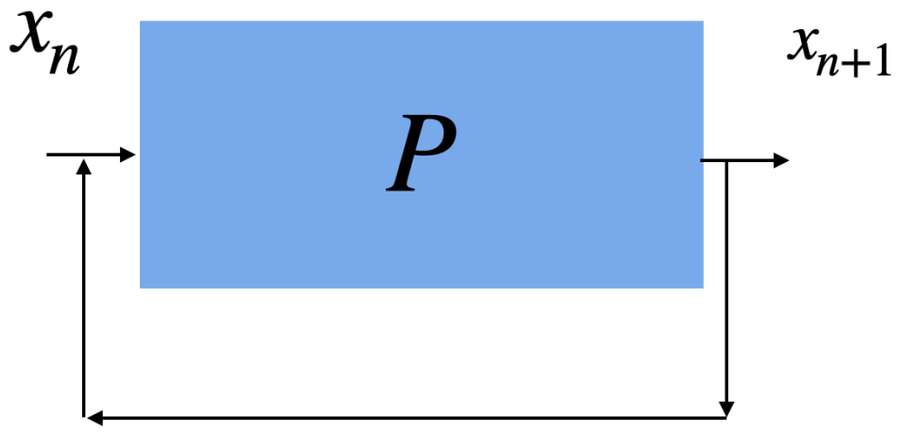
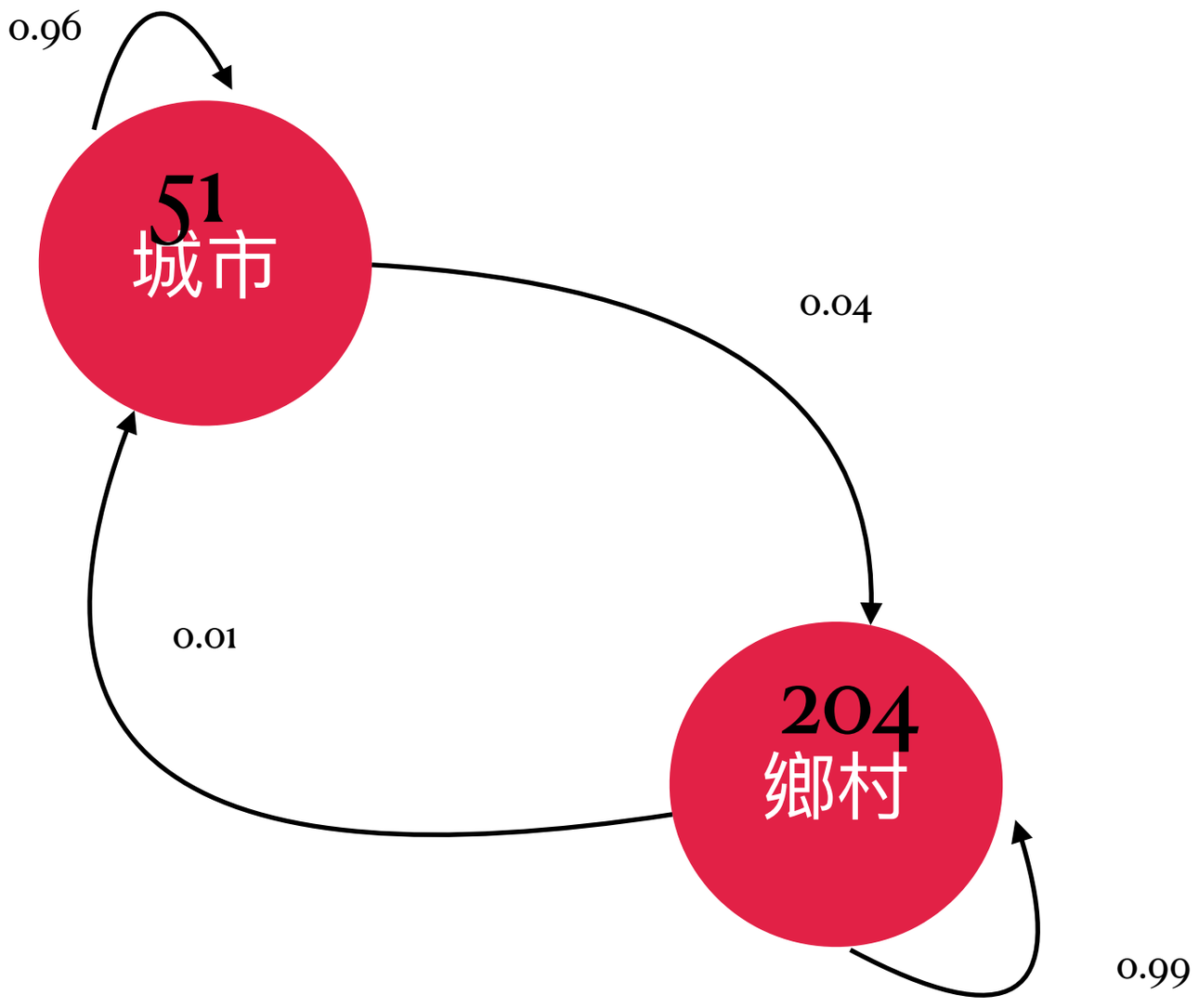
矩陣運算與Markov chain
長期預測

