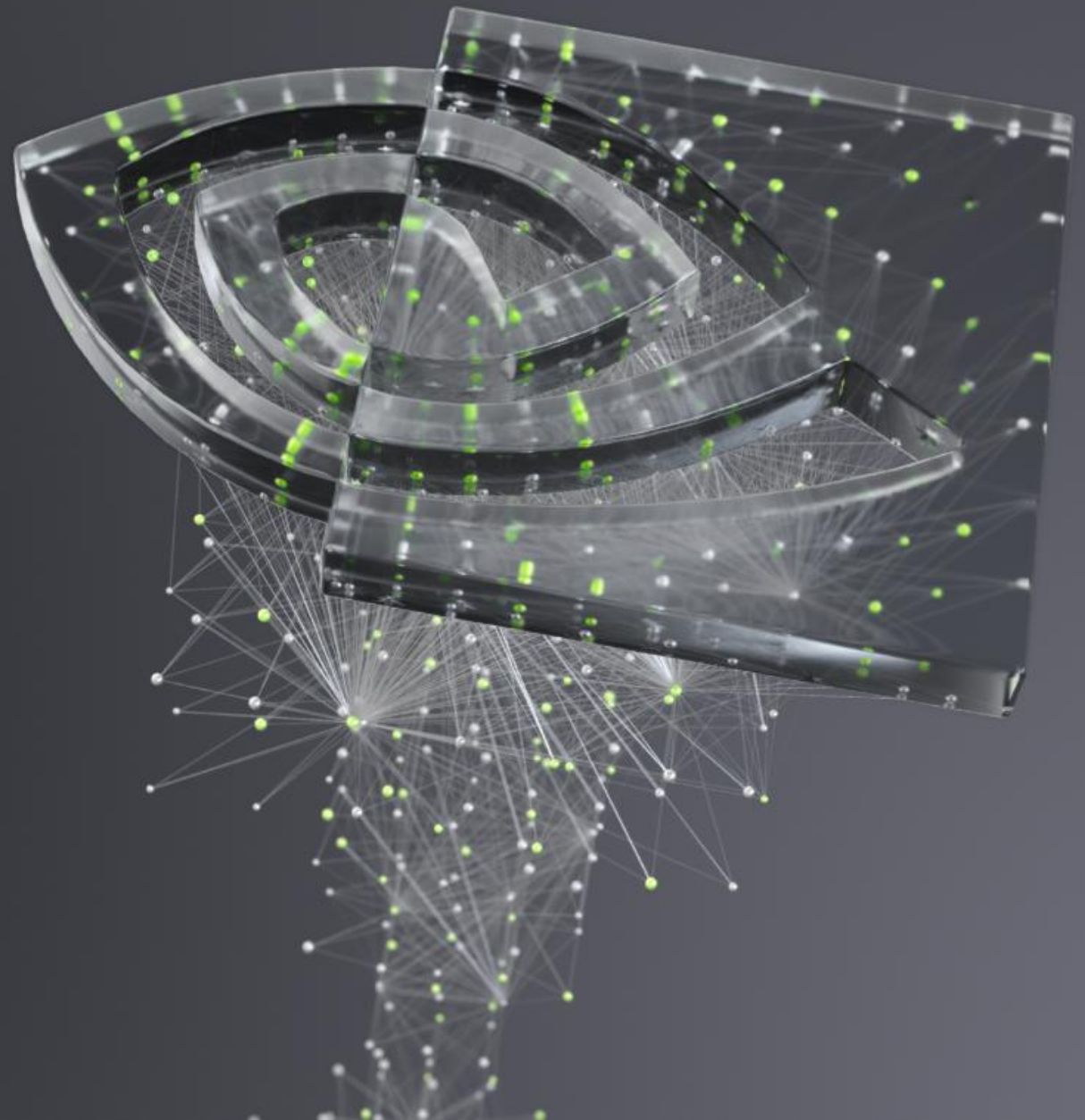
NVIDIA | DEEP LEARNING INSTITUTE



딥러닝의 기초

2부: 뉴럴 네트워크의 트레이닝 방식

자료 및 발표: DLI Ambassador 박제윤

목차

목차 – 2부

- 데이터 준비
- 뉴런의 학습 과정
- 뉴런에서 네트워크로
- 활성화 함수
- 과적합
- 회귀에서 분류로

목차 – 2부

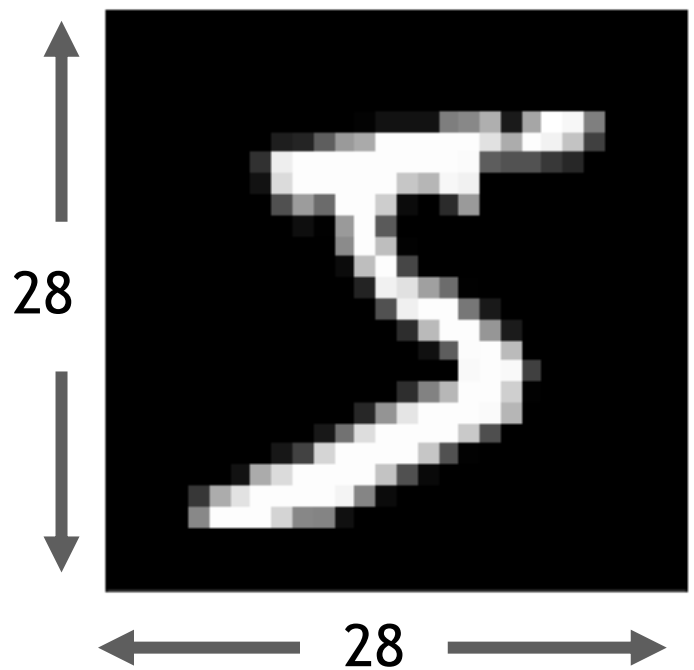
- **데이터 준비**
- 뉴런의 학습 과정
- 뉴런에서 네트워크로
- 활성화 함수
- 과적합
- 회귀에서 분류로

데이터 준비

데이터 준비 (DATA PREPARATION)

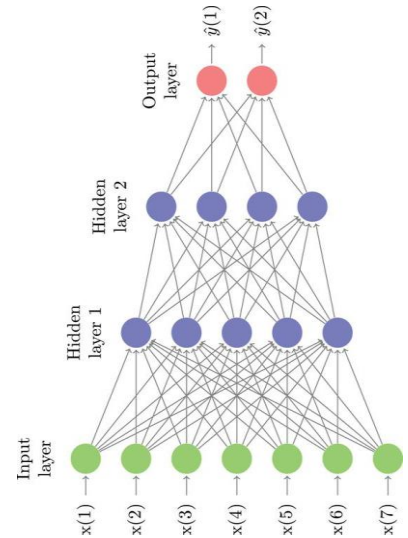
배열(벡터)을 통한 입력

- 데이터 준비 (Data preparation)
- 데이터 평탄화 (Flattening)
- 데이터 정규화 (Normalizing)
- 범주형 인코딩 (Categorical encoding)

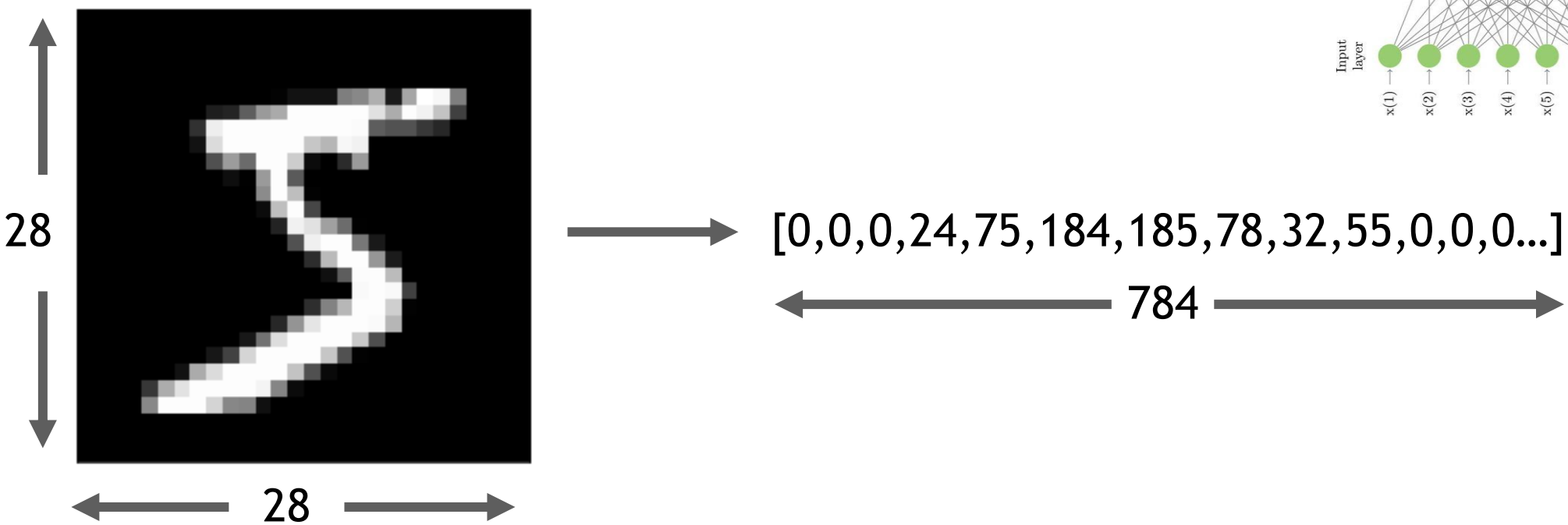


데이터 준비 (DATA PREPARATION)

배열(벡터)을 통한 입력



1) Flattening: 28x28 이미지를 길이가 784x1 인 배열로 만들음



데이터 준비 (DATA PREPARATION)

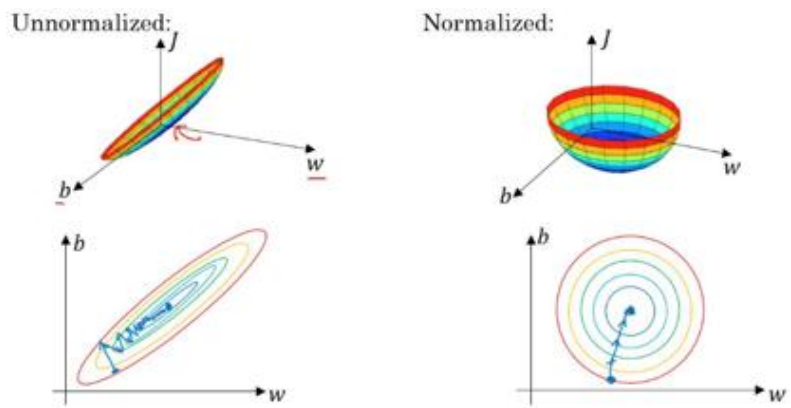
배열(벡터)을 통한 입력

2) Normalizing: 픽셀의 최대값인 255로 나누어 데이터를 0-1 사이로 정규화

- 학습시간이 줄어들고 local minimum에 빠질 가능성이 낮아짐

[0,0,0,24,75,184,185,78,32,55,0,0,0...] / 255

784



데이터 준비 (DATA PREPARATION)

배열(벡터)을 통한 입력

3) Categorical encoding: (1) Label encoding, and (2) One-hot encoding

3-1) Label encoding: 레이블에 숫자를 할당해주는 레이블링 방식

- 언제 쓸까? → 레이블 순서 의미를 넣고 싶을때

- 다시말해, 레이블 순서의 의미가 없어야 되는 task는 쓰면 안됨

- e.g.) MNIST task에서 5를 맞출때 4라고 대답하는게 9라고 대답한 것보다 낫다고 할 수 있을까?

