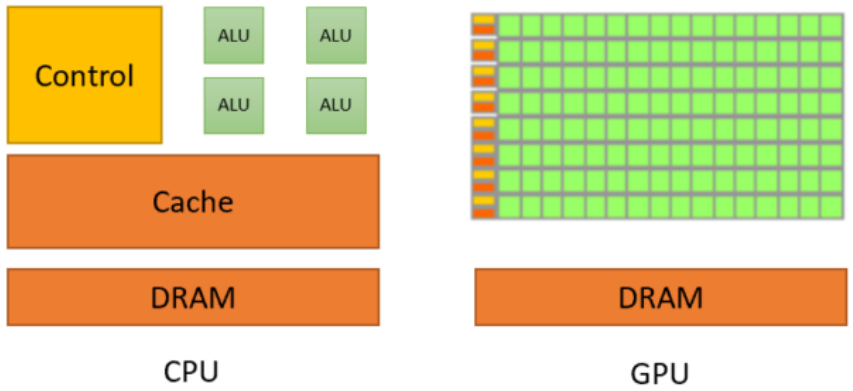
인공지능의 기초

2주차

Review

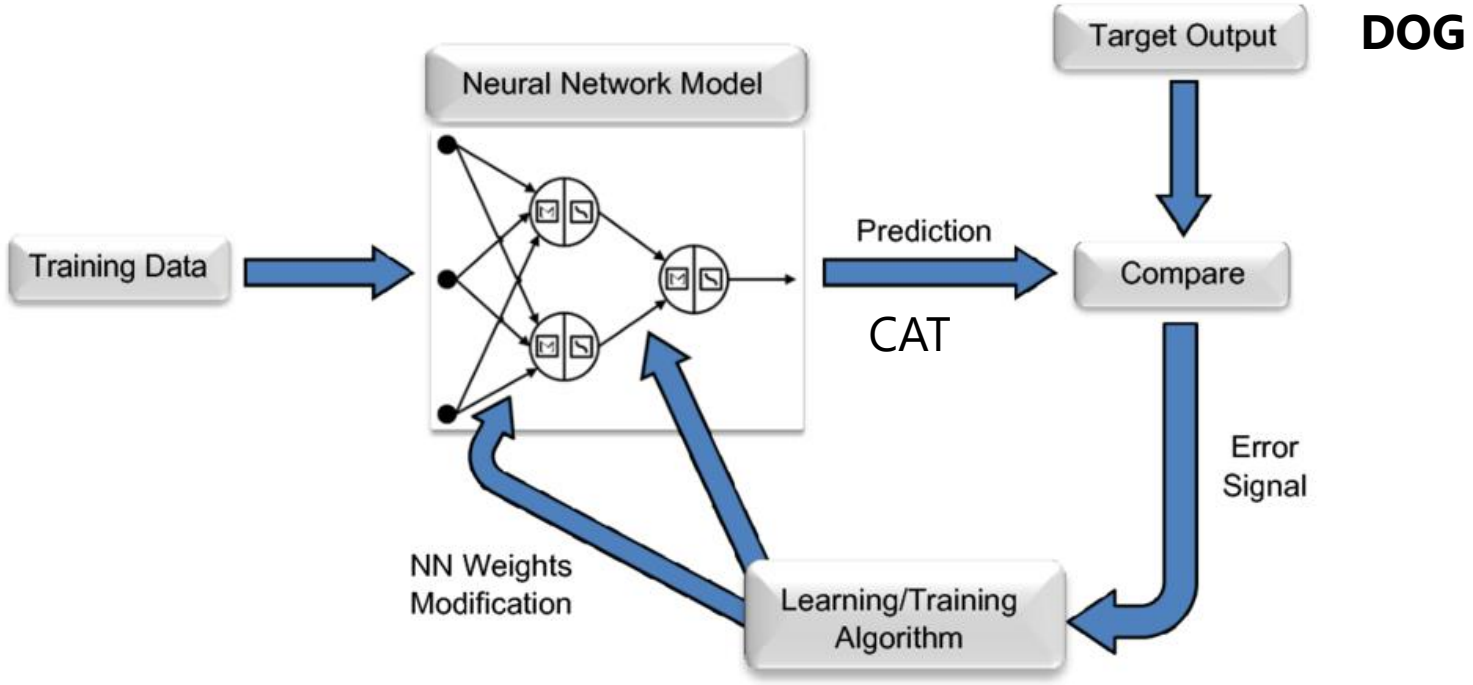
- Pytorch
  - PyTorch는 Python을 위한 오픈소스 머신 러닝 라이브러리이다
- Tensor
  - Pytorch에서의 Tensors는 Numpy의 배열과 비슷한데, 추가로 Tensors도 CUDA를 지원하는 GPU에 사용할 수 있다.
- CUDA
  - CUDA는 GPU에서 수행하는(병렬 처리) 알고리즘을 C 프로그래밍 언어를 비롯한 산업 표준 언어를 사용하여 작성할 수 있도록 하는 기술



Today

- 딥러닝 학습 과정에 대해 알아본다.
- 인공신경망에 대해 알아본다.