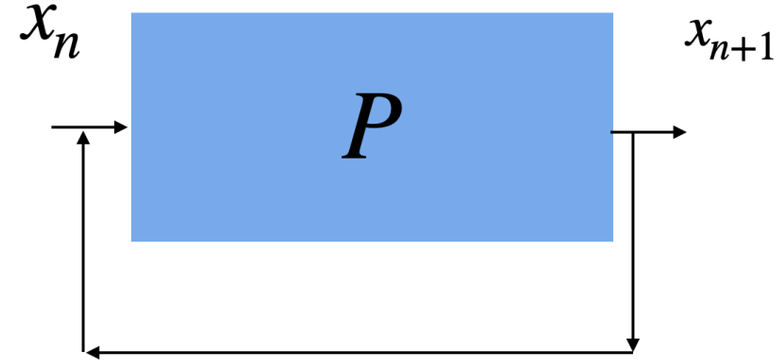


馬可夫鏈的移轉機率
Markov chain



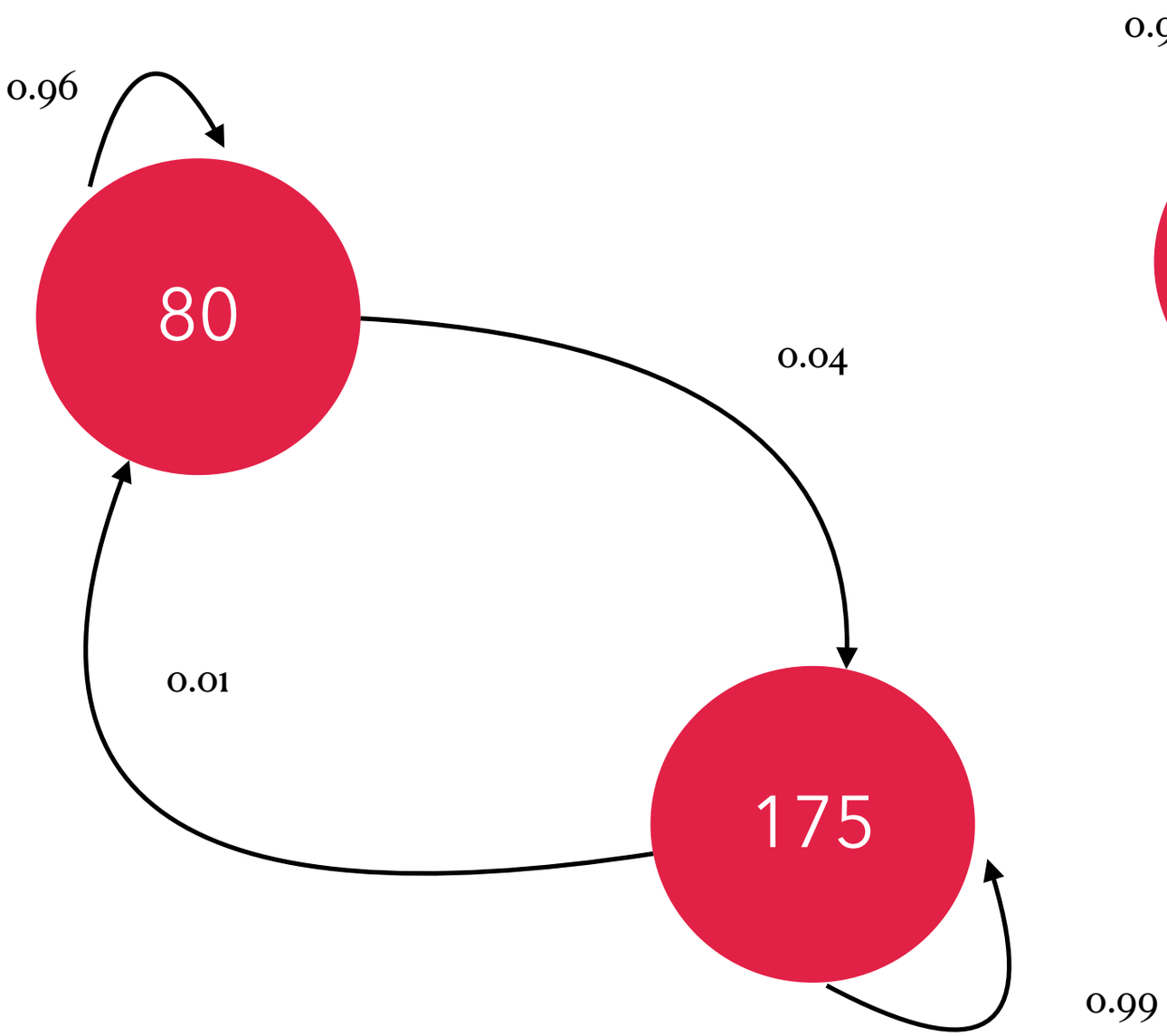
$$P = \begin{matrix} \text{(from)} \\ \text{city} \quad \text{suburb} \quad \text{(to)} \end{matrix} \begin{bmatrix} 0.96 & 0.01 \\ 0.04 & 0.99 \end{bmatrix} \begin{matrix} \\ \text{city} \\ \text{suburb} \end{matrix}$$

