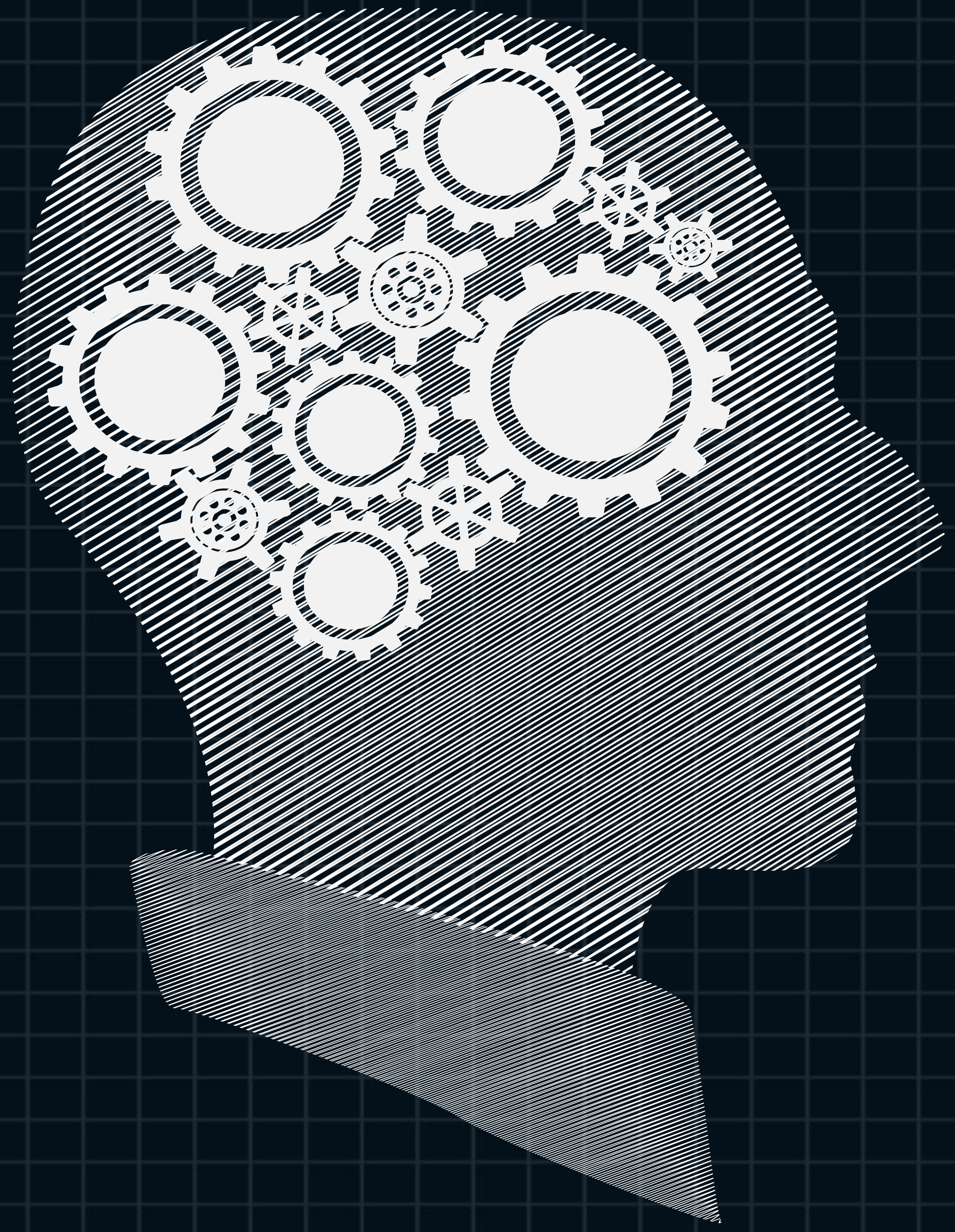

NumPy Master Class

Lecture.12
Matrix Operations

Dot Product

$$\vec{u} \cdot \vec{v} = \sum_{i=1}^{n} u_i v_i$$

`numpy.dot(a, b)`

```python
import numpy as np

u = np.random.uniform(0, 5, (4, ))
v = np.random.uniform(0, 5, (4, ))

sum_hadamard = np.sum(u*v)
np_dot = np.dot(u, v)

print(f"sum_hadamard: {sum_hadamard.round(2)}")
print(f"np_dot: {np_dot.round(2)}")
```

```
sum_hadamard: 38.67
np_dot: 38.67
```

- Vector Operations

Operations of a Artificial Neuron

```
import numpy as np

x = np.random.uniform(0, 5, (4, ))

w = np.random.uniform(0, 5, (4, ))
b = np.random.uniform(0, 5, ())

affine = np.dot(x, w) + b
activation = 1/(1 + np.exp(-affine))
```

- Matrix Operations

Matrix-vector Multiplication

$$M \cdot \overrightarrow{u}[i] = R_i \cdot \overrightarrow{u}$$

```python
import numpy as np

M = np.random.uniform(0, 5, (3, 4))
u = np.random.uniform(0, 5, (4, ))

mat_vec_mul = np.empty((3, ))
for row_idx, row in enumerate(M):
  mat_vec_mul[row_idx] = np.dot(row, u)

np_matmul = np.matmul(M, u)

print(f"mat_vec_mul: \n {mat_vec_mul.round(2)}")
print(f"np_matmul: \n {np_matmul.round(2)}")
```

```
mat_vec_mul:
 [ 7.12 17.09 16.17]
np_matmul:
 [ 7.12 17.09 16.17]
```

- Matrix Operations

Matrix-matrix Multiplication

$$M \cdot N[i, j] = R_i \cdot C_j$$

`numpy.matmul`(*x1*, *x2*)

```python
import numpy as np

M = np.random.uniform(0, 5, (3, 4))
N = np.random.uniform(0, 5, (4, 5))

mat_mat_mul = np.empty((3, 5))
for M_row_idx in range(3):
  for N_col_idx in range(5):
    dot_prod = np.dot(M[M_row_idx, :], N[:, N_col_idx])
    mat_mat_mul[M_row_idx, N_col_idx] = dot_prod

np_matmul = np.matmul(M, N)

print(f"mat_mat_mul: \n {mat_mat_mul.round()}")
print(f"np_matmul: \n {np_matmul.round()}")
```

```
mat_mat_mul:
 [[34. 33. 37. 32. 27.]
 [53. 53. 51. 48. 38.]
 [45. 47. 46. 41. 36.]]
np_matmul:
 [[34. 33. 37. 32. 27.]
 [53. 53. 51. 48. 38.]
 [45. 47. 46. 41. 36.]]
```

- Matrix Operations

Operations of Artificial Neurons

$$Z = X \cdot W + \overrightarrow{b}$$

$$A = g(X \cdot W + \overrightarrow{b})$$

```
import numpy as np

X = np.random.uniform(0, 5, (3, 4))

W = np.random.uniform(0, 5, (4, 5))
b = np.random.uniform(0, 5, (5, ))

affine = np.matmul(X, W) + b
activation = 1/(1 + np.exp(-affine))
```

NumPy Master Class

Lecture.12
Matrix Operations