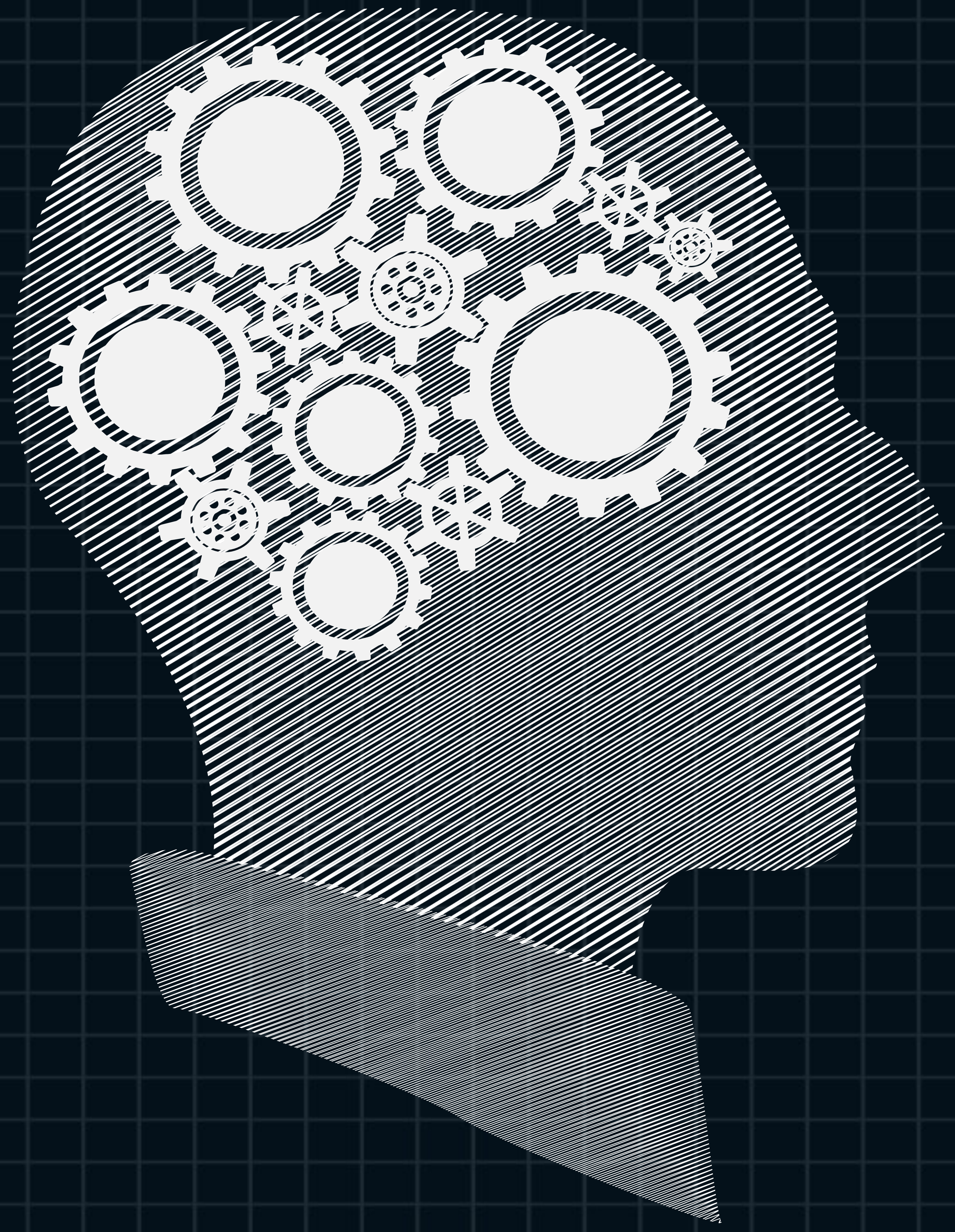
