MLDS_HW4

May 6, 2019

0.1 Problem 1 (Markov chains)

You will rank 767 college football teams based on the scores of every game in the 2018 season. The data provided in CFB2018 scores.csv contains the result of one game on each line inthe format

Team A index, Team A points, Team B index, Team B points

If Team A has more points than Team B, then Team A wins, and vice versa. The index of a team refers to the row of "TeamNames.txt" where that team's name can be found. Construct a 767Œ767 random walk matrix M on the college football teams. First construct the unnor-malized matrix M' with values initialized to zeros. For one particular game, let i be the index of Team A and j the index of Team B. Then update M'

After processing all games, let M be the matrix formed by normalizing the rows of M' so they sum toone. Letwtbe the 1Œ767 state vector at stept. Set w0 to the uniform distribution. Therefore,wt is the marginal distribution on each state after t steps given that the starting state is chosen uniformly atrandom.

```
In [329]: import pandas as pd
          import numpy as np
          data=pd.read_csv("/Users/zhejindong/Desktop/hw4_data/CFB2018_scores.csv",header=None)

In [295]: data.shape

Out[295]: (4208, 4)

In [296]: data=np.array(data)

In [297]: import numpy as np
          size=data.max()

In [298]: # Initialize M'
          M=np.zeros((size+1,size+1))

In [299]: # update M'
          for d in data:
              i=d[0]
              j=d[2]
              if d[1]>d[3]:
                  # i wins
                  M[i][i]=M[i][i]+1+d[1]*1.0/(d[1]+d[3])
                  M[j][j]=M[j][j]+d[3]*1.0/(d[1]+d[3])
```

```
                M[j][i]=M[j][i]+1+d[1]*1.0/(d[1]+d[3])
                M[i][j]=M[i][j]+d[3]*1.0/(d[1]+d[3])
            else:
                #j wins
                M[i][i]=M[i][i]+d[1]*1.0/(d[1]+d[3])
                M[j][j]=M[j][j]+1+d[3]*1.0/(d[1]+d[3])
                M[i][j]=M[i][j]+1+d[3]*1.0/(d[1]+d[3])
                M[j][i]=M[j][i]+d[1]*1.0/(d[1]+d[3])
```

In [300]:
```
M[1:,1:]
```

Out[300]:
```
array([[11.85876045,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  8.33501705,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  9.23978481, ...,  0.        ,
         0.        ,  0.        ],
       ...,
       [ 0.        ,  0.        ,  0.        , ..., 11.96262772,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
        10.21090202,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  9.1557783 ]])
```

In [301]:
```
# normalize the matrix:
M_1=(M[1:]/np.sum(M[1:],axis=1).reshape(-1,+1))
M=np.vstack((M[0],M_1))
```

In [302]:
```
w=np.ones((1,size))/size
```

0.1.1 a)

Use wt to rank the teams by sorting in decreasing value according to this vector. List the top 25 team names (see accompanying file) and their corresponding values inwtfort= 10,100,1000,10000.

In [303]:
```
name=pd.read_csv("/Users/zhejindong/Desktop/hw4_data/TeamNames.txt",header=None)
```

In [304]:
```
def rank(t):
    w=np.ones((1,size))/size
    x=w.dot(np.linalg.matrix_power(M[1:,1:],t))
    top_25=np.argsort(x)[0][-25:]
    n=name.iloc[list(reversed(top_25)),0].values
    score=x[0][list(reversed(top_25))]
    return pd.DataFrame(score,index=n,columns=["rank={t}".format(t=t)])
```

