Plan

1 Theory: Non-negative matrices
2 Application: Ranking models
3 Application: Population growth models

(1) Non-negative matrices

$A = (a_{ij})$ $m \times n$ matrix

Def: A is non-negative ($A \geq 0$) if $a_{ij} \geq 0$ for all i,j

A is positive ($A > 0$) if $a_{ij} > 0$ for all i,j

$\underline{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$ ✓ n-vector

non-negative: $\underline{v} \geq 0$ if $v_i \geq 0$ for all i

positive: $\underline{v} > 0$ if $v_i > 0$ — " —

Assume: $A = (a_{ij})$ $n \times n$ matrix, $A \geq 0$

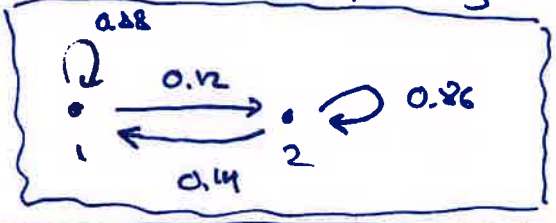
$A = (a_{ij})$ ⇝ Γ graph (directed graph)

Nodes: $1, 2, 3, \dots, n$

Edges (arrows): arrow from $j \to i$ if $a_{ij} \neq 0$ (>0)

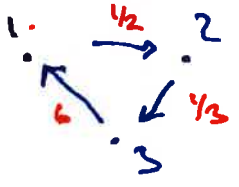
Ex: $A = \begin{pmatrix} 0.88 & 0.14 \\ 0.12 & 0.86 \end{pmatrix}$ (out from ①, out from ②) → Γ:

$a_{21} = 0.12$



col's: outgoing
row's: incoming

Ex: $A = \begin{pmatrix} 0 & 0 & 6 \\ 1/2 & 0 & 0 \\ 0 & 1/3 & 0 \end{pmatrix}$ ⇝ Γ: $A \geq 0$

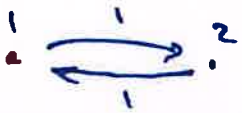


The graph Γ is called strongly connected if you get from any node to any other node in a finite number of steps.

Defn: A is called irreducible if its graph is strongly connected.

Alternatively: A is irreducible if for any indices (i,j), there is an m such that $(A^m)_{ij} > 0$.

Ex: $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ ✓



$A^2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$

$A^3 = I \cdot A = A$

$= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$A^4 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

irreducible
not primitive
no limit for A^m when $m \to \infty$

Defn: A is called primitive if there is a positive integer m such that $A^m > 0$.

$A > 0 \Rightarrow A$ primitive

primitive ⇓ irreducible

If A is a Markov chain matrix, then:

primitive = regular

A primitive (regular) Markov chain: $\lim_{m \to \infty} A^m = (\underline{v}|\underline{v}|\cdots|\underline{v})$

$A \geq 0 \iff a_{ij} \geq 0$ for all ij

Thm (Perron-Frobenius)

Let A be a non-negative $n \times n$-matrix that is irreducible, then:

1) there is a dominant eigenvalue $\lambda_A > 0$ such that $|\lambda| \leq \lambda_A$ for all eigenvalues λ of A (real or complex)
   (abs-value or modulus)
2) There is a positive eigenvector $\underline{v}_A > 0$ with eigenvalue λ_A
3) λ_A has multiplicity 1, and any (non-zero) eigenvector $\underline{v} \geq 0$ of A is a positive multiple of $\underline{v}_A$.
4) all eigenvalues of A with $|\lambda| = \lambda_A$ are all solutions of $\lambda^k = \lambda_A^k$ for some positive integer k.

Ex. 1: $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

irreducible not primitive (regular)

$\begin{vmatrix} -\lambda & 1 \\ 1 & -\lambda \end{vmatrix} = 0$

$\lambda^2 - 1 = 0$

$\lambda = 1, \lambda = -1$

$\lambda_A = 1$ dominant eigenvalue

$\underline{v}_A$: $\left(\begin{array}{cc|c} -1 & 1 & 0 \\ 1 & -1 & 0 \end{array}\right)$ $x = y$, y free

$\underline{v}_1 = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y \\ y \end{pmatrix} = y \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$\underline{v}_A = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ or $\underline{v}_A = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}$ stable vector