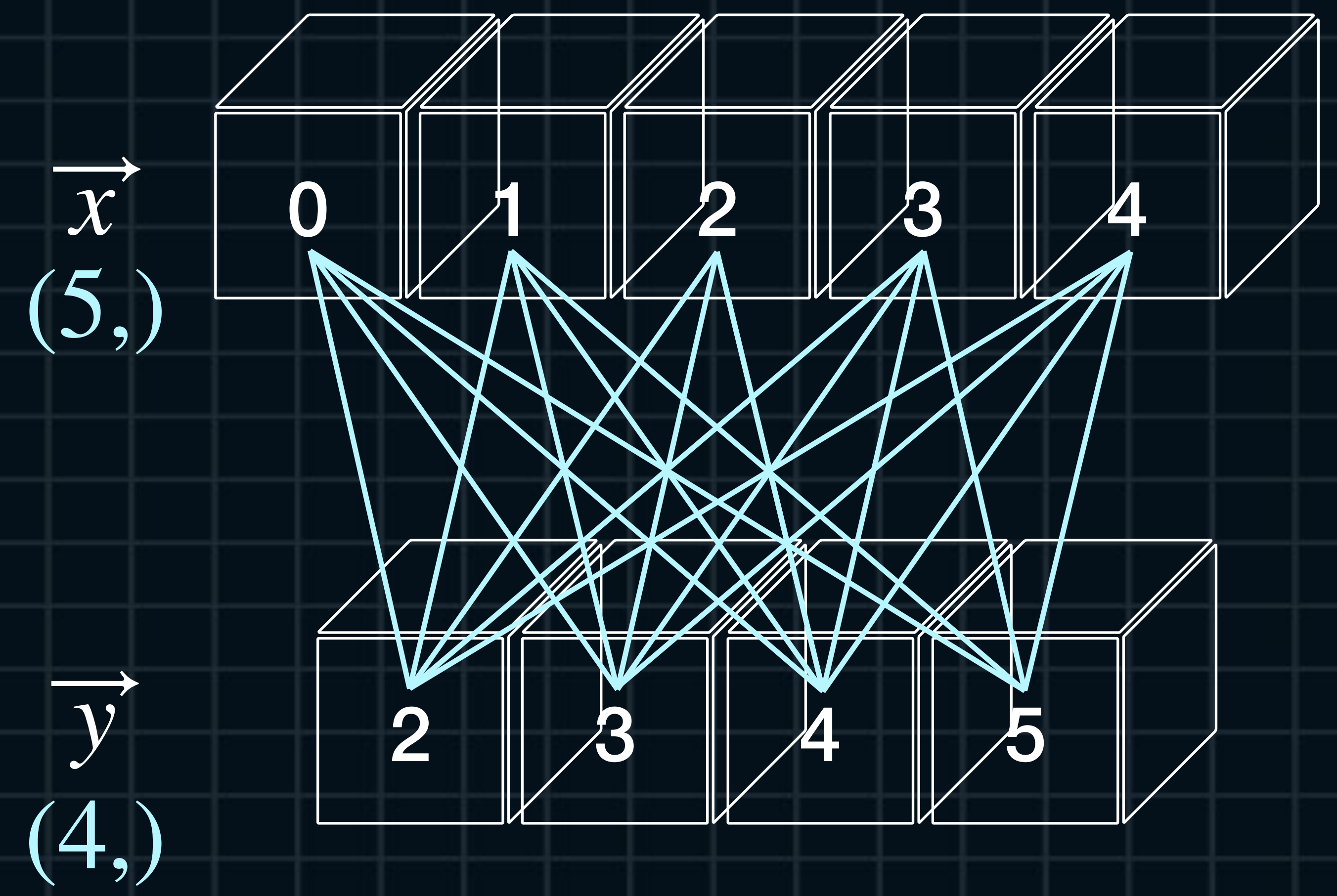