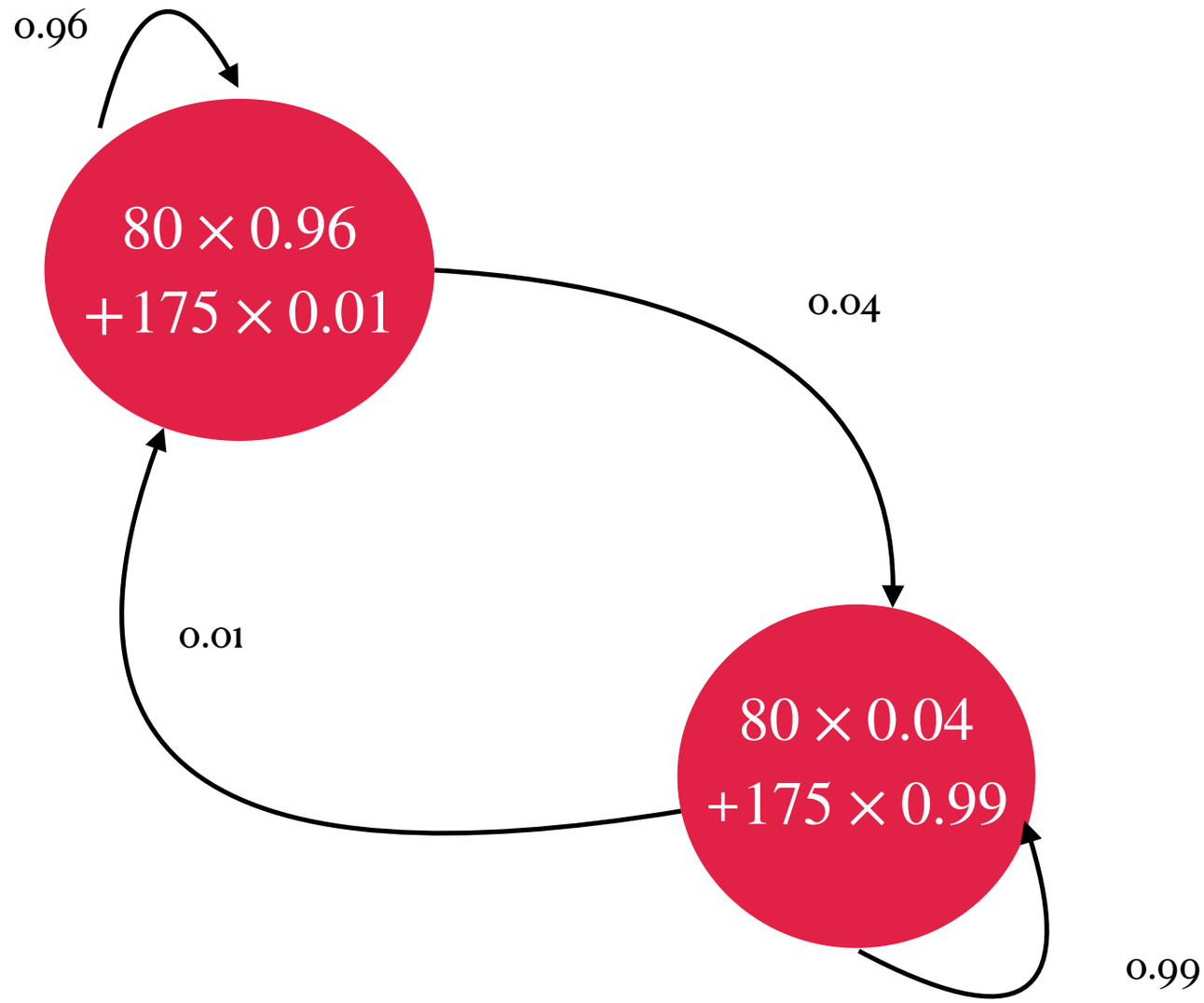
x_0代表一開始的人口
單位：百萬



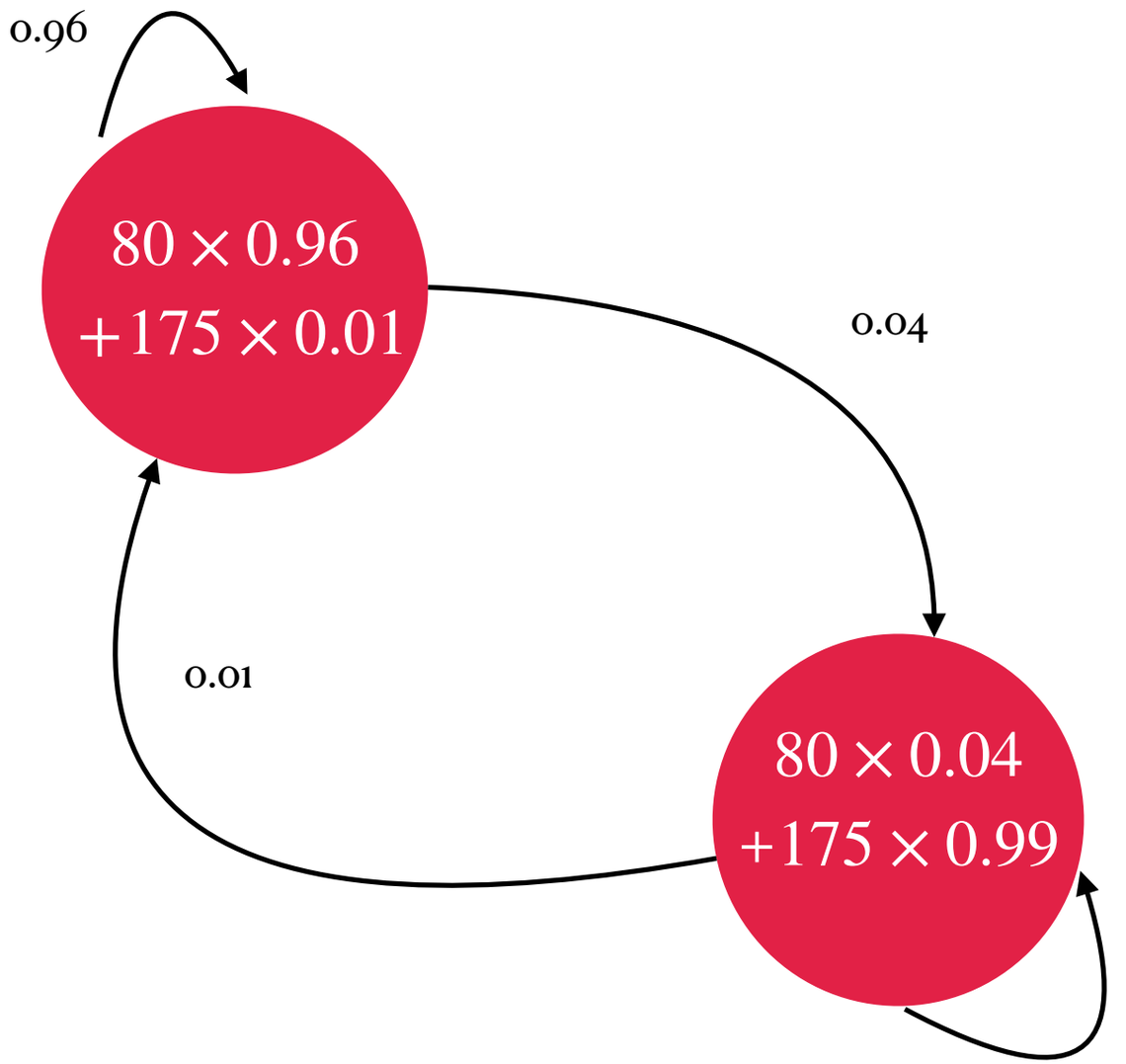
$$\mathbf{x}_0 = \begin{matrix}\text{Initial} \\ \text{populations}\end{matrix} \begin{bmatrix} 80 \\ 175 \end{bmatrix} \begin{matrix}\text{city} \\ \text{suburb}\end{matrix}$$

$$x_0 \to x_1$$


0.96
80
0.04
0.01
175
0.99


0.96
80 × 0.96
+175 × 0.01
0.04
0.01
80 × 0.04
+175 × 0.99
0.99

$$x_0 \rightarrow x_1$$



$$\begin{bmatrix} 0.96 & 0.01 \end{bmatrix} \begin{bmatrix} 80 \\ 175 \end{bmatrix}$$

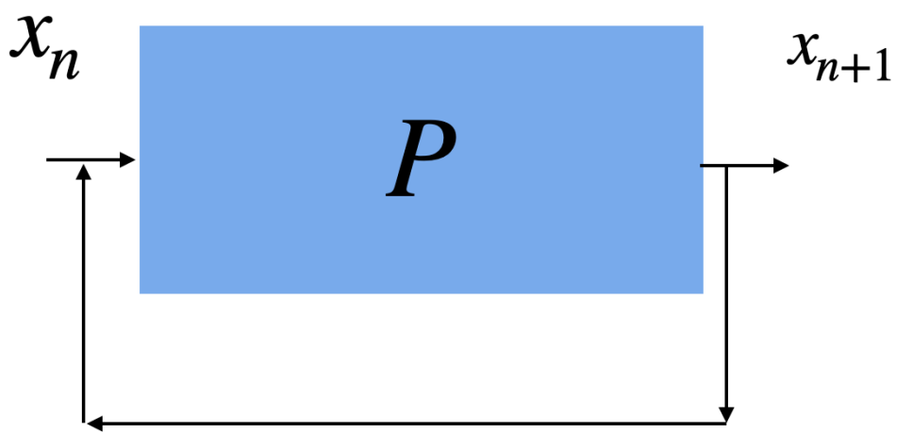