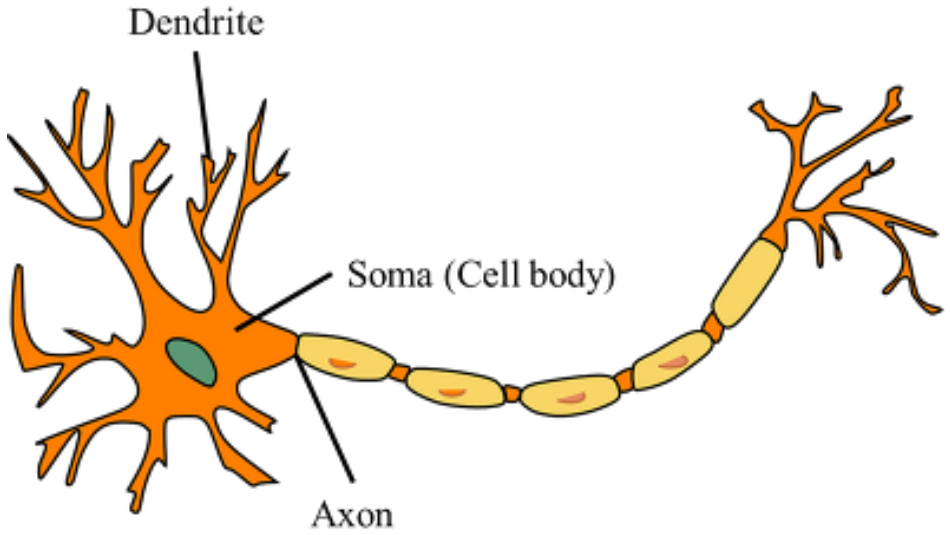
딥러닝 학습 과정

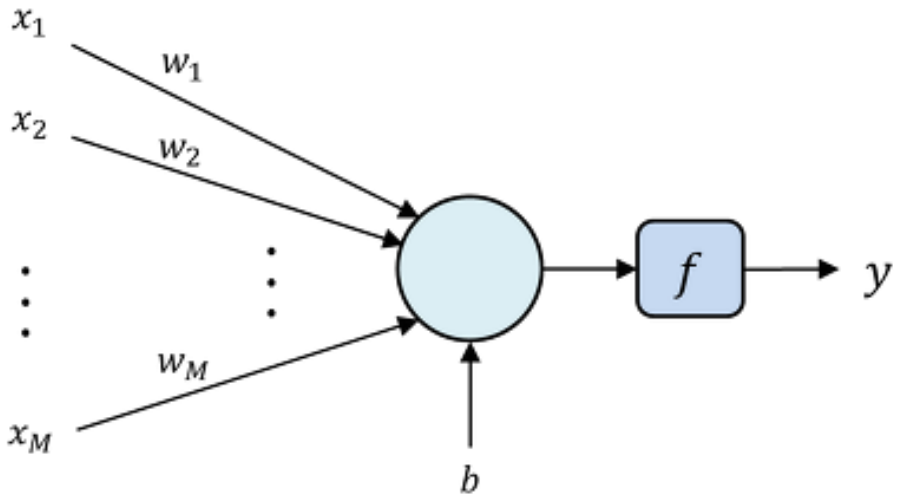


- Data
- Model -> 인공신경망(neural network)
- Training algorithm -> 오차역전파 (back propagation)

인공신경망(artificial neural network)

정의: 소프트웨어적으로 인간의 뉴런 구조를 본떠 만든 기계학습 모델로 인공지능을 구현하기 위한 기술 중 한 형태이다.





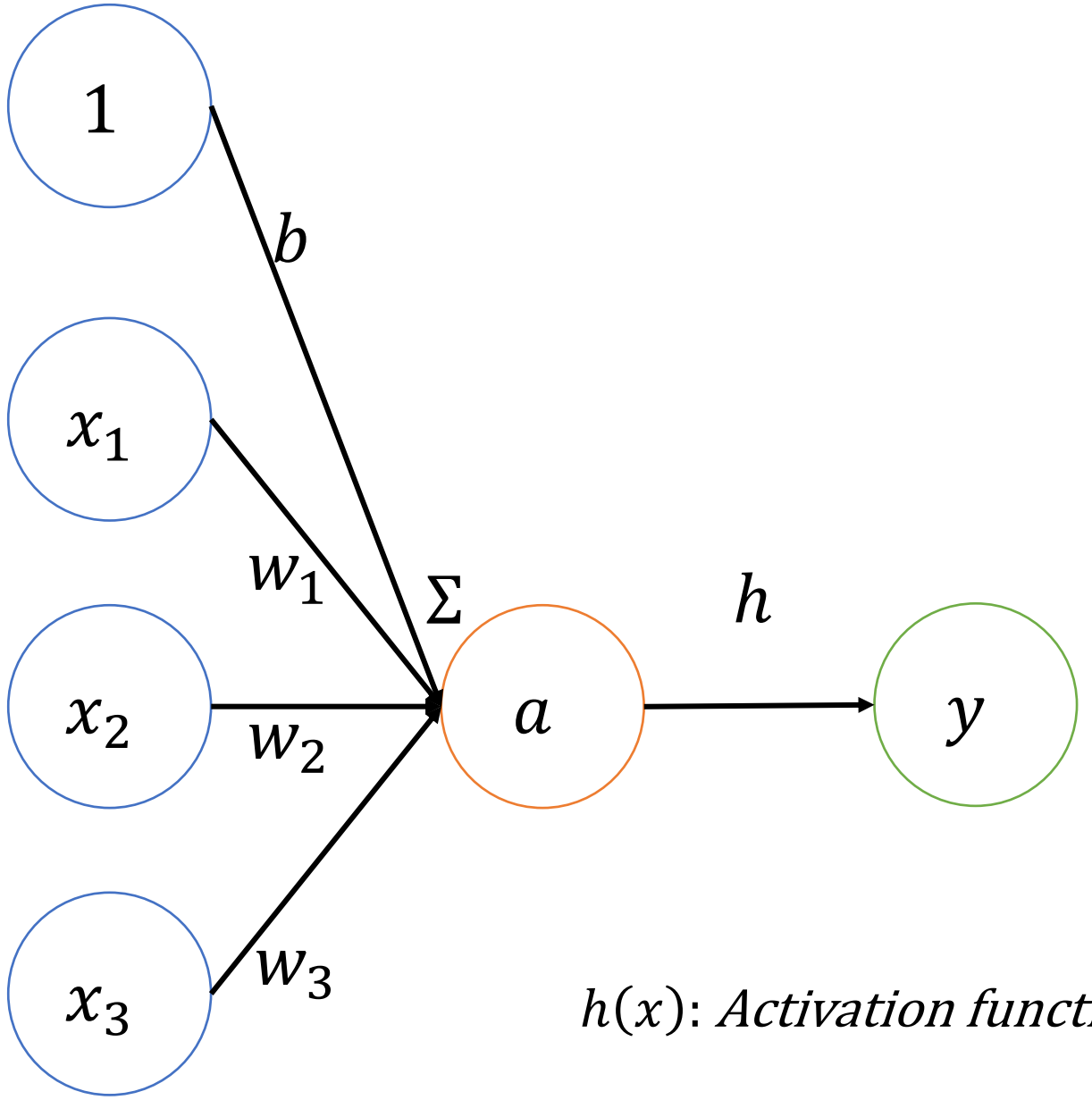
인공신경망

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$a = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