NumPy Master Class

Lecture.16
Tricks for Fully-connected Operations

- Fully-connected Operations

What's Fully-connected Operations?



- Fully-connected Operations

Imp. of FC Operations

```python
import numpy as np

x = np.arange(5)
y = np.arange(2, 6)

print(f"x: {x}")
print(f"y: {y}\n")
```

```
x: [0 1 2 3 4]
y: [2 3 4 5]
```

```python
for x_ in x:
  for y_ in y:
    print(x_ + y_, end=' ')
  print()
```

```
2 3 4 5
3 4 5 6
4 5 6 7
5 6 7 8
6 7 8 9
```

```python
import numpy as np

x = np.arange(5)
y = np.arange(2, 6)

print(f"x: {x}")
print(f"y: {y}\n")
```

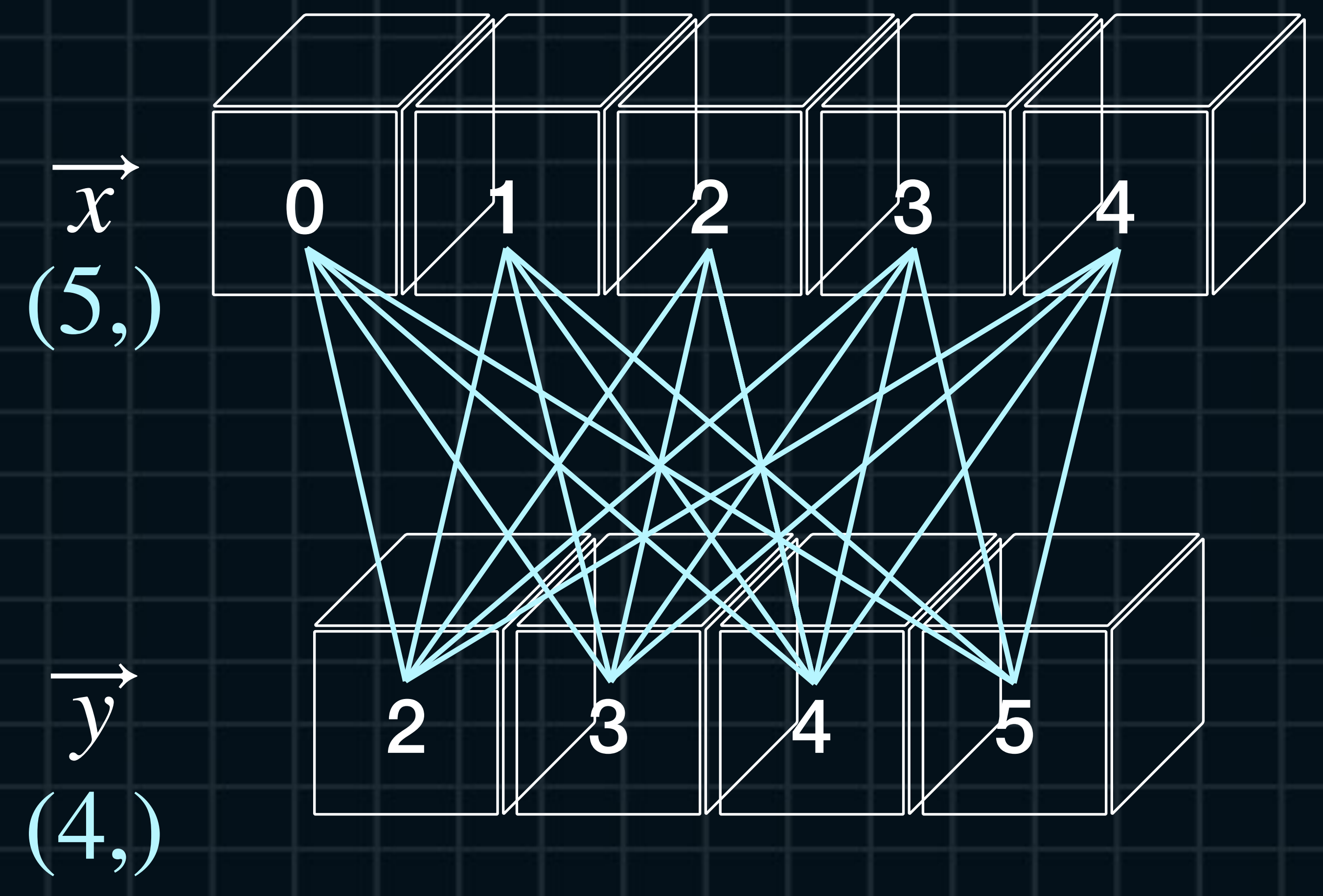
```
x: [0 1 2 3 4]
y: [2 3 4 5]
```

```python
for x_ in x:
  print(x_ + y)
```

```
[2 3 4 5]
[3 4 5 6]
[4 5 6 7]
[5 6 7 8]
[6 7 8 9]
```

- Fully-connected Operations for Scalars

Scalar Case



	y[0] = 2	y[1] = 3	y[2] = 4	y[3] = 5
x[0] = 0	2	3	4	5
x[1] = 1	3	4	5	6
x[2] = 2	4	5	6	7
x[3] = 3	5	6	7	8
x[4] = 4	6	7	8	9

