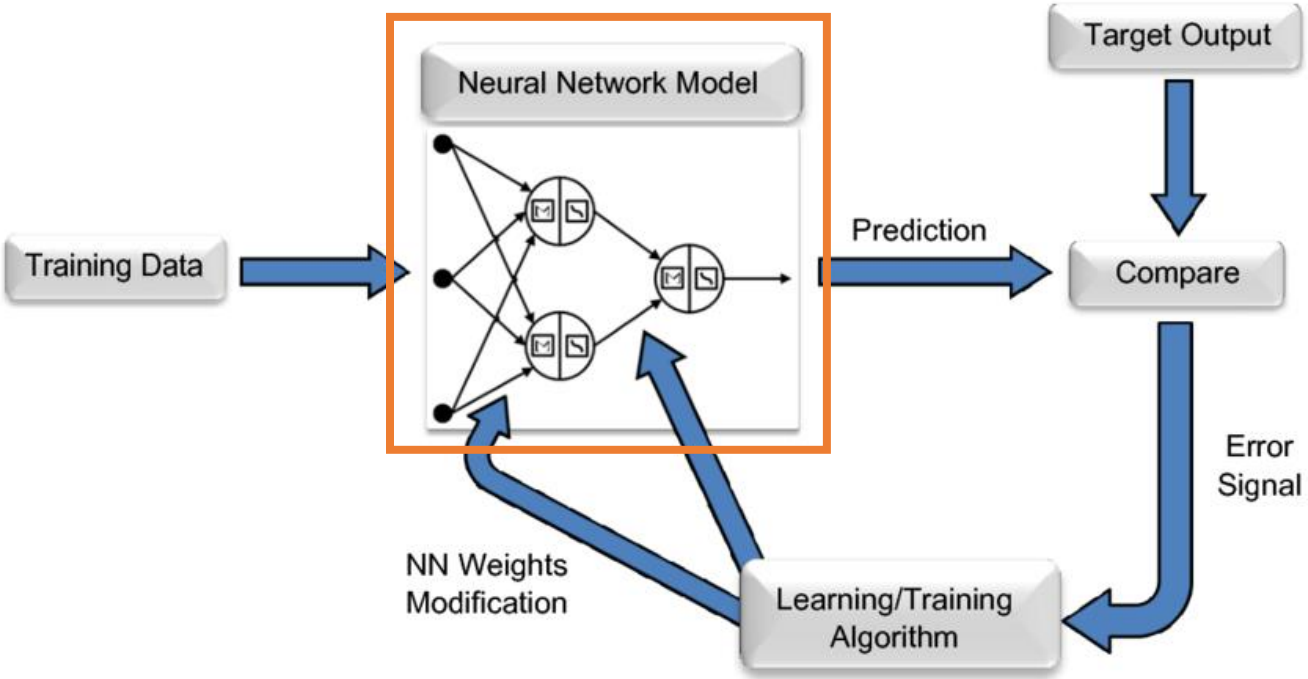
인공지능의 기초

3주차

Review

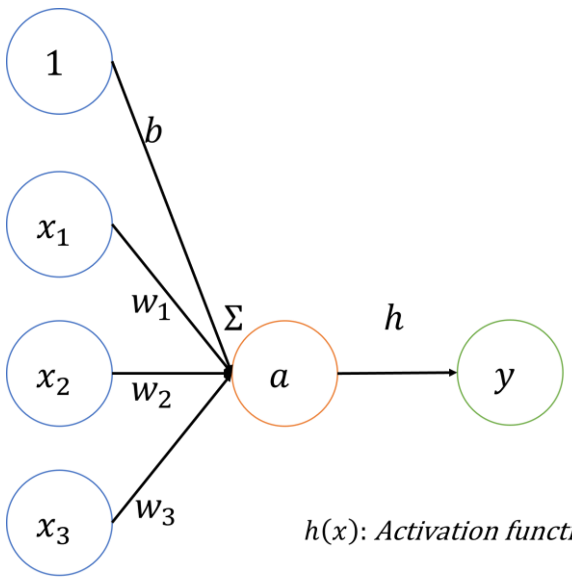
- 딥러닝 학습 과정



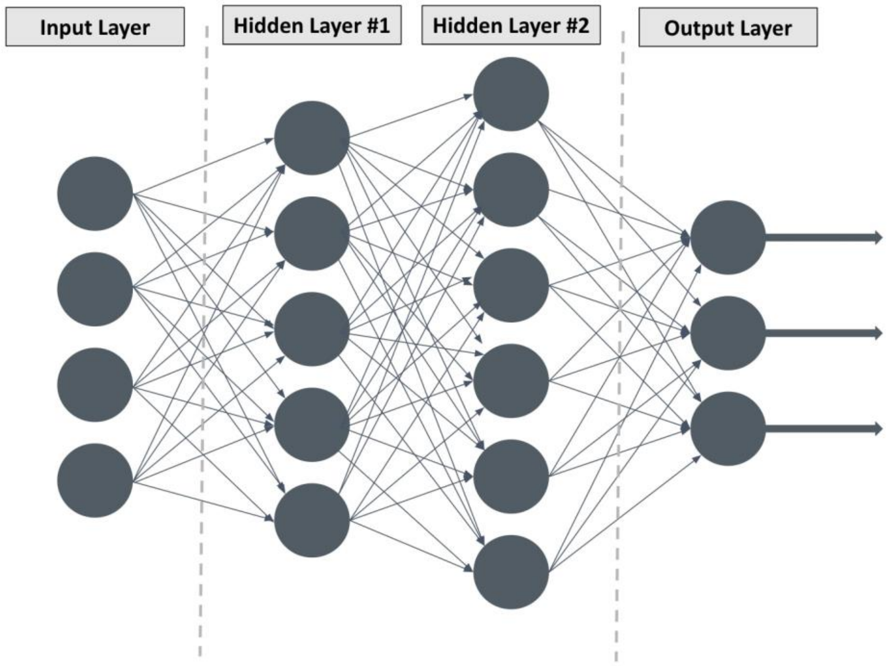
Review

- Deep Neural Network

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

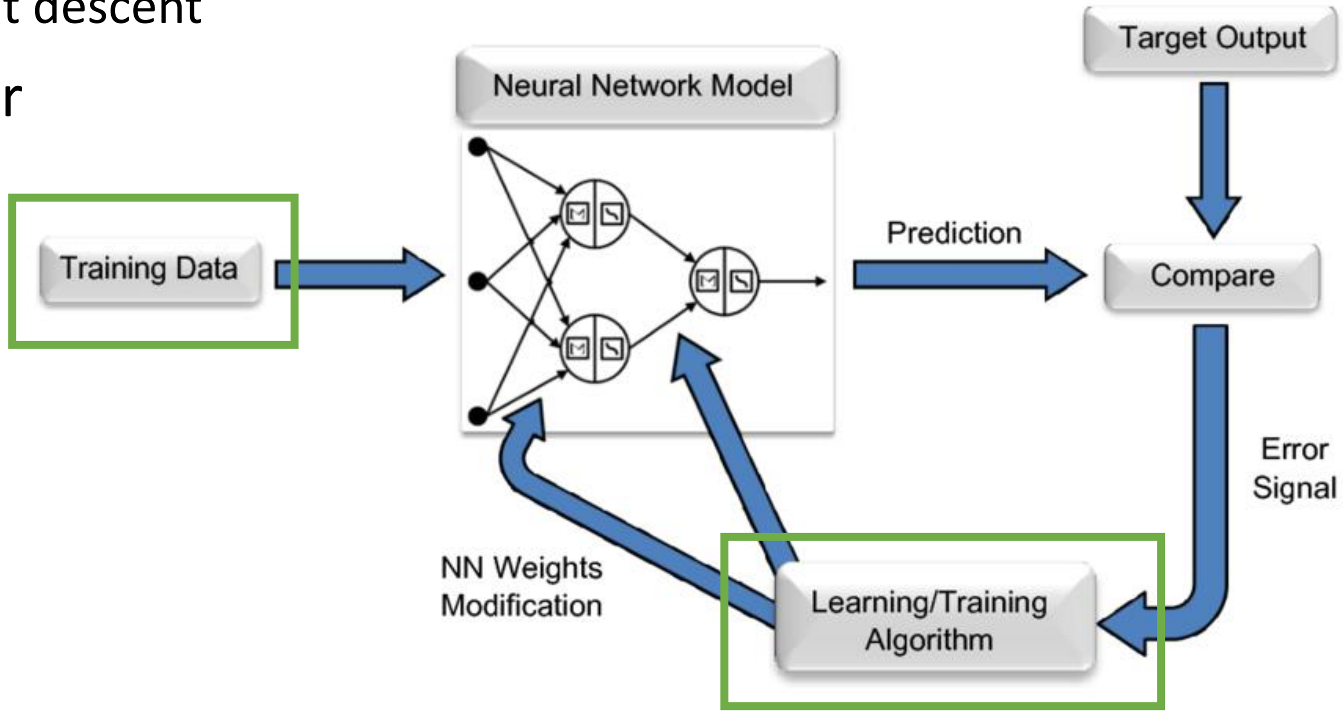


$h(x)$: Activation function



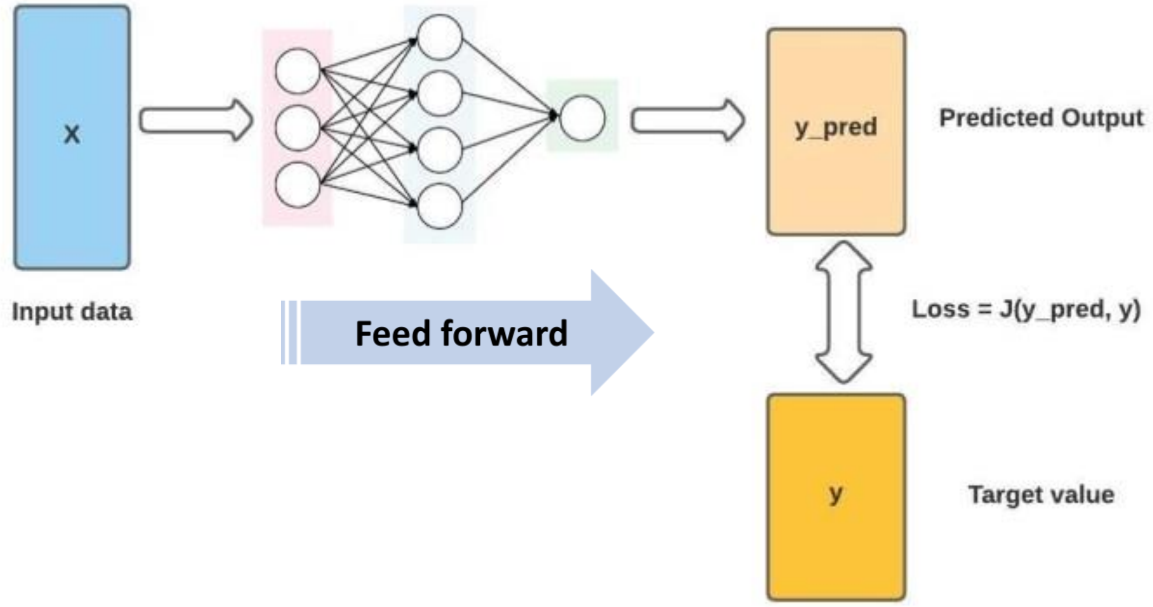
Today

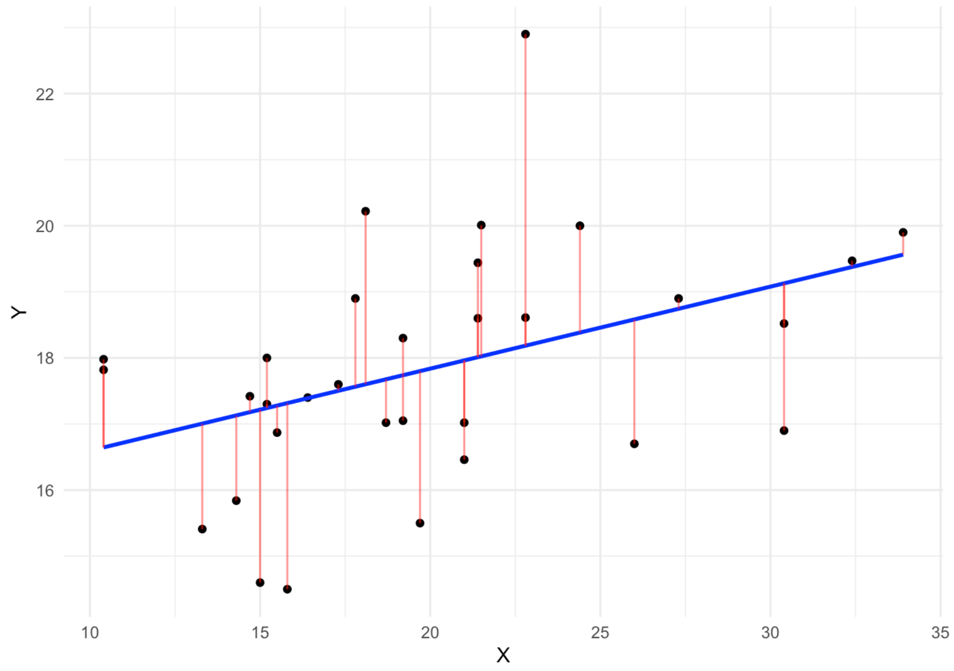
- Backpropagation
