CSE 250 A

NAME: Guanghao chen

PID: A53276390

Email: guc001@eng.ucsd.edu

4.1 Maximum likelihood estimation of a multinomial distribution

(a) Log-likelihood

$$\begin{aligned} P(data) &= \prod^{T} P(X = x^{(t)}) \\ &= \prod^{2n} P(X = n)^{C_n} \\ &= \prod^{2n} p_n^{C_n} \end{aligned}$$

Therefore, the log-likelihood can be denoted by inserting the P(data).

$$\begin{aligned} L &= log(P(data)) \\ &= log(\prod^{2n} p_n^{C_n}) \\ &= \sum^{2n} log(p_n^{C_n}) \\ &= \sum^{2n} C_n log(p_n) \end{aligned}$$

(b) Maximum likelihood estimate

As the hint says, we can use the Lagrange multiplier to calculate the maximum.

So the target function can be denoted by

$$L(p_i, \lambda) = \sum^{2n} C_n log(p_n) + \lambda(1 - \sum^{2n} p_n)$$

Then, calculate the partial derivative of the target function with respect to p_i and λ

$$\begin{cases} \dfrac{\partial L}{\partial p_i} = \dfrac{c_i}{p_i} - \lambda = 0 \\ \dfrac{\partial L}{\partial \lambda} = (1 - \sum^{2n} p_n) = 0 \end{cases}$$

Therefore we can first denote p_i with λ, namely

$$p_i = \frac{c_i}{\lambda}$$

Then, we substitute the fomula into the second equation and produce

$$\lambda = \sum^{2n} c_i$$

Therefore,

$$p_i = \frac{c_i}{\sum^{2n} c_i}$$

(c)

Since the tosses are identically, independent distributed, the probability for even tosses and odd ones can be denoted by

$P(even) = P(x = 2\ or\ x = 4\ or \ldots or\ 2n) = P(2) + P(4) + \ldots + P(2n) = p_2 + p_4 + \ldots p_{2n}$

$P(odd) = P(x = 1\ or\ x = 3\ or \ldots or\ 2n-1) = P(1) + P(3) + \ldots + P(2n-1) = p_1 + p_3 + \ldots p_{2n-1}$

The given equation also can be denoted by

$$\begin{aligned}\sum_{n=1}^{2N} (-1)^n p_n &= (p_2 + p_4 + \ldots + p_{2n}) - (p_1 + p_3 + \ldots + p_{2n-1}) \\ &= P(even) - P(odd) = 0\end{aligned}$$

Therefore, it shows that $P(even) = P(odd)$.

(d)

According to Lagrange multiplier, the target function can be denoted as follow.

$$L = \sum_{i=1}^{2N} c_i log(p_i) + \lambda_1 (1 - \sum_{i=1}^{2N} p_i) + \lambda_2 (-1)^i p_i$$

Then, by calculating the partial derivitive of the target function with respect to p_i, λ_1 and λ_2, it has

$$\begin{cases} \dfrac{\partial L}{\partial p_i} = \dfrac{c_i}{p_i} - \lambda_1 + (-1)^i \lambda_2 & = 0 \\ \dfrac{\partial L}{\partial \lambda_1} = 1 - \sum_{i=1}^{2N} p_i & = 0 \\ \dfrac{\partial L}{\partial \lambda_2} = \sum_{i=1}^{2N} (-1)^i p_i & = 0 \end{cases}$$

First we multiply $p_i$to the first equation to produce

$$
\begin{aligned}
c_i - \lambda_1 p_i + (-1)^i \lambda_2 p_i &= 0 \\
\sum_{i=1}^{2N} c_i - \lambda_1 \sum_{i=1}^{2N} p_i + \lambda_2 \sum_{i=1}^{2N} (-1)^i p_i &= 0 \\
\sum_{i=1}^{2N} c_i - \lambda_1 &= 0 \\
\lambda_1 &= \sum_{i=1}^{2N} c_i
\end{aligned}
$$

