STATS_HW2_Q3

March 8, 2019

```
In [1]: import pandas as pd
        data=pd.read_csv("/Users/zhejindong/Downlp1*(1-p1)/n1 + p2*(1-p2)/n2oads/CentralPark.cs
```

0.1 Q3 Task discription

The data set Central Park.csv consists of precipitation data from weather station at Central Park, New York. The data was collected from the National Oceanic and Atmospheric Administration (NOAA). The variable PRCP shows the observed amount of rain at time t in mm. Consider a first order MarkovChain model with a two dimensional state space corresponding to the states{0,1}={“rainy day”,“no rain”}, where we define a rain day as one with a PRCP of at least 1.5 mm. Suppose the estimated transition probability matrices obtained using data collected above.

```
<tr>
  <th>rainy</th>
  <th>nonrainy</th>
</tr>

<tr>
  <td>a<sub>1</sub></td>
  <td>a<sub>2</sub></td>
</tr>
<tr>
  <td>a<sub>3</sub></td>
  <td>a<sub>4</sub></td>
</tr>
```

0.1.1 1. Interpret the meaning of ai

For the first order Markov Chain mode, the weather of today only depends on yesterday. Thus:

a1= P(a0=0 | a0=0) which means given yesterday is rainy, the probality of today being rainy is a1.

a2= P(a1=0 | a0=0) which means given yesterday is rainy, the probality of today being nonrainy is a2.

a3= P(a0=0 | a1=0) which means given yesterday is nonrainy, the probality of today being rainy is a3.

a4= P(a1=0 | a1=0) which means given yesterday is nonrainy, the probality of today being rainy is a3.

0.1.2 2. What's the long-term probability of observing a rainy day in Central Park. (Use ai to express the result)

To solve this problem, we need to get the stationary distribution:

pTP=pT p0: the probability of raining in long term p1: the probability of not raining in long term

a1p0+ a3p1=p0

p0+p1=1

we get p0=a3/(a3-a1+1)

Thus the long-term probability of observing a rainy day is a3/(a3-a1+1)

0.1.3 3. Can you estimate ai for the month of July using the historical Central Park data?

```
In [137]: June_slice=data['DATE'].str.split("/").str[0].astype(int)==6
          June_data=data[June_slice]
          June_data['Status']=June_data['PRCP'].apply(lambda x: 0 if x >1.5 else 1)
          #June_data
          c_00=0
          c_01=0
          c_10=0
          c_11=0
          for i in range(len(June_data)-1):
              date=June_data.iloc[i]['DATE'].split("/")[1]
              if date!='30':
                  if June_data.iloc[i]['Status']==0:
                      if June_data.iloc[i+1]['Status']==0:
                          c_00=1+c_00
                      else:
                          c_01=1+c_01
                  else:
                      if June_data.iloc[i+1]['Status']==0:
                          c_10=1+c_10
                      else:
                          c_11=c_11+1
          # estimate ai

          a0=float(c_00/(c_00+c_01))
          a1=float(c_01/(c_00+c_01))
          a2=float(c_10/(c_10+c_11))
          a3=float(c_11/(c_10+c_11))


          print("a0 is {0:.2f}".format(a0))
          print("a1 is {0:.2f}".format(a1))
          print("a2 is {0:.2f}".format(a2))
          print("a3 is {0:.2f}".format(a3))
```

```
/anaconda2/envs/py36/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until


a0 is 0.33
a1 is 0.67
a2 is 0.23
a3 is 0.77
```

0.1.4 4. Are the probability laws of “Xt+1|Xt=1” and “1Xt+1|Xt= 0” significantlydifferent in Central Park?

X1=Xt+1|Xt=1~Bern(p1)
X2=1Xt+1|Xt=0~Bern(p2)
According to CLT:
$\bar{X}1$ ~ N(p1,p1*(1-p1)/n1)
$\bar{X}2$ ~ N(p2,p2*(1-p2)/n2)
$\bar{X}1$-$\bar{X}2$ ~ N(p1-p2, p1*(1-p1)/n1 + p2*(1-p2)/n2)
H0: p1-p2=0 H1: p1!=p2
If H0 holds, $\bar{X}1$-$\bar{X}2$ ~ N(0, p1*(1-p1)/n1 + p2*(1-p2)/n2)
p1’=float(c_11/(c_11+c_10))=0.77 p2’=float(c_00/(c_00+c_01))=0.67
$\bar{X}1$-$\bar{X}2$ ~ N(0, p1’*(1-p1’)/n1 + p2’*(1-p2’)/n2)
p_value is 2*P(($\bar{X}1$-$\bar{X}2$) > (0.77-0.67))=2.4696743317065284e-08 <0.05. Reject H0.
Thus, the probability laws of “Xt+1|Xt=1” and “1Xt+1|Xt= 0” are significantly different in Central Park