In [305]:
```
rank(10)
```

```
Out[305]:                      rank=10
          Mary Hardin-Baylor  0.017640
          Clemson             0.014032
          Mount Union         0.012249
          Morningside         0.011157
          North Dakota St     0.010944
          Valdosta St         0.010343
          St John's MN        0.009912
          Alabama             0.009898
          UW-Whitewater       0.009683
          Ferris St           0.009671
          Johns Hopkins       0.009448
          Brockport St        0.006859
          Princeton           0.006811
          Minn St-Mankato     0.006735
          Benedictine KS      0.006551
          Kansas Wesleyan     0.005917
          Ohio State          0.005844
          Marian IN           0.005659
          Bethel MN           0.005655
          Muhlenberg          0.005540
          Tarleton St         0.005476
          Ouachita Baptist    0.005418
          Georgia             0.005409
          Notre Dame          0.005299
          Notre Dame OH       0.005249

In [306]: rank(100)

Out[306]:                      rank=100
          Mary Hardin-Baylor  0.060847
          Clemson             0.048803
          Alabama             0.027211
          Mount Union         0.021396
          St John's MN        0.014651
          Morningside         0.013446
          Valdosta St         0.012792
          North Dakota St     0.012502
          Georgia             0.011877
          Ohio State          0.011871
          Notre Dame          0.011583
          UW-Whitewater       0.011481
          Ferris St           0.011187
          Oklahoma            0.010029
          Johns Hopkins       0.009890
          Texas A&M           0.008446
          LSU                 0.007800
          Florida             0.007782
```

```
Kentucky            0.007376
Texas               0.007278
Michigan            0.007193
Syracuse            0.006816
Central Florida     0.006538
Washington St       0.006380
Washington          0.006269
```

```
In [307]: rank(1000)
```

```
Out[307]:                    rank=1000
Clemson                       0.097376
Alabama                       0.053666
Georgia                       0.023191
Ohio State                    0.022592
Notre Dame                    0.022337
Oklahoma                      0.019113
Texas A&M                     0.016575
LSU                           0.015152
Florida                       0.015051
North Dakota St               0.014736
Kentucky                      0.014335
Texas                         0.013914
Michigan                      0.013782
Syracuse                      0.013223
Central Florida               0.012441
Mary Hardin-Baylor            0.011884
Washington                    0.011317
Washington St                 0.011222
Penn State                    0.010118
Auburn                        0.009977
Missouri                      0.009722
Iowa                          0.009648
Northwestern                  0.009192
West Virginia                 0.009174
Fresno St                     0.008901
```

```
In [373]: rank(10000)
```

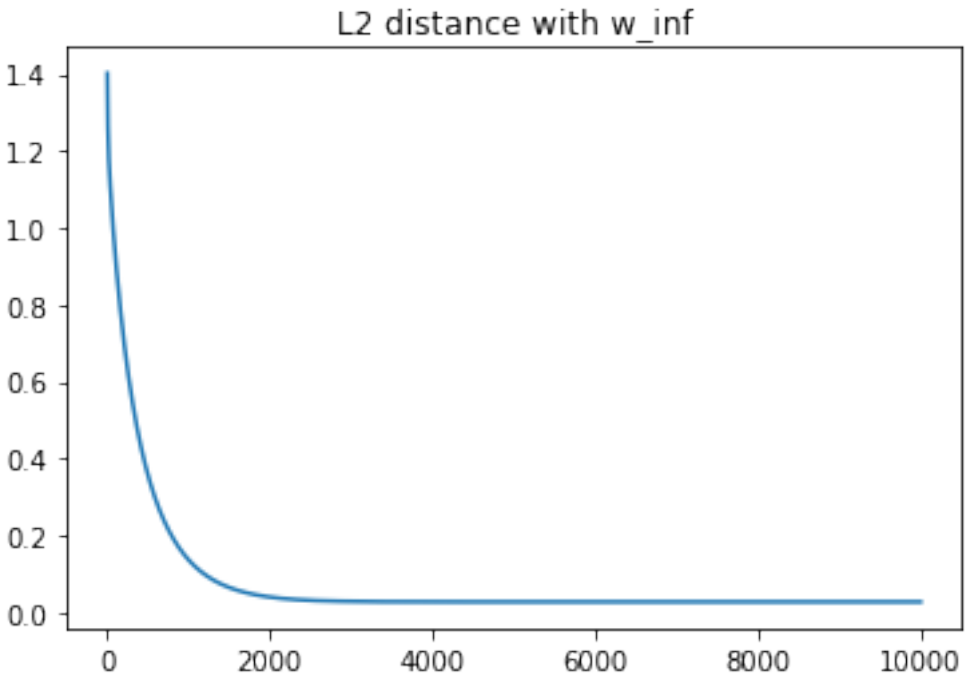
```
Out[373]:                   rank=10000
Clemson                       0.103882
Alabama                       0.057223
Georgia                       0.024716
Ohio State                    0.024055
Notre Dame                    0.023797
Oklahoma                      0.020352
Texas A&M                     0.017669
LSU                           0.016144
Florida                       0.016036
```

```
North Dakota St    0.015302
Kentucky           0.015276
Texas              0.014817
Michigan           0.014679
Syracuse           0.014091
Central Florida    0.013244
Washington         0.012022
Washington St      0.011908
Penn State         0.010773
Auburn             0.010624
Missouri           0.010359
Iowa               0.010259
Northwestern       0.009784
West Virginia      0.009764
Fresno St          0.009452
South Carolina     0.008896
```