  - Loss function
  - Weight update
  - Gradient descent
- Dataloader



사전지식

- Loss function
- 우리의 목표는 loss를 줄이는 것!





$$MAE = \frac{1}{N}\sum_{i=1}^{n}\left|\hat{y_i} - y_i\right|$$

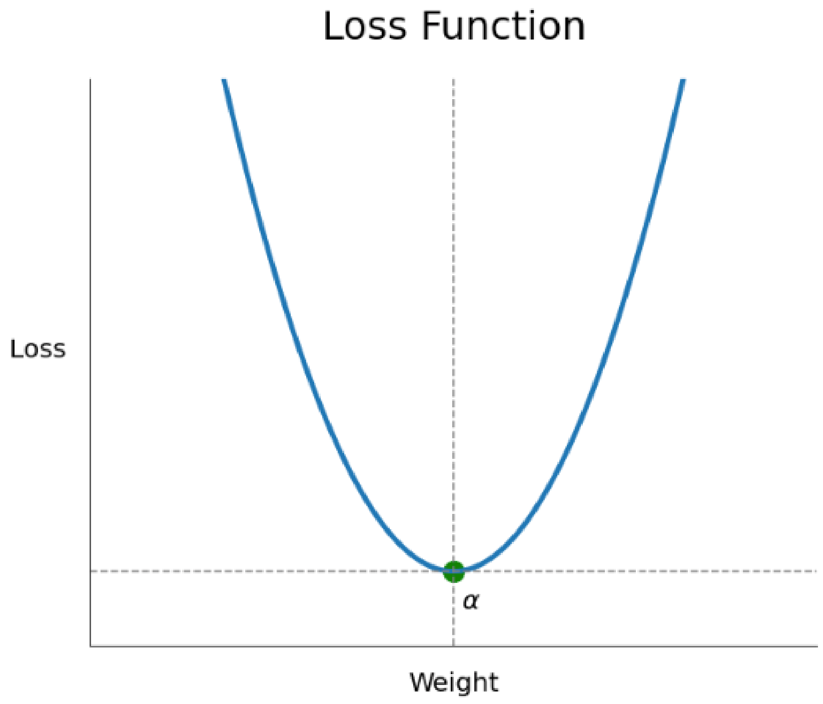
$$MSE = \frac{1}{N}\sum_{i=1}^{n}(\hat{y_i} - y_i)^2$$

사전지식

- Weight update
- Loss를 최소화하기 위한 weight를 찾자!

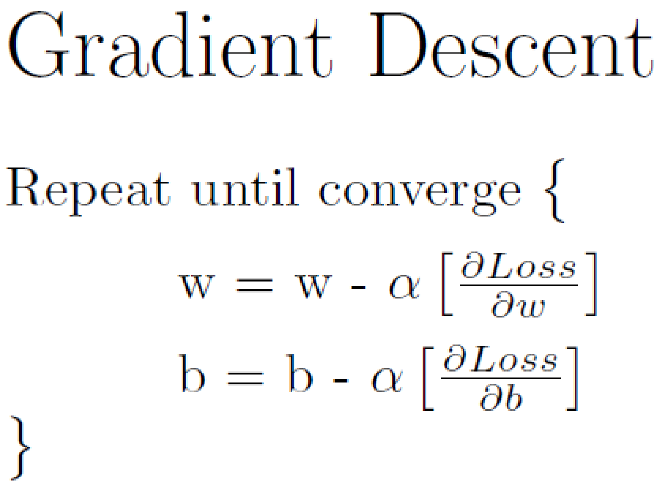
$$MSE = \frac{1}{N}\sum_{i=1}^{n}(\hat{y_i} - y_i)^2$$