Basing on the similar process mentioned above, we can multiply $(-1)^i p_i$ to the first equation again.

$$
\begin{aligned}
(-1)^i c_i - (-1)^i \lambda_1 p_i + \lambda_2 p_i &= 0 \\
\sum_{i=1}^{2N} (-1)^i c_i - \lambda_1 \sum_{i=1}^{2N} (-1)^i p_i + \lambda_2 \sum_{i=1}^{2N} p_i &= 0 \\
\sum_{i=1}^{2N} (-1)^i c_i + \lambda_2 &= 0 \\
\lambda_2 &= -\sum_{i=1}^{2N} (-1)^i c_i
\end{aligned}
$$

Then we substitute λ_1 and λ_2 into the first equation again to deduce p_i.

$$
\begin{aligned}
\frac{c_i}{p_i} - \lambda_1 + (-1)^i \lambda_2 &= 0 \\
\frac{c_i}{p_i} - \sum_{i=1}^{2N} c_i - (-1)^i \sum_{i=1}^{2N} (-1)^i c_i &= 0 \\
p_i &= \frac{c_i}{\sum_{i=1}^{2N} c_i + (-1)^i \sum_{i=1}^{2N} (-1)^i c_i}
\end{aligned}
$$

4.2 Maximum likelihood estimation in belief networks

(a)

$P(x_i = x^{'} | x_{i-1} = x) = \frac{count_{i-1}(x, x^{'})}{count_{t-1}(x)}$

(b)

$P(x_{i-1} = x | x_i = x^{'}) = \frac{count_{i-1}(x, x^{'})}{count_{t-1}(x^{'})}$

(c)

The joint distribution for G1:

$$
\begin{aligned}
P(x_1, x_2, \ldots, x_n) &= P(X_n|X_{n-1})P(X_{n-1}|X_{n-2})\ldots P(X_2|X_1)P(X_1) \\
&= \frac{count_{n-1}(x_{n-1}, x_n)}{count_{n-1}(x_{n-1})} \cdot \frac{count_{n-2}(x_{n-2}, x_{n-1})}{count_{n-2}(x_{n-2})} \cdots \frac{count_1(x_1, x_2)}{count_1(x_1)} \cdot \frac{count_1(x_1)}{|data|} \\
&= \frac{count_{n-1}(x_{n-1}, x_n)}{count_{n-1}(x_{n-1})} \cdot \frac{count_{n-2}(x_{n-2}, x_{n-1})}{count_{n-2}(x_{n-2})} \cdots \frac{count_1(x_1, x_2)}{|data|}
\end{aligned}
$$

The joint distribution for G2:

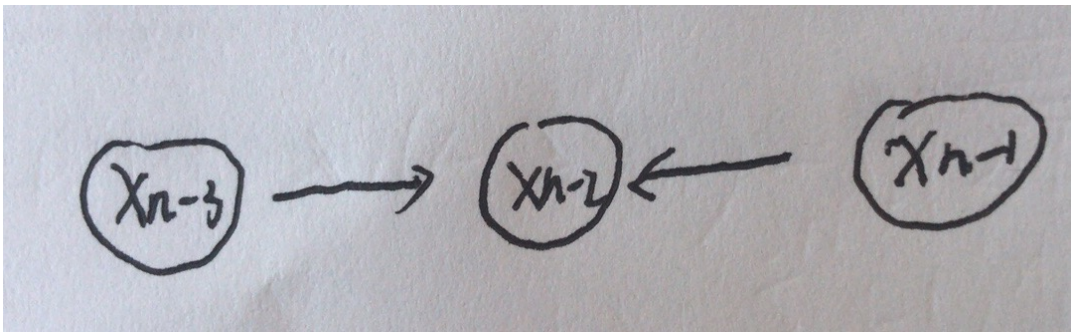
$$
\begin{aligned}
P(x_1, x_2, \ldots, x_n) &= P(X_1|X_2)P(X_2|X_3)\ldots P(X_{n-1}|X_n)P(x_n) \\
&= \frac{count_1(x_1, x_2)}{count_2(x_2)} \cdot \frac{count_2(x_2, x_3)}{count_3(x_3)} \cdots \frac{count_{n-1}(x_{n-1}, x_n)}{count_n(x_n)} \cdot \frac{count_n(x_n)}{|data|} \\
&= \frac{count_1(x_1, x_2)}{count_2(x_2)} \cdot \frac{count_2(x_2, x_3)}{count_3(x_3)} \cdots \frac{count_{n-1}(x_{n-1}, x_n)}{|data|}
\end{aligned}
$$