$$y = h(a)$$

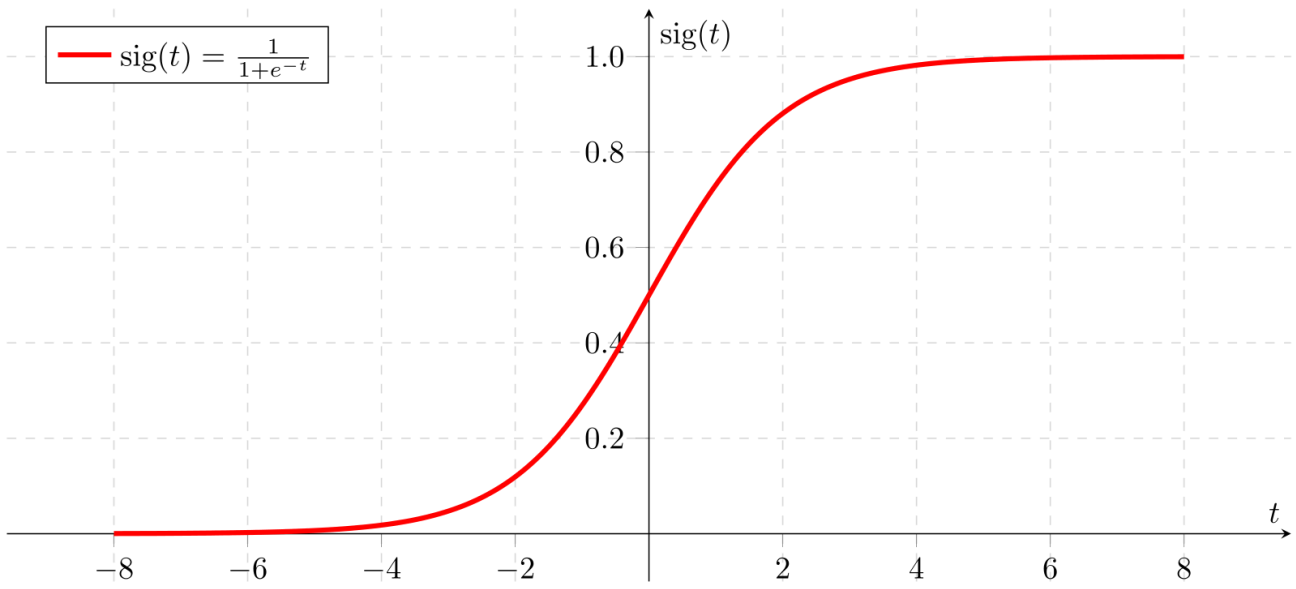
$$\rightarrow \; y = h\left(w^T x + b\right)$$