$\underline{E}_{-1}$: $\left(\begin{array}{cc|c} 1 & 1 & 0 \\ 1 & 1 & 0 \end{array}\right)$ ✓

$x = -y$, y free

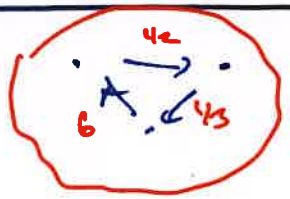
$\underline{v} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -y \\ y \end{pmatrix} = y \cdot \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

$k = 2$: $\lambda^k = 1$, $\lambda^2 = 1$, $\lambda = 1, -1$

minimum column sum of A $\leq \lambda_A \leq$ maximal column sum of A

Ex: $A = \begin{pmatrix} 0 & 0 & 6 \\ 1/2 & 0 & 0 \\ 0 & 1/3 & 0 \end{pmatrix}$

irreducible



irreducible

$$\begin{vmatrix} -\lambda & 0 & 6 \\ 1/2 & -\lambda & 0 \\ 0 & 1/3 & -\lambda \end{vmatrix} = 0$$

$$-\lambda \cdot (\lambda^2 - 0) + 6 \cdot (1/6 - 0) = 0$$

$$-\lambda^3 + 1 = 0$$

$$\lambda^3 - 1 = 0$$

$\lambda_A = 1$

$$(\lambda - 1)(\lambda^2 + \lambda + 1) = 0$$

$$\lambda = 1, \quad \lambda = \frac{-1 \pm \sqrt{1-4}}{2}$$

$$= \frac{-1 \pm \sqrt{3}\,i}{2}$$

$\lambda_1 = 1$ $\quad \lambda_2 = \frac{-1 + \sqrt{3}\,i}{2}$ $\quad \lambda_3 = \frac{-1 - \sqrt{3}\,i}{2}$

v_A:

$$\left(\begin{array}{ccc|c} -1 & 0 & 6 & 0 \\ 1/2 & -1 & 0 & 0 \\ 0 & 1/3 & -1 & 0 \end{array}\right)$$

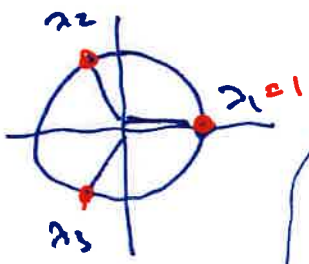
$$\vdots$$

$$v_A = \begin{pmatrix} 6 \\ 3 \\ 1 \end{pmatrix}$$

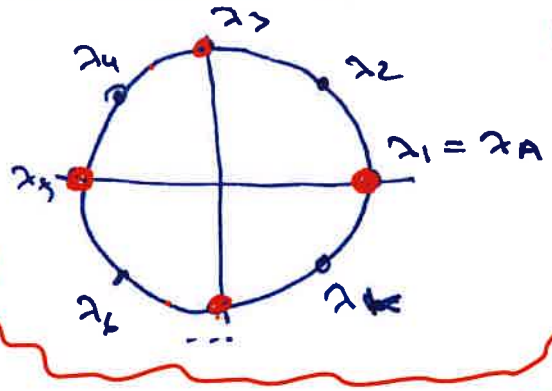
or

$$v_A = \begin{pmatrix} 0.6 \\ 0.3 \\ 0.1 \end{pmatrix}$$

$k = 3$: $\lambda^3 = 1$



General picture:



k: index of primitivity
=
# of eigenvalues λ with $|\lambda| = \lambda_A$

$\lambda^k = \lambda_A^k$

Sol. of $\lambda^k = \lambda_A^k$ (use polar coordinates)

$k = 1 \iff A$ is primitive

Untitled9

September 28, 2020

[1]:
```
import numpy as np
from numpy import linalg as LA
```

[2]:
```
A = np.array([[0.75,0.02,0.1],[0.2,0.9,0.2],[0.05,0.08,0.7]])
B = np.array([[0,0,6],[1/2,0,0],[0,1/3,0]])
```

[3]:
```
LA.eig(A)
```

[3]:
```
(array([1.  , 0.7 , 0.65]),
 array([[-1.88144174e-01, -8.08122036e-01, -7.07106781e-01],
        [-9.40720868e-01,  5.05076272e-01, -1.59384710e-15],
        [-2.82216261e-01,  3.03045763e-01,  7.07106781e-01]]))
```

