KWANGWOON UNIVERSITY

Artificial Neural Network

2020.11.17

Seoungjun Oh (sjoh@kw.ac.kr)
Junyoung Sung (jysung13mmlab@kw.ac.kr)
Gisu Hwang (kisu031@kw.ac.kr)

Multimedia LAB

VIA-Multimedia Center, Kwangwoon University

Contents

- Introduction
- Deep Learning
- Pytorch

Introduction

* 출처 : I-Korea 4.0 실현을 위한 인공지능 R&D 전략 (과기정통부, 2018.5)
* 본 그림은 2018.5 현재까지의 인공지능분야 주요 결과물과 향후 2030년까지의 주요 기술동향을 함께 표시함

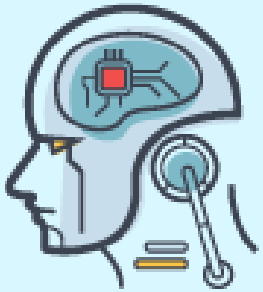
Introduction

❖ Artificial Intelligence

Artificial Intelligence

인공지능

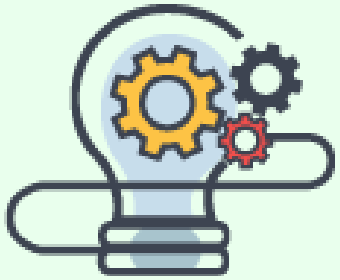
사고나 학습등 인간이 가진
지적 능력을 컴퓨터를 통해
구현하는 기술