In [147]:
```python
p1=0.77
p2=0.67
n1=c_10+c_11
n2=c_00+c_01
p1*(1-p1)/n1 + p2*(1-p2)/n2
from scipy.stats import norm
import math
mu, sigma = 0, p1*(1-p1)/n1 + p2*(1-p2)/n2 # mean and standard deviation
(1-norm.cdf(0.77-0.67, loc=mu, scale=math.sqrt(sigma)))*2
```

Out[147]:
```
2.4696743317065284e-08
```

0.1.5 5. Does a higher order chain improve the fit of the data?

In [148]:
```python
## second order markov chain
c_000=0
c_010=0
c_001=0
c_011=0
c_100=0
```

```
    c_110=0
    c_101=0
    c_111=0
    for i in range(len(June_data)-2):
        date=June_data.iloc[i]['DATE'].split("/")[1]
        if date!='29':
            if June_data.iloc[i]['Status']==0:
                if June_data.iloc[i+1]['Status']==0 and June_data.iloc[i+2]['Status']==0
                    c_000=1+c_000
                if June_data.iloc[i+1]['Status']==0 and June_data.iloc[i+2]['Status']==1
                    c_001=1+c_001
                if June_data.iloc[i+1]['Status']==1 and June_data.iloc[i+2]['Status']==0
                    c_010=1+c_010
                if June_data.iloc[i+1]['Status']==1 and June_data.iloc[i+2]['Status']==1
                    c_011=1+c_011
            else:
                if June_data.iloc[i+1]['Status']==0 and June_data.iloc[i+2]['Status']==0
                    c_100=1+c_100
                if June_data.iloc[i+1]['Status']==0 and June_data.iloc[i+2]['Status']==1
                    c_101=1+c_101
                if June_data.iloc[i+1]['Status']==1 and June_data.iloc[i+2]['Status']==0
                    c_110=1+c_110
                if June_data.iloc[i+1]['Status']==1 and June_data.iloc[i+2]['Status']==1
                    c_111=1+c_111
```

In [168]:
```
# second order
df={"0":{"c_00":c_000,"c_01":c_010,"c_10":c_100,"c_11":c_110},"1":{"c_00":c_001,"c_0
df2=pd.DataFrame(df)
df2['total']=df2['0']+df2['1']
df2
```

Out[168]:

	0	1	total
c_00	94	190	284
c_01	112	470	582
c_10	192	398	590
c_11	471	1523	1994

In [283]:
```
# first order
df1={"0":{"c_0":c_00,"c_1":c_10},"1":{"c_0":c_01,"c_1":c_11}}
df1=pd.DataFrame(df1)
df1['total']=df1['0']+df1['1']
df2['total']=df2['0']+df2['1']
```

In [284]:
```
df1
```

Out[284]:

	0	1	total
c_0	286	588	874
c_1	584	1993	2577

In [285]: df2

Out[285]:

	0	1	total
c_00	94	190	284
c_01	112	470	582
c_10	192	398	590
c_11	471	1523	1994

0.1.6 test statistics

H0: If second order chain can improve the peformance the likelihood test should converge to chi2 distribution

In [271]:
```
# test statistics:
p_df1=df1.iloc[:,:-1]/df1['total'].values.reshape(-1,+1)
p_df2=df2.iloc[:,:-1]/df2['total'].values.reshape(-1,+1)
```

In [286]: p_df1

Out[286]:

	0	1
c_0	0.327231	0.672769
c_1	0.226620	0.773380

In [287]: p_df2

Out[287]:

	0	1
c_00	0.330986	0.669014
c_01	0.192440	0.807560
c_10	0.325424	0.674576
c_11	0.236209	0.763791

In [296]:
```
import numpy as np
from scipy.stats import chi2
part1=(((np.log(p_df2)*df2.iloc[:,:-1]).values).sum())
part2=(((np.log(p_df1.iloc[0,:])*df2.iloc[[0,2],:-1]).values).sum())+(((np.log(p_df1
test_statistic=2*part1/part2
p_value=chi2.cdf(test_statistic, df=4)
```

In [297]: test_statistic

Out[297]: 1.997361373259808

In [291]: p_value

Out[291]: 0.25640213947908735

p_value is bigger than 0.05 so we cannot regect the H0. Thus, we conclude higer order cannot help leverage model performance.