[4]:
```
LA.matrix_power(A,5)
```

[4]:
```
array([[0.28940219, 0.09010763, 0.17337312],
       [0.55462   , 0.72269   , 0.55462   ],
       [0.15597781, 0.18720238, 0.27200687]])
```

[5]:
```
LA.matrix_power(A,10)
```

[5]:
```
array([[0.16077148, 0.12365308, 0.14730873],
       [0.64783498, 0.67608251, 0.64783498],
       [0.19139354, 0.20026441, 0.20485628]])
```

[6]:
```
LA.matrix_power(A,100)
```

[6]:
```
array([[0.13333333, 0.13333333, 0.13333333],
       [0.66666667, 0.66666667, 0.66666667],
       [0.2       , 0.2       , 0.2       ]])
```

[7]:
```
LA.eig(B)
```

[7]:
```
(array([-0.5+0.8660254j, -0.5-0.8660254j,  1. +0.j       ]),
 array([[ 0.88465174+0.j        ,  0.88465174-0.j        ,
         -0.88465174+0.j        ],
        [-0.22116293-0.38306544j, -0.22116293+0.38306544j,
         -0.44232587+0.j        ],
```

```
       [-0.07372098+0.12768848j, -0.07372098-0.12768848j,
        -0.14744196+0.j        ]]))
```

[8]:
```
LA.matrix_power(B,5)
```

[8]:
```
array([[0.        , 2.        , 0.        ],
       [0.        , 0.        , 3.        ],
       [0.16666667, 0.        , 0.        ]])
```

[9]:
```
LA.matrix_power(B,10)
```

[9]:
```
array([[0.        , 0.        , 6.        ],
       [0.5       , 0.        , 0.        ],
       [0.        , 0.33333333, 0.        ]])
```

[10]:
```
LA.matrix_power(B,100)
```

k=3

[10]:
```
array([[0.        , 0.        , 6.        ],
       [0.5       , 0.        , 0.        ],
       [0.        , 0.33333333, 0.        ]])
```

[11]:
```
LA.matrix_power(B,101)
```

[11]:
```
array([[0.        , 2.        , 0.        ],
       [0.        , 0.        , 3.        ],
       [0.16666667, 0.        , 0.        ]])
```

[12]:
```
LA.matrix_power(B,102)
```

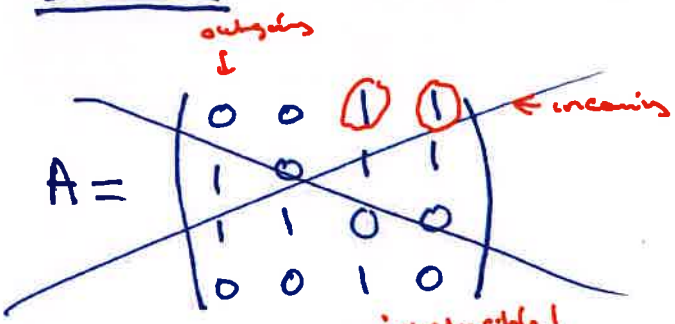
[12]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

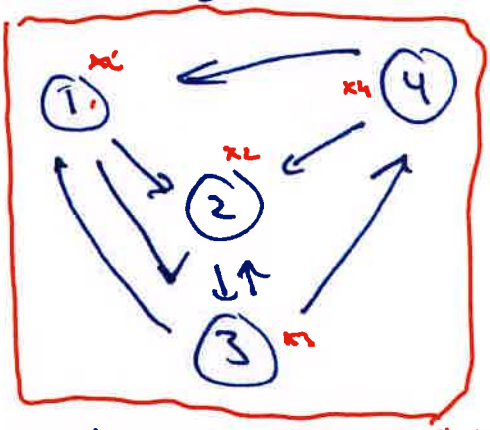
[ ]:

(2) Ranking models:

See note in lecture plan "Google's secret and linear algebra"