Machine Learning

머신러닝

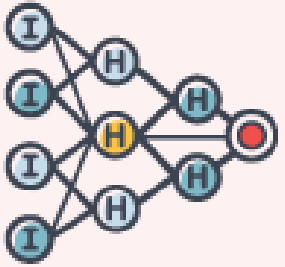
컴퓨터가 스스로 학습하여
인공지능의 성능을
향상 시키는 기술 방법



Deep Learning

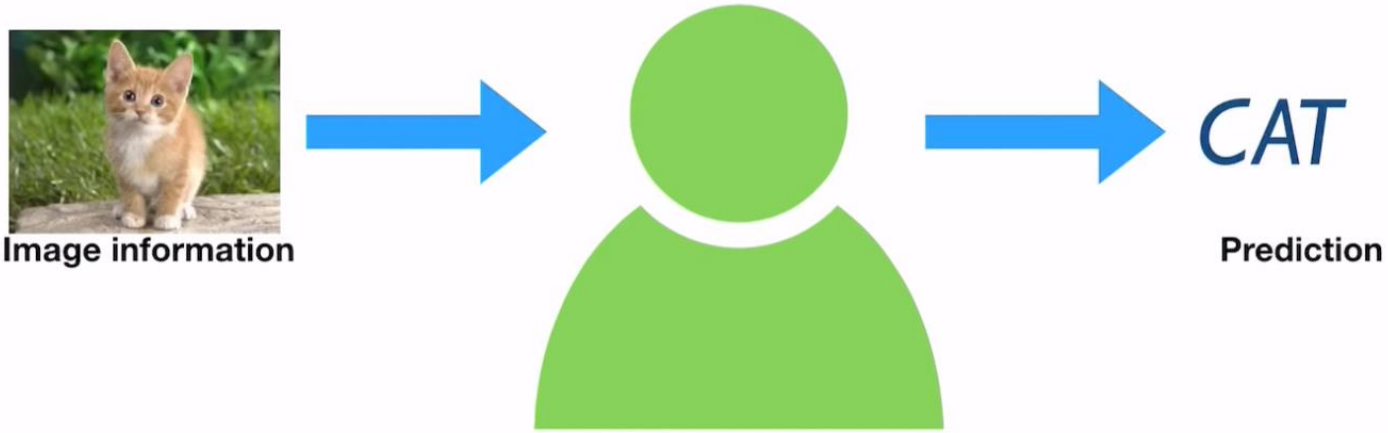
딥러닝

인간의 뉴런과 비슷한
인공신경망 방식으로
정보를 처리



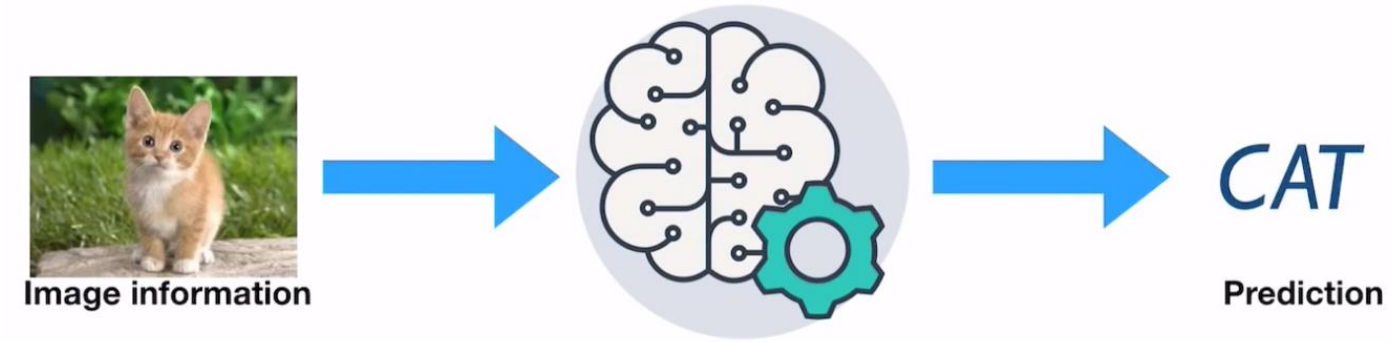
Introduction

- Human Intelligence



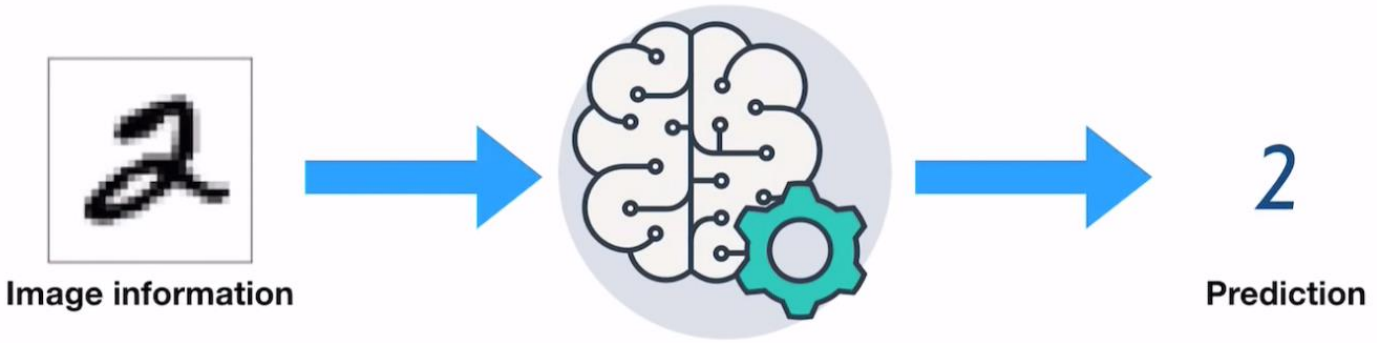
Introduction

❖ Artificial Intelligence



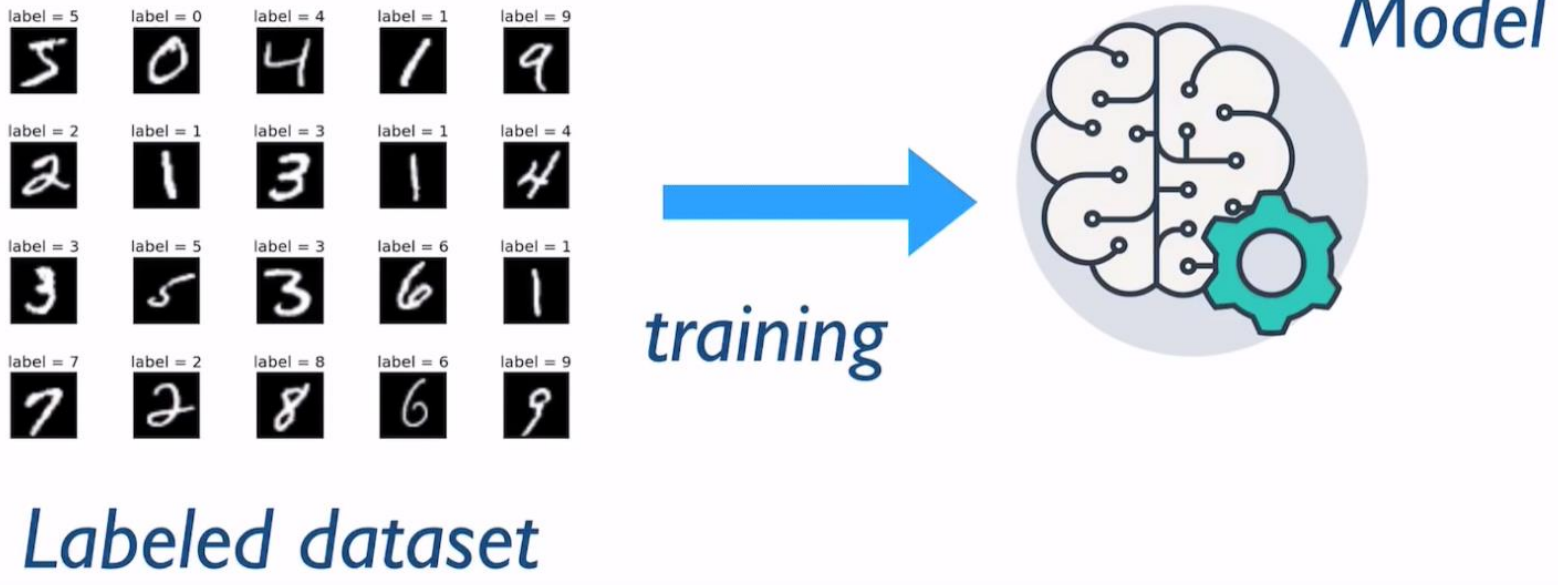
Introduction

❖ Artificial Intelligence



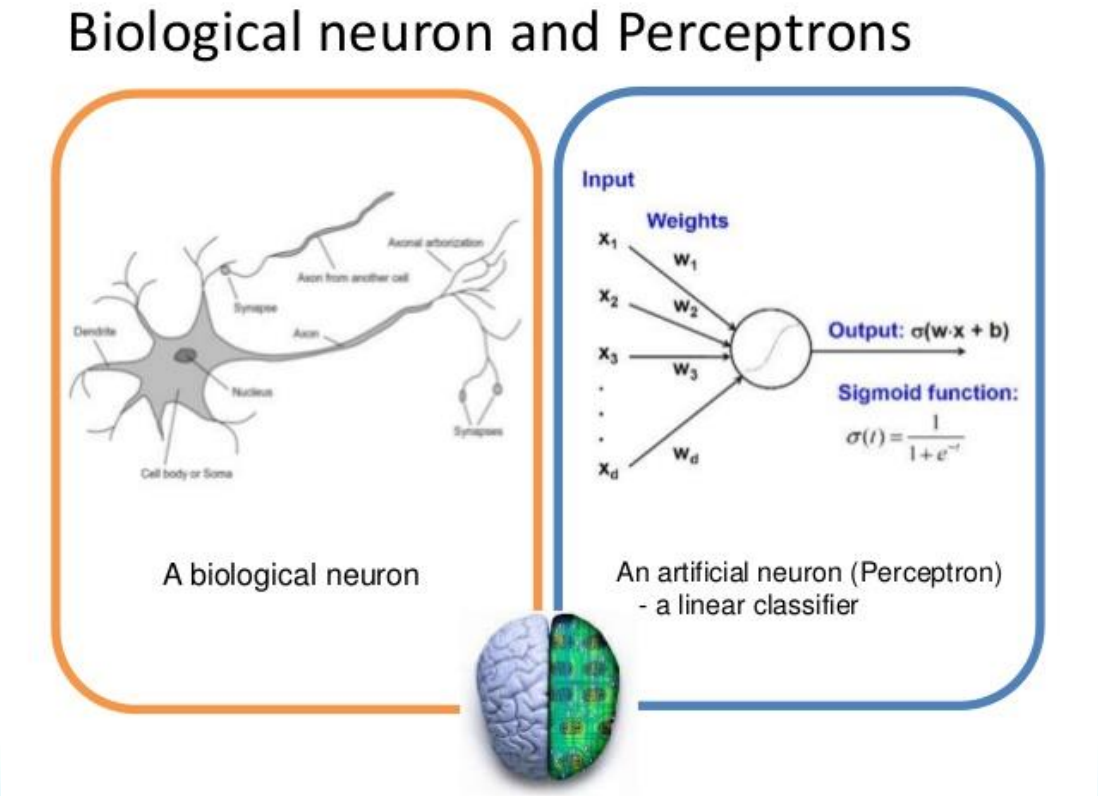
Introduction

- Artificial Intelligence



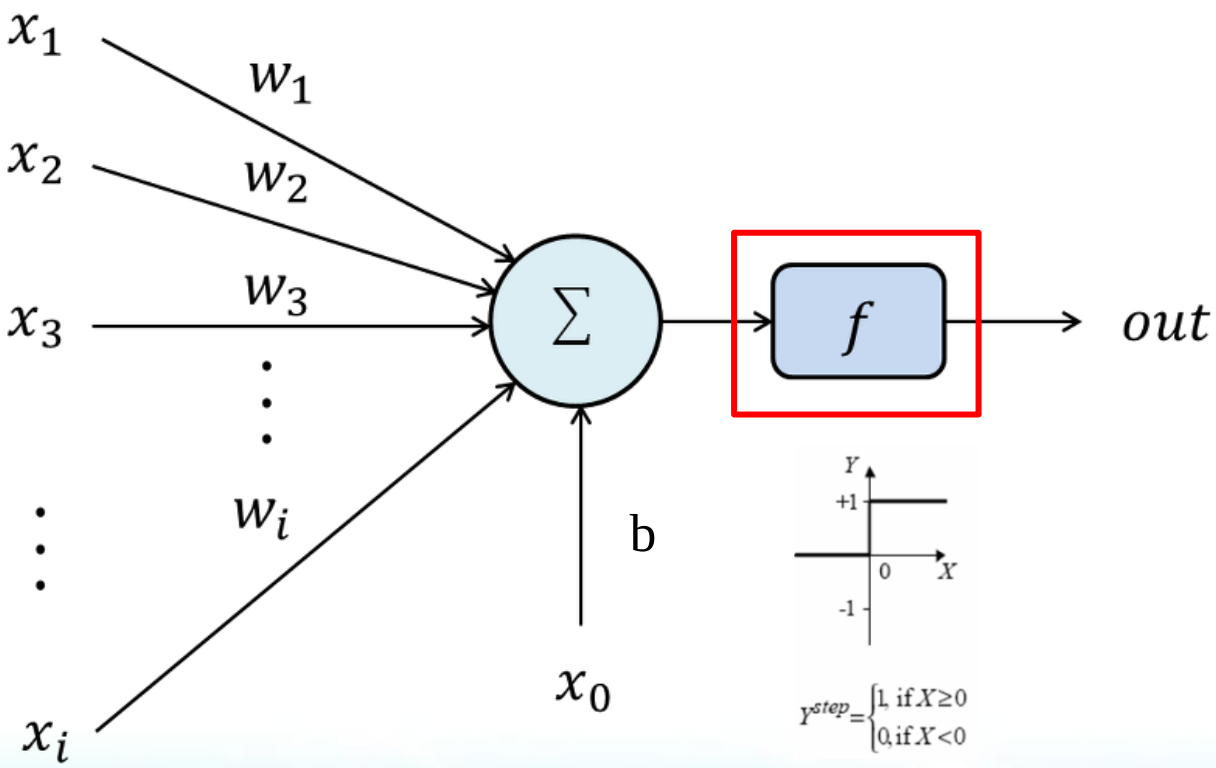
Deep Learning