$$y = f(w, x)$$



사전지식

- Gradient descent
- 주어진 함수의 극소점을 찾는 알고리즘



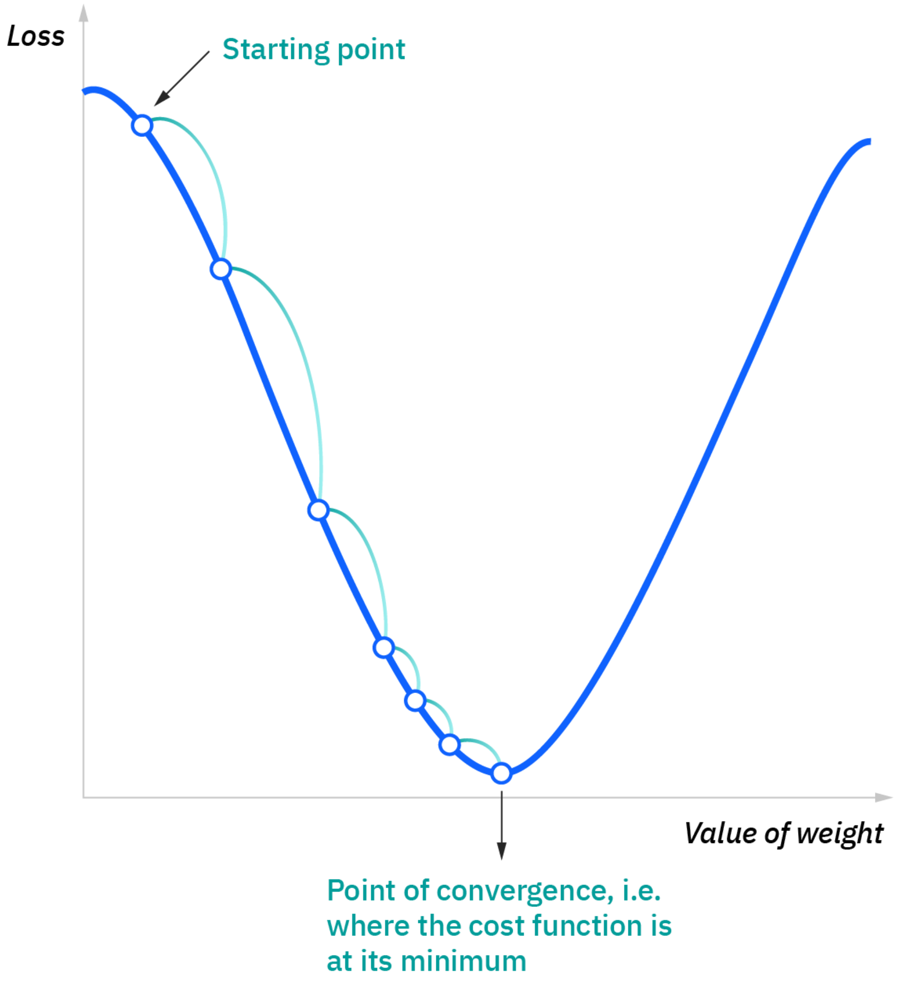
Gradient Descent

Repeat until converge {

$$w = w - \alpha \left[\frac{\partial Loss}{\partial w}\right]$$

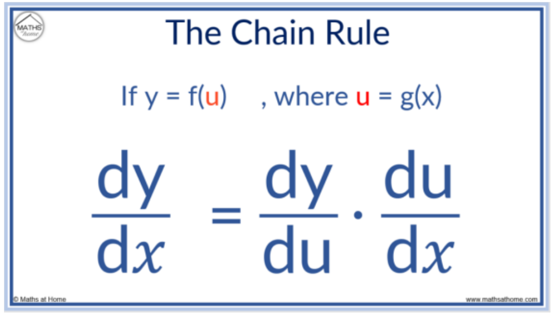
$$b = b - \alpha \left[\frac{\partial Loss}{\partial b}\right]$$

}


Loss
Starting point
Value of weight
Point of convergence, i.e. where the cost function is at its minimum

Backpropagation

- Feedforward 과정을 통하여 입력으로 출력을 계산해낸다.
- Loss를 구한다.
- 각 weight가 Loss에 얼마나 영향을 미치는지 추론한다.
  - Chain rule을 응용한다.
- Weight를 업데이트한다. (gradient descent)



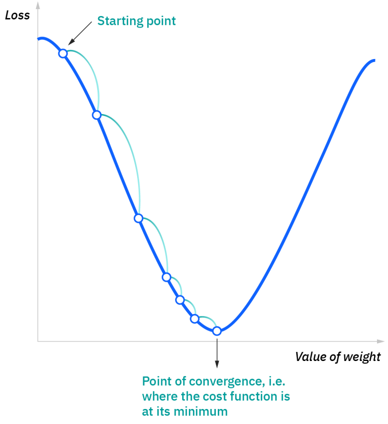
The Chain Rule

If y = f(u) , where u = g(x)