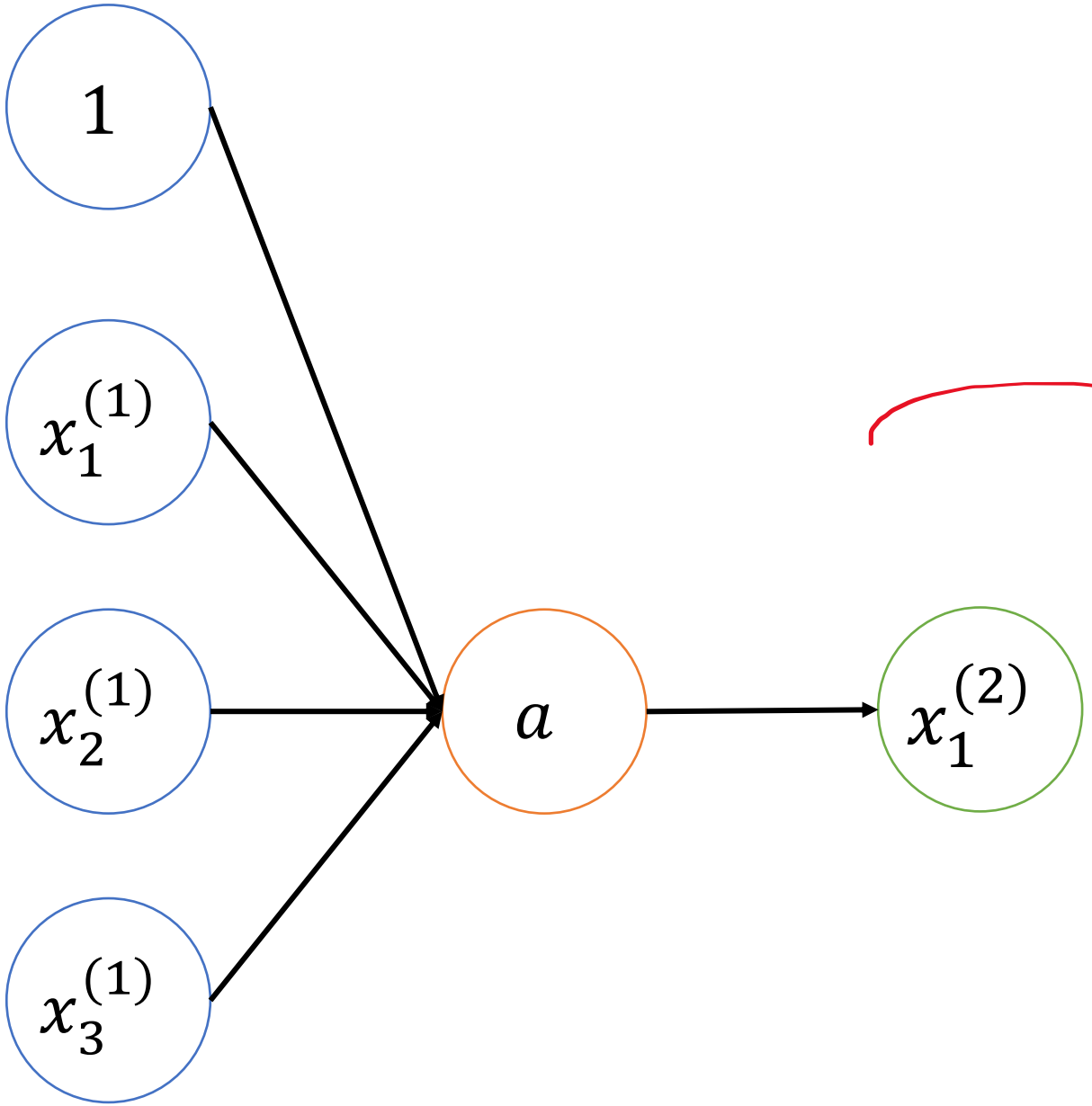


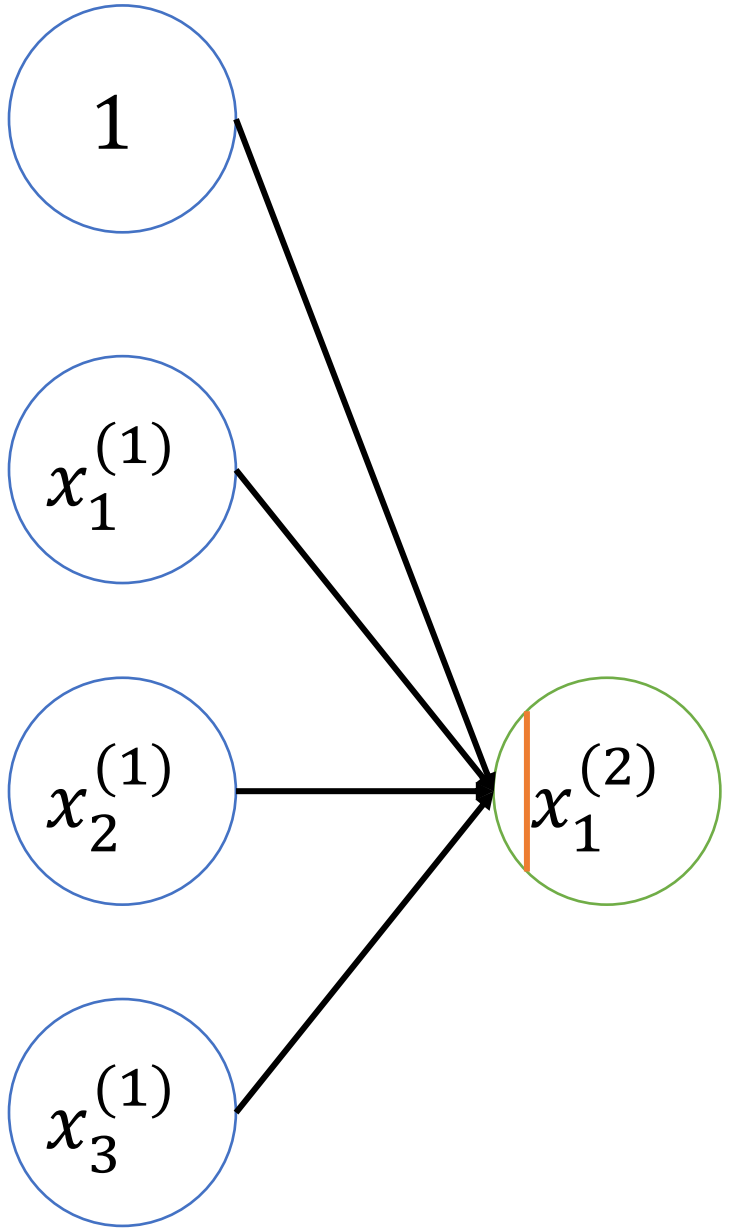
$h(x)$: *Activation function*

활성화 함수(activation function)