Imp. of Scalar Case with np.meshgrid

```python
import numpy as np

x = np.arange(5)
y = np.arange(2, 6)

X, Y = np.meshgrid(x, y)
Z = X + Y

X, Y, Z = X.T, Y.T, Z.T

print(f"X: \n{X}")
print(f"Y: \n{Y}")
print(f"Z: \n{Z}")
```

```
X:
[[0 0 0 0]
 [1 1 1 1]
 [2 2 2 2]
 [3 3 3 3]
 [4 4 4 4]]
```

```
Y:
[[2 3 4 5]
 [2 3 4 5]
 [2 3 4 5]
 [2 3 4 5]
 [2 3 4 5]]
```

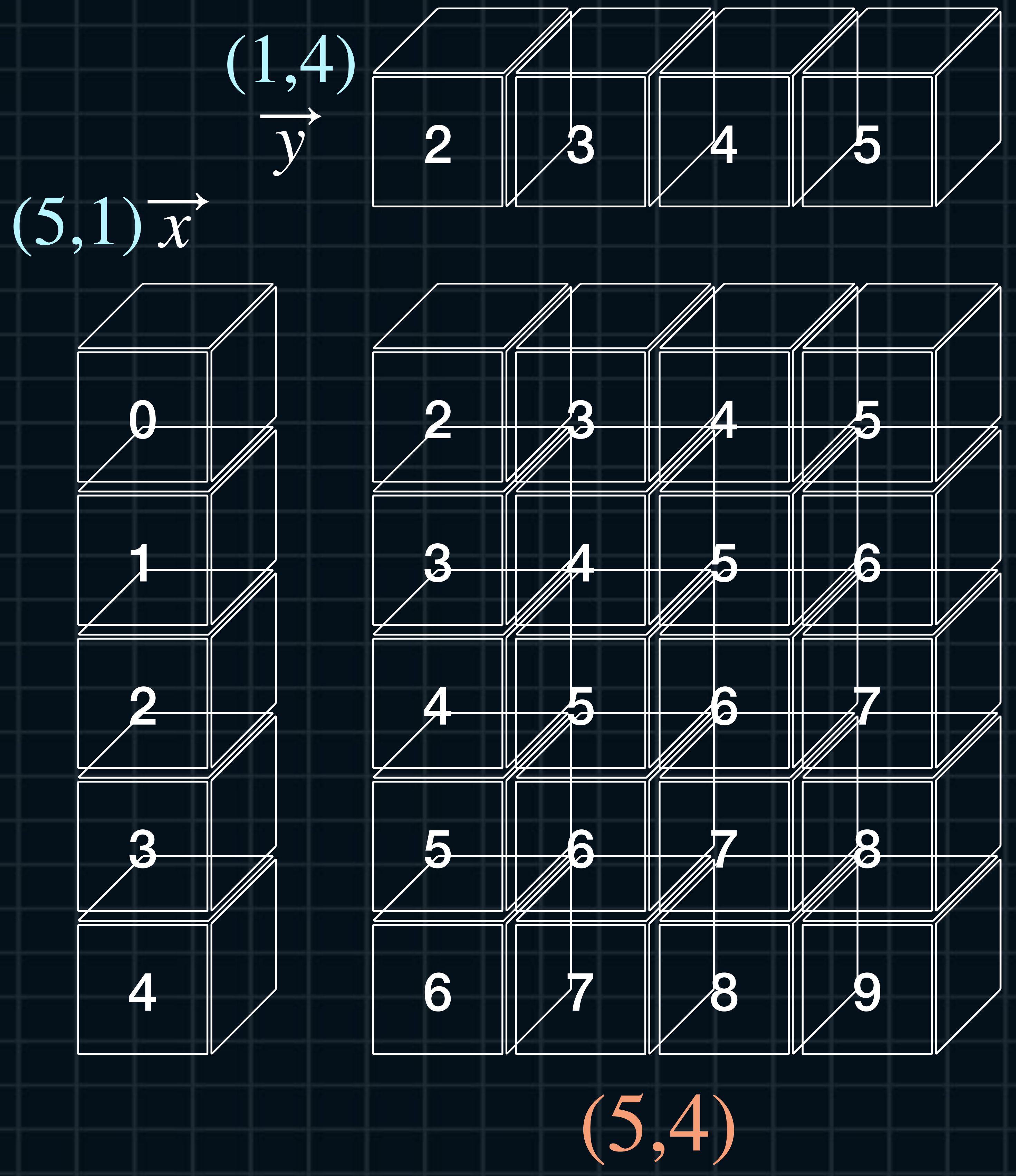
```
Z:
[[2 3 4 5]
 [3 4 5 6]
 [4 5 6 7]
 [5 6 7 8]
 [6 7 8 9]]
```