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

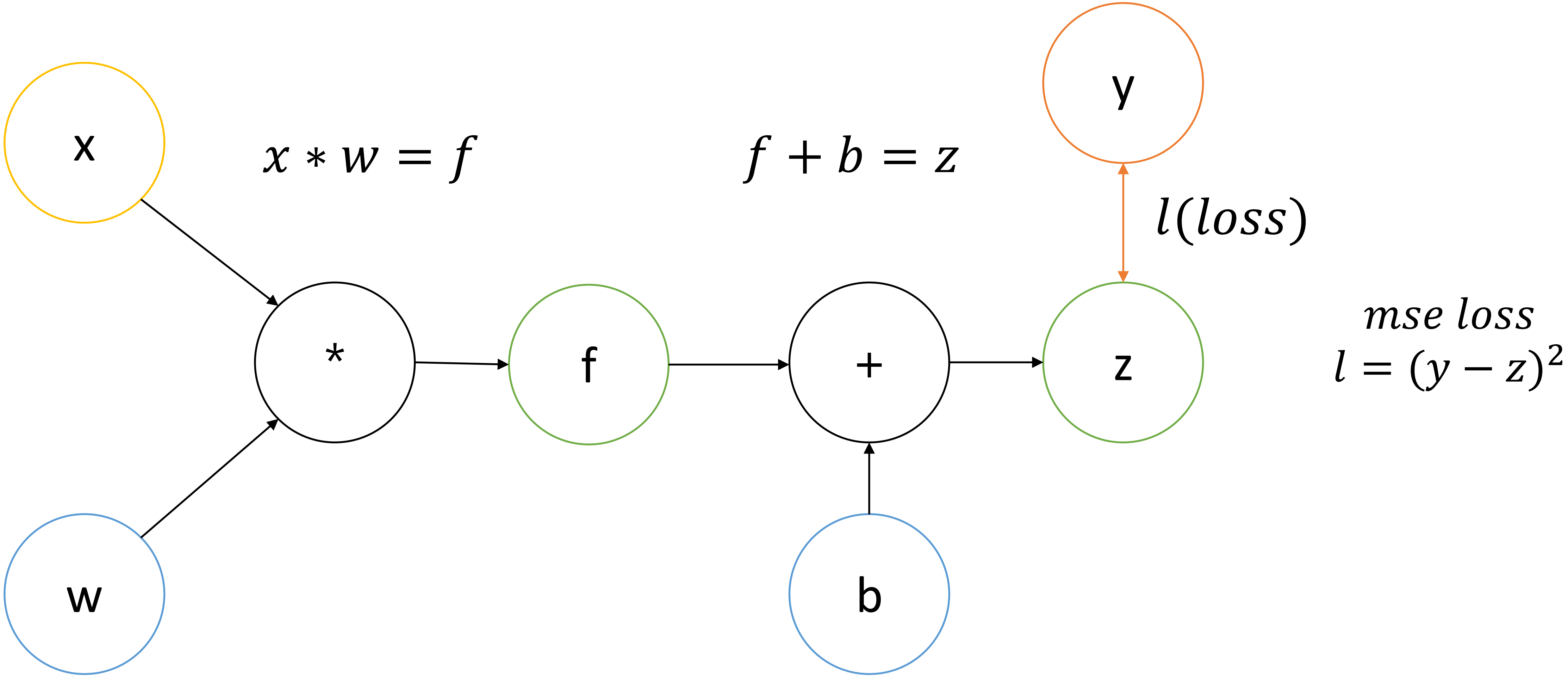
© Maths at Home

www.mathsathome.com



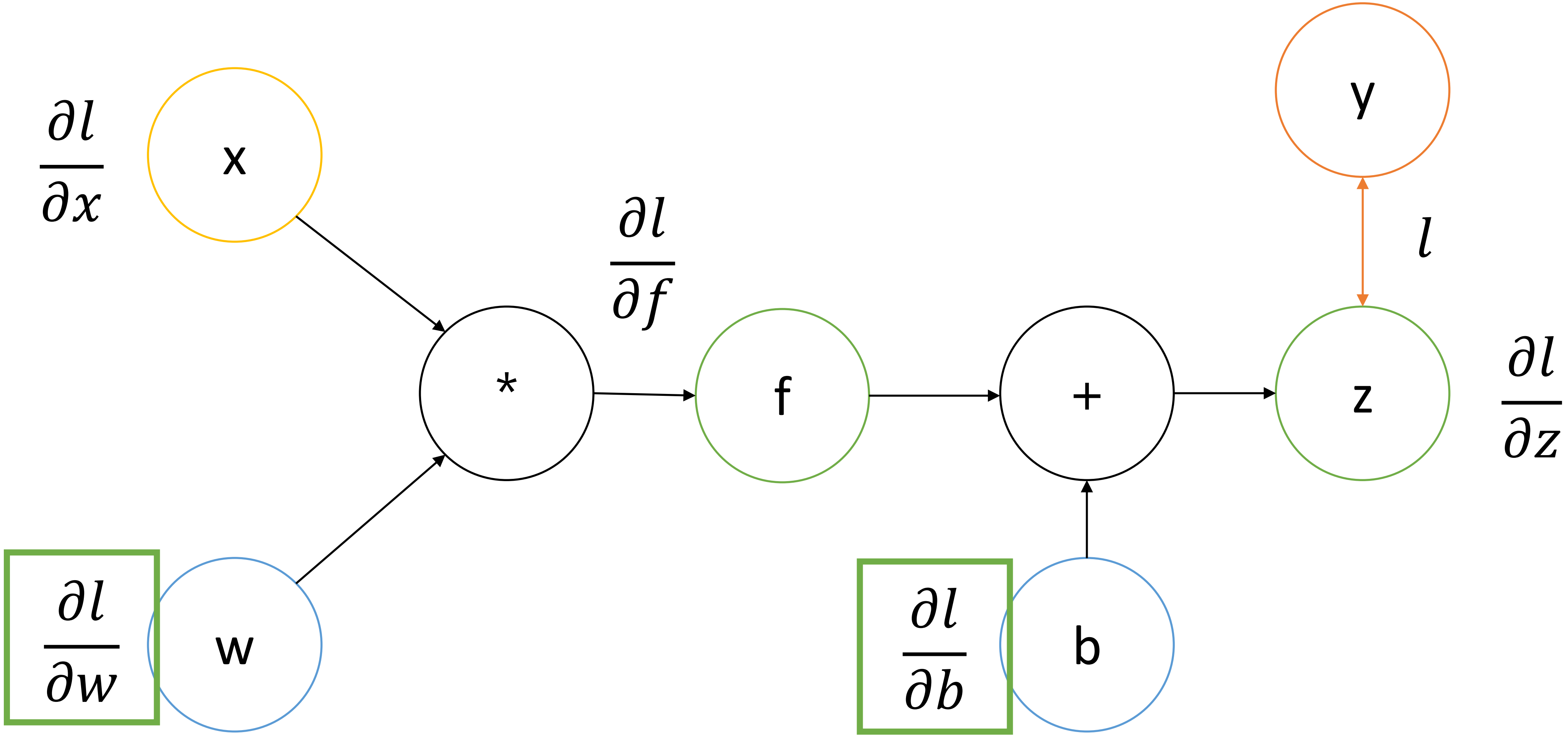
$$w = w - \alpha \boxed{\frac{\partial loss}{\partial w}}$$

Backpropagation



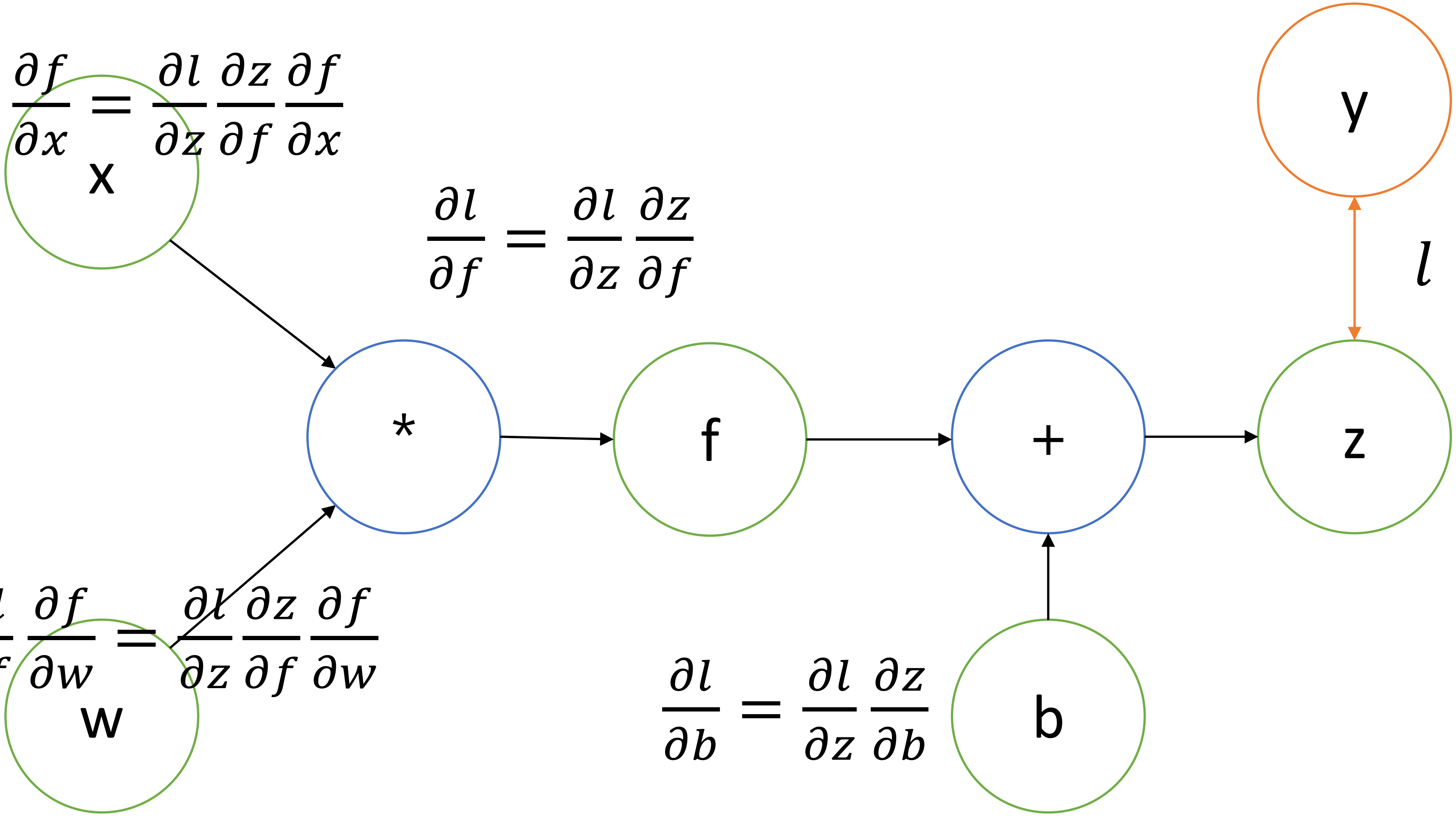
Backpropagation

$x * w = f \qquad f + b = z \qquad l = (y - z)^2$


$\frac{\partial l}{\partial x}$
x
$\frac{\partial l}{\partial f}$
*
f
+
y
l
z
$\frac{\partial l}{\partial z}$
$\frac{\partial l}{\partial w}$
w
$\frac{\partial l}{\partial b}$
b

$x * w = f \qquad f + b = z \qquad l = (y - z)^2$

Backpropagation



$$\frac{\partial l}{\partial x} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial x} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial f}\frac{\partial f}{\partial x}$$

$$\frac{\partial l}{\partial f} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial f}$$

$$\frac{\partial l}{\partial z}$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial w} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial f}\frac{\partial f}{\partial w}$$