Size
S
XL
M
S
XL
L
L

→

Size
0
3
1
0
3
2
2

데이터 준비 (DATA PREPARATION)

배열(벡터)을 통한 입력

3) Categorical encoding: (1) Label encoding, and (2) One-hot encoding

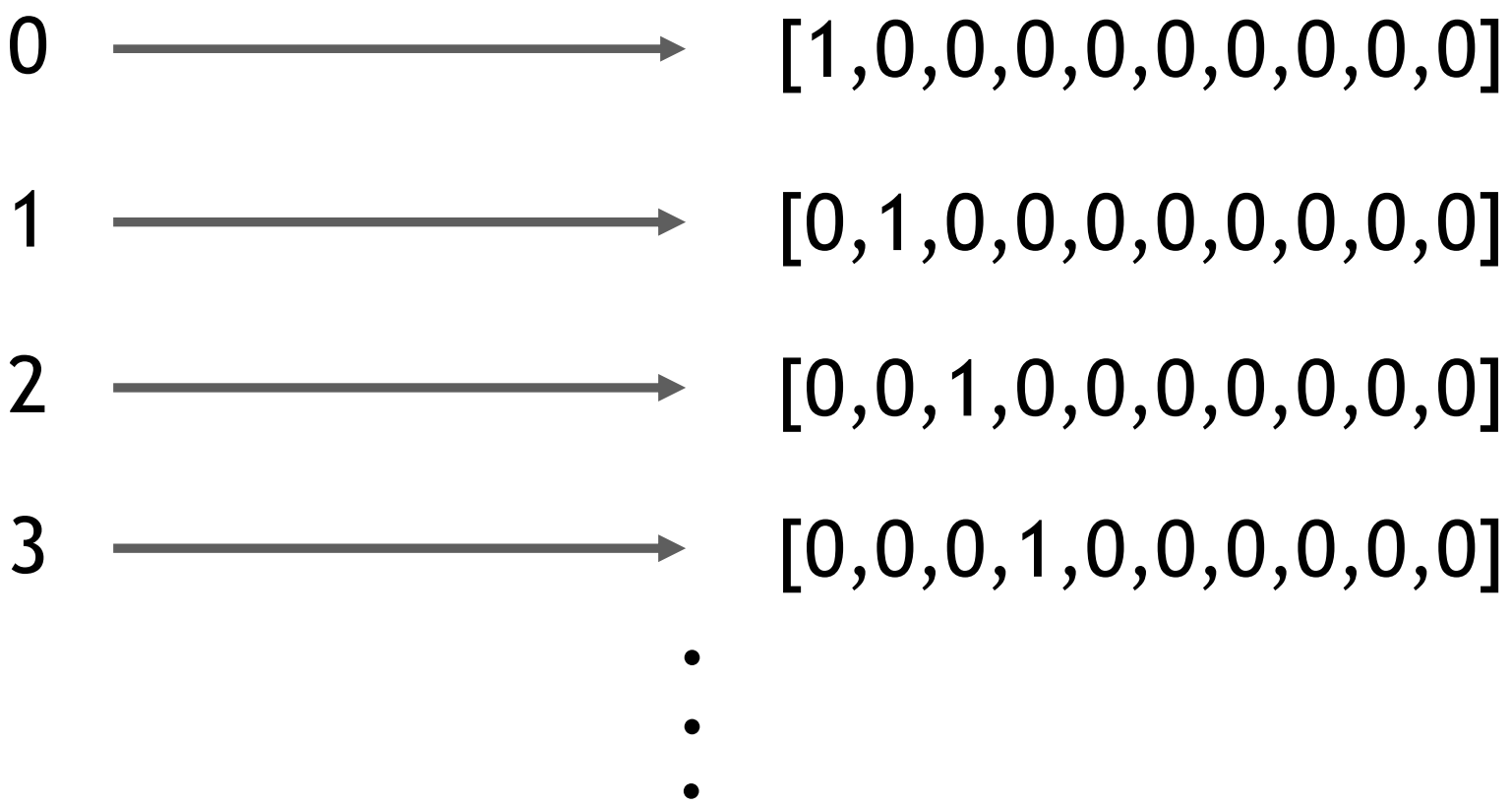
3-2) One-hot encoding: 데이터의 연속성 및 순서정보를 없애기 위해 하나의 값만 1을 가지고 나머지는 모두 0을 가지는 벡터형 데이터

- 1의 위치가 어느 범주에 속하는지 의미할 수 있음

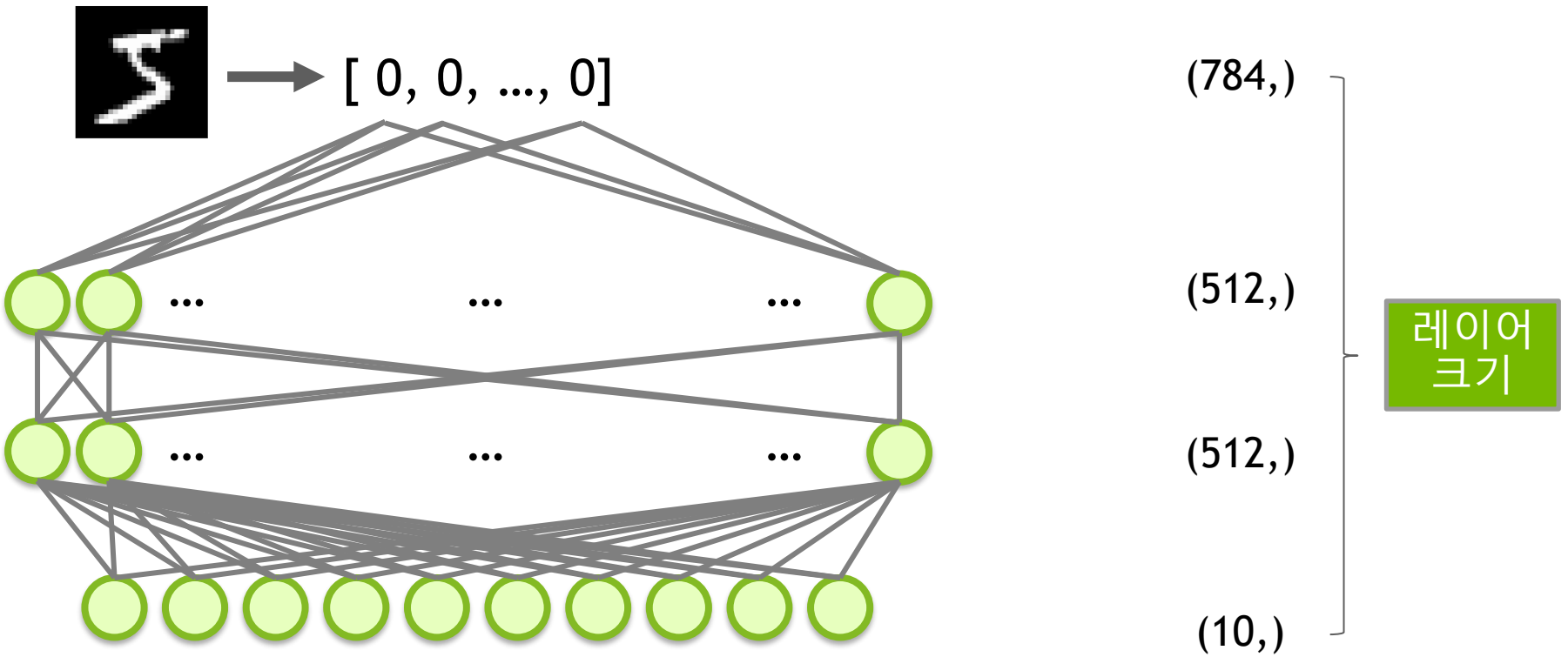
Actual Color	Is Red?	Is Blue?	Is Green?
Red	1	0	0
Green	0	0	1
Blue	0	1	0
Green	0	0	1

데이터 준비 (DATA PREPARATION)

범주형(categorical)으로 타깃팅



트레이닝되지 않은 모델




시작하겠습니다!
1) 01_mnist.ipynb

목차 – 2부

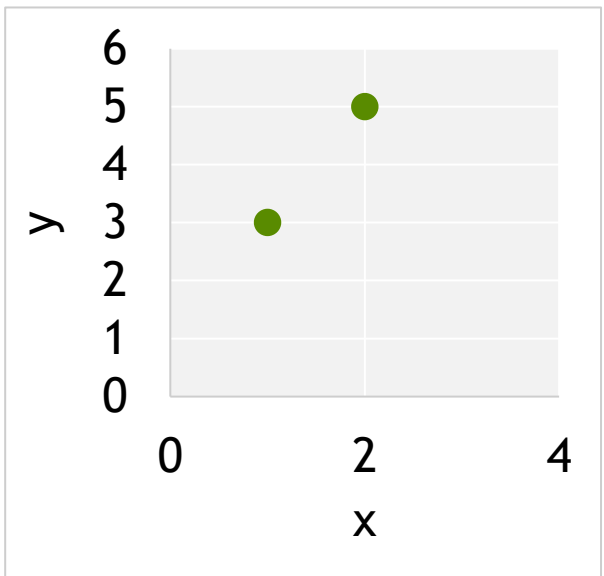
- 데이터 준비
- **뉴런의 학습 과정**
- 뉴런에서 네트워크로
- 활성화 함수
- 과적합
- 회귀에서 분류로