Therefore, the two joint distribution is equal.

(d)

Take node x_{n-2} for instance.

The subgraph arond node x_{n-2} shows as the following .



Accroding to the graph, we can conclude that the node $x_{n-3}$and x_{n-1} are not independent given node x_{n-2} because the arrow direction is not diversed.

Therefore, the joint distribution is not as the same as the previous two graphs G1 and G2.

4.3 Statistical language modeling

(a)

```
Word Starting With A,Prob
A,0.018407244690712494
AND,0.017863233925020615
AT,0.004312974000612439
AS,0.003991797167406474
```

```
AN,0.002999256673943544
ARE,0.0029896926709136874
ABOUT,0.0019256178376532746
AFTER,0.0013465675979453587
ALSO,0.0013100115812493978
ALL,0.001181814804064031
A.,0.0010256109080316418
ANY,0.0006318601694814718
AMERICAN,0.0006120961939108219
AGAINST,0.000595964582662253
ANOTHER,0.0004283866165304179
AMONG,0.0003742925175520858 5
AGO,0.0003565709825261751
ACCORDING,0.0003475451075440342
AIR,0.0003110013210309760 4
ADMINISTRATION,0.0002915186396670866
AGENCY,0.0002796553622515356
AROUND,0.0002768546503668335 5
AGREEMENT,0.0002627899400288089 5
AVERAGE,0.0002590719644264094 3
ASKED,0.0002582280818061279 5
ALREADY,0.0002490799049949608
AREA,0.0002310893059451922
ANALYSTS,0.0002260382404064060 4
ANNOUNCED,0.0002271511870505453 6
ADDED,0.0002212195483427698 6
ALTHOUGH,0.0002142605742711734 5
AGREED,0.0002117778471419395 7
APRIL,0.0002066900910544455 2
AWAY,0.0002020548517343487 8
```

(b)

```
The <UNK>       0.6150198100055118
The U.          0.013372499432610317
The FIRST       0.011720260675031612
The COMPANY     0.011658788055636611
The NEW         0.009451480076516552
```

(c)

Log-likelihoods of this sentence under unigram model: -64.50944034364878.

Log-likelihoods of this sentence under bigram model: -44.740469213403735.

Therefore, bigram model yields to the highest log-likelihood.

(d)

Log-likelihoods of this sentence under unigram model: -41.64345971649364.

The pairs $P(officials|nineteen)$ and $P(fire|sold)$ are not observed in the training corpus. The effect is that their probability will be 0 and further the log-likelihood will cause a domain error.

(e)

The optimal value is -39.953751575277174, where $\lambda = 0.4083469387755102$.

The plot is in the attachment.

4.4 Markov modeling

(a) Unigram model

	a	b	c	d
	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

(b) Bigram model

	a	b	c	d
a	$\frac{2}{3}$	0	0	$\frac{1}{3}$
b	0	$\frac{3}{4}$	$\frac{1}{4}$	0
c	0	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{1}{4}$
d	$\frac{1}{4}$	0	$\frac{1}{4}$	$\frac{2}{4}$

(c) Likelihoods

$P_U(S) = P_U(T_1)$

$P_U(S) = P_U(T_2)$

$P_U(S) = P_U(T_3)$

$P_B(S) > P_B(T_1)$

$P_B(S) > P_B(T_2)$

$P_B(T_2) = P_B(T_3)$

$P_B(S) > P_U(S)$

$P_B(T_1) = P_U(T_1)$

$P_B(T_2) < P_U(T_2)$

$P_B(T_3) < P_U(T_3)$

(d)

The plots for each sequence is D,A,C,B.