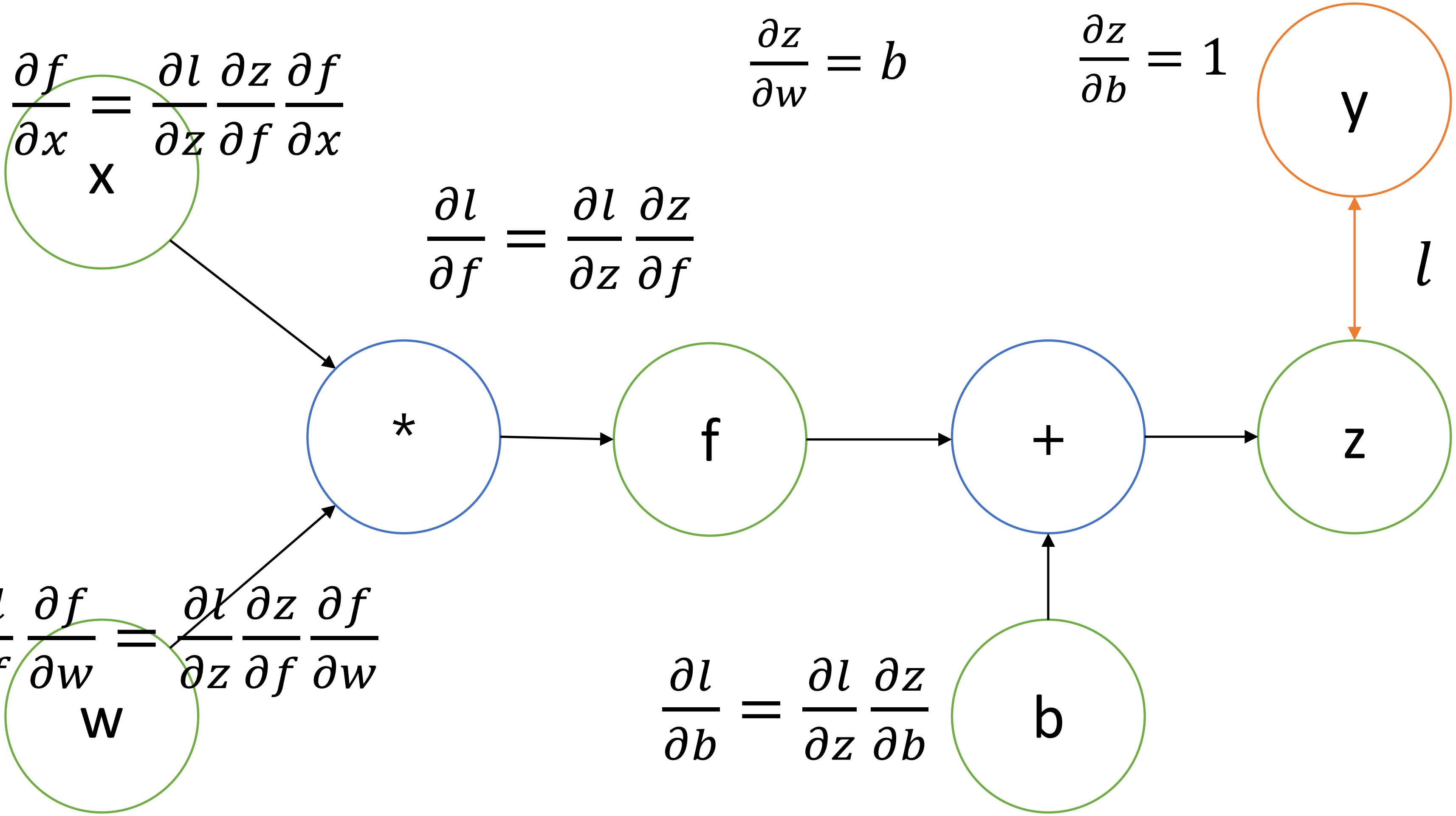
$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial b}$$

Backpropagation

$$x * w = f \qquad f + b = z \qquad l = (y - z)^2$$

$$\frac{\partial f}{\partial x} = w \qquad \frac{\partial z}{\partial f} = 1 \qquad \frac{\partial l}{\partial y} = 2y - 2z$$

$$\frac{\partial z}{\partial w} = b \qquad \frac{\partial z}{\partial b} = 1 \qquad \frac{\partial l}{\partial z} = 2z - 2y$$



$$\frac{\partial l}{\partial x} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial x} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial f}\frac{\partial f}{\partial x}$$

$$\frac{\partial l}{\partial f} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial f}$$

$$\frac{\partial l}{\partial z}$$

$$\frac{\partial l}{\partial w} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial w} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial f}\frac{\partial f}{\partial w}$$

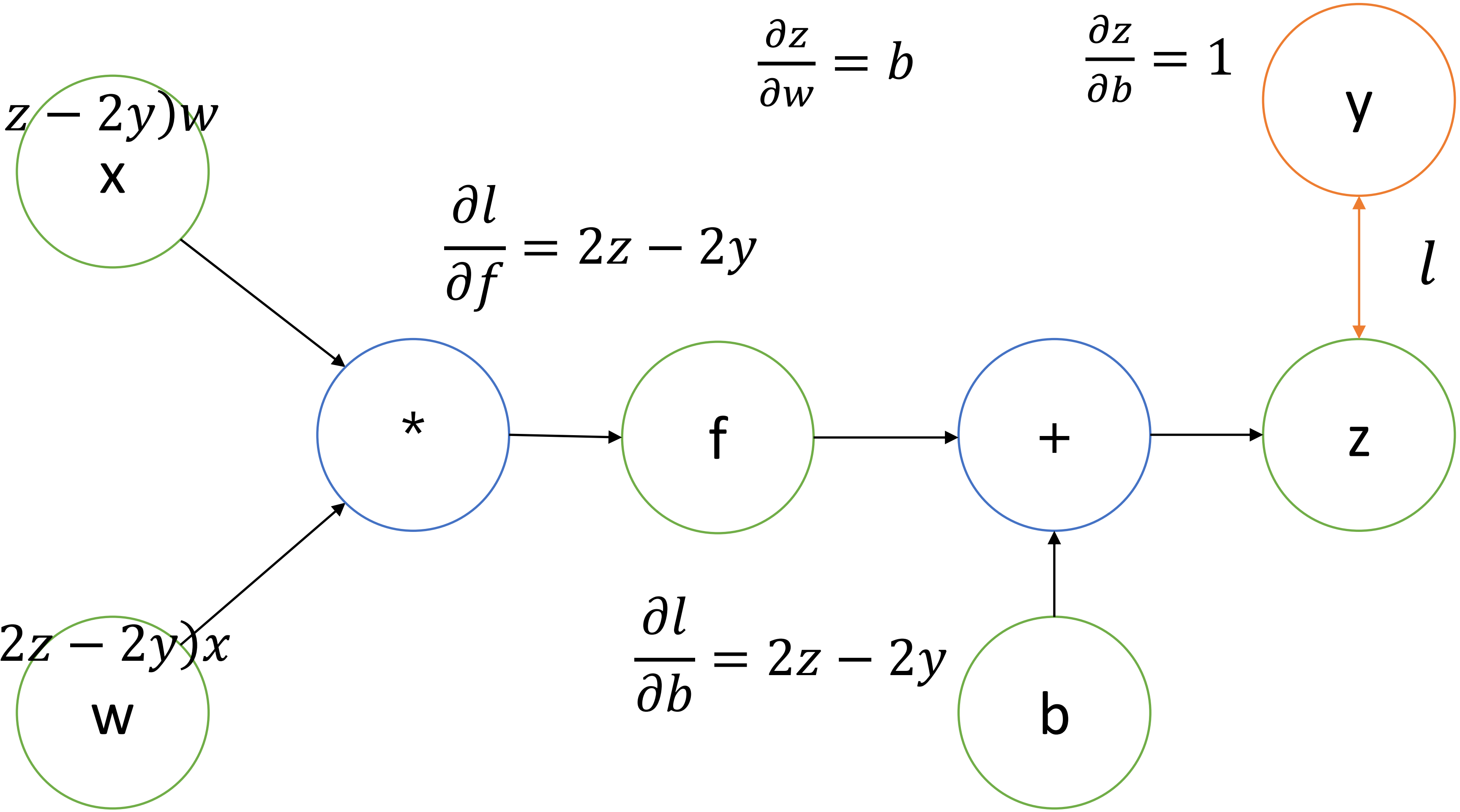
$$\frac{\partial l}{\partial b} = \frac{\partial l}{\partial z}\frac{\partial z}{\partial b}$$

Backpropagation

$$x * w = f \qquad f + b = z \qquad l = (y - z)^2$$

$$\frac{\partial f}{\partial x} = w \qquad \frac{\partial z}{\partial f} = 1 \qquad \frac{\partial l}{\partial y} = 2y - 2z$$

$$\frac{\partial z}{\partial w} = b \qquad \frac{\partial z}{\partial b} = 1 \qquad \frac{\partial l}{\partial z} = 2z - 2y$$