PageRank: Google algorithm for ranking web pages





x_i: positive number (higher number = higher rank)

node = webpage
arrow = link

$x_1 = c/31$ $x_2 = 9/31$ $x_3 = 12/31$ $x_4 = 4/31$

(3) (2) (1) (4)

$$\begin{aligned} x_1 &= K(1\cdot x_3 + 1\cdot x_4) \\ x_2 &= K(1\cdot x_1 + 1\cdot x_3 + 1\cdot x_4) \\ x_3 &= K(1\cdot x_1 + 1\cdot x_2) \\ x_4 &= K(1\cdot x_3) \end{aligned}$$

irreducible

$$\underline{x} = K\cdot(A\cdot\underline{x})$$

$$\frac{1}{K}\cdot\underline{x} = A\cdot\underline{x}$$

$$\lambda\cdot\underline{x} = A\underline{x} \qquad \underline{x} > 0$$

$\underline{x} = k\cdot v_A$

$$A = \begin{pmatrix} 0 & 0 & 1/3 & 1/2 \\ 1/2 & 0 & 1/3 & 1/2 \\ 1/2 & 1 & 0 & 0 \\ 0 & 0 & 1/3 & 0 \end{pmatrix}$$

$$\underline{x} = \tfrac{1}{4}x_4\cdot\begin{pmatrix} 6 \\ 9 \\ 12 \\ 4 \end{pmatrix} \Rightarrow v_A = \begin{pmatrix} 6/31 \\ 9/31 \\ 12/31 \\ 4/31 \end{pmatrix} \checkmark$$

$\lambda_A = 1$:

$$\begin{pmatrix} -1 & 0 & 1/3 & 1/2 \\ 1/2 & -1 & 1/3 & 1/2 \\ 1/2 & 1 & -1 & 0 \\ 0 & 0 & 1/3 & -1 \end{pmatrix} \xrightarrow{\cdot 1/2} \begin{pmatrix} -1 & 0 & 1/3 & 1/2 \\ 0 & -1 & 1/2 & 3/4 \\ 0 & 1 & -5/6 & 1/4 \\ 0 & 0 & 1/3 & -1 \end{pmatrix}$$

$$\underline{x} \to \left|\frac{\underline{v}}{\underline{v}}\right| \to \begin{pmatrix} -1 & 0 & 1/3 & 1/2 \\ 0 & -1 & 1/2 & 3/4 \\ 0 & 0 & -1/3 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

as many as

$-x_1 + \frac{1}{3}x_3 + \frac{1}{2}x_4 = 0 \quad x_1 = \frac{3}{2}x_4$

$-x_2 + \frac{1}{2}x_3 + \frac{3}{4}x_4 = 0 \quad x_2 = \frac{9}{4}x_4$

$-\frac{1}{3}x_3 + x_4 = 0 \Rightarrow x_3 = 3x_4$

Extra example: Sports ratings

Six teams play 21 matches (6 against teams in the same "conference"/geographic area, 3 against teams in the other "conference"). Number of wins of team i over team j: position (i,j)

	1	2	3	4	5	6	
1	0	3	0	–	1	2	6
2	3	0	2	2	2		10
3	6	4	0	2	1	1	14
4	3	1	1	0	2	2	9
5	2	1	2	1	0	2	11
6	1	2	2	1	1	0	13

Team 2: won 3 matches against team 1, 2 — 1 — 3, 2 — 1 — 4, 2 — 1 — 5, 1 — 1 — 6

Total wins. Team 3 seems "best"

Better method:

$$A = \begin{pmatrix} 0 & 3 & 0 & 0 & 1 & 2 \\ 3 & 0 & 2 & 2 & 2 & 1 \\ 6 & 4 & 0 & 2 & 1 & 1 \\ 3 & 1 & 1 & 0 & 2 & 2 \\ 2 & 1 & 2 & 1 & 0 & 2 \\ 1 & 2 & 2 & 1 & 1 & 0 \end{pmatrix}$$

Look for $\underline{x}$ such that $A\cdot\underline{x} = \lambda\underline{x}$

where $x_1\ x_2\ x_3\ x_4\ x_5\ x_6$ are pos. numbers

x_i: measure of how good team i is

$$x_1 = K\cdot(3x_2 + 1\cdot x_3 + 2x_6)$$
$$x_2 = K(3x_1 + 2x_3 + 2x_4 + 2x_5 + 1\cdot x_6)$$
$$\vdots$$

$$\underline{x} = K\cdot A\underline{x} \iff \frac{1}{K}\underline{x} = A\underline{x}$$
$$\lambda\underline{x} = A\underline{x}$$