0.1.2 b)

We saw thatwis related to the first eigenvector ofMT. That is, we can findwby getting the-first eigenvector and eigenvalue ofMTand post-processing:MTu1=1u1, w=uT1/[ju1(j)]This is be-causeuT1u1= 1by convention. Also, we observe that1= 1for this specific matrix.Plotwtw1as a func-tion oftfort= 1,...,10000.

```
In [308]: from numpy import linalg as LA

In [309]: w,v = LA.eig(M[1:,1:].T)
          v = v.T

In [310]: rearrage=sorted(zip(w,v),key=lambda x:x[0],reverse=True)

In [311]: w_inf=rearrage[0][1]/(rearrage[0][1].sum())

In [317]: temp=[]
          w=np.ones((1,size))/size
          m=M[1:,1:]
          for i in range(10000):
              x=w.dot(m)
              m=m.dot(M[1:,1:])
              score=x[0]
              temp.append(LA.norm(score-w_inf, 1))

In [323]: import matplotlib.pyplot as plt
          plt.plot(temp)
          plt.title("L2 distance with w_inf")
          plt.show()
```



0.2 Problem 2 (Nonnegative matrix factorization)

In this problem you will factorize an NŒM matrix X into a rank-K approximation WH, where W is NŒK, H is KŒM and all values in the matrices are nonnegative. Each value in W and H can be initialized randomly to a positive number, e.g., from a Uniform(1,2) distribution. The data to be used for this problem consists of 8447 documents from The New York Times. (See belowfor how to process the data.)

The vocabulary size is 3012 words. You will need to use this data to construct the matrix X, where Xij is the number of times word i appears in document j. Therefore,X is 3012Œ8447 and most values in X will equal zero.

```
In [2]: corpus=pd.read_csv("/Users/zhejindong/Desktop/hw4_data/nyt_vocab.dat",header=None)

In [3]: f = open("/Users/zhejindong/Desktop/hw4_data/nyt_data.txt", "r")
        text=f.readlines()

In [4]: x=np.zeros((len(corpus)+1,len(text)+1))

In [5]: for j,t in enumerate(text,1):
            temp=t.strip().split(',')
            for w_no,word in enumerate(temp):
                x[int(word.split(':')[0])][j]=int(word.split(':')[1])
```