Sigmoid $$h(x) = \frac{1}{1 + e^{-x}}$$




1
$x_1^{(1)}$
$x_2^{(1)}$
$x_3^{(1)}$
a
$x_1^{(2)}$

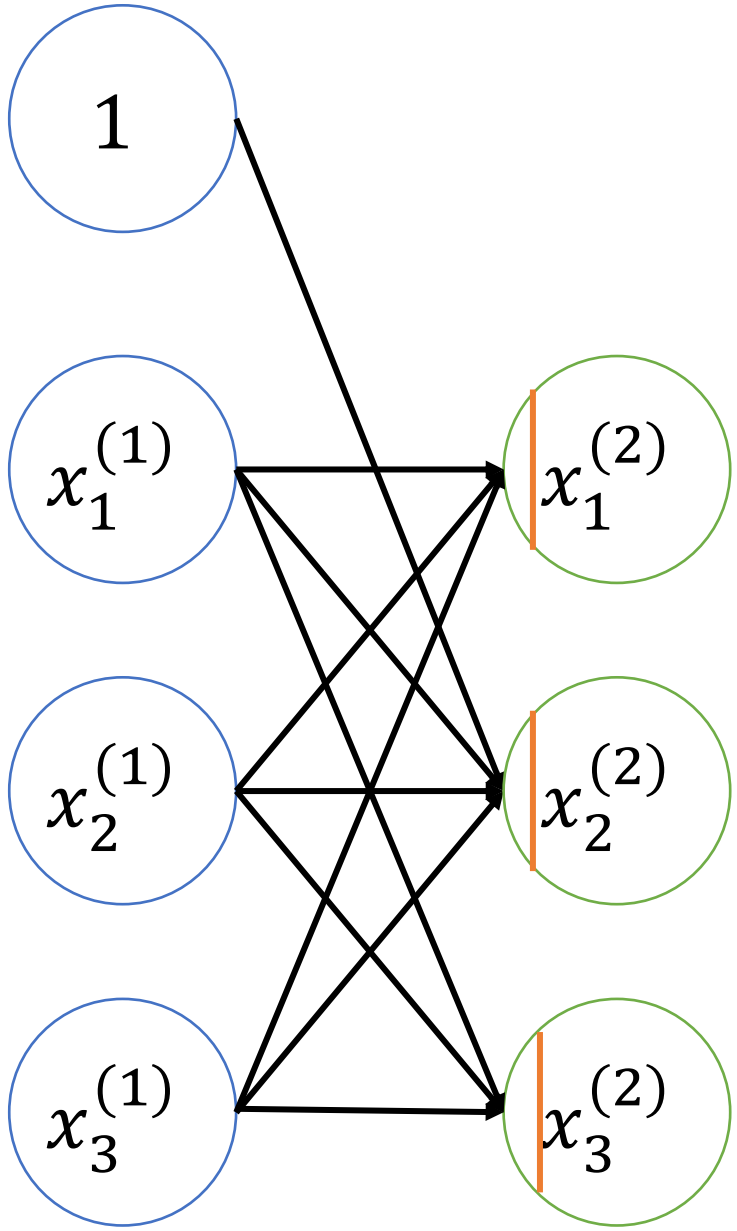

1
$x_1^{(1)}$
$x_2^{(1)}$
$x_3^{(1)}$
$x_1^{(2)}$

인공신경망

$$x^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix}, x^{(2)} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix}$$

$$w^{(1)} = \begin{bmatrix} w_{11}^{(1)} w_{12}^{(1)} w_{13}^{(1)} \\ w_{21}^{(1)} w_{22}^{(1)} w_{23}^{(1)} \\ w_{31}^{(1)} w_{32}^{(1)} w_{33}^{(1)} \end{bmatrix}$$

$$x^{(2)} = h\left(w^{(1)^T} x^{(1)} + b^{(1)}\right)$$

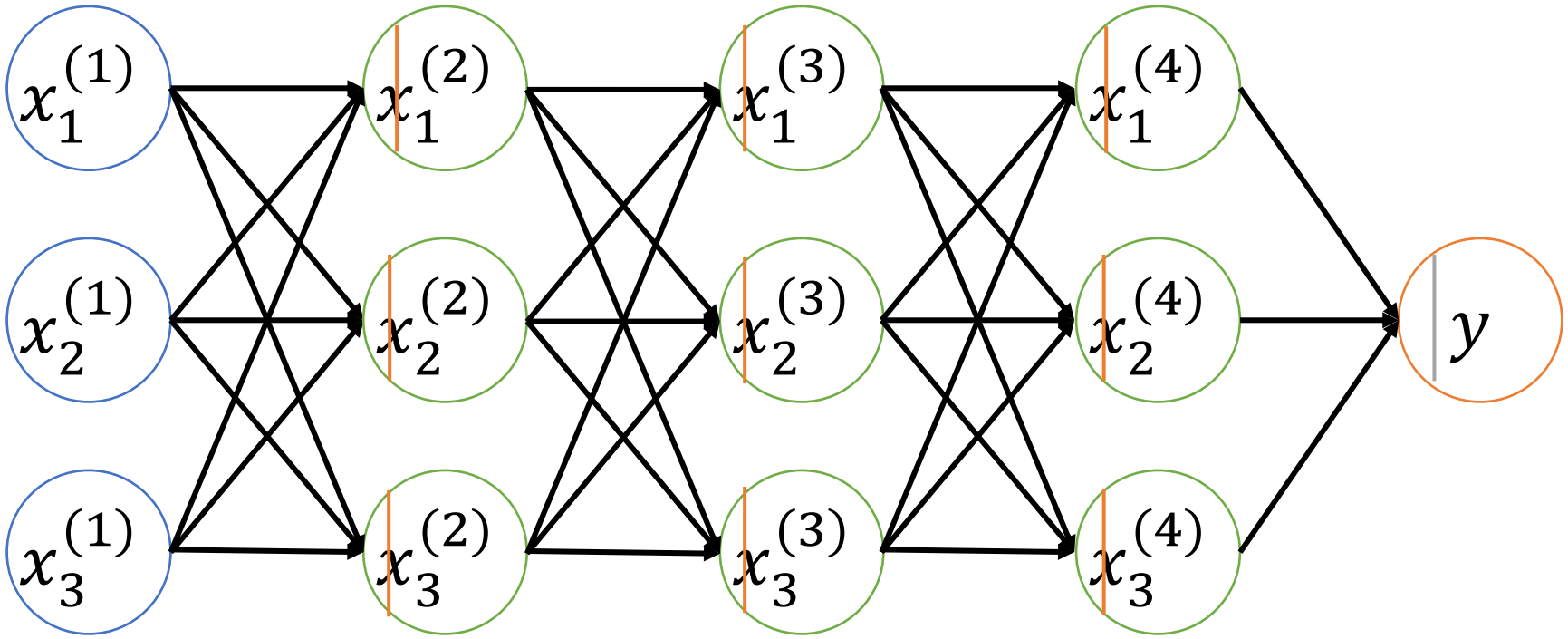


인공신경망

심층신경망(Deep Neural Network)