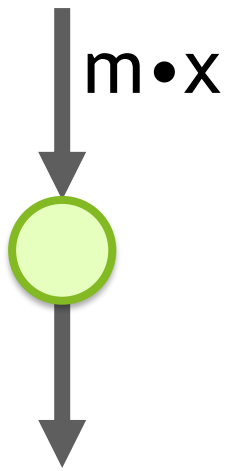
뉴런의 학습 과정

뉴런의 학습 과정

$$y = mx + b$$

x	y
1	3
2	5





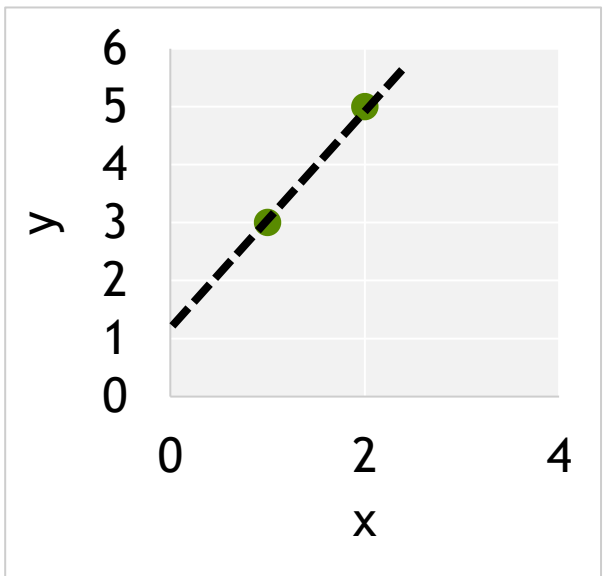
$m = ?$

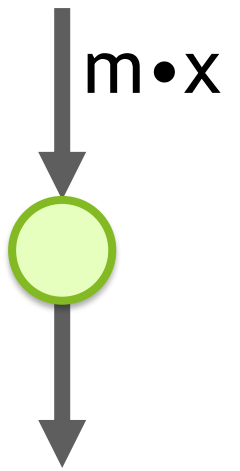
$b = ?$

뉴런의 학습 과정

$$y = mx + b$$

x	y
1	3
2	5





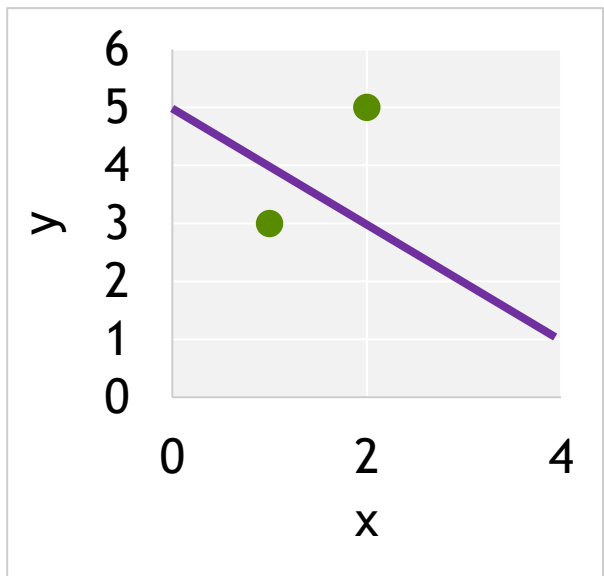
$m = ?$

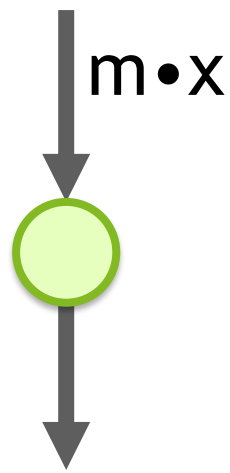
$b = ?$

뉴런의 학습 과정

$$y = mx + b$$

x	y	$\hat{y}$
1	3	4
2	5	3





무작위로 시작

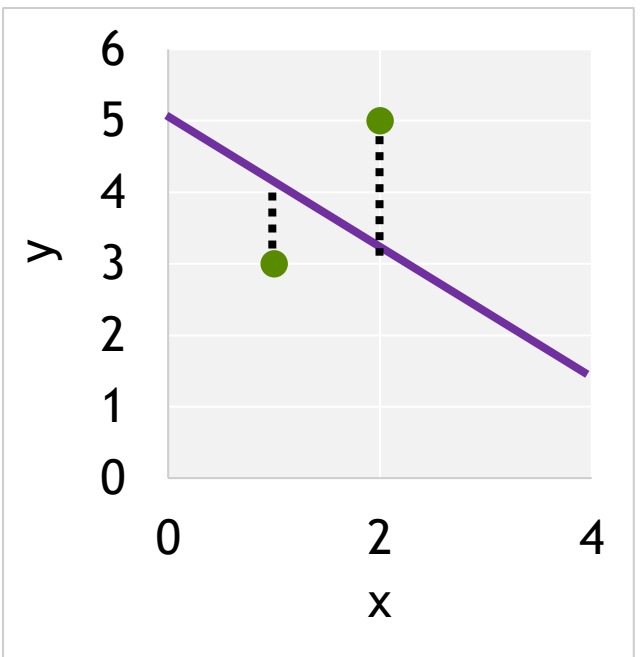
$$m = -1$$

$$b = 5$$

뉴런의 학습 과정

$$y = mx + b$$

x	y	$\hat{y}$	err^2
1	3	4	1
2	5	3	4
		MSE =	2.5
		RMSE =	1.6



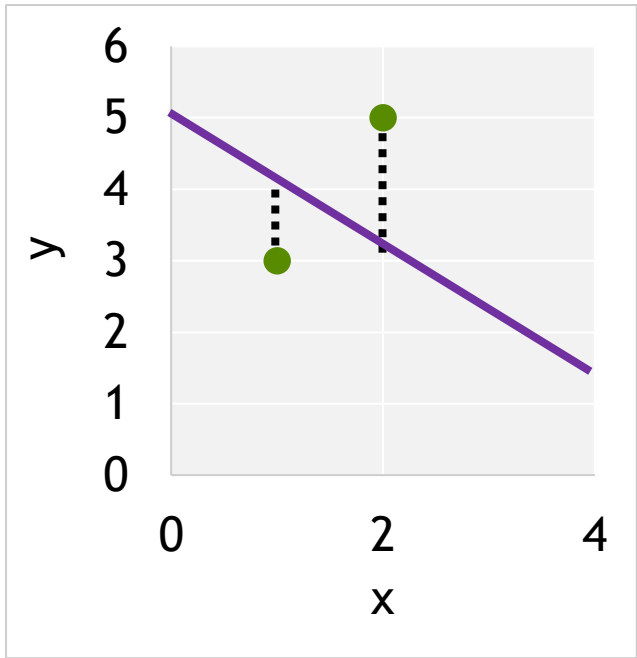
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2}$$

뉴런의 학습 과정

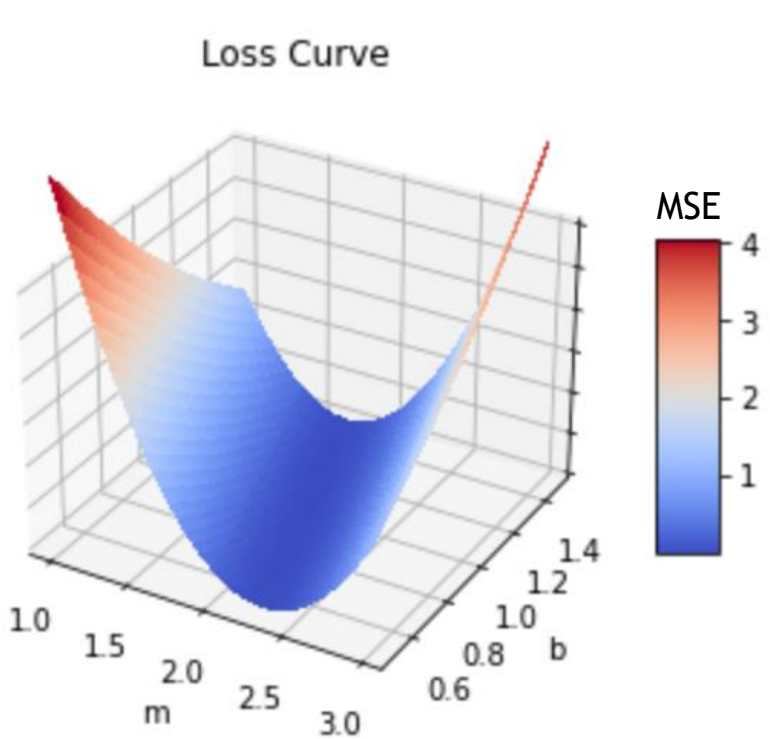
$$y = mx + b$$

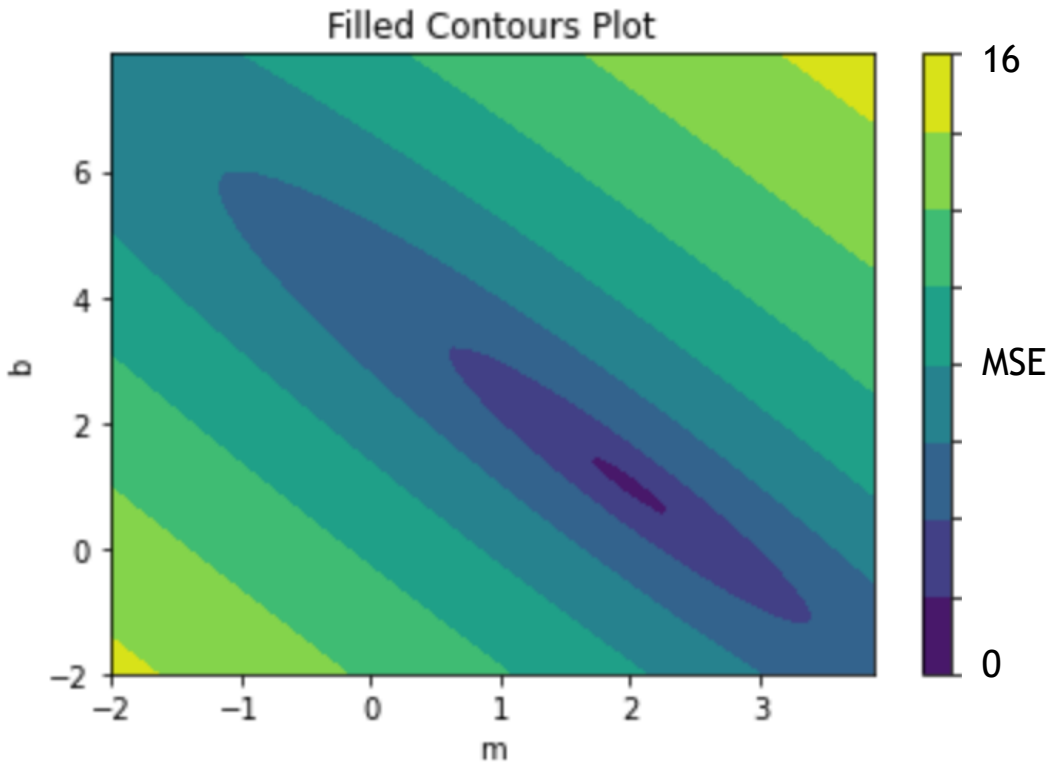
x	y	$\hat{y}$	err^2
1	3	4	1
2	5	3	4
		MSE =	2.5
		RMSE =	1.6



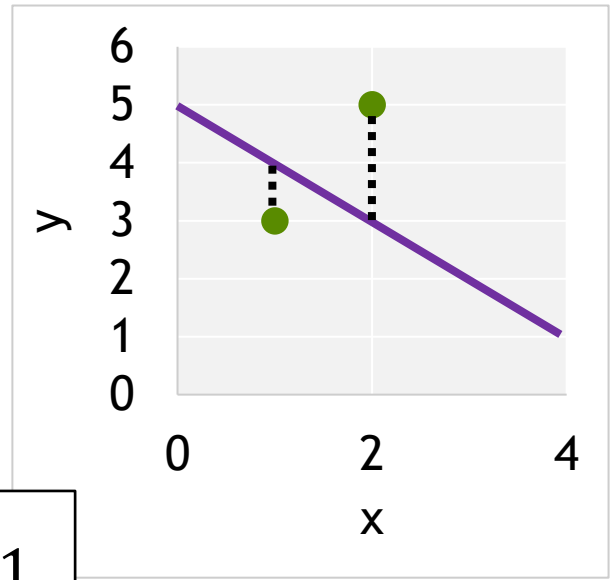

```python
data = [(1, 3), (2, 5)]
m = -1
b = 5

def get_mse(data, m, b):
    """Calculates Mean Square Error"""
    n = len(data)
    squared_error = 0
    for x, y in data:
        # Find predicted y
        y_hat = m*x+b
        # Square difference between
        # prediction and true value
        squared_error += (
            y - y_hat)**2
    # Get average squared difference
    mse = squared_error / n
    return mse
```

손실 곡선 (THE LOSS CURVE)



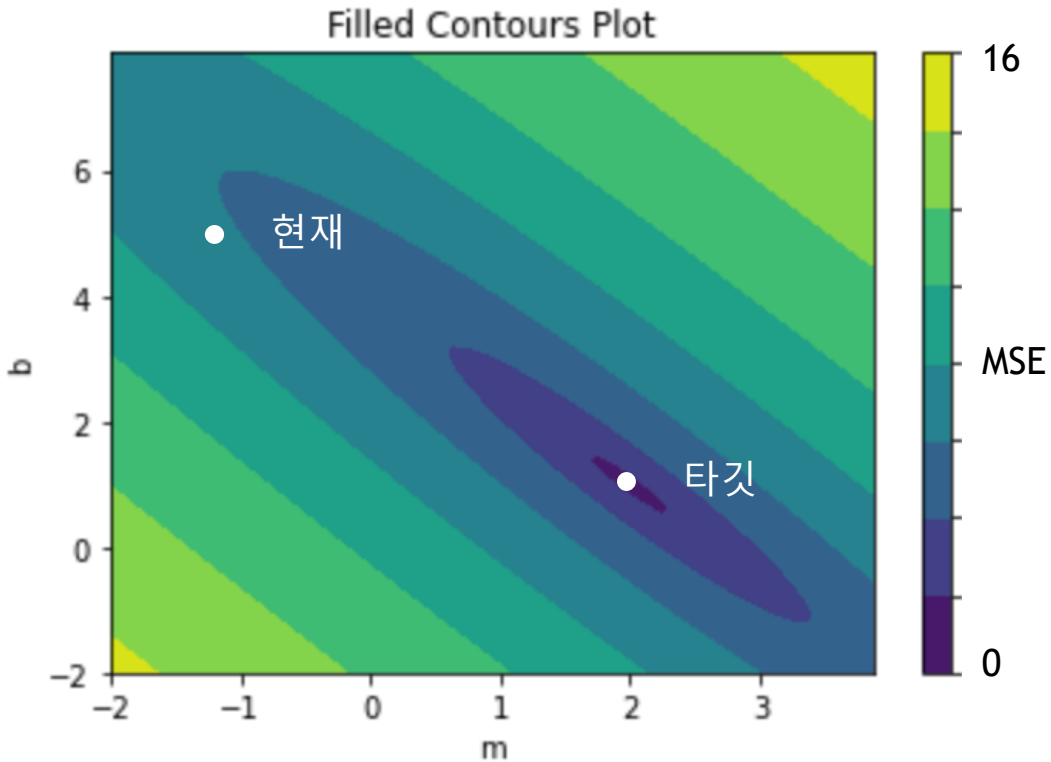


손실 곡선 (THE LOSS CURVE)