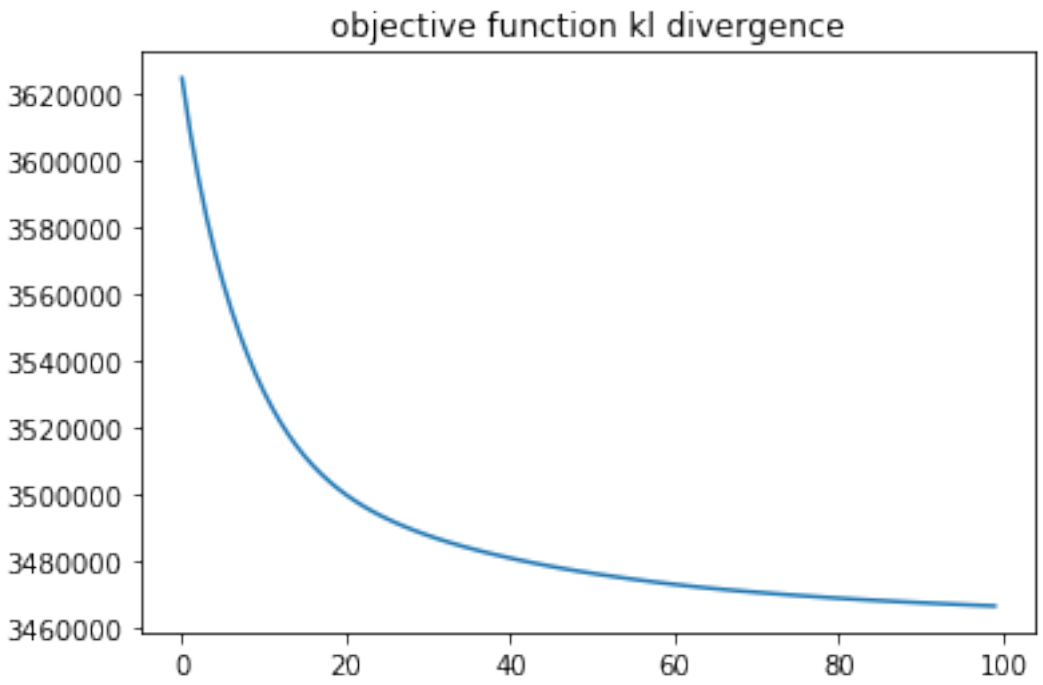
0.3 a)

Implement and run the NMF algorithm on this data using the divergence penalty. Set the rank to 25 and run for 100 iterations. This corresponds to learning 25 topics. Plot the objective as a function of iteration.

```
In [173]: # Initialize the matrix W and H
          rank=25
          W=np.random.rand(x[1:,1:].shape[0],rank)
          H=np.random.rand(rank,x[1:,1:].shape[1])
```

```
In [183]: obj=[]
          # update h and w
          for iteration in range(100):
                  H=H*W.T.dot(x[1:,1:]/(W.dot(H)+1e-16))/W.sum(axis=0,keepdims=True).T
                  W=W*(x[1:,1:]/(W.dot(H)+1e-16)).dot(H.T)/H.sum(axis=1,keepdims=True).T
                  OB=(np.log(1/(W.dot(H)+1e-16))*x[1:,1:]+W.dot(H)).sum()
                  obj.append(OB)
```

```
In [324]: import matplotlib.pyplot as plt
          plt.plot(obj)
          plt.title('objective function kl divergence')
          plt.show()
```



0.4 b)