- Perceptron



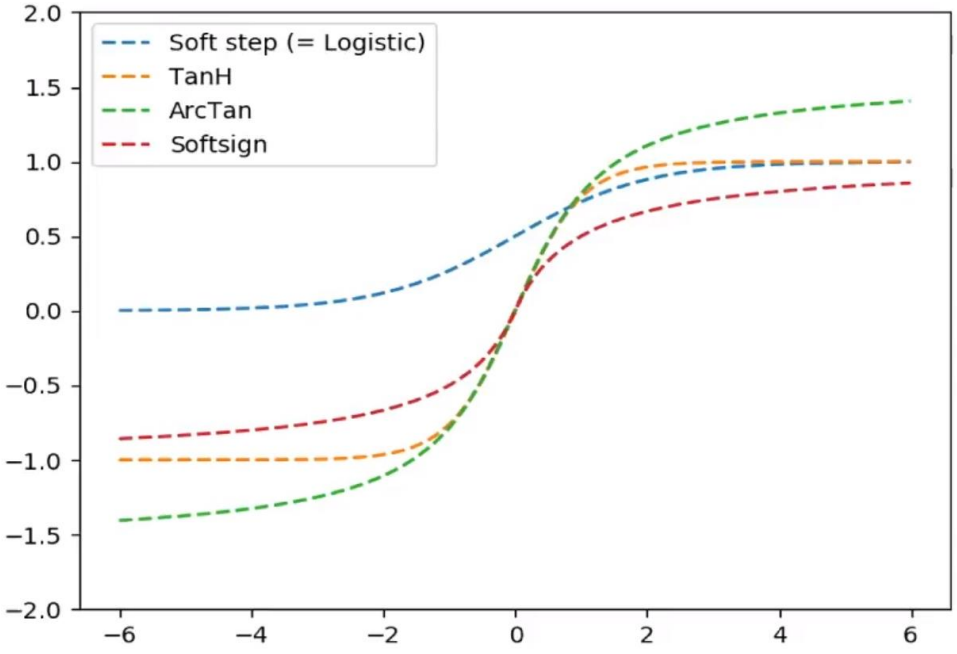
Deep Learning

- Perceptron



Deep Learning

❖ Activation Function



Logistic Function

$$y = k \ / \ 1 + me^{-x} \ (k,m \text{ are constant})$$

TanH(Hyperbolic Tangent) Function

$$sinhx \ / coshx = e^{x} + e^{-x} \ / \ 2$$

ArcTan(Arc Tangent) Function

$$y = tan^{-1}x = arctan\, x$$

Soft sign Function

$$y = x \ / \ 1 + |x|$$

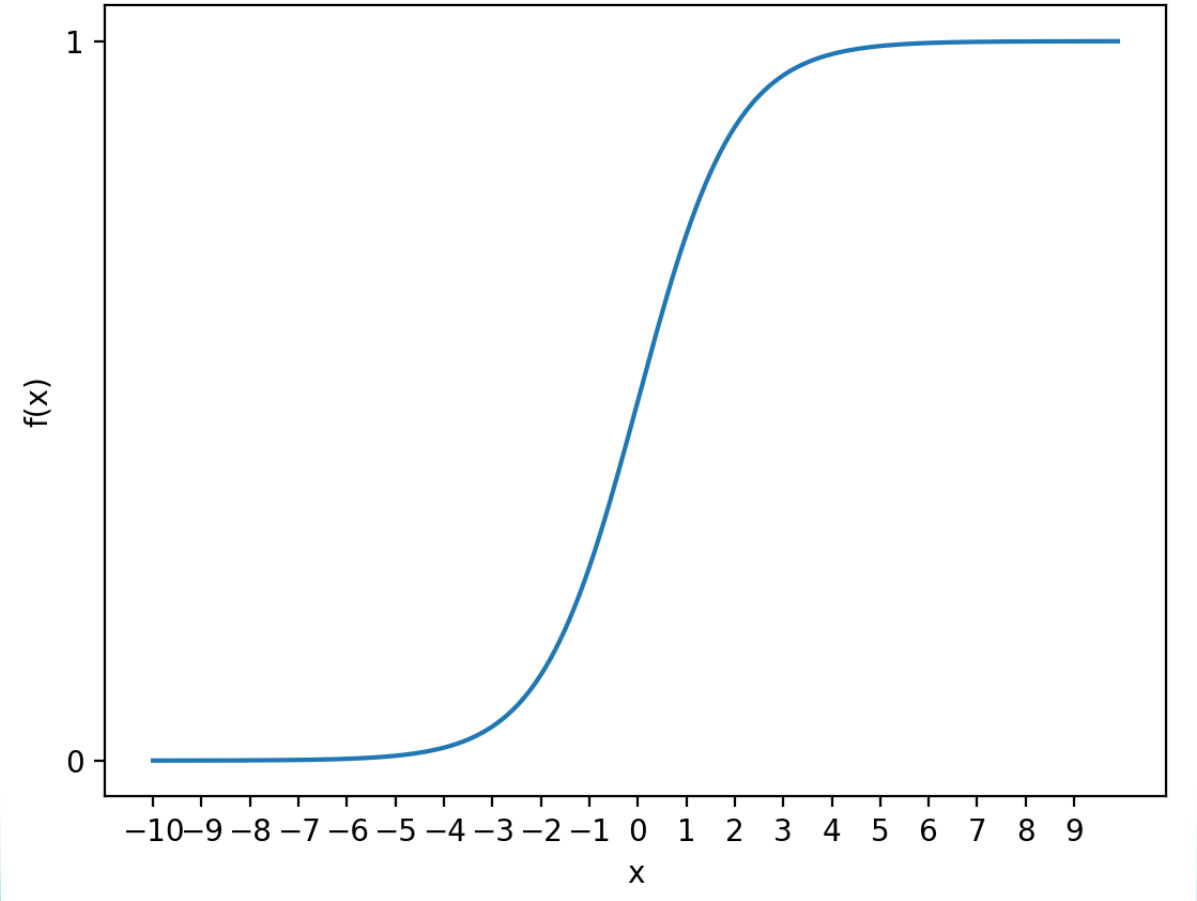
Deep Learning

❖ Activation Function

```
import numpy as np
import matplotlib.pylab as plt

fig = plt.figure()                  # figure 객체 생성