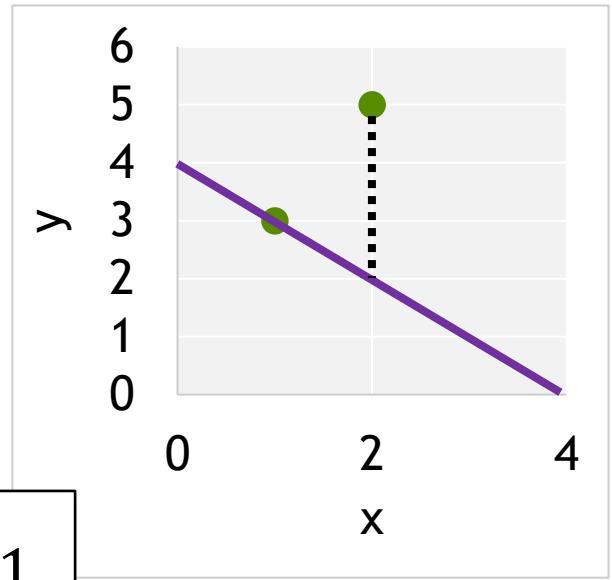


$m = -1$

$b = 5$

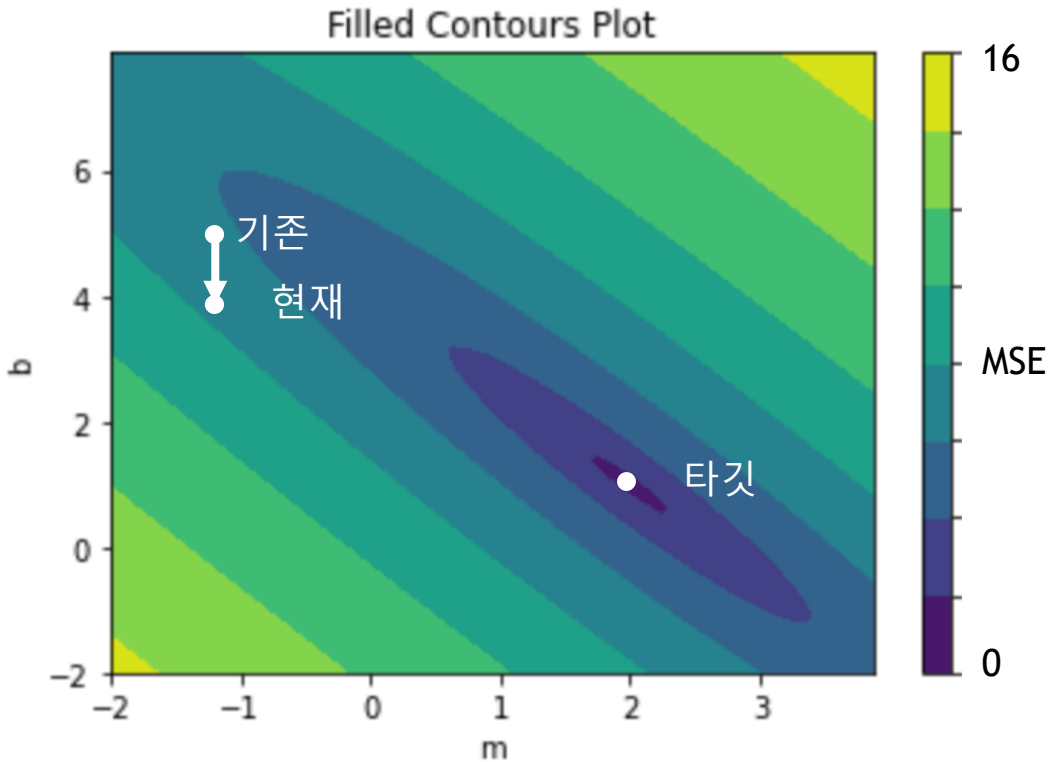


손실 곡선 (THE LOSS CURVE)

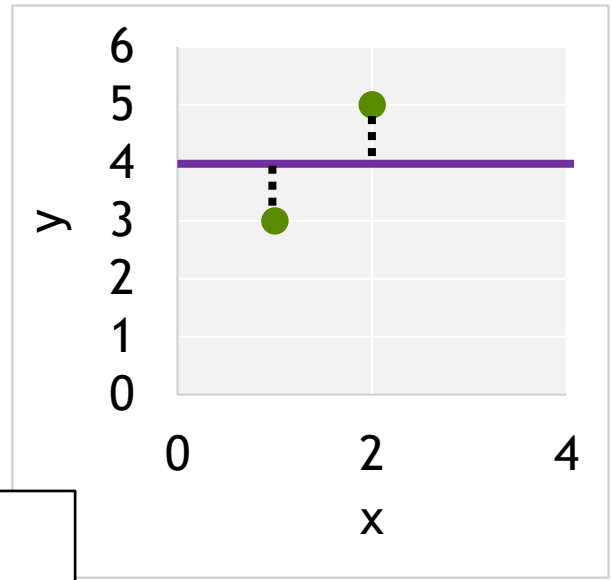


$m = -1$

$\mathrm{b} = 4$

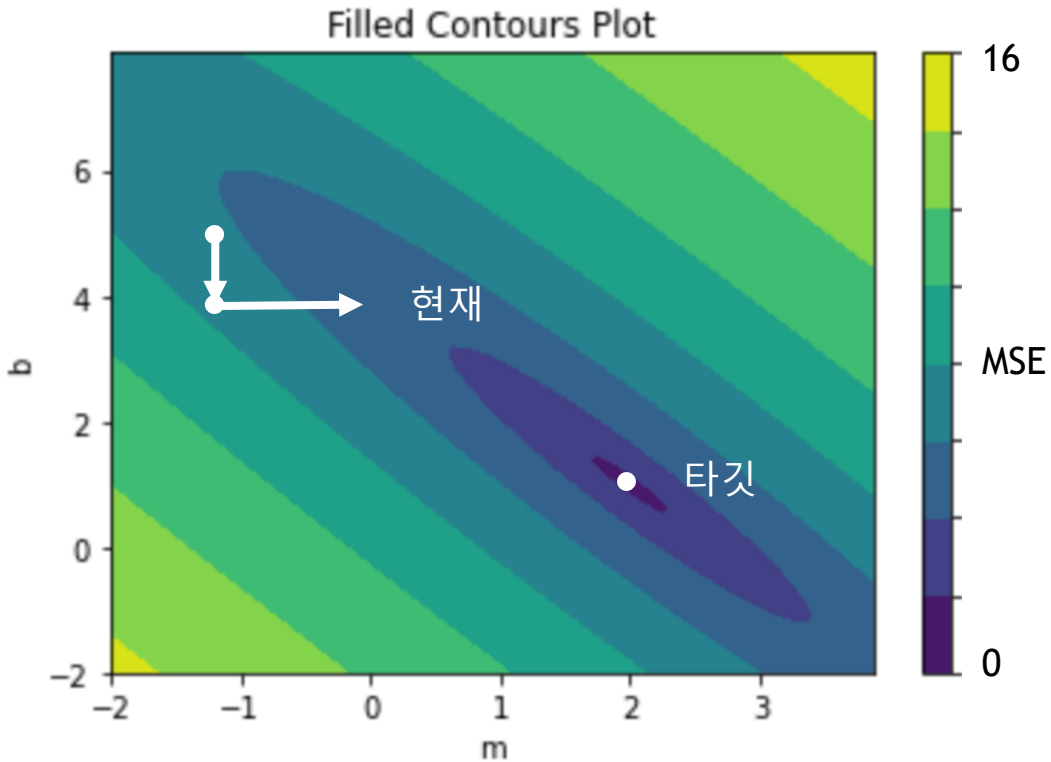


손실 곡선 (THE LOSS CURVE)



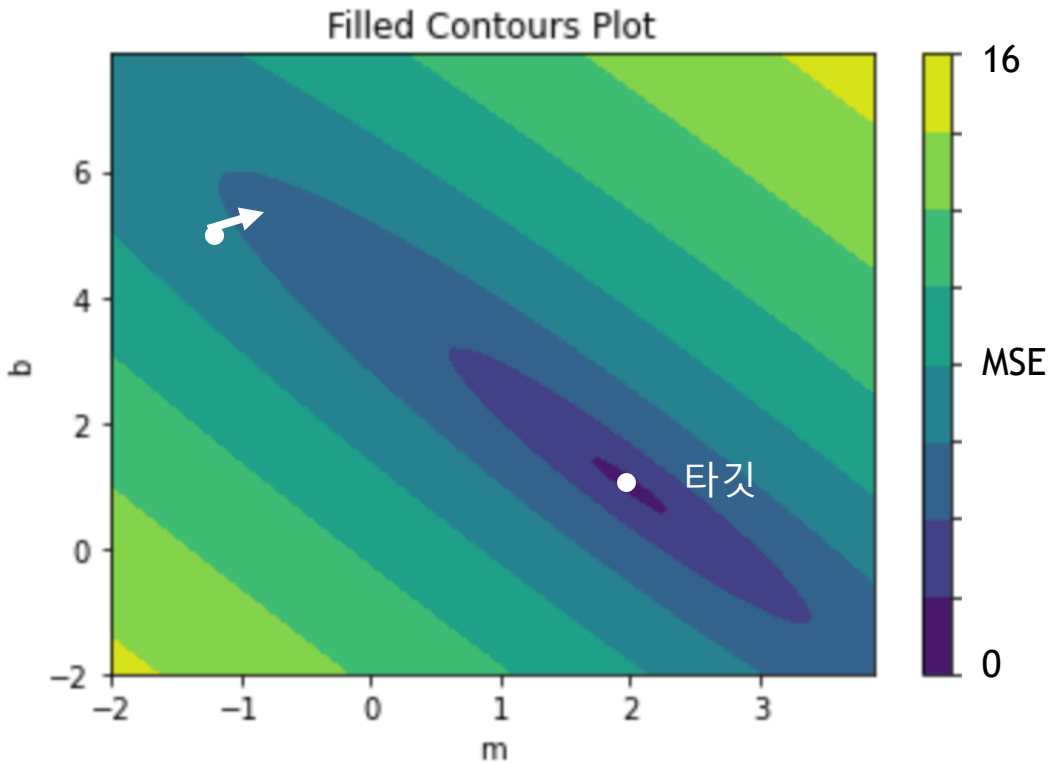
$m = 0$

$b = 4$



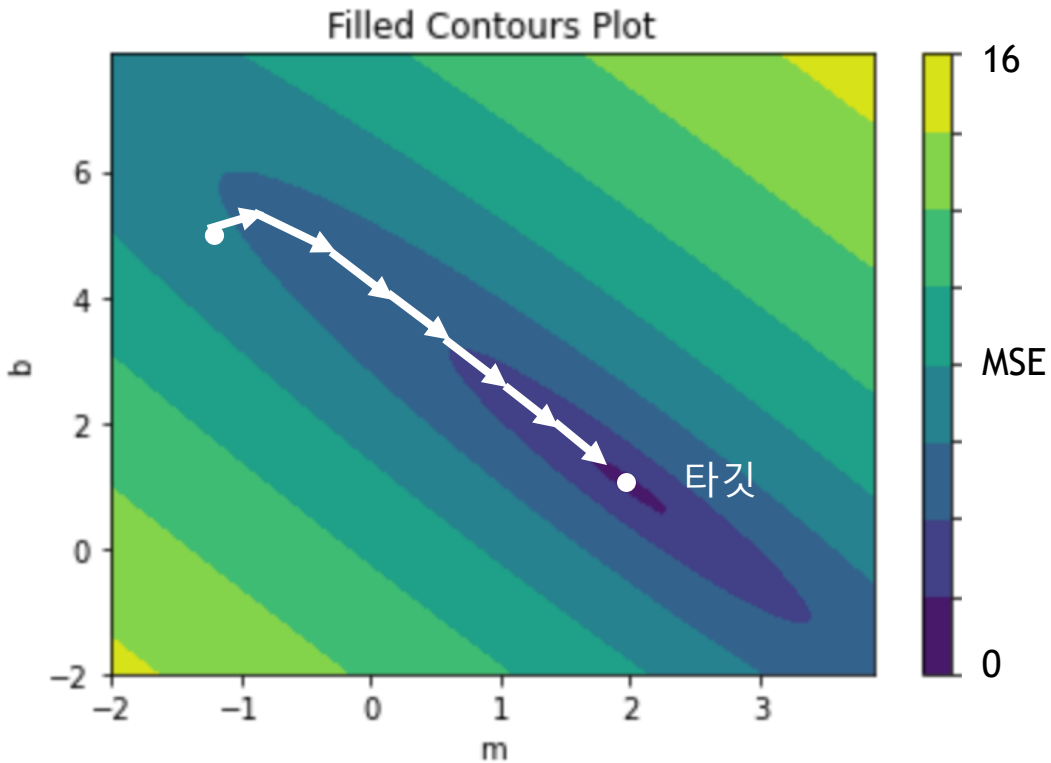
손실 곡선 (THE LOSS CURVE)