4.5 Stock Market Prediction

(a)

$$a1 = 0.9506733661404788$$
$$a2 = 0.01560133077286545$$
$$a3 = 0.03189568516025264$$

(b)

2000's Mean Square Error is 13902.401076367885 2001's Mean Square Error is 2985.0979241108607

Therefore, I won't recommend this linear model for stock prediction because of the high mean squared error.

(c)

See the attachment.

hw4_4.3

October 29, 2018

In [2]:
```
import numpy as np
from functools import reduce
import math
from matplotlib import pyplot as plt
```

1 Preprocessing

In [3]:
```
#vocab
vocab = None
with open("./hw4_vocab.txt",'r')as f:
    vocab = f.readlines()
for i in range(len(vocab)):
    vocab[i] = vocab[i][:-1]
#unigram
unigram = None
with open("./hw4_unigram.txt","r") as f:
    unigram = f.readlines()
for i in range(len(unigram)):
    unigram[i] = int(unigram[i][:-1])

#bigram
bigram = []
with open("./hw4_bigram.txt","r") as f:
    temp = f.readlines()
    for i in temp:
        bigram.append(i.split("\t"))
for i in range(len(bigram)):
    bigram[i][0] = int(bigram[i][0])
    bigram[i][1] = int(bigram[i][1])
    bigram[i][2] = int(bigram[i][2])
```

2 Problem (a)

In [5]:
```
unig = np.zeros((len(unigram),))
for i in range(len(unigram)):
    unig[i] = int(unigram[i])
```

```
        unig = unig/np.sum(unig)
        print("{},{}".format("Word Starting With A","Prob"))
        for i in range(len(vocab)):
            if vocab[i][0]=='A':
                print("{},{}".format(vocab[i][::],unig[i]))
```

```
Word Starting With A,Prob
A,0.018407244690712494
AND,0.017863233925020615
AT,0.004312974000612439
AS,0.003991797167406474
AN,0.002999256673943544
ARE,0.0029896926709136874
ABOUT,0.0019256178376532746
AFTER,0.0013465675979453587
ALSO,0.0013100115812493978
ALL,0.001181814804064031
A.,0.0010256109080316418
ANY,0.0006318601694814718
AMERICAN,0.0006120961939108219
AGAINST,0.000595964582662253
ANOTHER,0.0004283866165304179
AMONG,0.00037429251755208585
AGO,0.0003565709825261751
ACCORDING,0.0003475451075440342
AIR,0.00031100132103097604
ADMINISTRATION,0.0002915186396670866
AGENCY,0.0002796553622515356
AROUND,0.00027685465036683335
AGREEMENT,0.00026278994002880895
AVERAGE,0.0002590719644264 0943
ASKED,0.00025822808180612795
ALREADY,0.0002490799049949608
AREA,0.0002310893059451922
ANALYSTS,0.00022603824040640604
ANNOUNCED,0.00022715118705054536
ADDED,0.00022121954834276986
ALTHOUGH,0.00021426057427117345
AGREED,0.00021177784714193957
APRIL,0.00020669009105444552
AWAY,0.00020205485173434878
```