$$x^{(2)} = h\left(w^{(1)^T} x^{(1)} + b^{(1)}\right)$$
$$x^{(3)} = h\left(w^{(2)^T} x^{(2)} + b^{(2)}\right)$$
$$x^{(4)} = h\left(w^{(3)^T} x^{(3)} + b^{(3)}\right)$$
$$y = h\left(w^{(4)^T} x^{(4)} + b^{(4)}\right)$$

$$y = f_{w,b}(x)$$

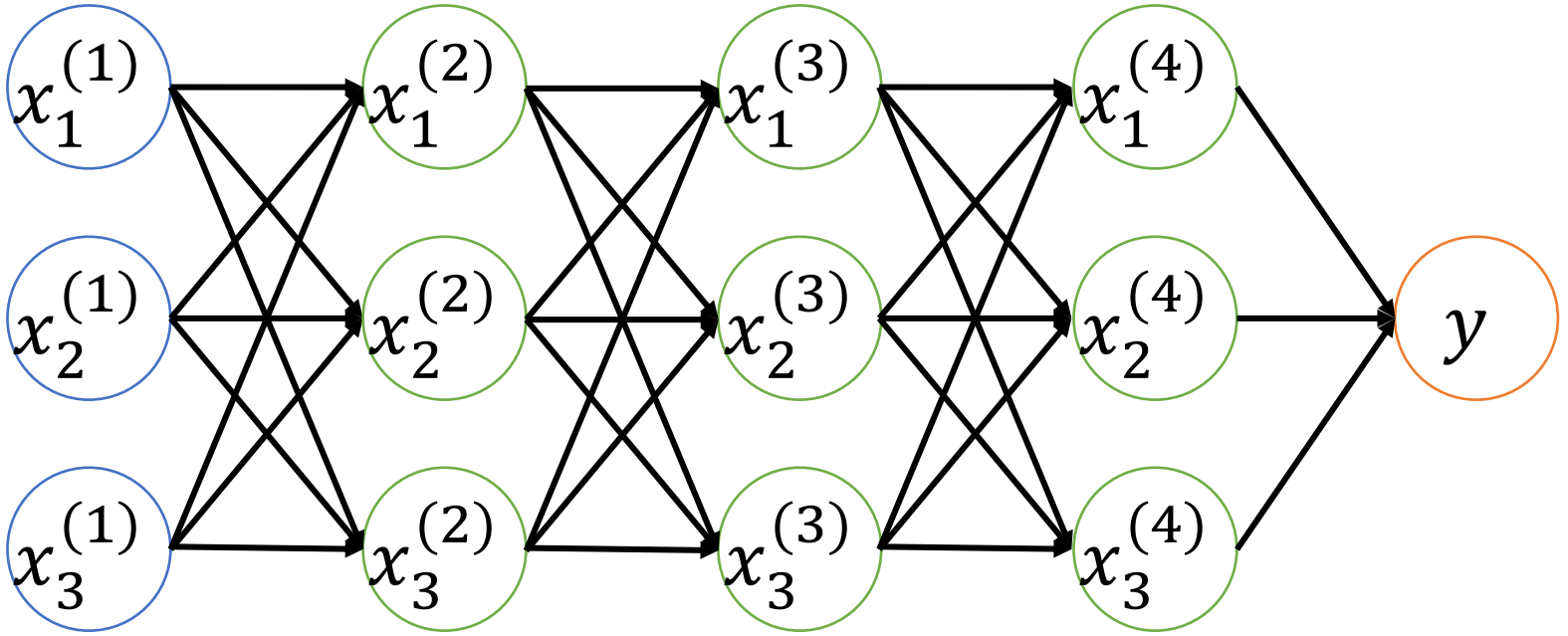


심층신경망(Deep Neural Network)

$$x^{(2)} = w^{(1)^T} x^{(1)} + b^{(1)}$$
$$x^{(3)} = w^{(2)^T} x^{(2)} + b^{(2)}$$
$$x^{(4)} = w^{(3)^T} x^{(3)} + b^{(3)}$$
$$y = w^{(4)^T} x^{(4)} + b^{(4)}$$

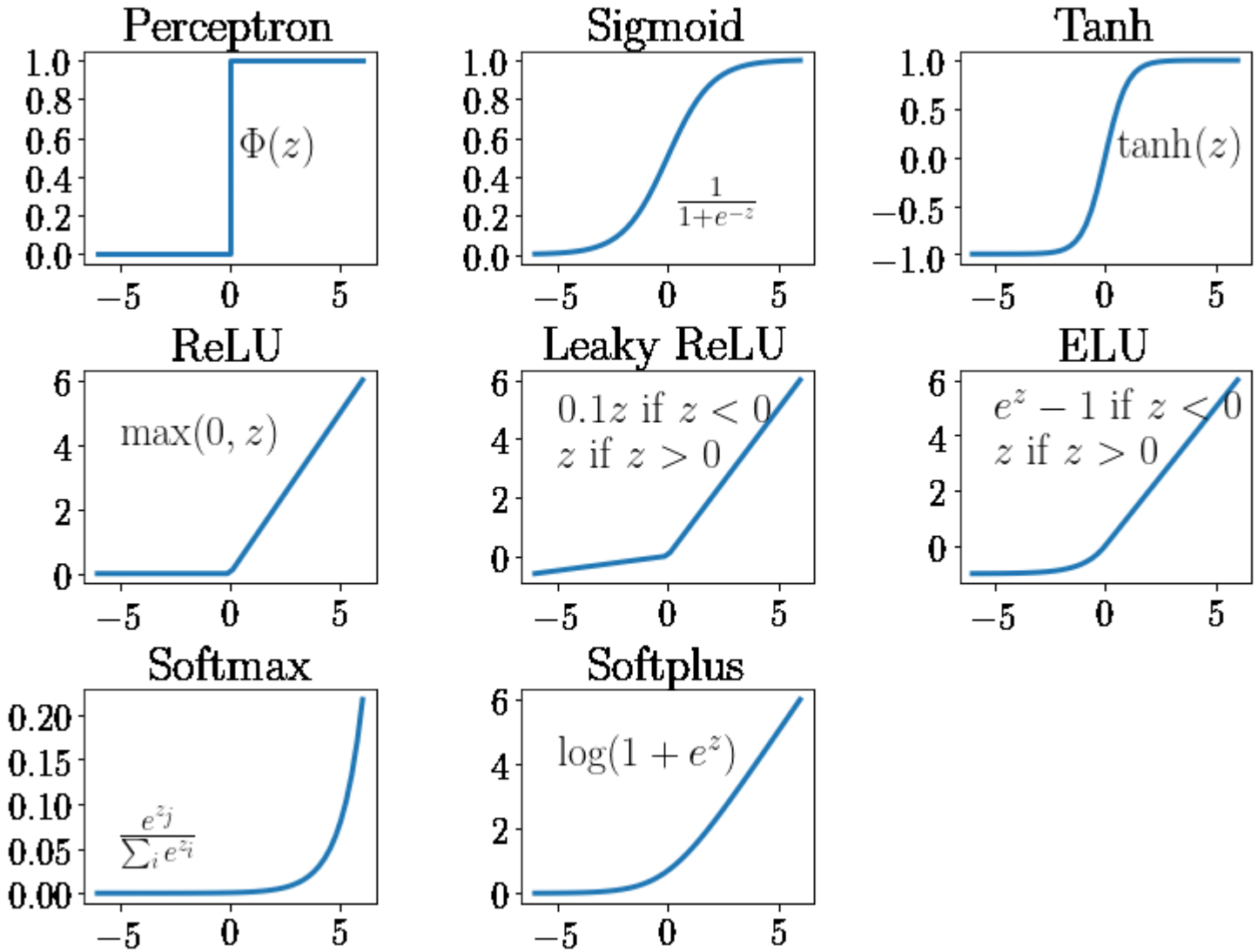
$$y = w^{(4)^T} w^{(4)^T} w^{(4)^T} w^{(4)^T} x^{(1)} + b$$