그래디언트 (Gradient)	어떤 방향으로 손실이 가장 많이 감소하는가
λ: 학습률 (Learning rate)	이동 거리
에포크 (Epoch)	전체 데이터세트로 모델 업데이트
배치 (Batch)	전체 데이터세트 중 특정 샘플
단계 (Step)	가중치 매개변수에 대한 업데이트

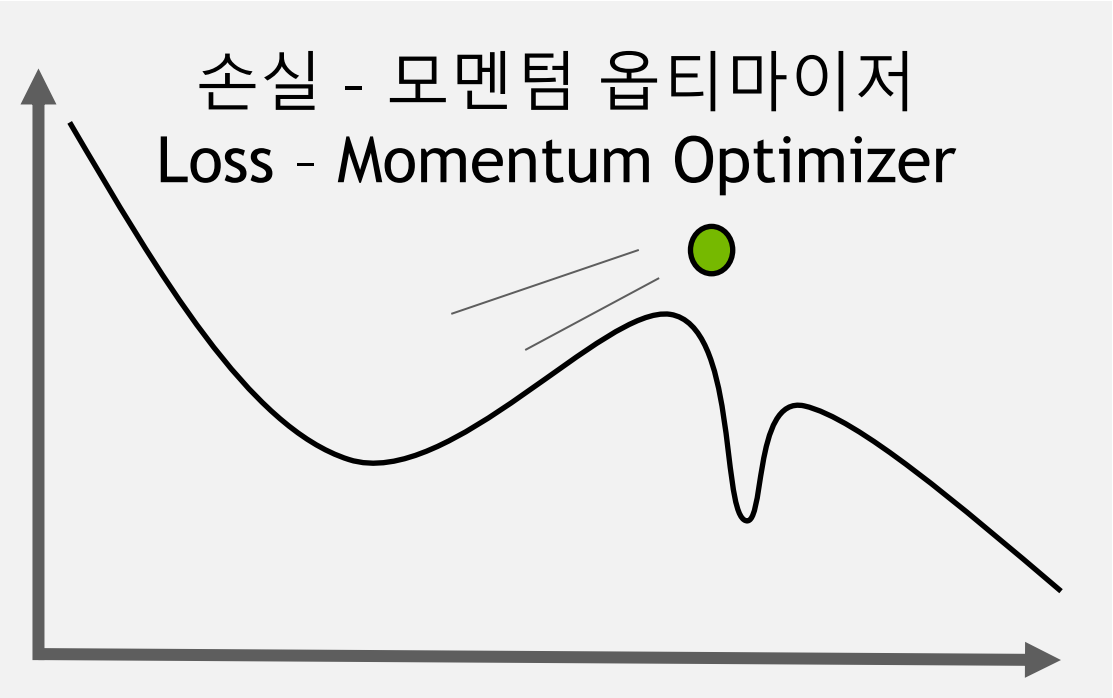


손실 곡선 (THE LOSS CURVE)

그래디언트 (Gradient)	어떤 방향으로 손실이 가장 많이 감소하는가
λ: 학습률 (Learning rate)	이동 거리
에포크 (Epoch)	전체 데이터세트로 모델 업데이트
배치 (Batch)	전체 데이터세트 샘플
단계 (Step)	가중치 매개변수에 대한 업데이트



옵티마이저 (OPTIMIZERS)



- Adam
- Adagrad
- RMSprop
- SGD

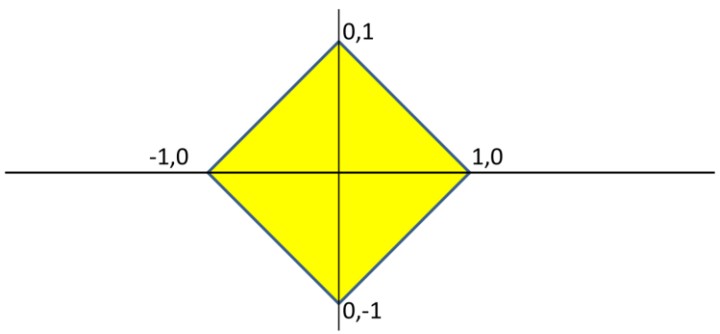
목차 – 2부

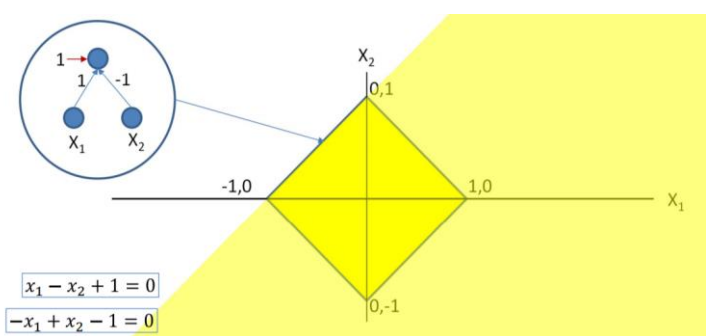
- 데이터 준비
- 뉴런의 학습 과정
- **뉴런에서 네트워크로**
- 활성화 함수
- 과적합
- 회귀에서 분류로