3 Problem (b)

```
In [6]: prob = []
        index_the = 0
```

```
        #find "The" index
        for i in range(len(vocab)):
            if(vocab[i]=="THE"):
                index_the = i
                break
        #The Bigram
        bigram_the = []
        for i in range(len(bigram)):
            if(bigram[i][0]==(index_the+1)):
                bigram_the.append(bigram[i])

        #calculate the ML prob
        prob_the_w = np.zeros((len(vocab),))
        for i in range(len(bigram_the)):
            w_index = bigram_the[i][1] -1
            count_the_w = bigram_the[i][2]
            prob_the_w[w_index] = count_the_w/unigram[index_the]
        #search 5 max
        max_prob = []
        for i in range(prob_the_w.shape[0]):
            if(len(max_prob)<5):
                max_prob.append((i,prob_the_w[i]))
                max_prob = sorted(max_prob,key = lambda x: x[1])
            else:
                if(prob_the_w[i]>max_prob[0][1]):
                    max_prob[0] = (i,prob_the_w[i])
                    max_prob = sorted(max_prob,key = lambda x: x[1])
        #print_out
        for i in max_prob[::-1]:
            print("The {}\t\t{}".format(vocab[i[0]],i[1]))
```

```
The <UNK>               0.6150198100055118
The U.                0.013372499432610317
The FIRST               0.011720260675031612
The COMPANY               0.011658788055636611
The NEW                0.009451480076516552
```

4 Problem (c)

```
In [7]: index_last = vocab.index("LAST")
        index_week = vocab.index("WEEK")
        index_stock = vocab.index("STOCK")
        index_market = vocab.index("MARKET")
        index_fell = vocab.index("FELL")
        index_by = vocab.index("BY")
        index_one = vocab.index("ONE")
```

```
        index_hundred = vocab.index("HUNDRED")
        index_points = vocab.index("POINTS")
        index_s = vocab.index("<s>")
        product_unig = reduce((lambda x, y: x * y), [unig[index_last],unig[index_week],unig[inc
                                                     unig[index_market],unig[index_fell],unig[index_
                                                    unig[index_hundred],unig[index_points]])
        log_likelihoods_uni = math.log(product_unig)
        print("Log-likelihoods of this sentence under unigram model: {}".format(log_likelihoods

        bi_mat = np.zeros((len(vocab),len(vocab)))
        for i in range(len(bigram)):
            x = bigram[i][0] - 1
            y = bigram[i][1] - 1
            bi_mat[x,y] = bigram[i][2]

        P_last_s = bi_mat[index_s,index_last]/unigram[index_s]
        P_week_last = bi_mat[index_last,index_week]/unigram[index_last]
        P_the_week = bi_mat[index_week,index_the]/unigram[index_week]
        P_stock_the = bi_mat[index_the,index_stock]/unigram[index_the]
        P_market_stock = bi_mat[index_stock,index_market]/unigram[index_stock]
        P_fell_market = bi_mat[index_market,index_fell]/unigram[index_market]
        P_by_fell = bi_mat[index_fell,index_by]/unigram[index_fell]
        P_one_by = bi_mat[index_by,index_one]/unigram[index_by]
        P_hundred_one = bi_mat[index_one,index_hundred]/unigram[index_one]
        P_points_hundred = bi_mat[index_hundred,index_points]/unigram[index_hundred]
        product_bigr = reduce((lambda x, y: x * y), [P_last_s,P_week_last,P_the_week,P_stock_th
                                                     P_fell_market,P_by_fell,P_one_by,P_hundred_
        log_likelihoods_bigr = math.log(product_bigr)
        print("Log-likelihoods of this sentence under bigram model: {}".format(log_likelihoods_
```

```
Log-likelihoods of this sentence under unigram model: -64.50944034364878
Log-likelihoods of this sentence under bigram model: -44.740469213403735
```