- Fully-connected Operations for Scalars

Imp. of Scalar Case with Broadcasting

```
import numpy as np

x = np.arange(5)
y = np.arange(2, 6)

X = x.reshape((-1, 1))
Y = y.reshape((1, -1))
Z = X + Y

print(f"X: {X.shape}\n{X}")
print(f"Y: {Y.shape}\n{Y}")
print(f"Z: {Z.shape}\n{Z}")
```

```
X: (5, 1)       Y: (1, 4)       Z: (5, 4)
[[0]            [[2 3 4 5]]     [[2 3 4 5]
 [1]                             [3 4 5 6]
 [2]                             [4 5 6 7]
 [3]                             [5 6 7 8]
 [4]]                            [6 7 8 9]]
```

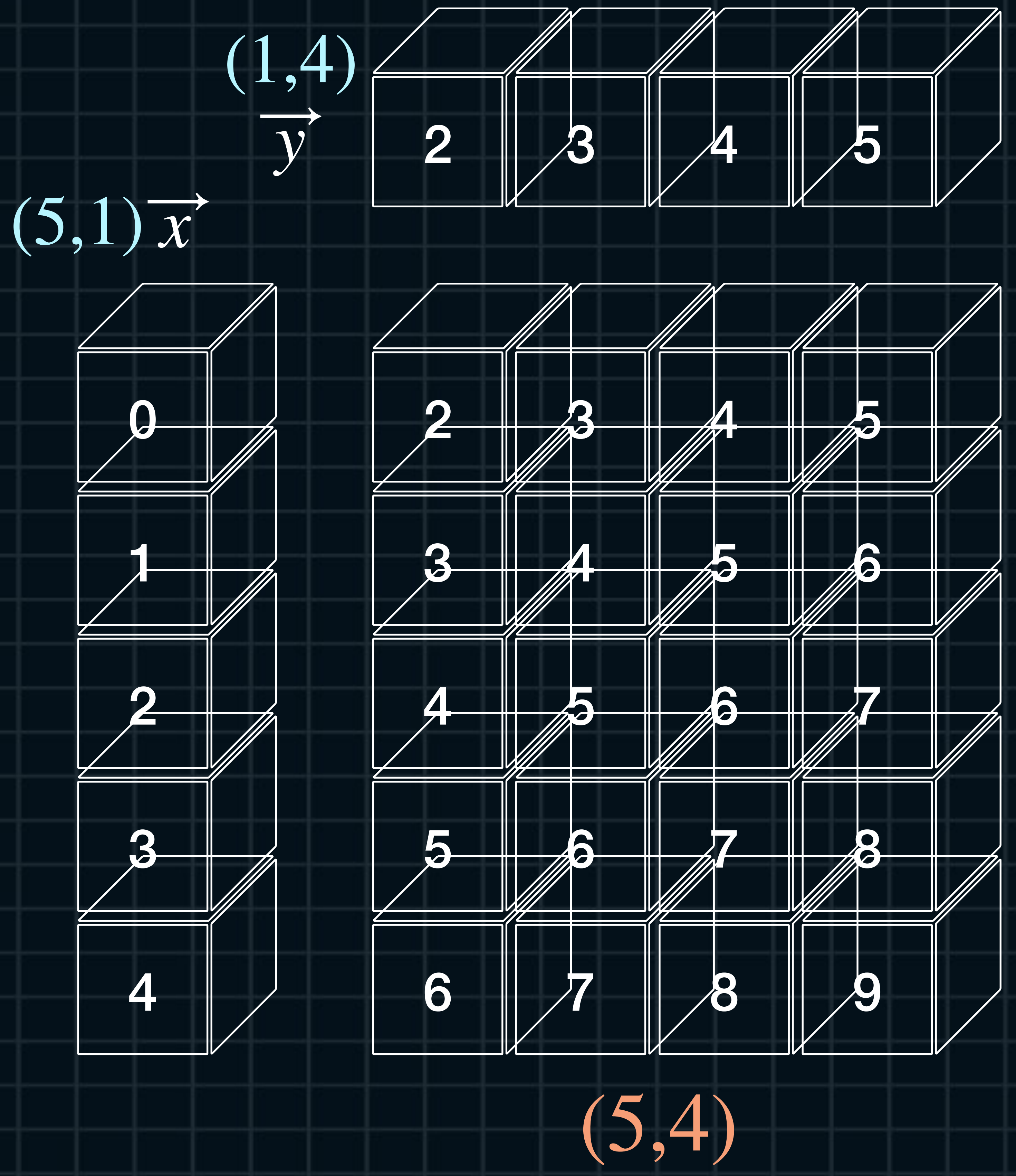


- Fully-connected Operations for Scalars

Imp. of Scalar Case with Broadcasting

```python
import numpy as np

x = np.arange(5)
y = np.arange(2, 6)

X = x.reshape((-1, 1))
Y = y.reshape((1, -1))
Z = X + Y

print(f"x[0] + y: {Z[0, :]}")
print(f"x[3] + y: {Z[3, :]}\n")


print(f"y[0] + x: {Z[:, 0]}")
print(f"y[2] + x: {Z[:, 2]}")
```

```
x[0] + y: [2 3 4 5]
x[3] + y: [5 6 7 8]

y[0] + x: [2 3 4 5 6]
y[2] + x: [4 5 6 7 8]
```



- Fully-connected Operations for Vectors

Imp. of Vector Case with For Loop

```
import numpy as np

X = np.random.uniform(-5, 5, (4, 2))
Y = np.random.uniform(-5, 5, (3, 2))

for x in X:
  for y in Y:
    add = x + y
    print(f"{add}", end='   ')
  print()
```

```
[ 5.45 -2.85]  [ 1.56 -2.27]  [1.42 6.37]
[ 7.22 -8.44]  [ 3.33 -7.87]  [3.19 0.77]