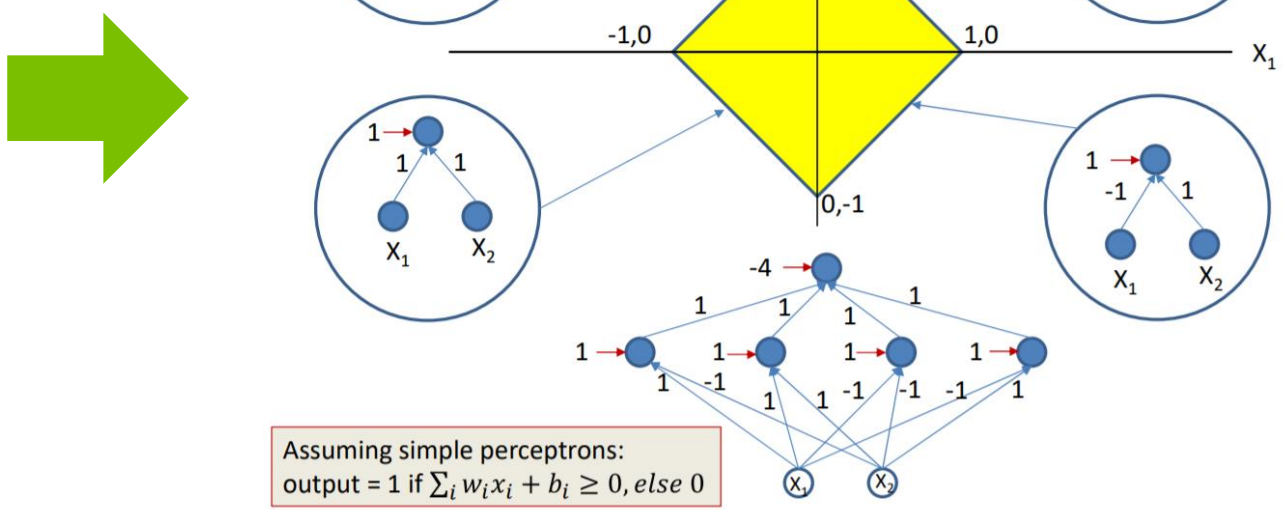

뉴런에서 네트워크로

Detour: Deep Learning

1. 퍼셉트론 ≫ decision boundary

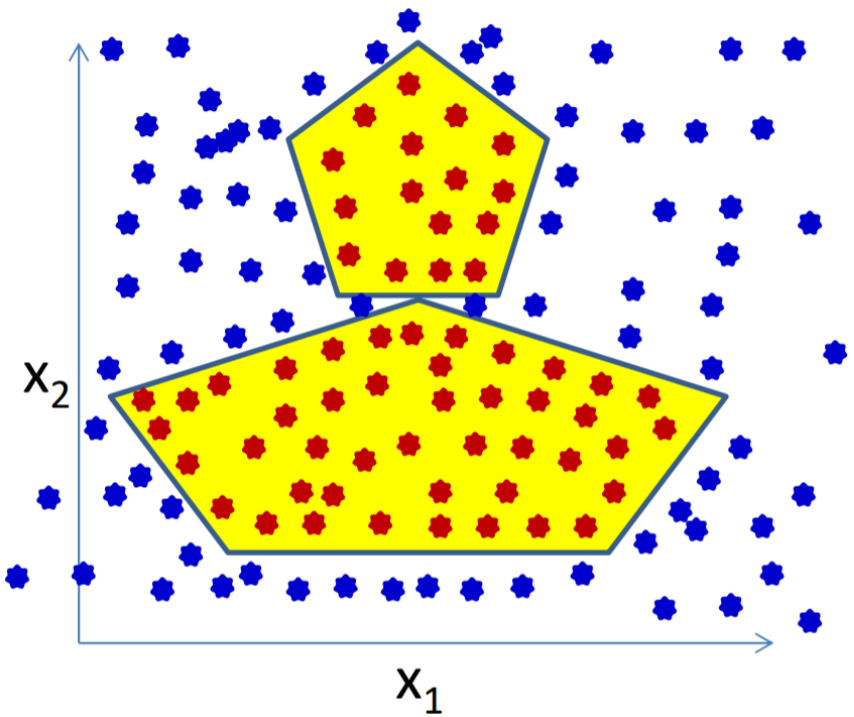


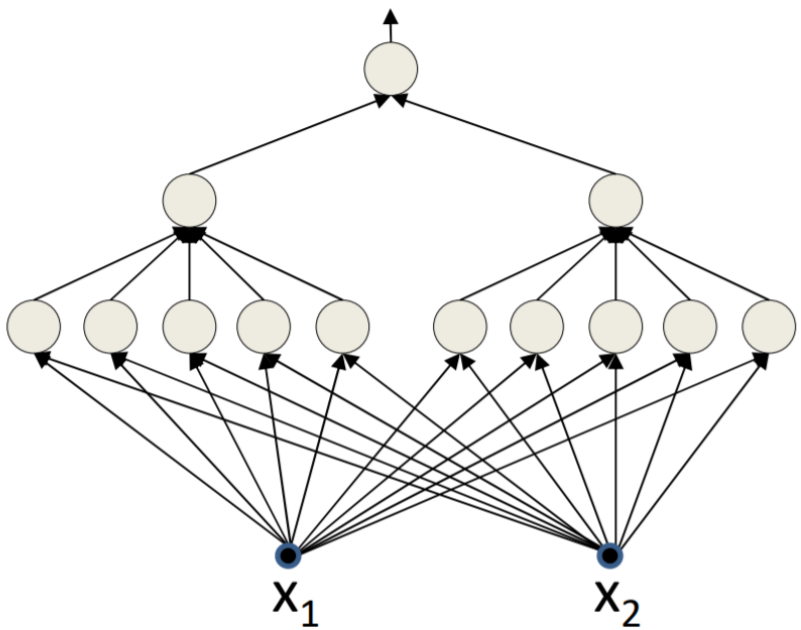




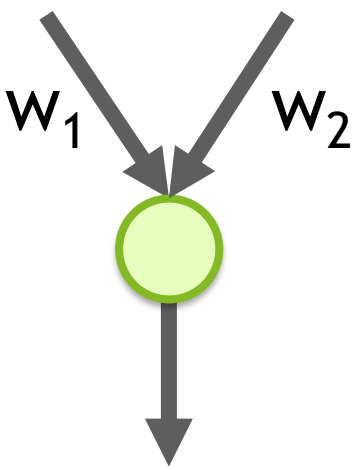
Detour: Deep Learning

2. More complex decision boundary (not linearly separable)



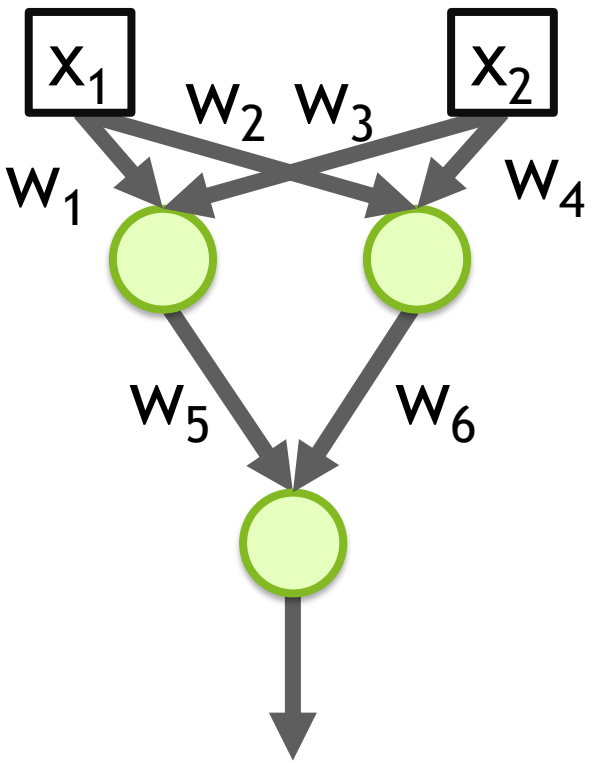


네트워크 구축



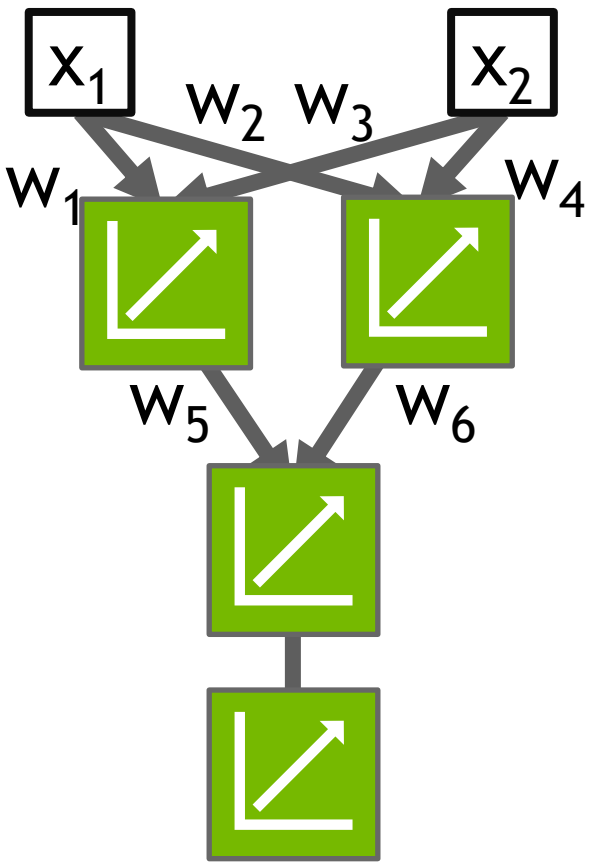
- 더 많은 변수로의 확장

네트워크 구축



- 더 많은 변수로의 확장
- 뉴런들간의 연결

네트워크 구축



- 더 많은 변수로의 확장
- 뉴런들간의 연결
- 모든 회귀가 선형이면 출력 결과도 선형 회귀

목차 – 2부

- 데이터 준비
- 뉴런의 학습 과정
- 뉴런에서 네트워크로
- **활성화 함수**
- 과적합
- 회귀에서 분류로

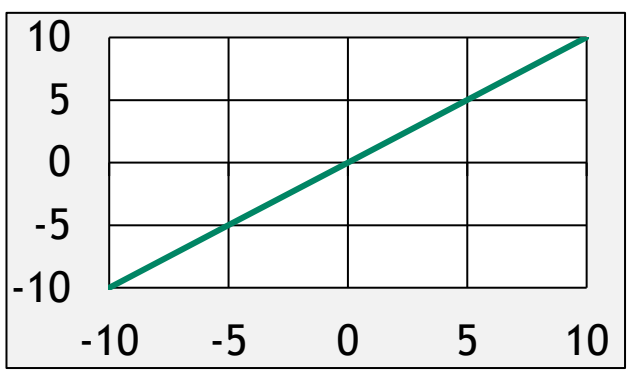
활성화 함수
ACTIVATION FUNCTIONS

활성화 함수 (ACTIVATION FUNCTIONS)

Linear(선형)

$$\hat{y} = wx + b$$

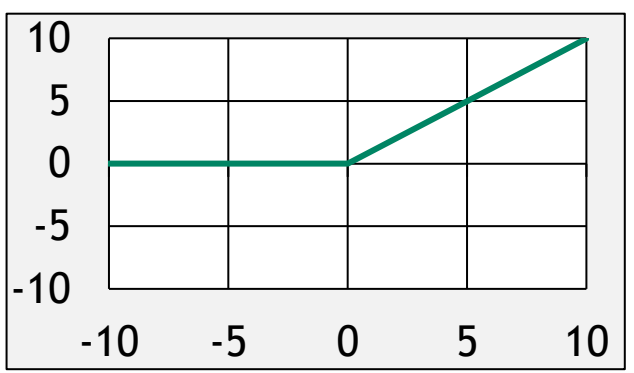
```
# Multiply each input
# with a weight (w) and
# add intercept (b)
y_hat = wx+b
```



ReLU

$$\hat{y} = \begin{cases} wx + b \; if \; wx + b > 0 \\ 0 \; otherwise \end{cases}$$

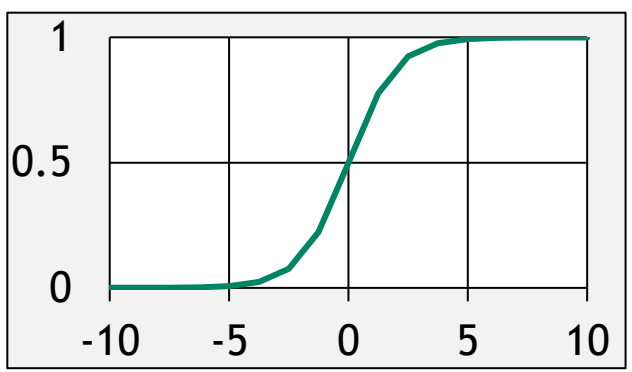
```
# Only return result
# if total is positive
linear = wx+b
y_hat = linear * (linear > 0)
```



Sigmoid(시그모이드)

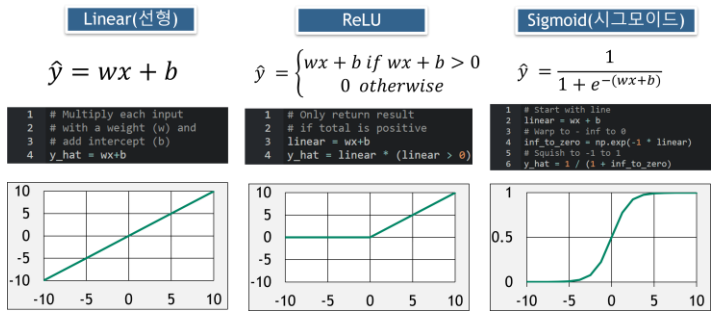
$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

```
# Start with line
linear = wx + b
# Warp to - inf to 0
inf_to_zero = np.exp(-1 * linear)
# Squish to -1 to 1
y_hat = 1 / (1 + inf_to_zero)
```



활성화 함수 (ACTIVATION FUNCTIONS)

- 활성화 함수란? 활성화 함수를 사용하는 이유는?



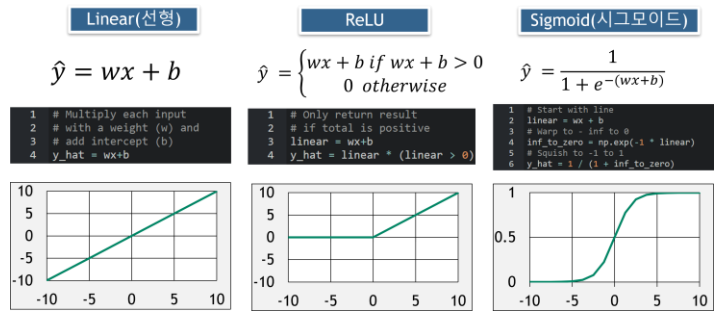
활성화 함수 (ACTIVATION FUNCTIONS)

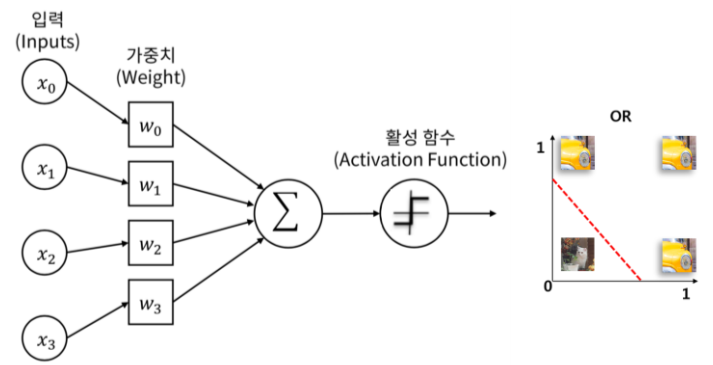
- 활성화 함수란? 활성화 함수를 사용하는 이유는?

- 앞에서 나왔던 XOR 문제등을 풀기위해

$$h = WX + b$$

라고 표현되는 hidden layer에 nonlinearity정보를 넣어주기 위해 사용함



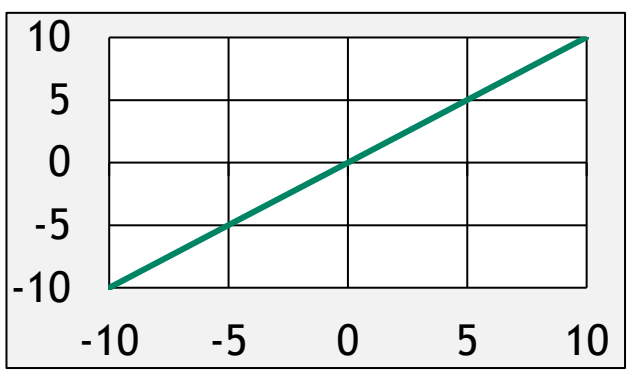


활성화 함수 (ACTIVATION FUNCTIONS)

Linear(선형)

$$\hat{y} = wx + b$$

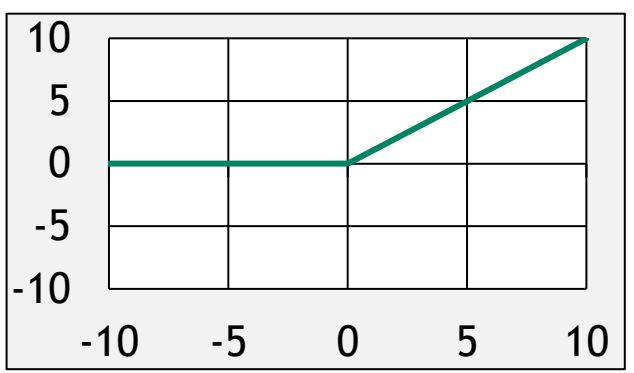
```
# Multiply each input
# with a weight (w) and
# add intercept (b)
y_hat = wx+b
```



ReLU

$$\hat{y} = \begin{cases} wx + b \; if \; wx + b > 0 \\ 0 \; otherwise \end{cases}$$

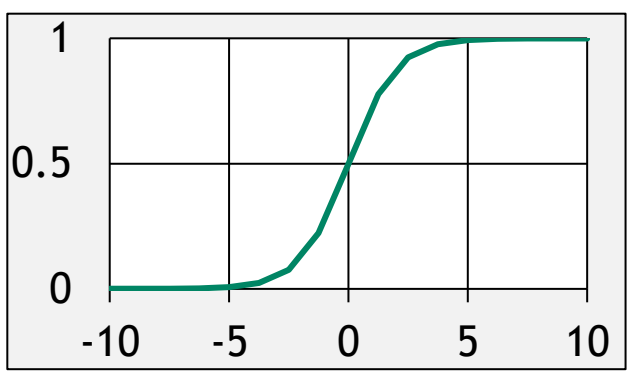
```
# Only return result
# if total is positive
linear = wx+b
y_hat = linear * (linear > 0)
```



Sigmoid(시그모이드)

$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

```
# Start with line
linear = wx + b
# Warp to - inf to 0
inf_to_zero = np.exp(-1 * linear)
# Squish to -1 to 1
y_hat = 1 / (1 + inf_to_zero)
```



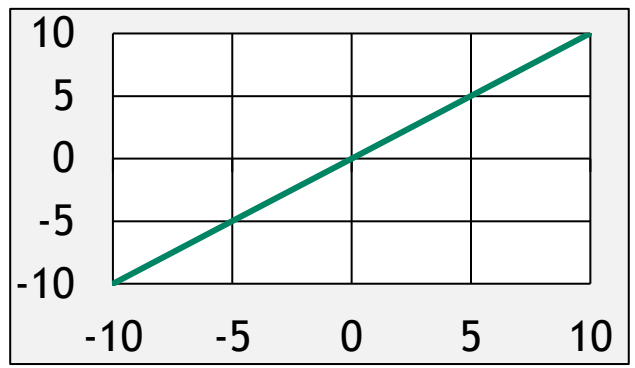
활성화 함수 (ACTIVATION FUNCTIONS)

- 활성화 함수로 선형함수를 쓰면 안되는 이유?