After running the algorithm, normalize the columns of W so they sum to one. For each column of W, list the 10 words having the largest weight and show the weight. The ith row of W corresponds to the ith word in the "dictionary" provided with the data. Organize these lists in a 5Œ5table.

```
In [228]: # normalize the column of W
          W1=W/(W.sum(axis=0,keepdims=True)+1e-16)

In [346]: p=[]
          for i in range(25):
              index=np.argsort(W1[:,i])[-10:]
              index=list(reversed(index))
              p.append(pd.DataFrame(W1[index,i],index=corpus.iloc[index][0],columns=["column={
              print(p[-1])

             column=0
0
music        0.020434
art          0.013619
artist       0.011785
performance  0.008954
play         0.008919
dance        0.008622
song         0.008526
perform      0.008384
sing         0.006863
present      0.006506
           column=1
0
president  0.055225
executive  0.037806
vice       0.026799
director   0.024724
father     0.024570
graduate   0.021500
chief      0.020366
mrs        0.019280
name       0.018601
son        0.018188
          column=2
0
book      0.011994
life      0.011850
write     0.010902
editor    0.010214
world     0.009450
history   0.008696
american  0.007848
```

```
society   0.006468
writer    0.006454
great     0.006367
            column=3
0
states      0.010057
policy      0.009916
country     0.009299
meeting     0.008877
government  0.008645
american    0.008597
official    0.007726
leader      0.006964
plan        0.006913
issue       0.006853
             column=4
0
list         0.019890
article      0.015463
information  0.013972
site         0.012802
write        0.012611
service      0.012343
editor       0.010962
newspaper    0.010557
name         0.009977
offer        0.009535
            column=5
0
military    0.015924
war         0.014678
attack      0.011900
force       0.011857
government  0.011335
kill        0.010216
official    0.010176
police      0.009092
american    0.008886
leader      0.007451
         column=6
0
game     0.022390
hit      0.021053
score    0.017546
second   0.017540
play     0.014704
third    0.013585
shot     0.013565
```

```
point    0.013291
ball     0.011755
victory  0.011374
         column=7
0
win      0.035644
second   0.022814
final    0.016345
race     0.015699
finish   0.012725
match    0.011176
world    0.010678
victory  0.010543
winner   0.009979
states   0.009629
          column=8
0
company   0.020039
sale      0.019240
market    0.018891
percent   0.017882
sell      0.016821
price     0.016117
industry  0.015909
business  0.013353
product   0.011368
buy       0.009806
               column=9
0
official       0.020251
spokesman      0.011567
report         0.011365
comment        0.009956
yesterday      0.009103
office         0.008615
statement      0.008486
investigation  0.008407
member         0.008354
accord         0.008009
            column=10
0
company      0.018550
stock        0.018089
percent      0.017269
share        0.016534
market       0.015271
bank         0.012496
price        0.011644
```

```
investor     0.011279
investment   0.009926
financial    0.009433
          column=11
0
building   0.021704
city       0.020313
build      0.014515
area       0.011921
house      0.010284
project    0.009979
resident   0.009976
space      0.009871
open       0.009555
street     0.009488
            column=12
0
vote         0.021161
campaign     0.020625
political    0.017400
republican   0.014942
election     0.014818
party        0.014255
candidate    0.014085
state        0.012215
democratic   0.011632
leader       0.010960
            column=13
0
computer     0.019674
system       0.017006
technology   0.015274
company      0.011775
design       0.011538
machine      0.009257
program      0.007927
device       0.007491
equipment    0.007406
develop      0.007339
        column=14
0
family   0.019366
home     0.017801
live     0.015798
woman    0.015654
house    0.013731
man      0.012935
friend   0.012854
```

```
room     0.012365
child    0.011677
wife     0.011564
        column=15
0
mile     0.012992
car      0.012000
travel   0.010171
water    0.010078
driver   0.009750
fly      0.009624
air      0.009614
hour     0.009580
plane    0.009348
flight   0.008630
        column=16
0
pay      0.021922
money    0.017857
cost     0.012561
state    0.012432
tax      0.011985
budget   0.010669
cut      0.010584
union    0.010017
worker   0.009986
plan     0.009752
        column=17
0
thing    0.020294
feel     0.014007
ask      0.013145
tell     0.012622
really   0.011022
lot      0.009830
little   0.009386
happen   0.008497
keep     0.007813
put      0.007749
         column=18
0
far       0.008114
level     0.007829
result    0.007359
change    0.007252
number    0.006760
grow      0.006390
small     0.006144
```

```
large     0.005961
problem   0.005953
recent    0.005691
          column=19
0
team       0.042993
player     0.030369
game       0.027140
season     0.025199
play       0.023520
coach      0.018081
league     0.010923
baseball   0.009723
football   0.009518
contract   0.008978
            column=20
0
film         0.017047
character    0.014827
movie        0.014696
play         0.012615
television   0.012008
story        0.010837
star         0.008289
man          0.007787
woman        0.007592
director     0.007463
           column=21
0
drug        0.020565
health      0.019042
doctor      0.016780
medical     0.015657
patient     0.013586
treatment   0.012664
hospital    0.011737
study       0.011417
cause       0.011022
care        0.010797
            column=22
0
food         0.010085
red          0.010013
white        0.009205
serve        0.007142
green        0.006952
black        0.006673
small        0.006555
```

```
restaurant   0.006388
taste        0.006381
color        0.006263
        column=23
0
court    0.022581
case     0.022289
lawyer   0.019733
charge   0.018554
law      0.017257
judge    0.015007
police   0.012527
legal    0.011462
trial    0.010572
state    0.009858
          column=24
0
school      0.047123
student     0.032614
child       0.028522
parent      0.016550
class       0.014839
education   0.014291
program     0.014251
college     0.013461
teacher     0.012463
community   0.011685
```