session 2 notebook

The notebook has been written during the second session
please watch the video on "Course Materials" section of iLearn for the full description

April 15, 2020

1 Session 2

1.0.1 Introduction to applied data science

TA: Nima Chartab

1.0.2 For the detailed info look at the full documentation of numpy

1.0.3 First import the needed packages (numpy and matplotlib for this session)

[2]:
```
import numpy as np
import matplotlib.pylab as plt
```

1.0.4 Scalar-array and array-array operations

[3]:
```
#let's make a 2*3 array
A=np.array([[1,2,3],
            [4,5,6]])
A
```

[3]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

1.0.5 note that arithmetic operations work element-wise

[4]:
```
2*A
```

[4]:
```
array([[ 2,  4,  6],
       [ 8, 10, 12]])
```

[5]:
```
A+4
```

[5]:
```
array([[ 5,  6,  7],
       [ 8,  9, 10]])
```

```
[6]: B=np.array([[7,9,10],[4,7,2]])
     B
```

```
[6]: array([[ 7,  9, 10],
            [ 4,  7,  2]])
```

```
[7]: B-A
```

```
[7]: array([[ 6,  7,  7],
            [ 0,  2, -4]])
```

```
[9]: B*A # note that this does not return matrix multiplication
```

```
[9]: array([[ 7, 18, 30],
            [16, 35, 12]])
```

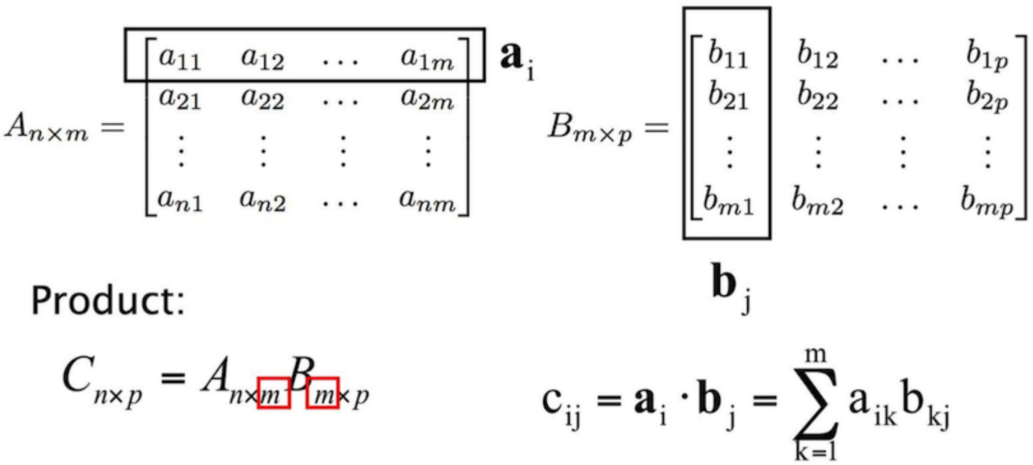
```
[10]: # len()
      #number of row
      len(A)
```

```
[10]: 2
```

```
[15]: #number of column
      len(A[0])
```

```
[15]: 3
```

1.1 Matrix multiplication



$$A_{n\times m} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \mathbf{a}_i \qquad B_{m\times p} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{bmatrix}$$

$$\mathbf{b}_j$$

Product:

$$C_{n\times p} = A_{n\times \boxed{m}} B_{\boxed{m}\times p} \qquad c_{ij} = \mathbf{a}_i \cdot \mathbf{b}_j = \sum_{k=1}^{m} a_{ik} b_{kj}$$

Source: https://web.stanford.edu/class/cs231a/section/section1.pdf

1.1.1 Let's write a function for matrix multiplication without using numpy

```
[17]: #zip function
      A=[2,4]
      B=[[3,4],
         [5,8]]
```

```
[18]: list(zip(A,B))
```

```
[18]: [(2, [3, 4]), (4, [5, 8])]
```

```
[19]: list(zip(*B))
```

```
[19]: [(3, 5), (4, 8)]
```

```
[20]: #let's define A,B
      A=[[1,2,3],
          [4,5,6]]
      B=[[3,4],
         [5,6],
         [4,7]]
```

```
[25]: def MatrixMulti(A,B):
          """
          A function which computes matrix multiplication of the first
          and second inputs. Inputs should be list-like.
          """

          if len(A[0])==len(B):
              R=[[sum(x*y for x,y in zip(A_row,B_col)) for B_col in zip(*B)] for A_row␣
        ↪in A]
              return R
          else:
              print('Matrix dimensions do not match')
```

```
[26]: MatrixMulti(A,B)
```

```
[26]: [[25, 37], [61, 88]]
```