5 Problem (d)

```
In [8]: "The nineteen officials sold fire insurance"
        index_nineteen = vocab.index("NINETEEN")
        index_officials = vocab.index("OFFICIALS")
        index_sold = vocab.index("SOLD")
        index_fire = vocab.index("FIRE")
        index_insurance = vocab.index("INSURANCE")
        product_d_unig = reduce((lambda x,y:x*y),[unig[index_the],unig[index_nineteen],unig[inc
        print("Log-likelihoods of this sentence under unigram model: {}".format(math.log(produc

        P_the_s = bi_mat[index_s,index_the]/unigram[index_s]
        P_nineteen_the = bi_mat[index_the,index_nineteen]/unigram[index_the]
```

```
P_officials_nineteen = bi_mat[index_nineteen,index_officials]/unigram[index_nineteen]
P_sold_officials = bi_mat[index_officials,index_sold]/unigram[index_officials]
P_fire_sold = bi_mat[index_sold,index_fire]/unigram[index_sold]
P_insurance_fire = bi_mat[index_fire,index_insurance]/unigram[index_fire]
product_d_bigr = reduce((lambda x, y: x * y), [P_the_s,P_nineteen_the,P_officials_ninet
                                               P_fire_sold,P_insurance_fire])
print([P_the_s,P_nineteen_the,P_officials_nineteen,P_sold_officials,
                                               P_fire_sold,P_insurance_fire])
print("Log-likelihoods of this sentence under bigram model: {}".format(math.log(product
```

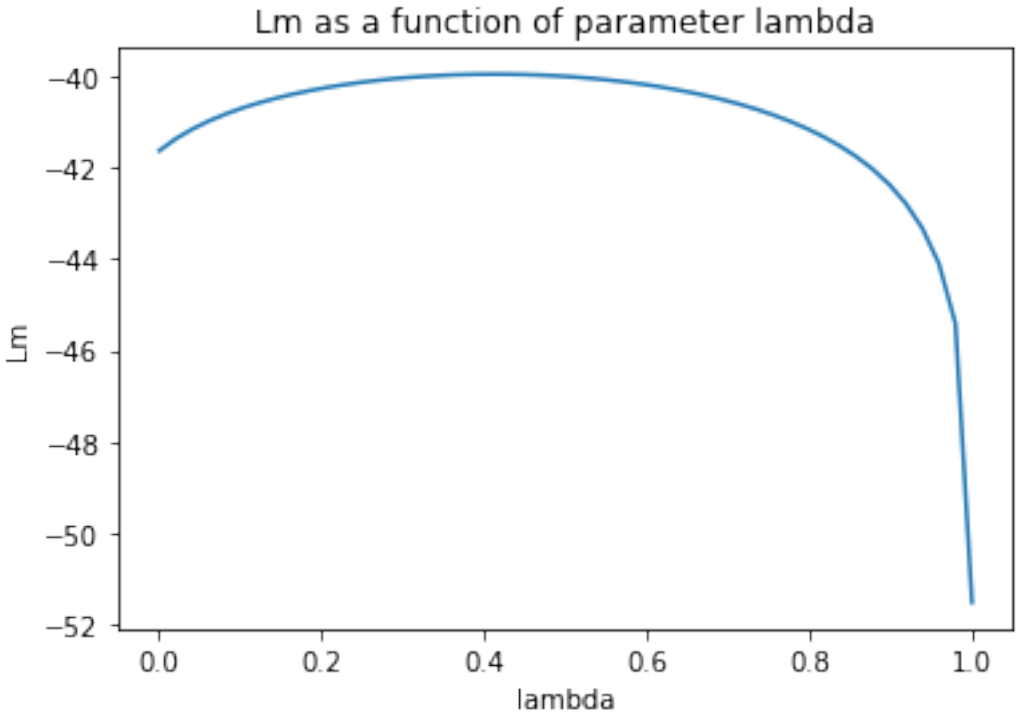
```
Log-likelihoods of this sentence under unigram model: -41.64345971649364
[0.15865263383617936, 0.006650714911000876, 0.0, 9.162207725573554e-05, 0.0, 0.003052399525182
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-8-f5b85366f4b8> in <module>()
 18 print([P_the_s,P_nineteen_the,P_officials_nineteen,P_sold_officials,
 19                                                P_fire_sold,P_insurance_fire])
---> 20 print("Log-likelihoods of this sentence under bigram model: {}".format(math.log(pr


ValueError: math domain error
```