$$\begin{bmatrix} 0.04 & 0.99 \end{bmatrix} \begin{bmatrix} 80 \\ 175 \end{bmatrix}$$

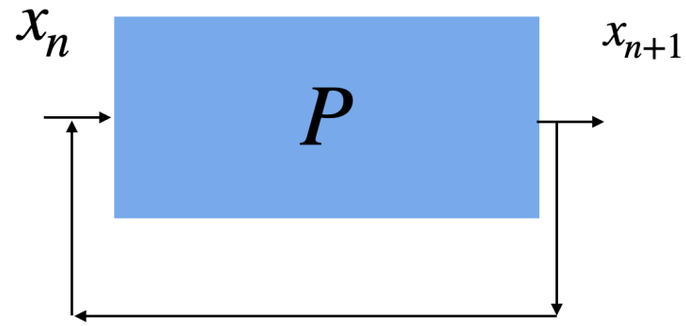
$$\begin{bmatrix} 0.96 & 0.01 \\ 0.04 & 0.99 \end{bmatrix} \begin{bmatrix} 80 \\ 175 \end{bmatrix}$$

$$\begin{bmatrix} 0.96 & 0.01 \\ 0.04 & 0.99 \end{bmatrix} \begin{bmatrix} 80 \\ 175 \end{bmatrix}$$

$$Px_0 \rightarrow x_1$$

$$P = \begin{array}{c} \begin{matrix} \text{(from)} \\ \text{city} \quad \text{suburb} \end{matrix} \quad \text{(to)} \\ \begin{bmatrix} 0.96 & 0.01 \\ 0.04 & 0.99 \end{bmatrix} \begin{matrix} \text{city} \\ \text{suburb} \end{matrix} \end{array}$$