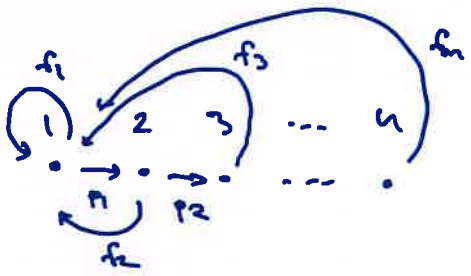
Computation: Problem set B, pb. 2

Plan:

1. Population growth models and Leslie matrices
2. Pandas in python / time series

1. Leslie matrices

$$A = \begin{pmatrix} f_1 & f_2 & f_3 & \cdots & f_{n-1} & f_n \\ p_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & p_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & p_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & p_{n-1} & 0 \end{pmatrix}$$



p_i: survival rate

$0 < p_i < 1$

f_i: birthrates

$f_i \geq 0$, $f_n \neq 0$

x_1: 0-1
x_2: 1-2
x_3: 2-3
$\vdots$
x_n: (n-1)-n

pop. in different age groups

$$\underline{x}_t = \begin{pmatrix} x_{1t} \\ x_{2t} \\ \vdots \\ x_{nt} \end{pmatrix}$$

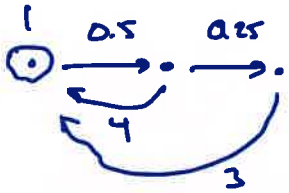
Assumptions:

- only females
- fertility up to age n

$$\underline{x}_{t+1} = A \cdot \underline{x}_t$$

Ex: Salmon, time in years, $n = 2$

$$A = \begin{pmatrix} 0 & 4 & 3 \\ 0.5 & 0 & 0 \\ 0 & 0.25 & 0 \end{pmatrix}$$



- non-negative, irreducible
- primitive (key): ok when there are several $f_i > 0$

$$\underline{v}_0 = \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix} \quad \underline{v}_1 = A \cdot \underline{v}_0 \quad \underline{v}_2 = A \cdot \underline{v}_1 \quad \cdots \quad \underline{v}_m = A \cdot \underline{v}_{m-1}$$

$$3 \left\{ \left(\underline{v}_0 \mid \underline{v}_1 \mid \underline{v}_2 \mid \cdots \mid \underline{v}_m \right) \right. \leftarrow \text{first row: } x_{10}\ x_{11}\ x_{12} \cdots x_{1m} \text{ (salmon 0+)}$$

$m+1$

Example Leslie matrices

October 6, 2020

[8]:
```
import numpy as np
import pandas as pd
```

[9]:
```
A = np.array([[0,4,3],[0.5,0,0],[0,0.25,0]])
v = np.array([[10],[10],[10]])
```

[3]:
```
eval, evect = np.linalg.eig(A)
```

[4]:
```
eval
```

[4]:
```
array([ 1.5       , -1.30901699, -0.19098301])
```

[5]:
```
w = evect[:,0]
```

[7]:
```
w/w.sum()
```

[7]:
```
array([0.72, 0.24, 0.04])
```

[10]:
```
rows = v.shape[0]
for i in range(12):
    last = v.shape[1]
    w = np.reshape(v[:,last-1],(rows,1))
    v = np.append(v,A.dot(w),axis=1)
```

[11]:
```
v
```

[11]:
```
array([[1.00000000e+01, 7.00000000e+01, 2.75000000e+01, 1.43750000e+02,
        8.12500000e+01, 2.97812500e+02, 2.16406250e+02, 6.26093750e+02,
        5.44492188e+02, 1.33333984e+03, 1.32376953e+03, 2.87086426e+03,
        3.14754150e+03],
       [1.00000000e+01, 5.00000000e+00, 3.50000000e+01, 1.37500000e+01,
        7.18750000e+01, 4.06250000e+01, 1.48906250e+02, 1.08203125e+02,
        3.13046875e+02, 2.72246094e+02, 6.66669922e+02, 6.61884766e+02,
        1.43543213e+03],
       [1.00000000e+01, 2.50000000e+00, 1.25000000e+00, 8.75000000e+00,
        3.43750000e+00, 1.79687500e+01, 1.01562500e+01, 3.72265625e+01,
        2.70507812e+01, 7.82617188e+01, 6.80615234e+01, 1.66667480e+02,
```

```
        1.65471191e+02]])
```

