += math.log(probs[word2])
print("Lb = ",sumi)
```

```
Lb =  -40.91813213378977
```

Bigram yields the higest log likelihood

4.3) D)

Unigram Lu

```
ip = "THE SIXTEEN OFFICIALS SOLD FIRE INSURANCE"
ip = ip.split()
```

```
# ip[0]
```

```
sumi = 0
for i in range(len(ip)):
  sumi += math.log(unigram[ip[i]]/sum(unigram.values()))
print("Lu = ",sumi)
```

```
Lu =  -44.291934473132606
```

Bigram Lb

```
ip = "<s> THE SIXTEEN OFFICIALS SOLD FIRE INSURANCE"
ip = ip.split()
```

```
sumi = 0
for i in range(0,len(ip)-1):
  word1,word2 = ip[i],ip[i+1]
  # print(word1,word2,end=" ")
  probs = dict(bigramProb(word1))
  # print("\t",probs[word2])
  if word2 not in probs:
    sumi = float("-inf")
    print(word2,"|",word1, " not present")
    continue
  sumi += math.log(probs[word2])
print("Lb = ",sumi)
```

```
OFFICIALS | SIXTEEN  not present
FIRE | SOLD  not present
Lb =  -inf
```

Using this model, the Bigram becomes -infinite because the word pairings are not present, hence log(0) = inf

Q4.3 E)

```
# my_default_dict = defaultdict(int,d)
```

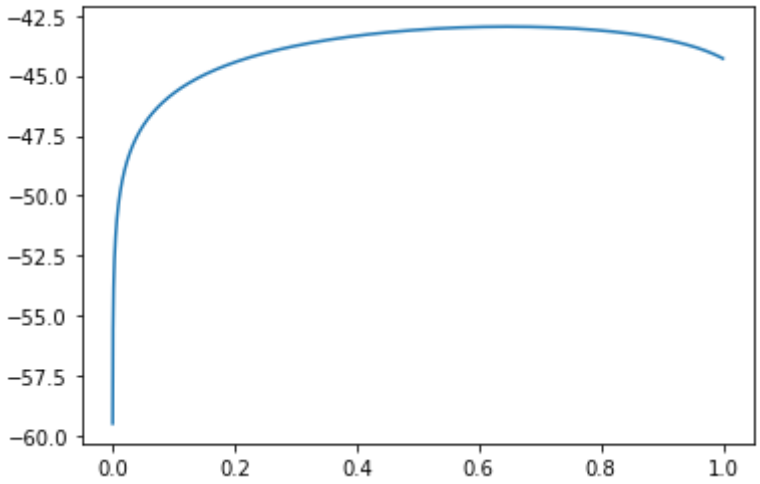
```
ip = "<s> THE SIXTEEN OFFICIALS SOLD FIRE INSURANCE"
ip = ip.split()
```

```
best_Lm = []
for lambdaa in tqdm(range(1,10001)):
  sumi = 0

  for i in range(0,len(ip)-1):
    word1,word2 = ip[i],ip[i+1]
    # print(word1,word2,end=" ")
    probs = dict(bigramProb(word1))
    # print("\t",probs[word2])
    # if word2 not in probs:
    #   sumi = float("-inf")
    #   print(word2,"|",word1, " not present")
```

```
    #   continue
    probs = defaultdict(int,probs)
    sumi += math.log((1-(lambdaa/10000))*(probs[word2])+(lambdaa/10000)*(unigram[word2]
  best_Lm.append(sumi)

print("best lambda = ", (best_Lm.index(max(best_Lm)))/10000)
print("best Lm = ", (max(best_Lm)))
```

```
100%                                                  10000/10000 [00:12<00:00, 814.13it/s]
best lambda =  0.6478
best Lm =  -42.964137160084704
```

```
plt.plot(np.linspace(0/10000,10000/10000,10000),best_Lm)
plt.show()
```



Q4.4

Q4.4 A)

```
s2000 = []
with open("/content/nasdaq00.txt") as file:
  f = file.readlines()
  # print(f)
  for i,l in enumerate(f):
    s2000.append(l)
    # break
# print(s2000[0])
for key in range(len(s2000)):
  s2000[key] = float(s2000[key][:-1])

# print(s2000[0])
# print(s2000[0])
```

```
s2001 = []
with open("/content/nasdaq01.txt") as file:
```

```
  f = file.readlines()
  # print(f)
  for i,l in enumerate(f):
    s2001.append(l)
    # break
# print(s2001[0])
for key in range(len(s2001)):
  s2001[key] = float(s2001[key][:-1])

# print(s2001[0])
# print(s2001[0])
```

```
s2000 = np.array(s2000)
y_00 = s2000[3:]
# print(y_00)
y_00 = y_00.reshape(-1,1)
# print(y_00.shape)
```

```
X_00 = []
for i in range(len(s2000)-3):
  tmp = []
  tmp.extend([s2000[i],s2000[i+1],s2000[i+2]])
  X_00.append(tmp)
X_00 = np.array(X_00)
# print((X_00.shape))
```

```
# y = Xb
b_00 = []
theta,residuals,rank,s = np.linalg.lstsq(X_00, y_00)
# print(theta)
print("Linear Coefficients:")
print("a1:",theta[2][0])
print("a2:",theta[1][0])
print("a3:",theta[0][0])
```

```
Linear Coefficients:
a1: 0.9506722769536844
a2: 0.015603326703986786
a3: 0.031894723175170614
```

Q4.4 B)

RMSE 00

```
y_00_pred = np.dot(X_00, theta)

MSE_00 = ((y_00 - y_00_pred)**2).mean()
print("MSE:",MSE_00)
print("RMSE 2000:",math.sqrt(MSE_00))
```

```
MSE: 13902.375020416126
RMSE 2000: 117.9083331254247
```

RMSE 01

```
X_01 = []
for i in range(len(s2001)-3):
  tmp = []
  tmp.extend([s2001[i],s2001[i+1],s2001[i+2]])
  X_01.append(tmp)

s2001 = np.array(s2001)
y_01 = s2001[3:]
# print(y_00)
y_01 = y_01.reshape(-1,1)
# print(y_01.shape)


X_01 = np.array(X_01)
# print((X_01.shape))
y_01_pred = np.dot(X_01, theta)

MSE_01 = ((y_01 - y_01_pred)**2).mean()
print("MSE:",MSE_01)
print("RMSE 2001:",math.sqrt(MSE_01))
```

```
MSE: 2985.0983142892514
RMSE 2001: 54.63605324590395
```

```
1
```

Colab paid products - Cancel contracts here

✓ 0s completed at 9:00 PM