[ 3.57 -3.6 ]  [-0.32 -3.03]  [-0.45  5.61]
[ 1.01 -7.72]  [-2.88 -7.15]  [-3.01  1.49]
```

- Fully-connected Operations for Vectors

Imp. of Vector Case with Broadcasting

```python
import numpy as np

X = np.random.uniform(-5, 5, (4, 2))
Y = np.random.uniform(-5, 5, (3, 2))

X = np.expand_dims(X, axis=1)
Y = np.expand_dims(Y, axis=0)
Z = X + Y

print("shapes: ")
print(f"X/Y/Z: {X.shape}/{Y.shape}/{Z.shape}\n")
```

```
shapes:
X/Y/Z: (4, 1, 2)/(1, 3, 2)/(4, 3, 2)
```

- Fully-connected Operations for Vectors

Imp. of Vector Case with Broadcasting

$$(4,2) \quad X = \big((x_1, y_1) \quad (x_2, y_2) \quad (x_3, y_3) \quad (x_4, y_4)\big)$$

$$(3,2) \quad U = \big((u_1, v_1) \quad (u_2, v_2) \quad (u_3, v_3)\big)$$

$$(1,3,2) \quad U = \big[(u_1, v_1) \qquad (u_2, v_2) \qquad (u_3, v_3)\big]$$

$$\underset{(4,1,2)}{X =} \begin{bmatrix} (x_1, y_1) \\ (x_2, y_2) \\ (x_3, y_3) \\ (x_4, y_4) \end{bmatrix} \qquad X + U = \begin{bmatrix} (x_1 + u_1, y_1 + v_1) & (x_1 + u_2, y_1 + v_2) & (x_1 + u_3, y_1 + v_3) \\ (x_2 + u_1, y_2 + v_1) & (x_2 + u_2, y_2 + v_2) & (x_2 + u_3, y_2 + v_3) \\ (x_3 + u_1, y_3 + v_1) & (x_3 + u_2, y_3 + v_2) & (x_3 + u_3, y_3 + v_3) \\ (x_4 + u_1, y_4 + v_1) & (x_4 + u_2, y_4 + v_2) & (x_4 + u_3, y_4 + v_3) \end{bmatrix}$$

$$(4,3,2)$$

Lecture.16 Tricks for Fully-connected Operations - Fully-connected Operations for Vectors