1.1.2 let's do it with numpy

```
[27]: #np.dot
      A=np.array(A)
      B=np.array(B)
      np.dot(A,B)
```

```
[27]: array([[25, 37],
              [61, 88]])
```

```
[28]: A*B
```

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

        ValueError                                Traceback (most recent call last)

        <ipython-input-28-47896efed660> in <module>
    ----> 1 A*B


        ValueError: operands could not be broadcast together with shapes (2,3)␣
    ↪(3,2)
```

```
[29]: #you can convert array to matrix and use arthmetic operations
      A=np.matrix(A)
      A
```

```
[29]: matrix([[1, 2, 3],
              [4, 5, 6]])
```

```
[30]: B=np.matrix(B)
      B
```

```
[30]: matrix([[3, 4],
              [5, 6],
              [4, 7]])
```

```
[31]: A*B
```

```
[31]: matrix([[25, 37],
              [61, 88]])
```

```
[44]: #Transpose
      v=np.ones(10)
      v
```

```
[44]: array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
[45]: v=np.matrix(v)
      v
```

```
[45]: matrix([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]])
```

[46]:

```
v=v.T
v
```

[46]:

```
matrix([[1.],
        [1.],
        [1.],
        [1.],
        [1.],
        [1.],
        [1.],
        [1.],
        [1.],
        [1.]])
```

[47]:

```
D=np.random.randint(0,2,(10,10))
D
```

[47]:

```
array([[1, 0, 1, 0, 0, 1, 1, 0, 1, 0],
       [0, 0, 1, 0, 1, 0, 0, 0, 0, 0],
       [1, 0, 1, 1, 1, 0, 0, 1, 0, 0],
       [1, 0, 1, 1, 1, 1, 1, 1, 1, 0],
       [0, 1, 1, 1, 0, 1, 1, 1, 1, 0],
       [0, 1, 1, 0, 1, 0, 0, 0, 0, 1],
       [1, 0, 0, 1, 0, 0, 0, 1, 1, 1],
       [1, 1, 1, 1, 0, 0, 0, 1, 1, 0],
       [0, 0, 0, 1, 1, 1, 0, 0, 0, 1],
       [1, 0, 0, 1, 0, 1, 1, 1, 1, 1]])
```

[48]:

```
D=np.matrix(D)
```

[49]:

```
#get Dv
D*v
```

[49]:

```
matrix([[5.],
        [2.],
        [5.],
        [8.],
        [7.],
        [4.],
        [5.],
        [6.],
        [4.],
        [7.]])
```

1.1.3 Example: Market share of technology companies

1.1.4 Financial analysis of rise and decline of three technology companies show that the monthly market shares of three companies A, B and C can be estimated by a transformation matrix P:

1.1.5

$$P = \begin{bmatrix} 0.8 & 0.03 & 0.2 \\ 0.1 & 0.95 & 0.05 \\ 0.1 & 0.02 & 0.75 \end{bmatrix}$$

1.1.6 For example: the first column of matrix P represents the share of Company A that will pass to Company A, Company B and Company C respectively. The initial market share of the three companies is

$$I = \begin{bmatrix} 30 \\ 15 \\ 55 \end{bmatrix}$$

.

1.1.7 Find the final market share of the 3 companies A, B and C.

[51]:
```
P=np.array([[0.8,0.03,0.2],[0.1,0.95,0.05],[0.1,0.02,0.75]])
P
```

[51]:
```
array([[0.8 , 0.03, 0.2 ],
       [0.1 , 0.95, 0.05],
       [0.1 , 0.02, 0.75]])
```

[52]:
```
I=np.array([[30],[15],[55]])
I
```

[52]:
```
array([[30],
       [15],
       [55]])
```

[53]:
```
#predicted market share after 1 month
I1=np.dot(P,I)
I1
```

[53]:
```
array([[35.45],
       [20.  ],
       [44.55]])
```

[61]:
```
#let's write a function which takes P and I and finds share after N months
def Market_Share(Trans, Init, N):
    """
    A function for market share problem.
    """
    S=np.dot(Trans,Init)
    for i in range(N-1):
        S=np.dot(Trans,S)
    return(S.T[0].tolist())
```

```
[62]: Market_Share(P,I,3)
```

```
[62]: [38.510675000000006, 29.18875, 32.300574999999995]
```

```
[96]: N=range(1,41)
      Share=[]
      for j in N:
          Share.append(Market_Share(P,I,j))
```