Linear(선형)

$$\hat{y} = wx + b$$

```
# Multiply each input
# with a weight (w) and
# add intercept (b)
y_hat = wx+b
```



활성화 함수 (ACTIVATION FUNCTIONS)

- 활성화 함수로 선형함수를 쓰면 안되는 이유?

1) 여러 층으로 된 모델을 하나의 층만으로도 나타낼 수 있기 때문

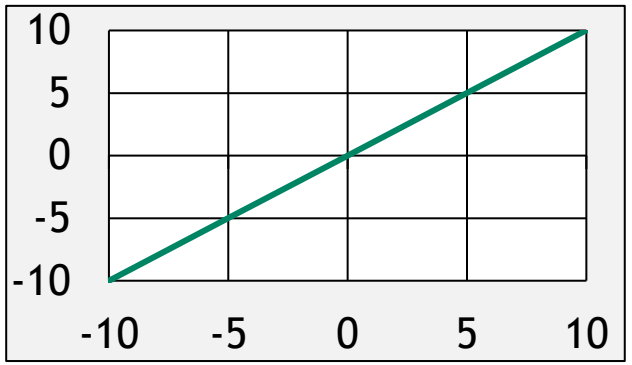
e.g.) $y = w\big(w(wx + b)\big) + b = w'x + b'$

2) 따라서 층을 쌓는 혜택을 얻고 싶다면 activation function은 반드시 nonlinear 함수를 사용해야함

Linear(선형)

$$\hat{y} = wx + b$$

```
# Multiply each input
# with a weight (w) and
# add intercept (b)
y_hat = wx+b
```

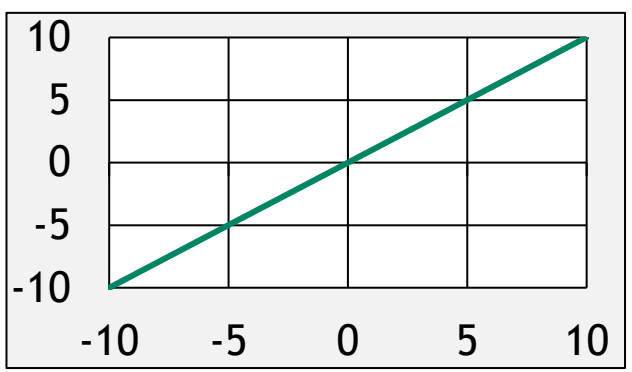


활성화 함수 (ACTIVATION FUNCTIONS)

Linear(선형)

$$\hat{y} = wx + b$$

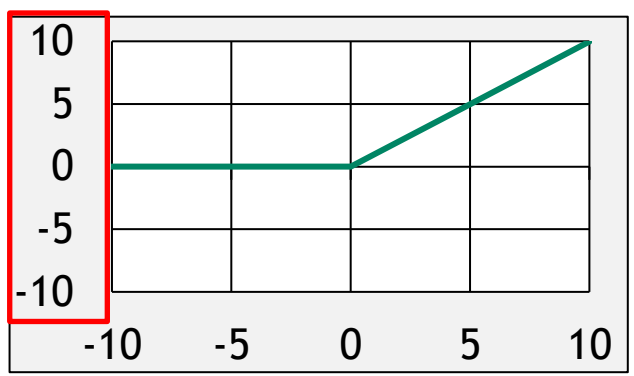
```
# Multiply each input
# with a weight (w) and
# add intercept (b)
y_hat = wx+b
```



ReLU

$$\hat{y} = \begin{cases} wx + b \; if \; wx + b > 0 \\ 0 \; otherwise \end{cases}$$

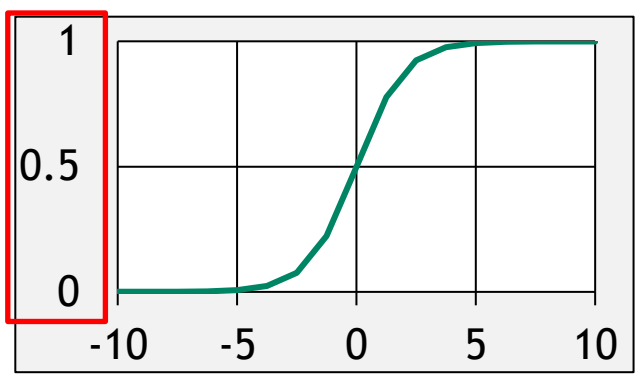
```
# Only return result
# if total is positive
linear = wx+b
y_hat = linear * (linear > 0)
```



Sigmoid(시그모이드)

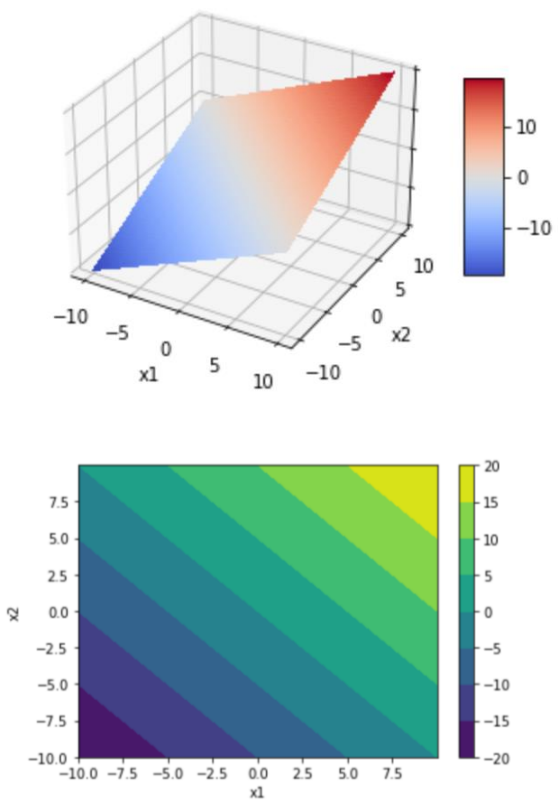
$$\hat{y} = \frac{1}{1 + e^{-(wx+b)}}$$

```
# Start with line
linear = wx + b
# Warp to - inf to 0
inf_to_zero = np.exp(-1 * linear)
# Squish to -1 to 1
y_hat = 1 / (1 + inf_to_zero)
```

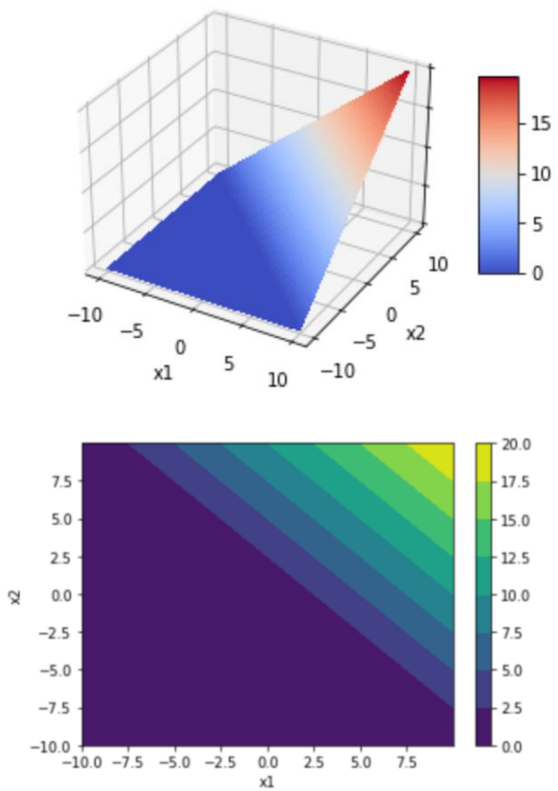


활성화 함수 (ACTIVATION FUNCTIONS)

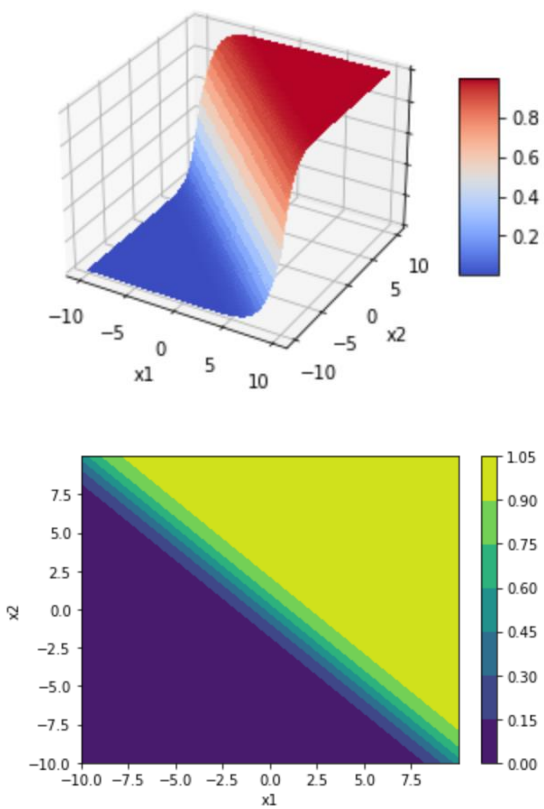
Linear(선형)