$$\frac{\partial l}{\partial x} = (2z - 2y)w$$

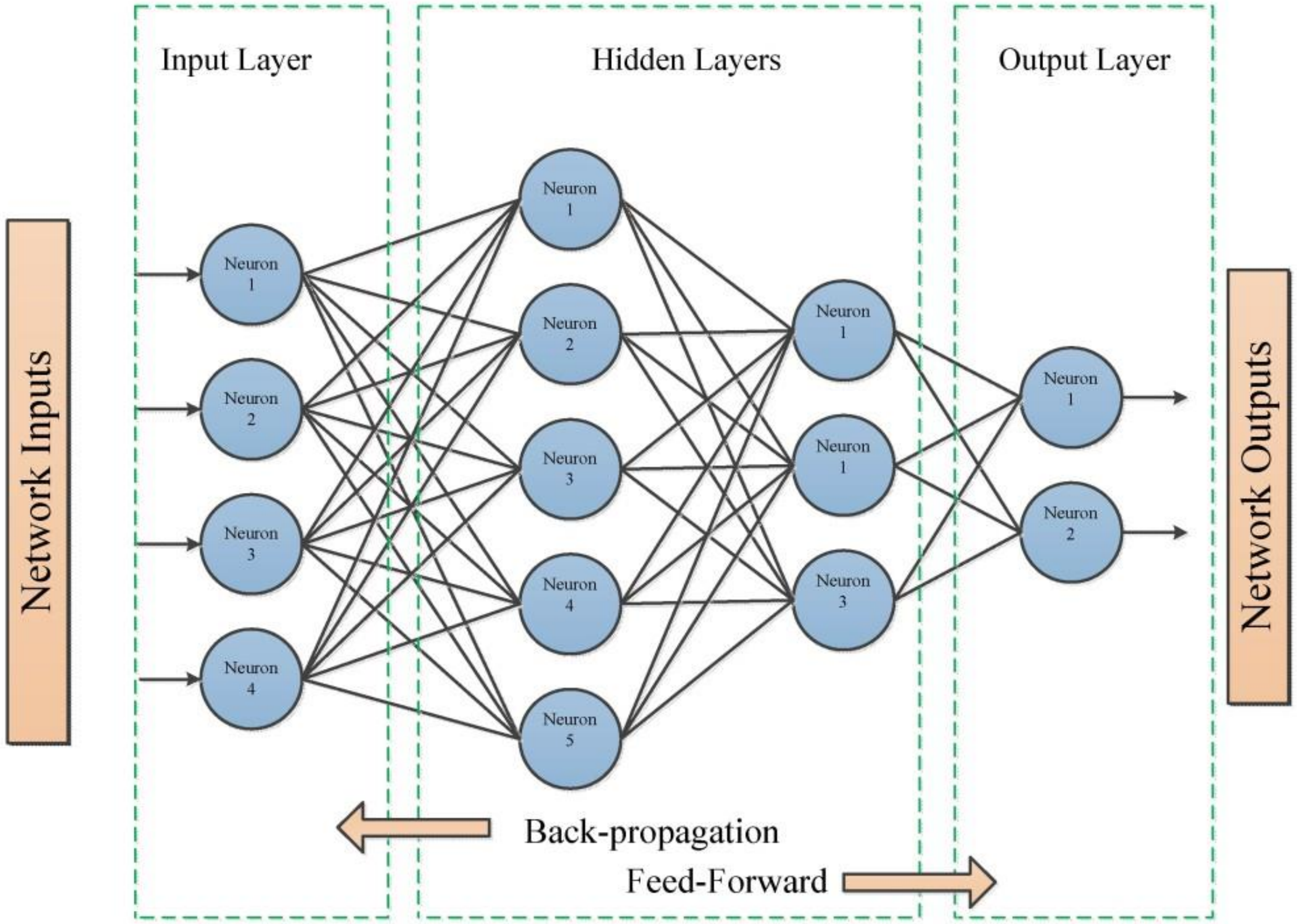
$$\frac{\partial l}{\partial f} = 2z - 2y$$

$$\frac{\partial l}{\partial z} = 2z - 2y$$

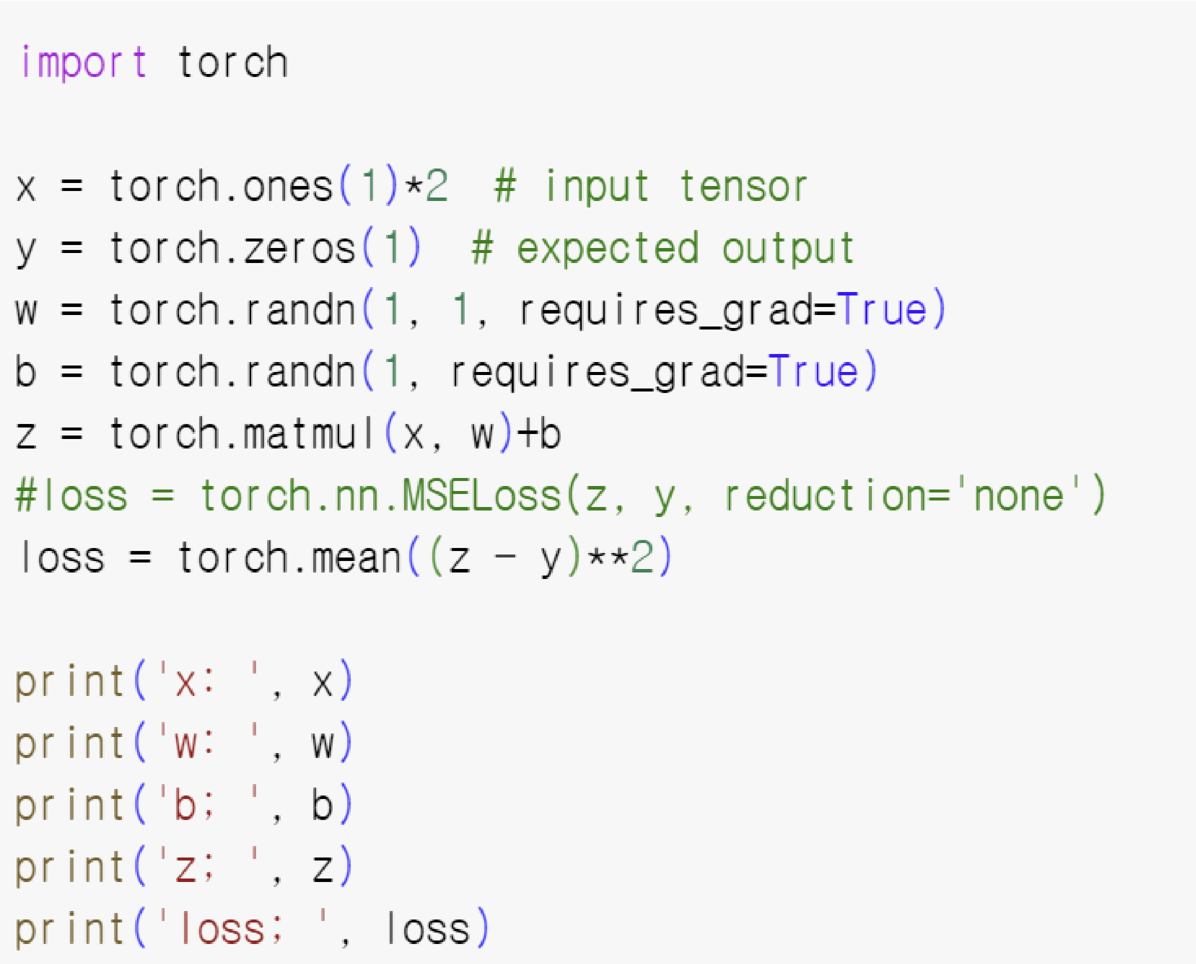
$$\frac{\partial l}{\partial w} = (2z - 2y)x$$

$$\frac{\partial l}{\partial b} = 2z - 2y$$

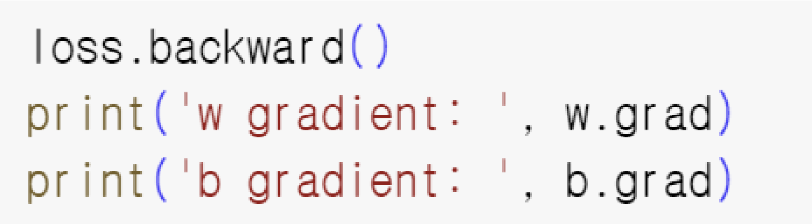
Backpropagation in Deep Neural Network



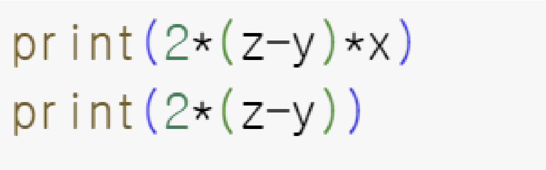
실습



```
import torch

x = torch.ones(1)*2  # input tensor
y = torch.zeros(1)  # expected output
w = torch.randn(1, 1, requires_grad=True)
b = torch.randn(1, requires_grad=True)
z = torch.matmul(x, w)+b
#loss = torch.nn.MSELoss(z, y, reduction='none')
loss = torch.mean((z - y)**2)

print('x: ', x)
print('w: ', w)
print('b; ', b)
print('z; ', z)
print('loss; ', loss)
```