6 Problem (e)

```
In [9]: alphas = np.linspace(0.001,0.999)
        res = []
        max_tuple = (1,-float('Inf'))
        for alpha in alphas:
            Pm_the_s = unig[index_the]*(1-alpha)+P_the_s*alpha
            Pm_nineteen_the = unig[index_nineteen]*(1-alpha)+P_nineteen_the*alpha
            Pm_officials_nineteen = unig[index_officials]*(1-alpha)+P_officials_nineteen*alpha
            Pm_sold_officials = unig[index_sold]*(1-alpha)+P_sold_officials*alpha
            Pm_fire_sold = unig[index_fire]*(1-alpha)+P_fire_sold*alpha
            Pm_insurance_fire = unig[index_insurance]*(1-alpha)+P_insurance_fire*alpha
            prod_alpha = reduce((lambda x,y:x*y),[Pm_the_s,Pm_nineteen_the,Pm_officials_ninete
                                                  Pm_fire_sold,Pm_insurance_fire])
            res.append(math.log(prod_alpha))
            if(math.log(prod_alpha)>max_tuple[1]):
                max_tuple = (alpha,math.log(prod_alpha))

        plt.title("Lm as a function of parameter lambda")
        plt.plot(alphas,res)
```

```
plt.xlabel("lambda")
plt.ylabel("Lm")
plt.show()
print("Max Lm:{}, where alpha={}".format(max_tuple[1],max_tuple[0]))
```




```
Max Lm:-39.953751575277174, where alpha=0.4083469387755102
```

In [ ]:

hw4_4.5

October 29, 2018

```
In [46]: import numpy as np
         from functools import reduce
```

1 Problem(a)

```
In [57]: with open("hw4_nasdaq00.txt","r") as f:
             price = f.readlines()
         price = list(map(lambda x:float(x.strip()),price))
         row_a1 = [np.array([price[i-1],price[i-2],price[i-3]]) * price[i-1] for i in range(3,
         row_a1 = np.sum(row_a1,axis=0)
         row_a2 = [np.array([price[i-1],price[i-2],price[i-3]]) * price[i-2] for i in range(3,
         row_a2 = np.sum(row_a2,axis=0)
         row_a3 = [np.array([price[i-1],price[i-2],price[i-3]]) * price[i-3] for i in range(3,
         row_a3 = np.sum(row_a3,axis=0)
         A = np.vstack((row_a1,row_a2))
         A = np.vstack((A,row_a3))

         b1 = reduce(lambda x,y:x+y,[price[i-1]*price[i] for i in range(3,len(price))])
         b2 = reduce(lambda x,y:x+y,[price[i-2]*price[i] for i in range(3,len(price))])
         b3 = reduce(lambda x,y:x+y,[price[i-3]*price[i] for i in range(3,len(price))])
         b = np.zeros((3,))
         b[0] = b1
         b[1] = b2
         b[2] = b3
         a_vec = np.linalg.solve(A,b)
         print("The coefficient: \na1 = {}\na2 = {}\na3 = {}".format(a_vec[0],a_vec[1],a_vec[2]
```

```
The coefficient:
a1 = 0.9506733661404788
a2 = 0.01560133077286545
a3 = 0.03189568516025264
```

2 Problem(b)

```
In [65]: with open("hw4_nasdaq01.txt","r") as f:
             price_2001=f.readlines()
```

```
        price_2001 = list(map(lambda x:float(x.strip()),price_2001))
        predict_2000 = [np.sum(np.array(price[i-3:i][::-1] * a_vec)) for i in range(3, len(pri
        predict_2001 = [np.sum(np.array(price_2001[i-3:i][::-1] * a_vec)) for i in range(3, le
        print ("2000\'s Mean Square Error is {}".format(np.mean(np.square(np.array(predict_200
        print ("2001\'s Mean Square Error is {}".format(np.mean(np.square(np.array(predict_200
```

```
2000's Mean Square Error is 13902.401076367885
2001's Mean Square Error is 2985.0979241108607
```