Imp. of Vector Case with Broadcasting

```python
import numpy as np
np.set_printoptions(sign='+')

X = np.random.uniform(-5, 5, (4, 2))
Y = np.random.uniform(-5, 5, (3, 2))

X = np.expand_dims(X, axis=1)
Y = np.expand_dims(Y, axis=0)
Z = X + Y
```

```python
print(f"X[0] + Y: \n{Z[0, :, :]}")
```

```
X[0] + Y:
[[+6.38 -3.86]
 [+4.65 -0.02]
 [+5.35 -0.77]]
```

```python
print(f"X[3] + Y: \n{Z[3, :, :]}\n")
```

```
X[3] + Y:
[[+4.15 -1.68]
 [+2.42 +2.16]
 [+3.12 +1.41]]
```

```python
print(f"Y[0] + X: \n{Z[:, 0, :]}")
```

```
Y[0] + X:
[[+6.38 -3.86]
 [+5.21 +0.84]
 [+0.25 -1.4 ]
 [+4.15 -1.68]]
```

```python
print(f"Y[2] + X: \n{Z[:, 2, :]}")
```

```
Y[2] + X:
[[+5.35 -0.77]
 [+4.18 +3.93]
 [-0.77 +1.69]
 [+3.12 +1.41]]
```

Fully-connected Operations for Euclidean Distances

$$(1,3,2) \quad U = \begin{bmatrix}(u_1, v_1) & (u_2, v_2) & (u_3, v_3)\end{bmatrix}$$

$$\underset{(4,1,2)}{X} = \begin{bmatrix}(x_1, y_1) \\ (x_2, y_2) \\ (x_3, y_3) \\ (x_4, y_4)\end{bmatrix} \quad X + U = \begin{bmatrix}
\sqrt{(x_1 - u_1)^2 + (y_1 - v_1)^2} & \sqrt{(x_1 - u_2)^2 + (y_1 - v_2)^2} & \sqrt{(x_1 - u_3)^2 + (y_1 - v_3)^2} \\
\sqrt{(x_2 - u_1)^2 + (y_2 - v_1)^2} & \sqrt{(x_2 - u_2)^2 + (y_2 - v_2)^2} & \sqrt{(x_2 - u_3)^2 + (y_2 - v_3)^2} \\
\sqrt{(x_3 - u_1)^2 + (y_3 - v_1)^2} & \sqrt{(x_3 - u_2)^2 + (y_3 - v_2)^2} & \sqrt{(x_3 - u_3)^2 + (y_3 - v_3)^2} \\
\sqrt{(x_4 - u_1)^2 + (y_4 - v_1)^2} & \sqrt{(x_4 - u_2)^2 + (y_4 - v_2)^2} & \sqrt{(x_4 - u_3)^2 + (y_4 - v_3)^2}
\end{bmatrix}$$

$$(4,3)$$

Fully-connected Operations for Euclidean Distances

$$(1,3,2) \quad U = \begin{bmatrix}(u_1, v_1) & (u_2, v_2) & (u_3, v_3)\end{bmatrix}$$

$$\underset{(4,1,2)}{X} = \begin{bmatrix}(x_1, y_1) \\ (x_2, y_2) \\ (x_3, y_3) \\ (x_4, y_4)\end{bmatrix} \qquad X + U = \underset{(4,3)}{\begin{bmatrix} L_2(\vec{x_1}, \vec{u_1}) & L_2(\vec{x_1}, \vec{u_2}) & L_2(\vec{x_1}, \vec{u_2}) \\ L_2(\vec{x_2}, \vec{u_1}) & L_2(\vec{x_2}, \vec{u_2}) & L_2(\vec{x_2}, \vec{u_2}) \\ L_2(\vec{x_3}, \vec{u_1}) & L_2(\vec{x_3}, \vec{u_2}) & L_2(\vec{x_3}, \vec{u_2}) \\ L_2(\vec{x_4}, \vec{u_1}) & L_2(\vec{x_4}, \vec{u_2}) & L_2(\vec{x_4}, \vec{u_2}) \end{bmatrix}}$$