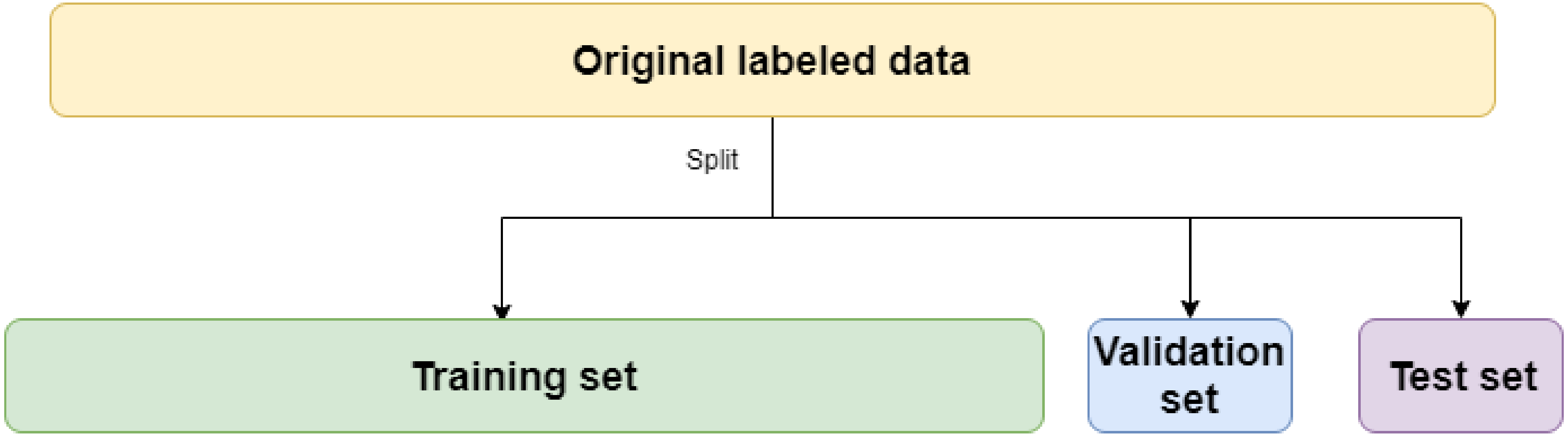


```
loss.backward()
print('w gradient: ', w.grad)
print('b gradient: ', b.grad)
```

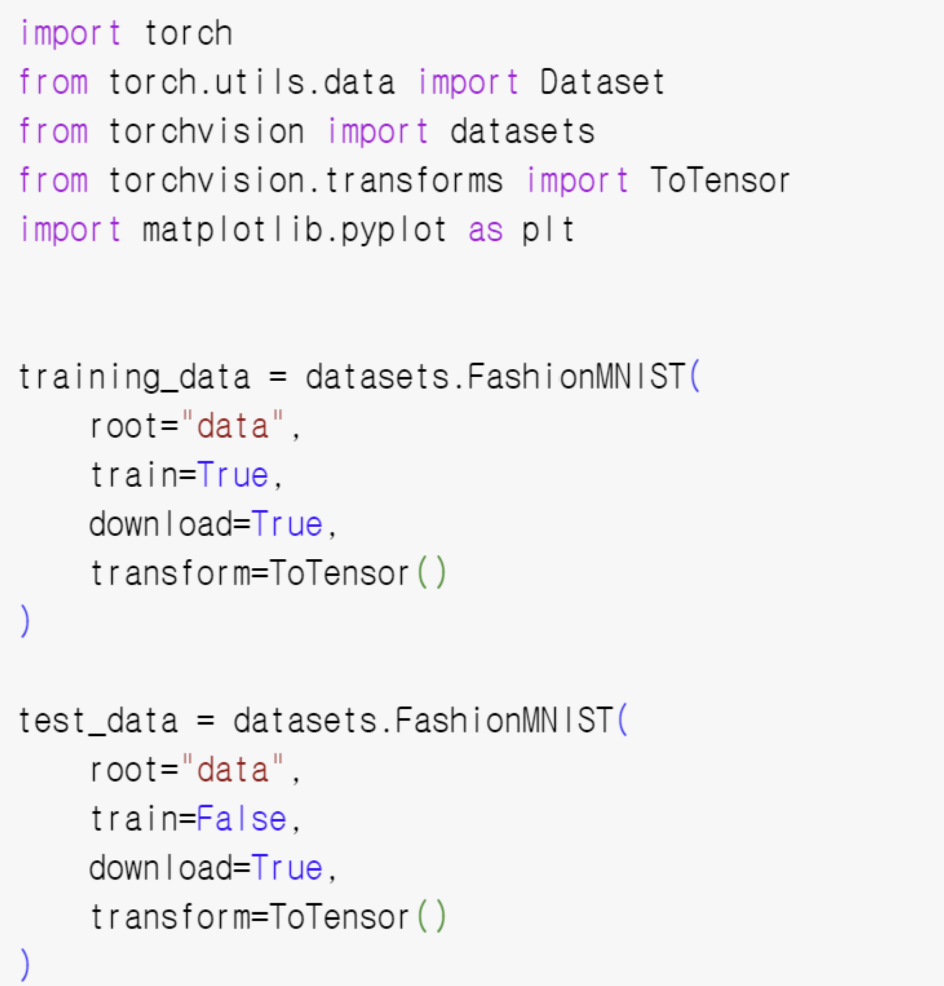


```
print(2*(z-y)*x)
print(2*(z-y))
```

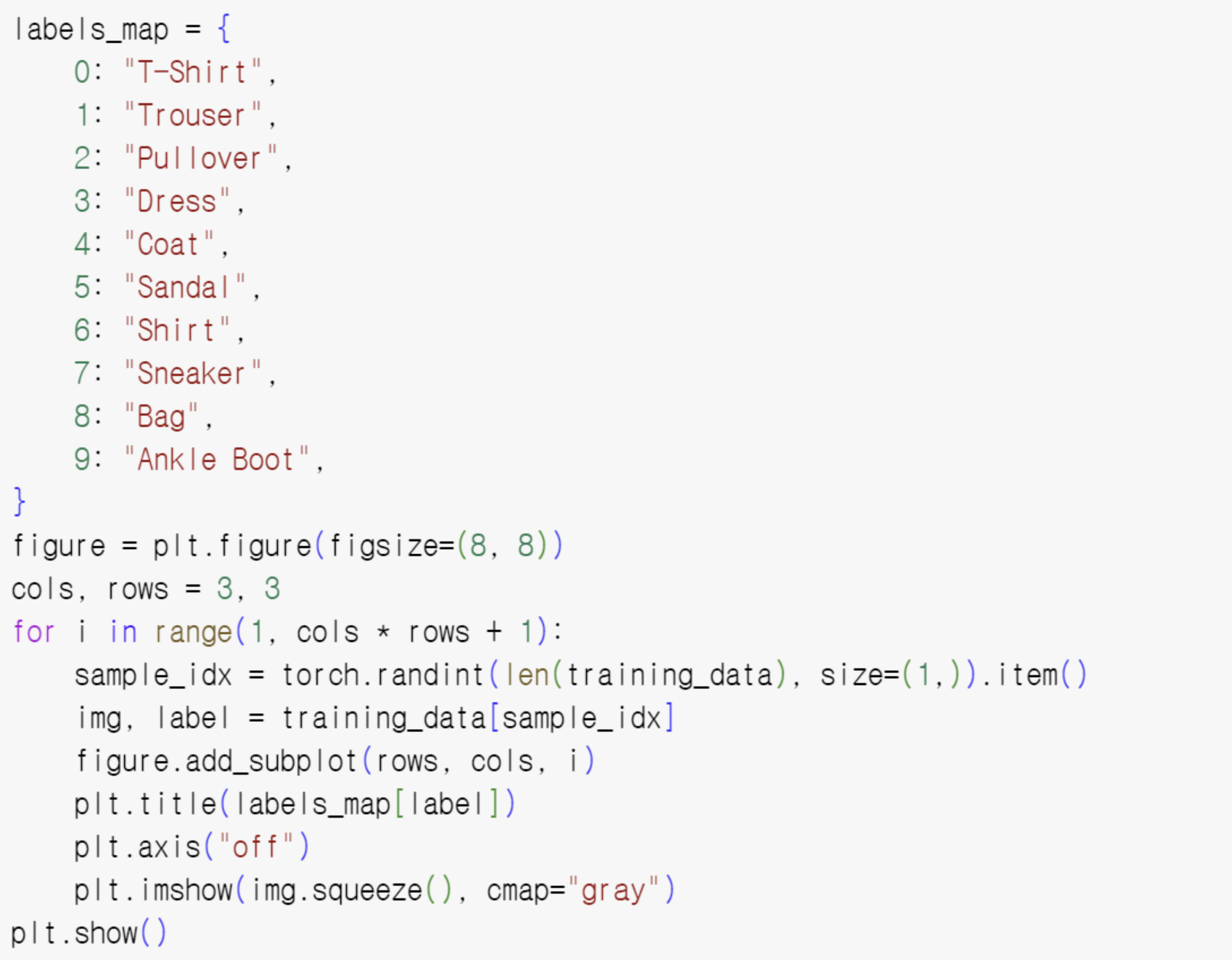
Dataloader


Original labeled data
Split
Training set
Validation set
Test set

실습

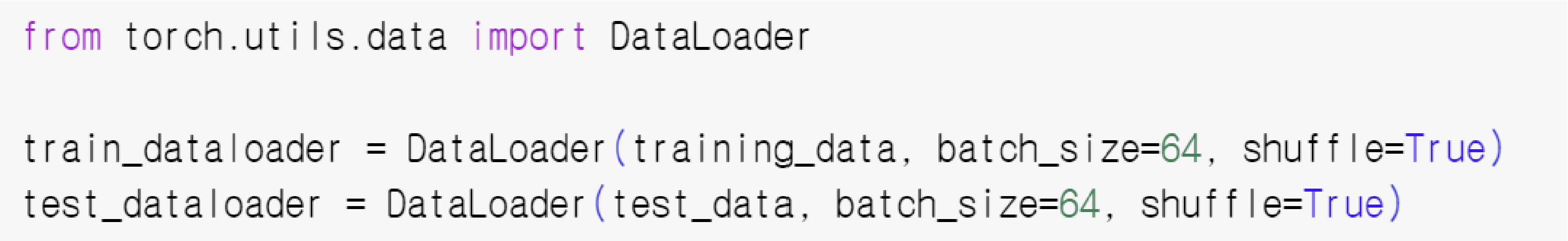


```python
import torch
from torch.utils.data import Dataset
from torchvision import datasets
from torchvision.transforms import ToTensor
import matplotlib.pyplot as plt


training_data = datasets.FashionMNIST(
    root="data",
    train=True,
    download=True,
    transform=ToTensor()
)

test_data = datasets.FashionMNIST(
    root="data",
    train=False,
    download=True,
    transform=ToTensor()
)
```