長期預測:是否會收斂？

$$x_0 \to x_1 \to \ldots \to x_n \to x_{n+1} \ldots$$

$$Px_n \to x_{n+1}$$

總會有一個n，使得
x_n和x_{n+1}都變成x

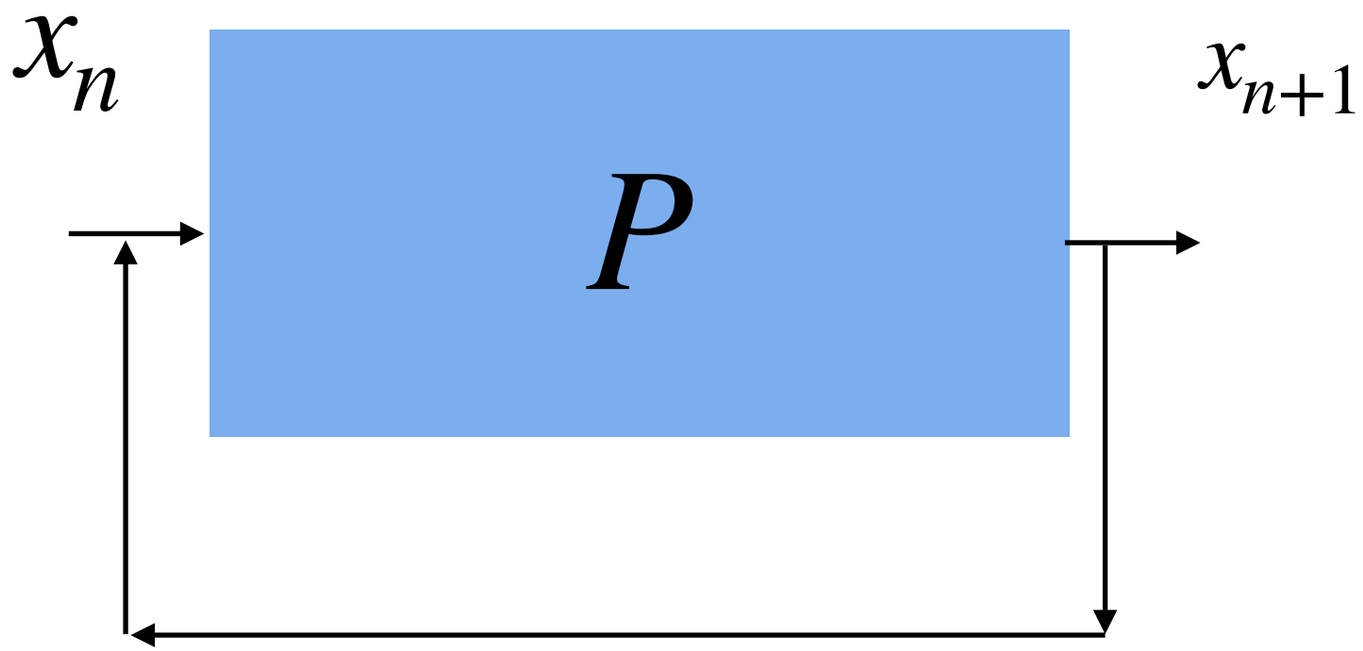
長期預測:如果收斂

$$x_0 \to x_1 \to \ldots \to x \to x$$

$$Px_n \to x_{n+1}$$

$$Px = x$$

Recurrent linear system



最重要的收斂特性

$$Px = x$$

x為P的特徵向量

一般的特徵向量寫法，會有一個特徵值

$$Px = \lambda x$$

$$Px - \lambda x = 0$$

$$Px - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} x = 0$$

$$(P - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix})x = 0$$

x為P的特徵向量

λ為P的特徵值

最重要的收斂特性

$$Px = x$$

x為P的特徵向量，
相對應的特徵值$\lambda = 1$

P矩陣，相對應特徵值$\lambda = 1$
的特徵向量，就代表最後的收斂狀態

$$Px = x$$

$$P = \begin{bmatrix} 0.96 & 0.01 \\ 0.04 & 0.99 \end{bmatrix}$$

Determine eigenvalues of P by solving polynomials

$$\text{Determinant of } \begin{bmatrix} 0.96 - \lambda & 0.01 \\ 0.04 & 0.99 - \lambda \end{bmatrix}$$

$$\text{Determinant of } \begin{bmatrix} 0.96-\lambda & 0.01 \\ 0.04 & 0.99-\lambda \end{bmatrix}$$

$$= (0.96-\lambda)(0.99-\lambda) - 0.01 \times 0.04$$

$$= \lambda^2 - (0.96+0.99)\lambda + 0.96 \times 0.99 - 0.01 \times 0.04$$

```
import numpy as np
coeff = [1, -(0.96+0.99), 0.96*0.99-0.01*0.04]
ans = np.roots(coeff)
print(ans)
```

Find roots of a polynomials

```
>>> import numpy as np
... coeff = [1, -(0.96+0.99), 0.96*0.99-0.01*0.04]
... ans = np.roots(coeff)
... print(ans)
[1.   0.95]
```