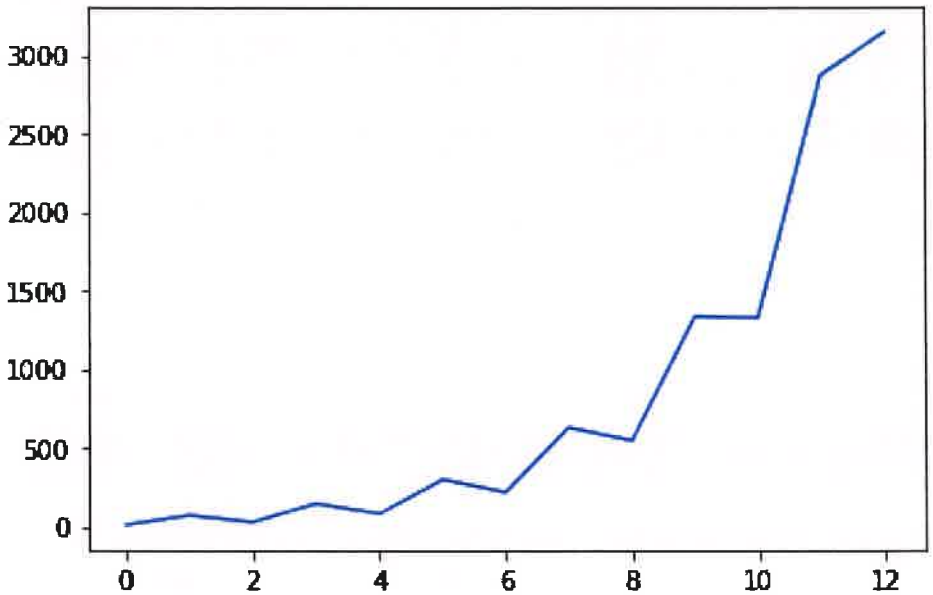
[12]:
```
u = pd.Series(v[0])
```

[13]:
```
u
```

[13]:
```
0       10.000000
1       70.000000
2       27.500000
3      143.750000
4       81.250000
5      297.812500
6      216.406250
7      626.093750
8      544.492188
9     1333.339844
10    1323.769531
11    2870.864258
12    3147.541504
dtype: float64
```

[14]:
```
u.plot()
```

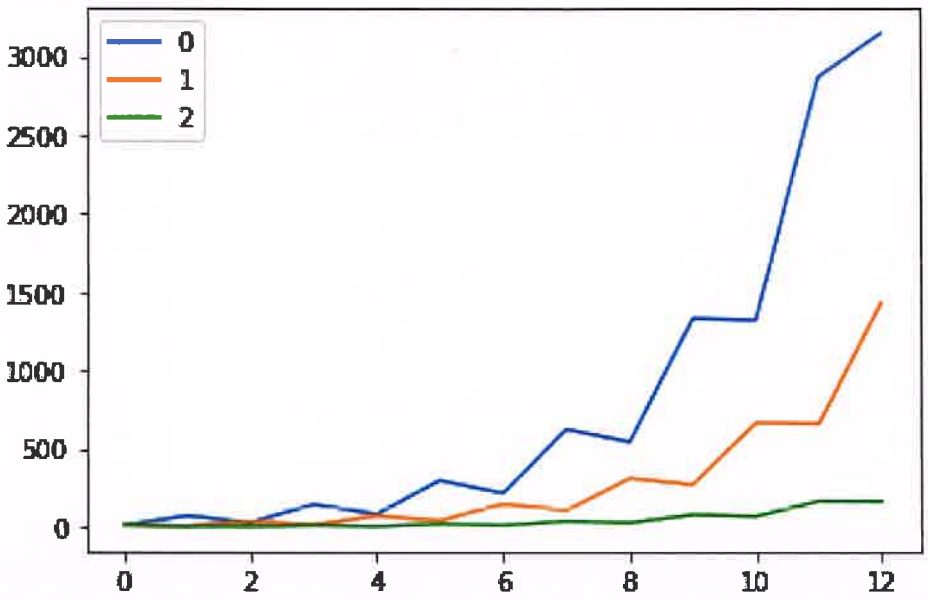
[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26ca0cf7490>
```



[17]:
```
u = pd.DataFrame(np.transpose(v))
```

[18]:
```
u.plot()
```

[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26ca0ed6250>
```



[ ]:

(2) Pandas : Python book (Ch. 5) – pd.Series(--)
– pd.DataFrame(--)