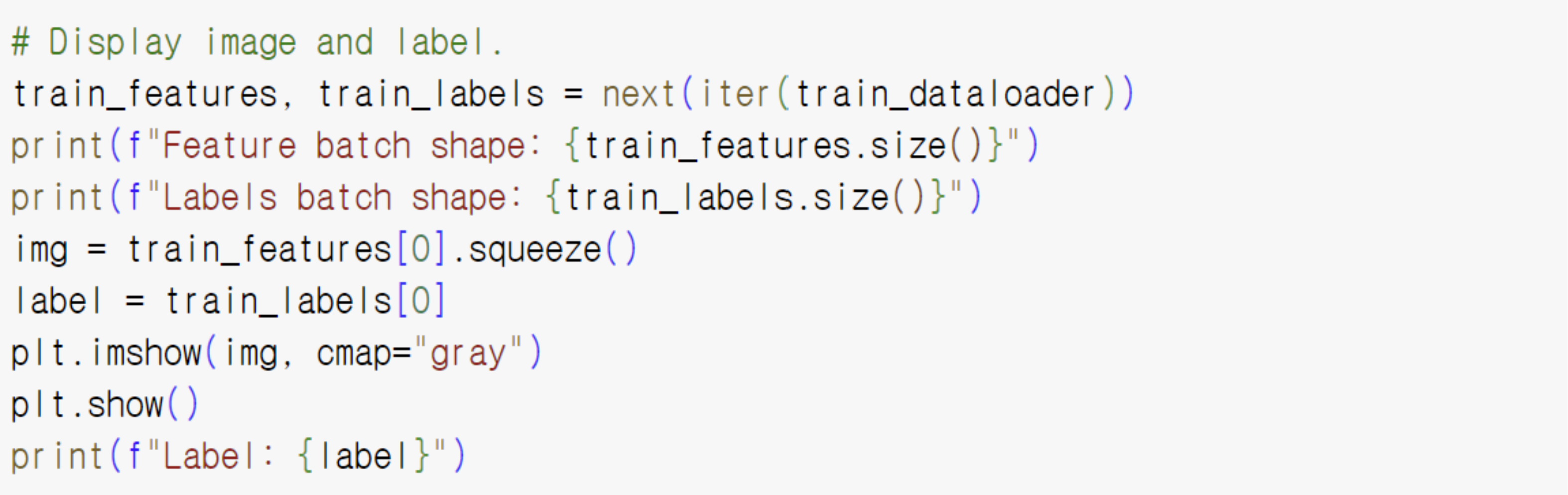


```python
labels_map = {
    0: "T-Shirt",
    1: "Trouser",
    2: "Pullover",
    3: "Dress",
    4: "Coat",
    5: "Sandal",
    6: "Shirt",
    7: "Sneaker",
    8: "Bag",
    9: "Ankle Boot",
}
figure = plt.figure(figsize=(8, 8))
cols, rows = 3, 3
for i in range(1, cols * rows + 1):
    sample_idx = torch.randint(len(training_data), size=(1,)).item()
    img, label = training_data[sample_idx]
    figure.add_subplot(rows, cols, i)
    plt.title(labels_map[label])
    plt.axis("off")
    plt.imshow(img.squeeze(), cmap="gray")
plt.show()
```

실습



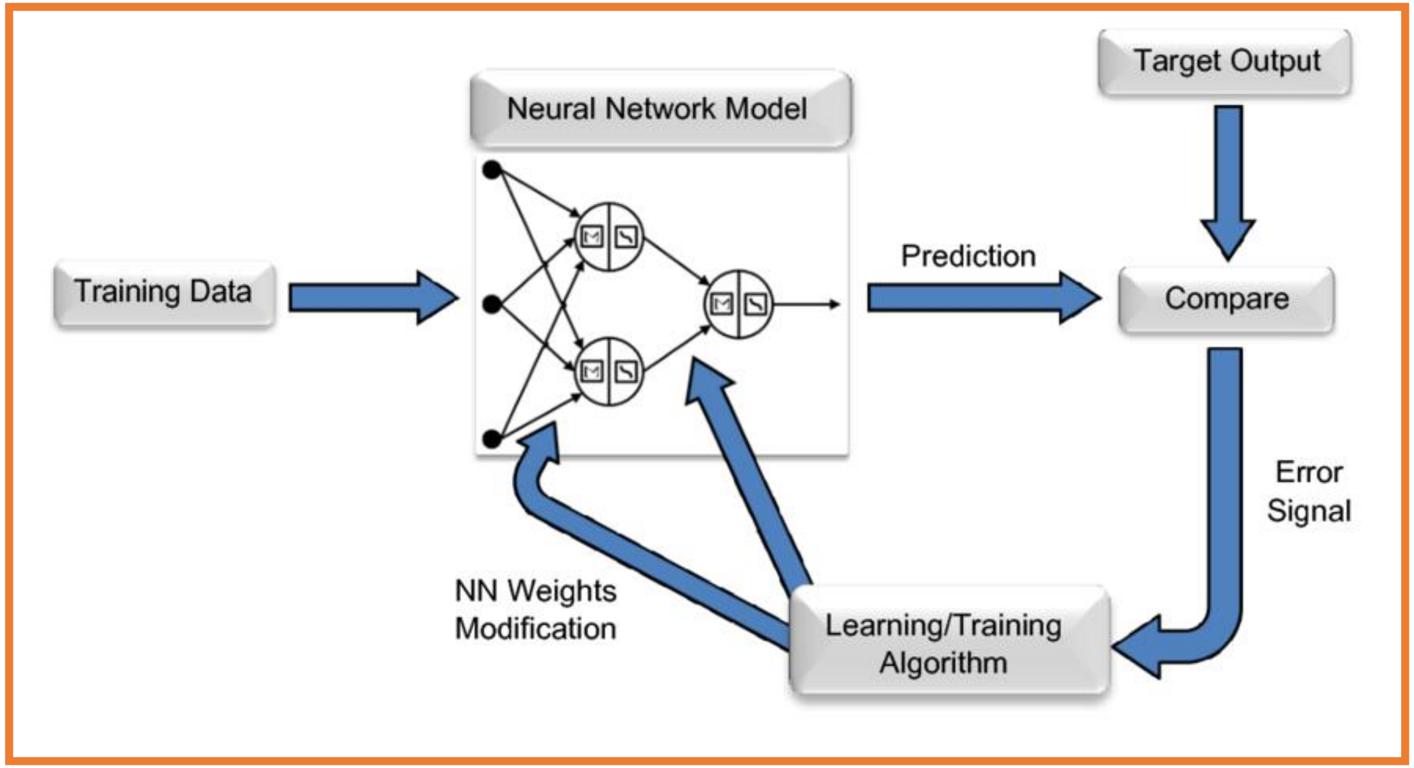
```python
from torch.utils.data import DataLoader

train_dataloader = DataLoader(training_data, batch_size=64, shuffle=True)
test_dataloader = DataLoader(test_data, batch_size=64, shuffle=True)
```



```python
# Display image and label.
train_features, train_labels = next(iter(train_dataloader))
print(f"Feature batch shape: {train_features.size()}")
print(f"Labels batch shape: {train_labels.size()}")
img = train_features[0].squeeze()
label = train_labels[0]
plt.imshow(img, cmap="gray")
plt.show()
print(f"Label: {label}")
```

예고

- Training



Target Output
Neural Network Model
Training Data
Prediction
Compare
Error
Signal
NN Weights
Modification
Learning/Training
Algorithm

감사합니다

시험 잘 보세요!!

출처

- https://pytorch.org/tutorials/beginner/basics/autogradqs_tutorial.html
- https://pytorch.org/tutorials/beginner/basics/data_tutorial.html