The largest eigenvalue is one

計算矩陣特徵值與特徵向量

```
from numpy.linalg import eig
```

$$Px = x$$

[-0.24253563
-0.9701425]

```
from numpy.linalg import eig
P = np.array([[0.96, 0.01],[0.04,0.99]])
value, vector = eig(P)
print(value)
print(vector[:,1])
```

使用eig計算
P矩陣的
特徵值與特徵向量

```
[0.95 1.  ]
[-0.24253563 -0.9701425 ]
```

$$Px = x$$

```
from numpy.linalg import eig
P = np.array([[0.96, 0.01],[0.04,0.99]])
value, vector = eig(P)
print(value)
print(vector[:,1])
```

value代表特徵值，第二個特徵值是1

Vector代表特徵向量，印出第二個特徵值向量

```
[0.95 1.  ]
[-0.24253563 -0.9701425 ]
```

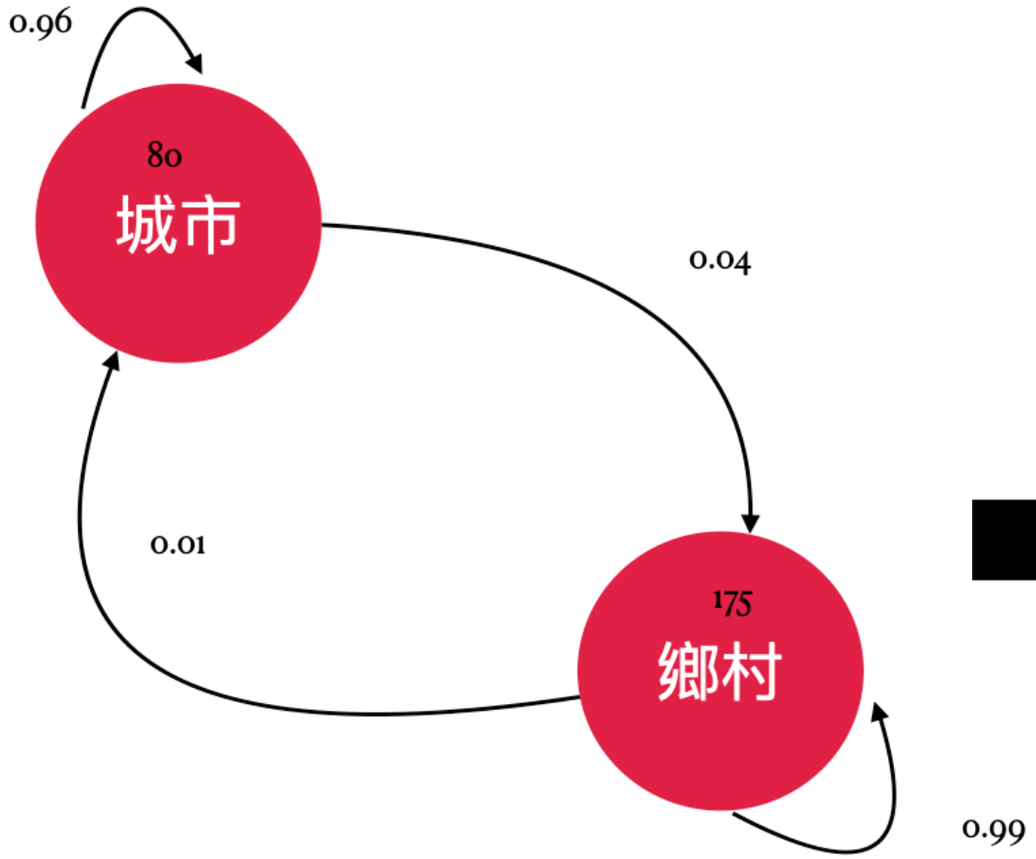
```
a,b = abs(vector[:,1])
x = np.array([[80],[175]])
r = a / (a+b)
print(np.sum(x) * r)
print(np.sum(x) * (1-r))
```