```
[97]: Share
```

```
[97]: [[35.45, 20.0, 44.55],
       [37.870000000000005, 24.7725, 37.357499999999995],
       [38.510675000000006, 29.18875, 32.300574999999995],
       [38.14431750000001, 33.19540875, 28.660273749999998],
       [37.243371012500006, 36.783083749999996, 25.973545237499998],
       [36.09289837000001, 39.966943925624996, 23.940157704375],
       [34.86135855464376, 42.774894451562496, 22.363746993793754],
       [33.64508307602063, 45.24047293413844, 21.114443989840943],
       [32.49616944680885, 47.39867979452563, 20.10515075866538],
       [31.439926103015956, 49.283620287413505, 19.276453609570552],
       [30.48574021294928, 50.92725456382295, 18.58700522322778],
       [29.63381085191967, 52.358816118088114, 18.007373029992223],
       [28.879287771076825, 53.604625048875285, 17.516087180047897],
       [28.214786404337296, 54.68812693254159, 17.09708666312111],
       [27.631890264070307, 55.6300535595043, 16.738056176425395],
       [27.122025053326453, 56.448642716757384, 16.429332229916163],
       [26.67694577014712, 57.15987969774797, 16.163174532104915],
       [26.28898791347112, 57.777739016480524, 15.933273070048358],
       [25.951177115280984, 58.314414510506026, 15.734408374212991],
       [25.657255802382572, 58.78053191521947, 15.562212282397963],
       [25.401663055842235, 59.185341513816645, 15.41299543034112],
       [25.179489776156515, 59.53689051522709, 15.283619708616396],
       [24.986422478105307, 59.84217595251221, 15.171401569382489],
       [24.818683574936113, 60.10727948116625, 15.074036943897642],
       [24.67297263316341, 60.33748571179643, 14.989541655040169],
       [24.546411008892658, 60.53738577227496, 14.916203218832397],
       [24.436491024048856, 60.7109677454921, 14.852541230459064],
       [24.3410300976956, 60.861695522145325, 14.797274380159024],
       [24.258129819852698, 60.99257747481557, 14.74929270533174],
       [24.186139721192976, 61.10622621832664, 14.707634060480387],
       [24.12362537560026, 61.204910582553616, 14.671464041846122],
       [24.069340426326043, 61.29060079307827, 14.64005878059569],
       [24.022202120972324, 61.36500773508674, 14.612790143940938],
       [23.98126995761865, 61.42961706762668, 14.58911297475467],
       [23.945727073074654, 61.48571885874494, 14.5685540681804],
       [23.914864037858152, 61.53443332652417, 14.550702635617665],
       [23.888064757205782, 61.57673319576465, 14.535202047029546],
       [23.864794211043474, 61.61346311404847, 14.521742674908033],
       [23.844587797237843, 61.64535651319579, 14.510055689566341],
       [23.827042071099413, 61.673050251738104, 14.499907677162454]]
```

[98]:

```
Share=np.array(Share)
Share=Share.T
```

[99]:

```
Share[0]
```

[99]:

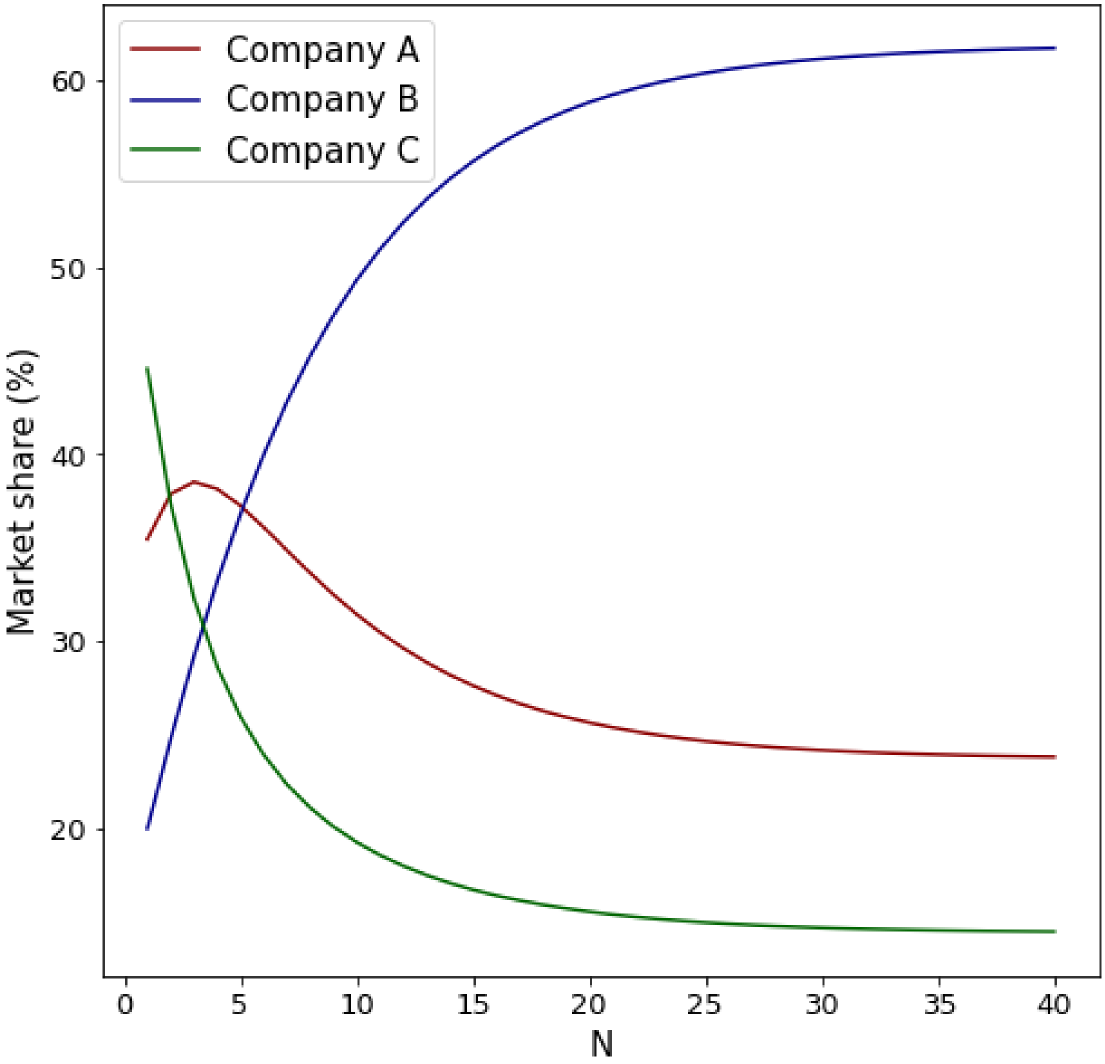
```
array([35.45      , 37.87      , 38.510675  , 38.1443175 , 37.24337101,
       36.09289837, 34.86135855, 33.64508308, 32.49616945, 31.4399261 ,
       30.48574021, 29.63381085, 28.87928777, 28.2147864 , 27.63189026,
       27.12202505, 26.67694577, 26.28898791, 25.95117712, 25.6572558 ,
       25.40166306, 25.17948978, 24.98642248, 24.81868357, 24.67297263,
       24.54641101, 24.43649102, 24.3410301 , 24.25812982, 24.18613972,
       24.12362538, 24.06934043, 24.02220212, 23.98126996, 23.94572707,
       23.91486404, 23.88806476, 23.86479421, 23.8445878 , 23.82704207])
```

[100]:

```
#plot data

plt.figure(figsize=(8,8))
plt.plot(N,Share[0],c='darkred',label="Company A")
plt.plot(N,Share[1],c='darkblue',label="Company B")
plt.plot(N,Share[2],c='darkgreen',label="Company C")
plt.xlabel('N', fontsize=15)
plt.ylabel('Market share (%)', fontsize=15)
plt.legend(fontsize=15)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)

plt.show()
```



```
[95]: #Share[0][99],Share[1][99],Share[2][99]
```

```
[95]: (23.711364426375237, 61.85563187485769, 14.433003698766914)
```

```
[101]: Share[0][39],Share[1][39],Share[2][39]
```

```
[101]: (23.827042071099413, 61.673050251738104, 14.499907677162454)
```

1.1.8 Is there any way to find final market share without estimating it month-by-month?

1.1.9 We reach the ultimate market share when $PS = S$, where S is the market share vector. So, to find the final market share we just need to find eigenvector of P with eigenvalue of 1

[103]:

```
E=np.linalg.eig(P)
E
```

[103]:

```
(array([0.6316784, 1.        , 0.8683216]),
 array([[-0.76769674,  0.34973152,  0.51126117],
        [ 0.14305534,  0.91234311, -0.80695536],
        [ 0.6246414 ,  0.21288006,  0.2956942 ]]))
```

[107]:

```
SF=E[1].T[1]
SF
```

[107]:

```
array([0.34973152, 0.91234311, 0.21288006])
```

[110]:

```
SF/np.sum(SF)*100
```

[110]:

```
array([23.71134021, 61.8556701 , 14.43298969])
```

[ ]:

```

```