Fully-connected Operations for Euclidean Distances

```python
import numpy as np

X = np.random.uniform(-5, 5, (4, 2))
Y = np.random.uniform(-5, 5, (3, 2))

for x in X:
  for y in Y:
    e_dist = np.sqrt(np.sum(np.square(x - y)))
    print(f"{e_dist:5.2f}", end='   ')
  print()
```

```
 8.79    3.61    6.37
 9.26    3.78    7.13
 4.94    4.17    5.63
 5.16    6.44    3.15
```

- Fully-connected Operations for Vectors

Fully-connected Operations for Euclidean Distances

```python
import numpy as np

X = np.random.uniform(-5, 5, (4, 2))
Y = np.random.uniform(-5, 5, (3, 2))

X = np.expand_dims(X, axis=1)
Y = np.expand_dims(Y, axis=0)

Z = np.sqrt(np.sum(np.square(X - Y), axis=-1))
print(Z)
```

```
[[1.32 1.45 4.89]
 [3.74 2.59 2.08]
 [8.58 7.81 4.04]
 [1.97 3.23 2.83]]
```

- Fully-connected Operations for Vectors

Fully-connected Operations for Euclidean Distances

$$L_2 = \sqrt{(x_1 - x_2)^2 + (y_1 + y_2)^2}$$

$$L_2 = \sqrt{(x_1 - x_2)^2 + (y_1 + y_2)^2 + (z_1 - z_2)^2}$$

- Fully-connected Operations for Vectors

Fully-connected Operations for Euclidean Distances

$$(\alpha,3) \quad X = \big((x_1, y_1, z_1) \quad (x_2, y_2, z_2) \quad \dots \quad (x_\alpha, y_\alpha, z_\alpha)\big)$$

$$(\beta,3) \quad U = \big((u_1, v_1, w_1) \quad (u_2, v_2, w_2) \quad \dots \quad (u_\beta, v_\beta, v_\beta)\big)$$

$$(1,\beta,3) \;\; U = \big[(u_1, v_1, z_1) \quad (u_2, v_2, z_2) \quad \dots \quad (u_\beta, v_\beta, z_\beta)\big]$$

$$\underset{(\alpha,1,3)}{X} = \begin{bmatrix} (x_1, y_1, z_1) \\ (x_2, y_2, z_2) \\ \vdots \\ (x_\alpha, y_\alpha, z_\alpha) \end{bmatrix} \quad X + U = \underset{(\alpha,\beta)}{\begin{bmatrix} L_2(\vec{x_1}, \vec{u_1}) & L_2(\vec{x_1}, \vec{u_2}) & \dots & L_2(\vec{x_1}, \vec{u_\beta}) \\ L_2(\vec{x_2}, \vec{u_1}) & L_2(\vec{x_2}, \vec{u_2}) & \dots & L_2(\vec{x_2}, \vec{u_\beta}) \\ \vdots & \vdots & \ddots & \vdots \\ L_2(\vec{x_\alpha}, \vec{u_1}) & L_2(\vec{x_\alpha}, \vec{u_2}) & \dots & L_2(\vec{x_\alpha}, \vec{u_\beta}) \end{bmatrix}}$$

- Fully-connected Operations of n ndarrays

Scalar Case

$$\begin{array}{ll} \overrightarrow{x} & (\alpha,) \\ \overrightarrow{y} & (\beta,) \\ \vec{z} & (\gamma,) \end{array} \longrightarrow \begin{array}{ll} \overrightarrow{x} & (\alpha,1,1) \\ \overrightarrow{y} & (1,\beta,1) \\ \vec{z} & (1,1,\gamma) \end{array} \longrightarrow \begin{array}{c} \overrightarrow{x} + \overrightarrow{y} + \vec{z} \\ (\alpha, \beta, \gamma) \end{array}$$

- Fully-connected Operations of n ndarrays

Vector Case

$$\begin{matrix} X & (\alpha,2) \\ Y & (\beta,2) \\ Z & (\gamma,2) \end{matrix} \longrightarrow \begin{matrix} X & (\alpha,1,1,2) \\ Y & (1,\beta,1,2) \\ Z & (1,1,\gamma,2) \end{matrix} \longrightarrow \begin{matrix} X+Y+Z \\ (\alpha,\beta,\gamma,2) \end{matrix}$$

NumPy Master Class

Lecture.16
Tricks for Fully-connected Operations