ax = fig.add_subplot(1, 1, 1)       # 1x1개의 서브플롯을 추가하고, 첫번째 서브 플롯을 사용

x = np.arange(-10, 10, 0.1)         # x값을 -10 ~ 10까지 0.1단위로 입력
f = 1/(1 + np.exp(-x))              # 함수 f(x) 선언

ax.plot(x, f)                       # x, y값 입력
ax.set_xticks(range(-10, 10))       # x축 눈금 범위 지정
ax.set_yticks(range(0, 2))          # y축 눈금 범위 지정
ax.set_xlabel('x')                  # x축 라벨 지정
ax.set_xlabel('f(x)')               # y축 라벨 지정
plt.show()
```

Deep Learning

❖ Activation Function



Pytorch

- Tensor
  - empty
  - rand
  - zeros
  - ones
  - tensor

```
import torch

a = torch.empty(5, 3)
b = torch.rand(5, 3)
c = torch.zeros(5, 3)
d = torch.ones(5, 3)
e = torch.tensor([5.5, 3])

print(a)
print(b)
print(c)
print(d)
print(e)
print(a.shape)
print(a.size())
```

CODE

```
tensor([[7.2551e-39, 8.4490e-39, 9.6429e-39],
        [8.4490e-39, 9.6429e-39, 9.2755e-39],
        [1.0286e-38, 9.0919e-39, 8.9082e-39],
        [9.2755e-39, 8.4490e-39, 1.0194e-38],
        [9.0919e-39, 8.4490e-39, 1.0286e-38]])
tensor([[0.3846, 0.9463, 0.7332],
        [0.5467, 0.4784, 0.3033],
        [0.7599, 0.9369, 0.8928],
        [0.1870, 0.4699, 0.7343],
        [0.6270, 0.0797, 0.2602]])
tensor([[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]])
tensor([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])
tensor([5.5000, 3.0000])
torch.Size([5, 3])
torch.Size([5, 3])
```

Result

Pytorch

- Array to Tensor
  - tensor
  - as_tensor
  - from_numpy

```python
import torch
import numpy as np

a = np.array([[1, 2], [3, 4]])
b = torch.tensor(a)
c = torch.as_tensor(a)
d = torch.from_numpy(a)

print(a)
print(b)
print(c)
print(d)
```

CODE

```
[[1 2]
 [3 4]]
tensor([[1, 2],
        [3, 4]], dtype=torch.int32)
tensor([[1, 2],
        [3, 4]], dtype=torch.int32)
tensor([[1, 2],
        [3, 4]], dtype=torch.int32)
```