$$y = w^T x^{(1)} + b$$

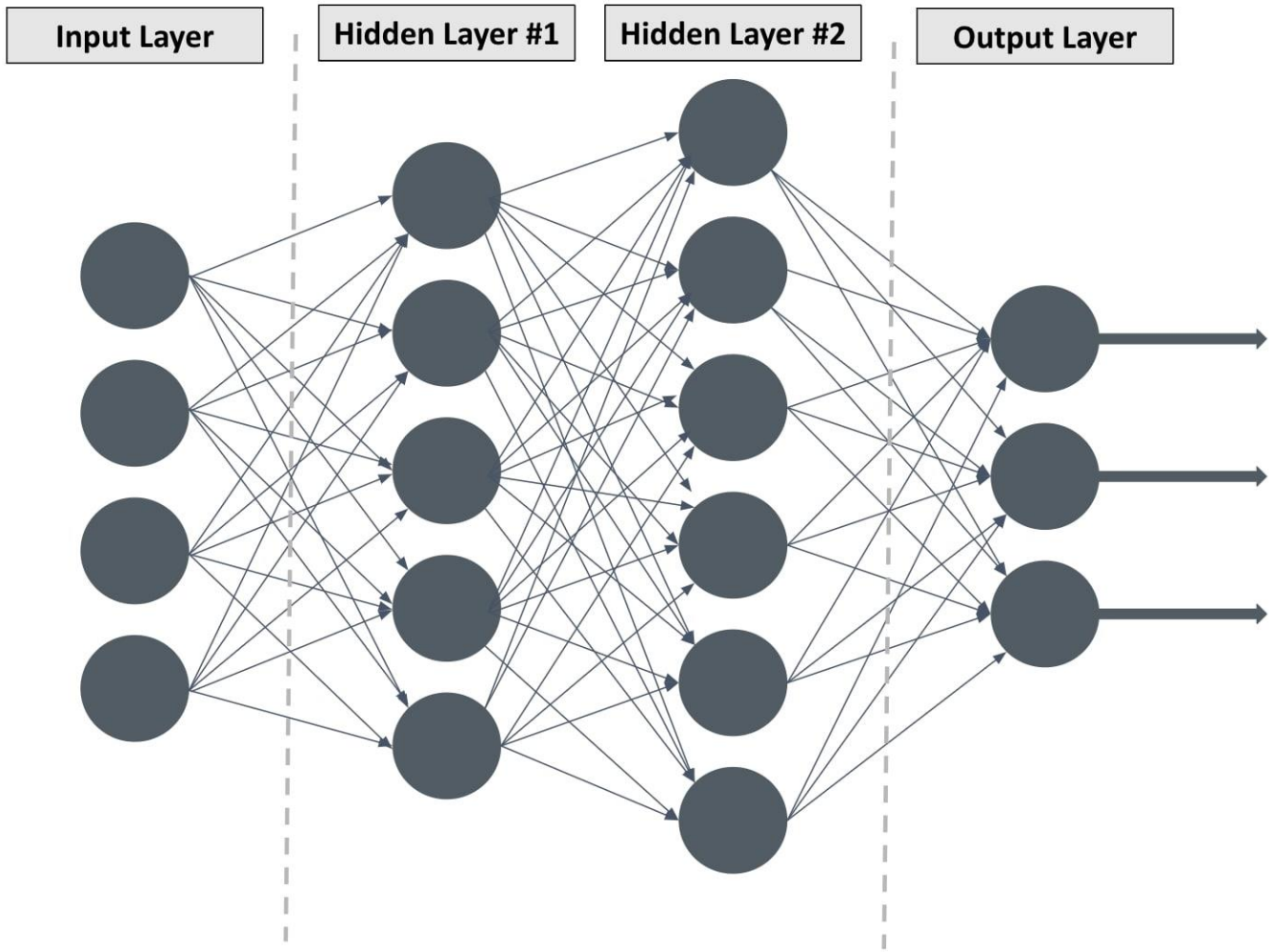


activation function은 중요!

