Python. Wektory i wartości własne

B.Kowal

2 lutego 2022

Wektory i wartości własne

$$\begin{bmatrix} A_1^1 & \dots & A_1^n \\ \vdots & \ddots & \vdots \\ A_n^1 & \dots & A_n^n \end{bmatrix} \begin{bmatrix} v^1 \\ \vdots \\ v^n \end{bmatrix} = \alpha \begin{bmatrix} v^1 \\ \vdots \\ v^n \end{bmatrix}$$

α - wartość własna związana z wektorem własnym $\begin{bmatrix} v^1 \\ \vdots \\ v^n \end{bmatrix}$

Wektory i wartości własne

```python
import numpy as np
#from numpy import linalg
from scipy import linalg

A = np.array([[1 ,2],[3,2]],dtype=np.float64)
print("macierz A:", '\n', A, '\n')
EigenValues, EigenVectors = linalg.eig(A)

print("wartosci wlasne:")
print(EigenValues,'\n')
print("wektory wlasne (kolumny macierzy):")
print(EigenVectors,'\n')

print("pierwszy wektor własny (pierwsza kolumna):")
print(EigenVectors[:,0],'\n')
#lub print(EigenVectors.transpose()[0],'\n')
print("drugi wektor własny (druga kolumna):")
print(EigenVectors[:,1],'\n')
#lub print(EigenVectors.transpose()[1],'\n')
```

Wektory i wartości własne

```
macierz A:
 [[1. 2.]
 [3. 2.]]

wartosci wlasne:
[-1.+0.j  4.+0.j]

wektory wlasne (kolumny macierzy):
[[-0.70710678 -0.5547002 ]
 [ 0.70710678 -0.83205029]]

pierwszy wektor własny (pierwsza kolumna):
[-0.70710678  0.70710678]

drugi wektor własny (druga kolumna):
[-0.5547002  -0.83205029]
```

Wektory i wartości własne

```
import numpy as np
#from numpy import linalg
from scipy import linalg

A = np.array([[1 ,2],[3,2]],dtype=np.float64)
print("macierz A:", '\n', A, '\n')

EigenValues, EigenVectors = linalg.eig(A)

print("wartosc wlasna a1:", EigenValues[0], '\n', "wektor własny v1:",EigenVectors[:,0])
print("A.v1:",np.dot(A,EigenVectors[:,0]))
print("a1*v1:",EigenValues[0]*EigenVectors[:,0],'\n')

print("wartosc wlasna a2:", EigenValues[1], '\n', "wektor własny v2:",EigenVectors[:,1])
print("A.v2:",np.dot(A,EigenVectors[:,1]))
print("a2*v2:",EigenValues[1]*EigenVectors[:,1])
```

Wektory i wartości własne

```
macierz A:
 [[1. 2.]
 [3. 2.]]

wartosc wlasna a1: (-1+0j)
 wektor własny v1: [-0.70710678  0.70710678]
A.v1: [ 0.70710678 -0.70710678]
a1*v1: [ 0.70710678-0.j -0.70710678+0.j]

wartosc wlasna a2: (4+0j)
 wektor własny v2: [-0.5547002  -0.83205029]
A.v2: [-2.21880078 -3.32820118]
a2*v2: [-2.21880078+0.j -3.32820118+0.j]
```

macierz trójdiagonalna

```python
import numpy as np
size=4
g = 2.0*np.ones(size-1);
d = np.array([3*(i+1) for i in range(0,size)])

print(np.diag(g, -1),'\n')
print(np.diag(d, 0),'\n')
print(np.diag(g, 1),'\n')
print(np.diag(g, -1)+np.diag(d, 0)+np.diag(g, 1))
```

```
[[0. 0. 0. 0.]
 [2. 0. 0. 0.]
 [0. 2. 0. 0.]
 [0. 0. 2. 0.]]

[[ 3  0  0  0]
 [ 0  6  0  0]
 [ 0  0  9  0]
 [ 0  0  0 12]]

[[0. 2. 0. 0.]
 [0. 0. 2. 0.]
 [0. 0. 0. 2.]
 [0. 0. 0. 0.]]

[[ 3.  2.  0.  0.]
 [ 2.  6.  2.  0.]
 [ 0.  2.  9.  2.]
 [ 0.  0.  2. 12.]]
```

sortowanie

```python
import numpy as np

values = np.array([3.0, 2.0, -1.0, 1.0, 5.0])

print(values.argsort())
#indeksy 3 najmniejszych wartosci
minimum_indexes= values.argsort()[0:3]
print(minimum_indexes)

#3 najmniejsze wartosci
print(values[minimum_indexes])
```

```
[2 3 1 0 4]
[2 3 1]
[-1.  1.  2.]
```