Result

Pytorch

- Array to Tensor
  - tensor
  - as_tensor
  - from_numpy

```python
import torch
import numpy as np

a = np.array([[1, 2], [3, 4]])
b = torch.tensor(a)
c = torch.as_tensor(a)
d = torch.from_numpy(a)

a[0, 0] = 1000
a[0, 1] = 2000

print(a)
print(b)
print(c)
print(d)
```

CODE

```
[[1000 2000]
 [   3    4]]
tensor([[1, 2],
        [3, 4]], dtype=torch.int32)
tensor([[1000, 2000],
        [   3,    4]], dtype=torch.int32)
tensor([[1000, 2000],
        [   3,    4]], dtype=torch.int32)
```

Result

Pytorch

❖ Tensor to Array

- numpy

```
import torch
import numpy as np

a = np.array([[1, 2], [3, 4]])
b = torch.tensor(a)
c = b.numpy()

b[0, 0] = 1000
b[0, 1] = 2000

print(a)
print(b)
print(c)
```

CODE

```
[[1 2]
 [3 4]]
tensor([[1000, 2000],
        [   3,    4]], dtype=torch.int32)
[[1000 2000]
 [   3    4]]
```

Result

Pytorch

- Tensor
  - view
  - squeeze
  - unsqueeze
  - cat

```
import torch

a = torch.ones(4, 3)
b = a.view(3, 4)
c = a.view(12)
d = torch.rand(1, 3, 3)
e = d.squeeze()
f = e.unsqueeze(1)

g = torch.ones(2, 3)
h = torch.zeros(3, 3)
i = torch.cat([g, h], dim=0)

print(a)
print(b)
print(c)
print(d.size())
print(e.size())
print(f.size())
print(i)
```

CODE

```
tensor([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
tensor([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
torch.Size([1, 3, 3])
torch.Size([3, 3])
torch.Size([3, 1, 3])
tensor([[1., 1., 1.],
        [1., 1., 1.],
        [0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]])
```

Result

Pytorch

- Tensor
  - chunk
  - split

```
import torch

a = torch.rand(3, 6)
b, c, d = torch.chunk(a, 3, dim=1)
print(a)
print(b)
print(c)
print(d)

e, f = torch.split(a, 3, dim=1)
print(e)
print(f)
```

CODE

```
tensor([[0.6924, 0.9834, 0.7211, 0.0598, 0.1183, 0.9953],
        [0.0709, 0.4422, 0.4562, 0.0612, 0.9949, 0.0778],
        [0.9317, 0.0744, 0.2314, 0.9160, 0.1614, 0.3617]])
tensor([[0.6924, 0.9834],
        [0.0709, 0.4422],
        [0.9317, 0.0744]])
tensor([[0.7211, 0.0598],
        [0.4562, 0.0612],
        [0.2314, 0.9160]])
tensor([[0.1183, 0.9953],
        [0.9949, 0.0778],
        [0.1614, 0.3617]])
tensor([[0.6924, 0.9834, 0.7211],
        [0.0709, 0.4422, 0.4562],
        [0.9317, 0.0744, 0.2314]])
tensor([[0.0598, 0.1183, 0.9953],
        [0.0612, 0.9949, 0.0778],
        [0.9160, 0.1614, 0.3617]])
```

Result

Assignment

```
import numpy as np
import matplotlib.pylab as plt

fig = plt.figure()                  # figure 객체 생성
ax = fig.add_subplot(1, 1, 1)       # 1x1개의 서브플롯을 추가하고, 첫번째 서브 플롯을 사용

x = np.arange(-10, 10, 0.1)         # x값을 -10 ~ 10까지 0.1단위로 입력

?

ax.set_xticks(range(-10, 10))       # x축 눈금 범위 지정
ax.set_yticks(range(0, 2))          # y축 눈금 범위 지정
ax.set_xlabel('x')                  # x축 라벨 지정
ax.set_ylabel('f(x)')               # y축 라벨 지정
plt.show()
```

$$Y = \frac{1}{1 + e^{-wx}}$$

np.exp()

$w = 0.5 \sim 3$

Assignment