활성화 함수(activation function)



심층신경망(Deep Neural Network)



실습

```
import torch
from torch import nn
```

[4]
```
if torch.cuda.is_available():
    device = 'cuda'
else:
    device = 'cpu'
print(f"Using {device} device")
```

```
Using cuda device
```

[5]
```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.deep_nn = nn.Sequential(
            nn.Linear(4, 5),
            nn.Sigmoid(),
            nn.Linear(5, 6),
            nn.Sigmoid(),
            nn.Linear(6, 3),
        )

    def forward(self, x):
        logits = self.deep_nn(x)
        return logits
```

[6] 4초

```
model = NeuralNetwork().to(device)
print(model)
```

```
NeuralNetwork(
  (deep_nn): Sequential(
    (0): Linear(in_features=4, out_features=5, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=5, out_features=6, bias=True)
    (3): Sigmoid()
    (4): Linear(in_features=6, out_features=3, bias=True)
  )
)
```

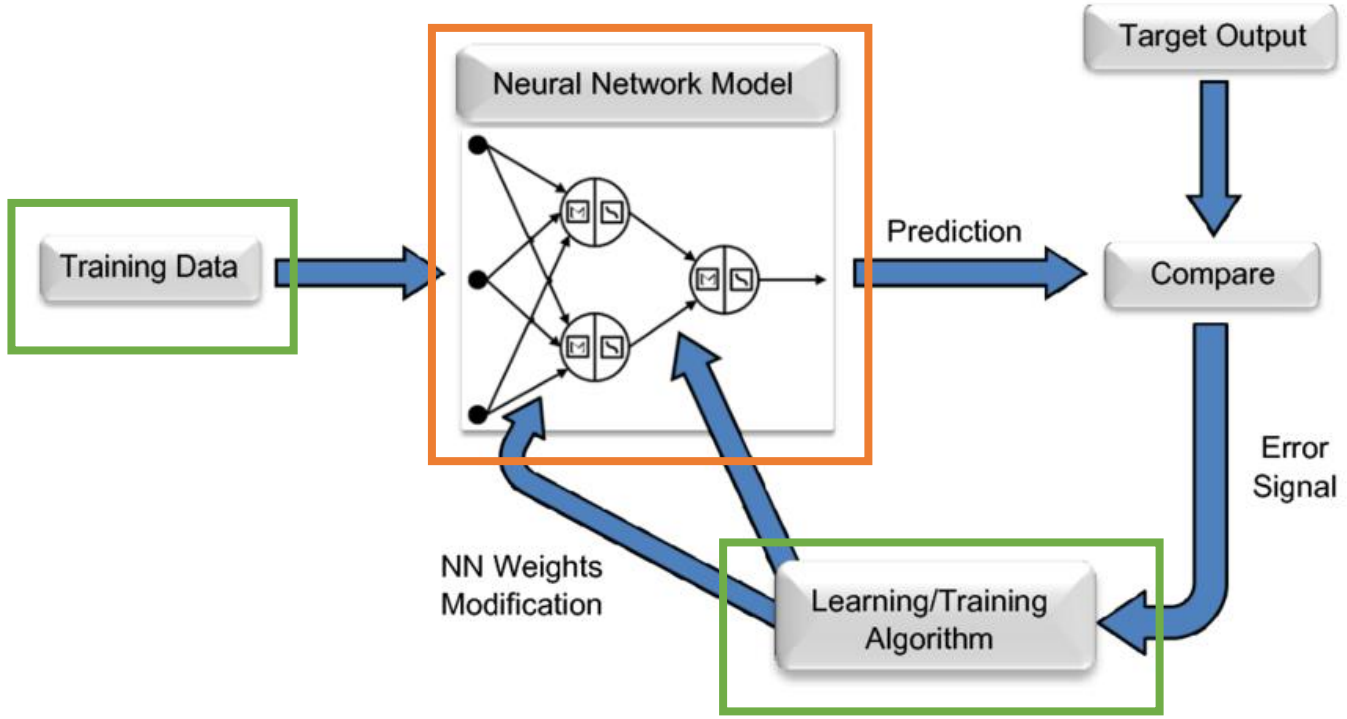
[ ]

```
for name, param in model.named_parameters():
    print(f"Layer: {name} | Size: {param.size()} | Values : {param[:2]} ₩n")
```

```
Layer: deep_nn.0.weight | Size: torch.Size([5, 4]) | Values : tensor([[ 0.1760,  0.4974, -0.0657,  0.3411],
        [ 0.4869, -0.3594,  0.0197,  0.2212]], device='cuda:0',
       grad_fn=<SliceBackward0>)

Layer: deep_nn.0.bias | Size: torch.Size([5]) | Values : tensor([-0.3495, -0.3794], device='cuda:0', grad_fn=<SliceBackward0>)

Layer: deep_nn.2.weight | Size: torch.Size([6, 5]) | Values : tensor([[-0.0791, -0.3209,  0.0879, -0.2188, -0.4224],
        [ 0.2214, -0.1003,  0.2277,  0.4271, -0.1109]], device='cuda:0',
       grad_fn=<SliceBackward0>)

Layer: deep_nn.2.bias | Size: torch.Size([6]) | Values : tensor([-0.1674,  0.3968], device='cuda:0', grad_fn=<SliceBackward0>)

Layer: deep_nn.4.weight | Size: torch.Size([3, 6]) | Values : tensor([[-0.1662, -0.2702, -0.3876, -0.0817,  0.1697, -0.2335],
        [-0.2957,  0.1465,  0.0656, -0.3857,  0.2925, -0.3490]],
       device='cuda:0', grad_fn=<SliceBackward0>)

Layer: deep_nn.4.bias | Size: torch.Size([3]) | Values : tensor([-0.0851,  0.2132], device='cuda:0', grad_fn=<SliceBackward0>)
```

예고

- Backpropagation
- Dataloader



감사합니다