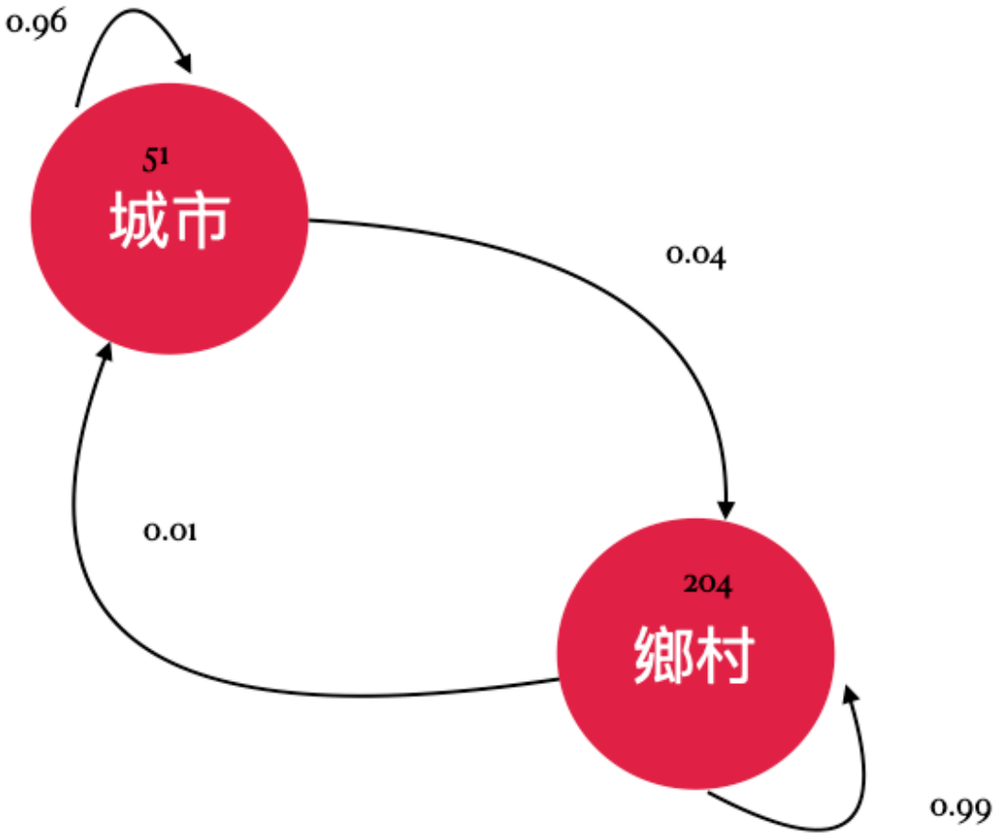
a,b代表第二個特徵向量的兩元素絕對值

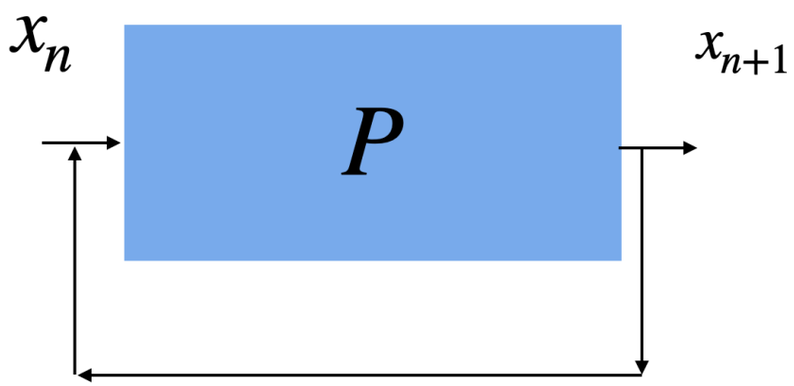
假設總人口不變，r代表收斂後的城市人口比例

```
51.00000000000004
203.99999999999994
```

長期預測的城市人口與鄉村人口


0.96
80
城市
0.04
0.01
175
鄉村
0.99


0.96
51
城市
0.04
0.01
204
鄉村
0.99


x_n
P
x_{n+1}

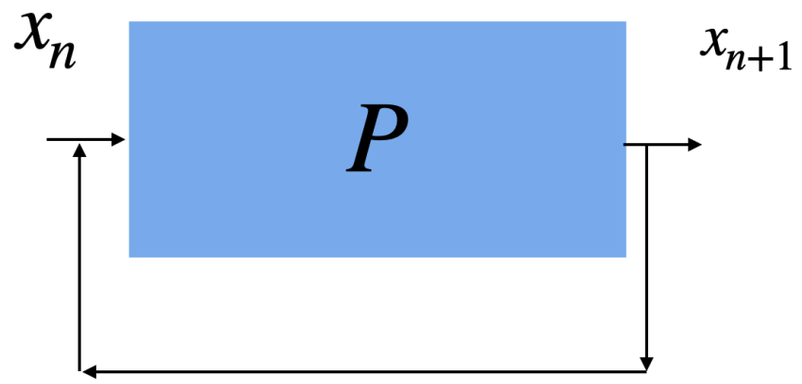
兩個步驟

```
from numpy.linalg import eig
P = np.array([[0.96, 0.01],[0.04,0.99]])
value, vector = eig(P)
print(value)
print(vector[:,1])
```

10.28

```
a,b = abs(vector[:,1])
x = np.array([[80],[175]])
r = a / (a+b)
print(np.sum(x) * r)
print(np.sum(x) * (1-r))
```

Iterative execution of state transition to verify convergence



Start

```
import numpy as np

P = np.array([[0.96, 0.01],[0.04,0.99]])
x = np.array([[80],[175]])
loop = 0
loop_max = 10000
Hc = False
```

while not Hc:

End

```
x_old = x
x = P @ x
change = np.sum(abs(x_old - x))
if change < pow(10,-4) or loop >= loop_max:
    Hc = True
loop = loop + 1
print("loop: ", loop, " change: ", change)
```

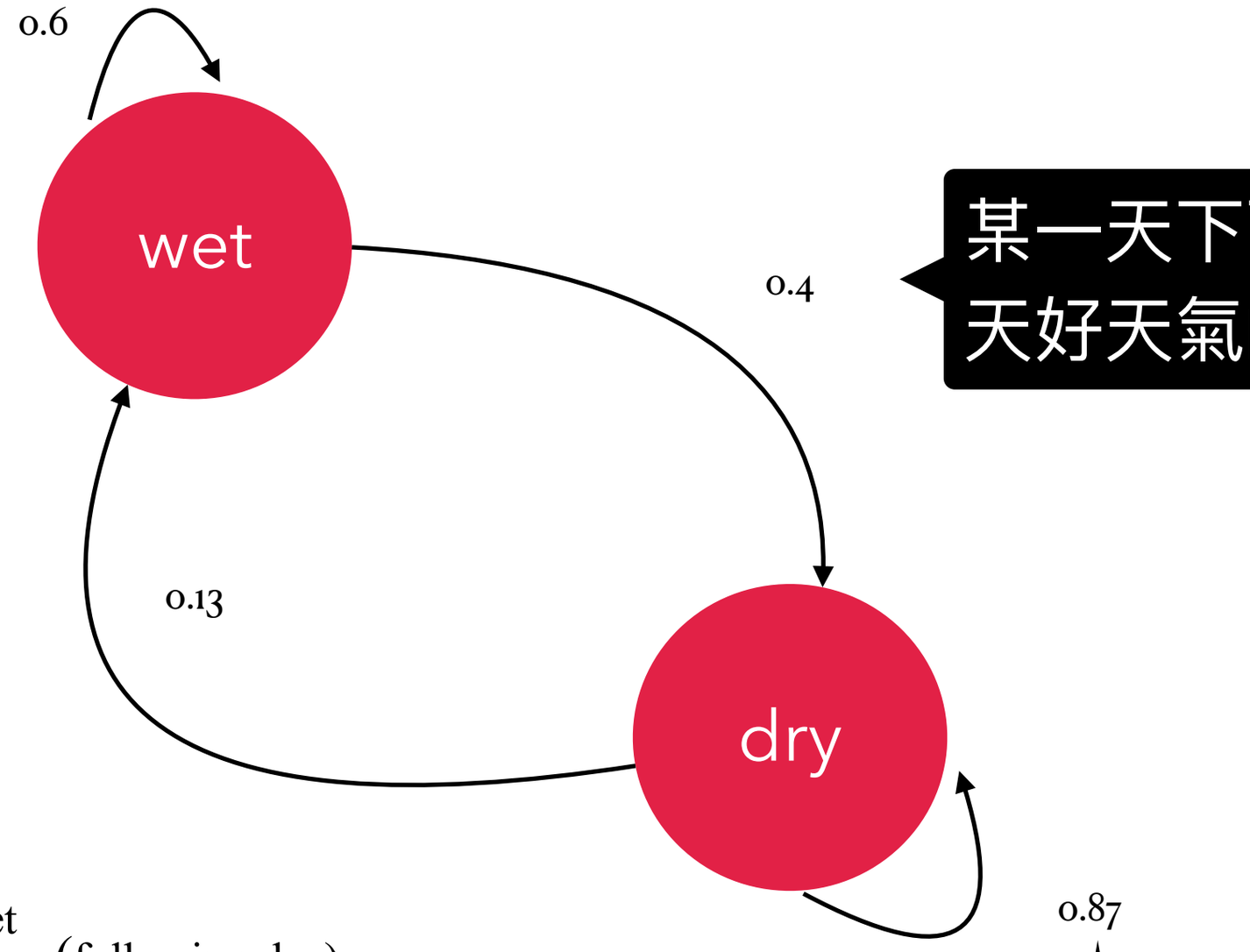
```python
import numpy as np

P = np.array([[0.96, 0.01],[0.04,0.99]])
x = np.array([[80],[175]])
loop = 0
loop_max = 10000
Hc = False
while not Hc:
    x_old = x
    x = P @ x
    change = np.sum(abs(x_old - x))
    if change < pow(10,-4) or loop >= loop_max:
        Hc = True
    loop = loop + 1
    print("loop: ", loop, " change: ", change)

print(round(x[0,0]),round(x[1,0]))
print(x[0,0],x[1,0])
```

```
loop:  200  change:  0.0001070028759073693 4
loop:  201  change:  0.0001016527321127114 2
51 204
```

天氣預測

預測好天氣的機率

馬可夫鏈的移轉機率
Markov chain



某一天下雨，隔天好天氣的機率

某一天好天氣，隔天好天氣的機率

$$P = \begin{array}{c} \text{(A given day)} \\ \text{wet} \quad \text{dry} \end{array} \begin{bmatrix} 0.6 & 0.13 \\ 0.4 & 0.87 \end{bmatrix} \begin{array}{l} \text{wet} \\ \text{dry} \end{array} \text{(following day)}$$

直接進行矩陣乘法模擬收斂

```python
import numpy as np

P = np.array([[0.96, 0.01],[0.04,0.99]])
x = np.array([[80],[175]])
loop = 0
loop_max = 10000
Hc = False
while not Hc:
    x_old = x
    x = P @ x
    change = np.sum(abs(x_old - x))
    if np.sum(abs(x_old-x)) < pow(10,-4) or loop >= loop_max:
        Hc = True
    loop = loop + 1
    print("loop: ", loop, " change: ", change)

print(round(x[0,0]),round(x[1,0]))
```

負鼠不同年齡數量的長期預測

AgeGroups(years)	0-1	1-2	2-3	3-4	4-5
InitialPopulation	194	82	55	22	6
BreedingRate	0	1.3	1.8	0.9	0.2
SurvivalRate	0.6	0.8	0.8	0.4	0

負鼠不同年齡數量的轉移矩陣

$$L=\begin{bmatrix} 0 & 1.3 & 1.8 & 0.9 & 0.2 \\ 0.6 & 0 & 0 & 0 & 0 \\ 0 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0 \end{bmatrix} \qquad X_0=\begin{bmatrix} 194 \\ 82 \\ 55 \\ 22 \\ 6 \end{bmatrix}$$

4-5