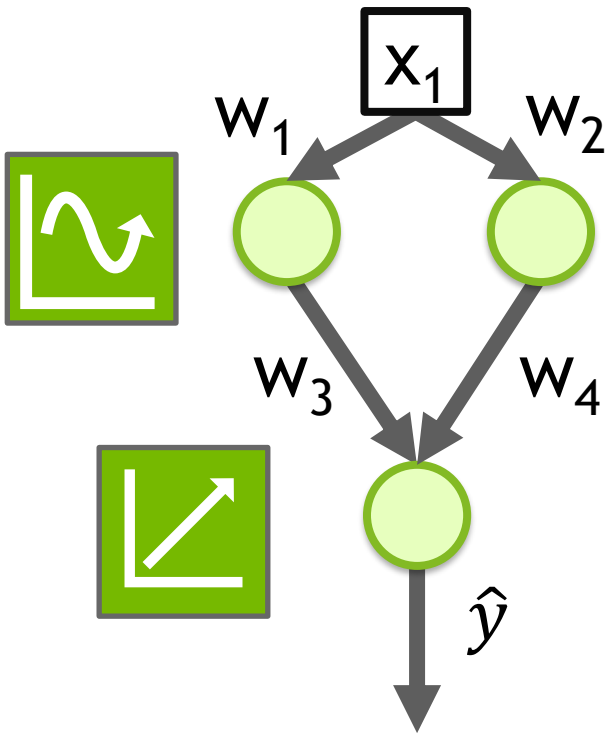
ReLU

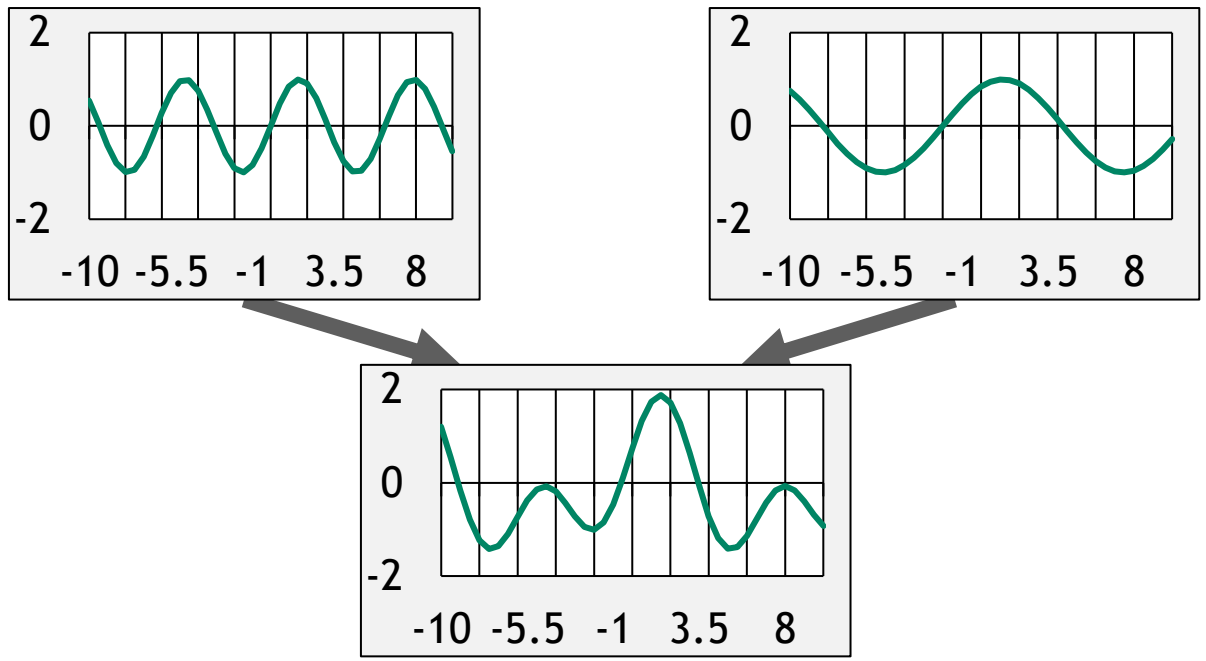


Sigmoid(시그모이드)



활성화 함수 (ACTIVATION FUNCTIONS)





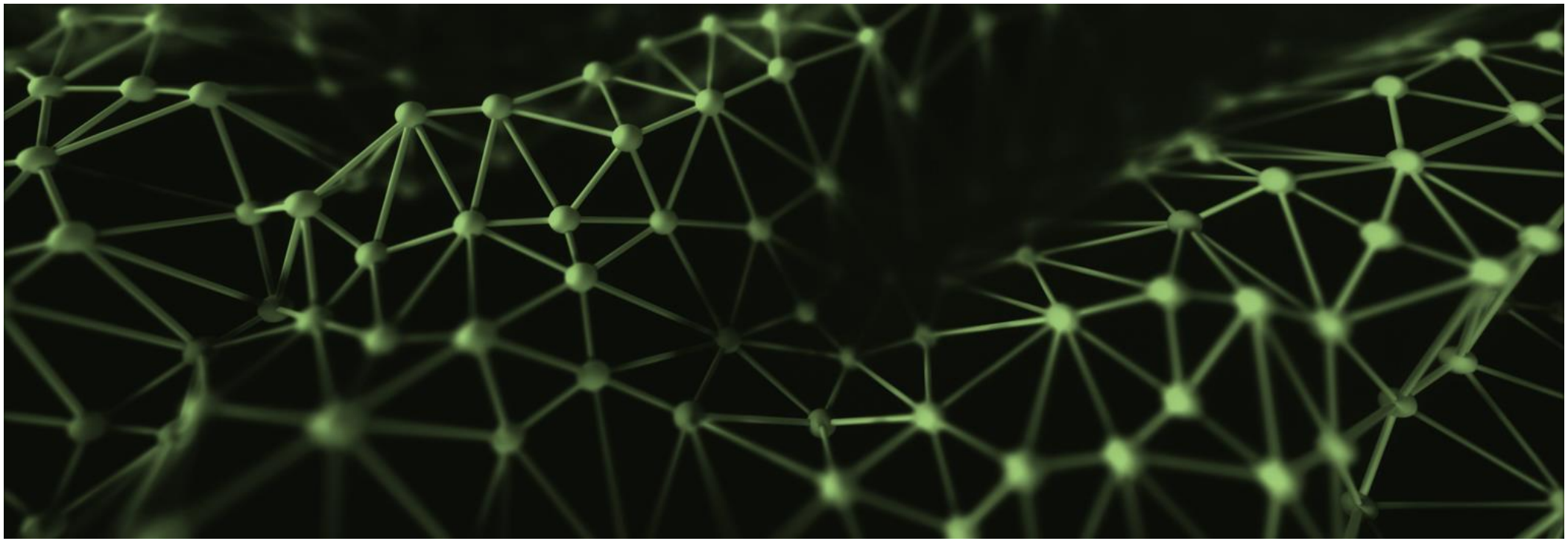
목차 – 2부

- 데이터 준비
- 뉴런의 학습 과정
- 뉴런에서 네트워크로
- 활성화 함수
- **과적합**
- 회귀에서 분류로

과적합
OVERFITTING

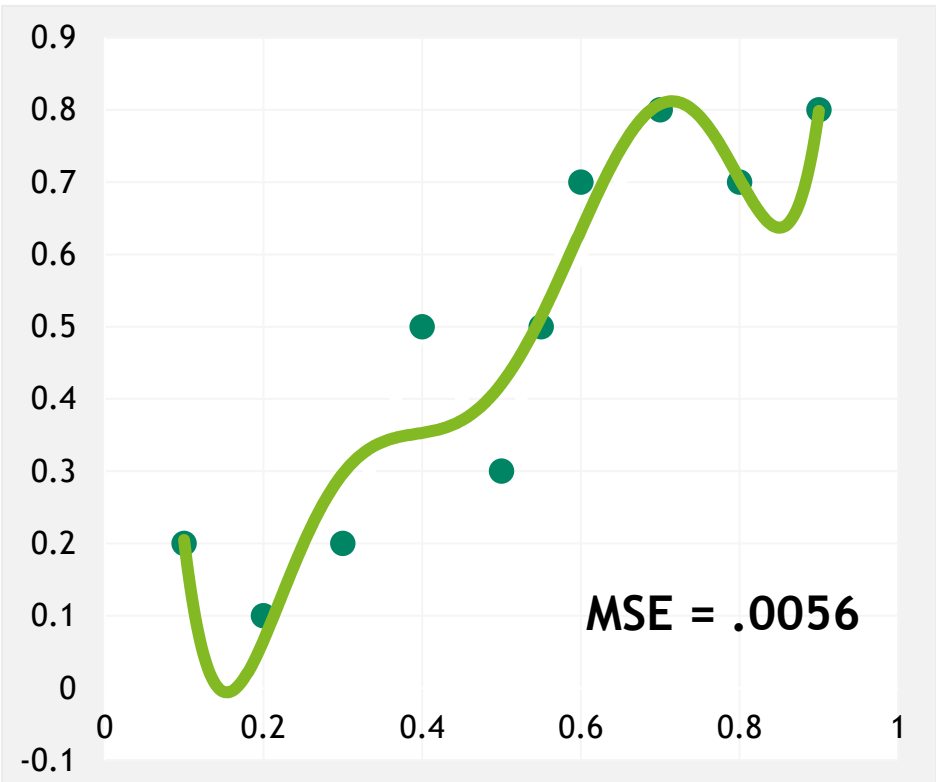
과적합 (OVERFITTING)

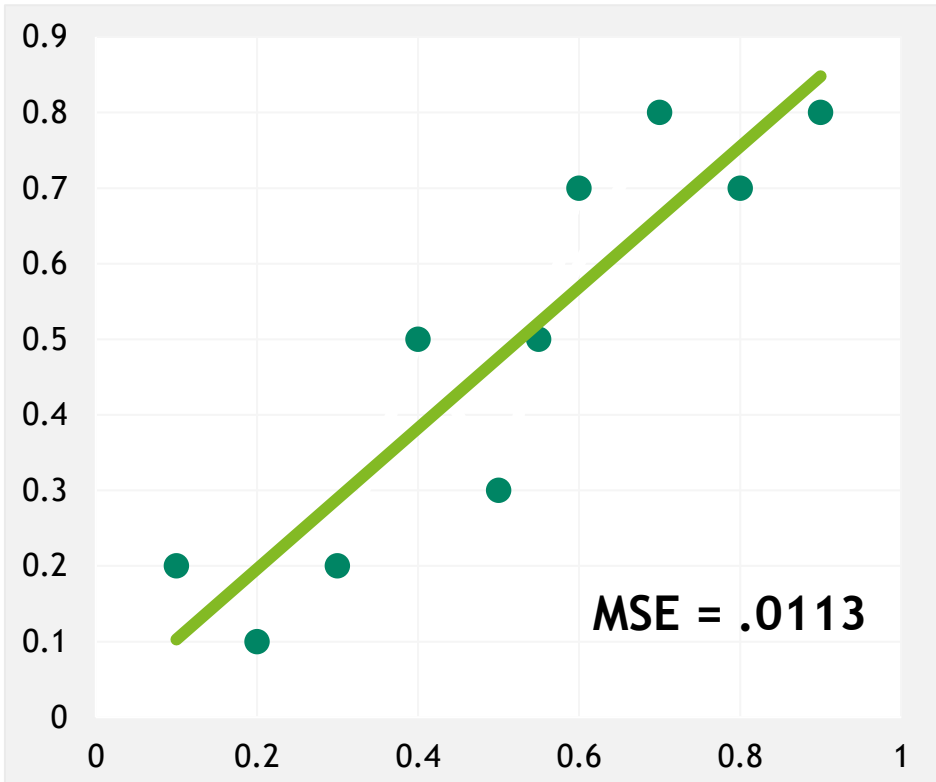
거대한 뉴럴 네트워크를 구축하면 되지 않을까?



과적합 (OVERFITTING)

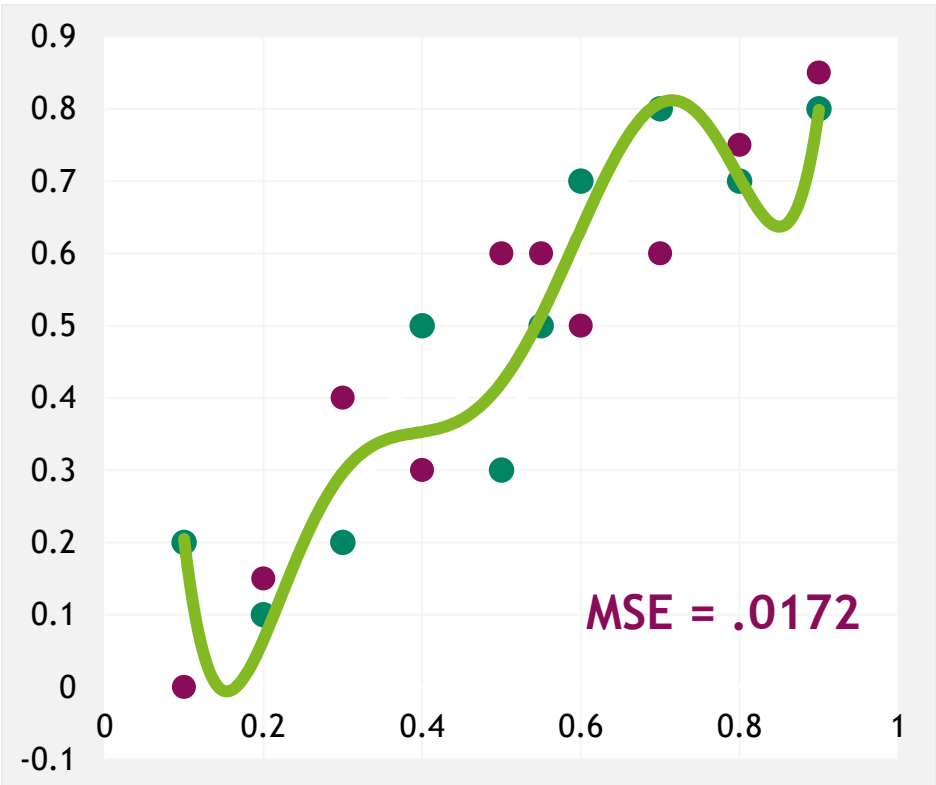
어떤 추세선이 더 나은가?

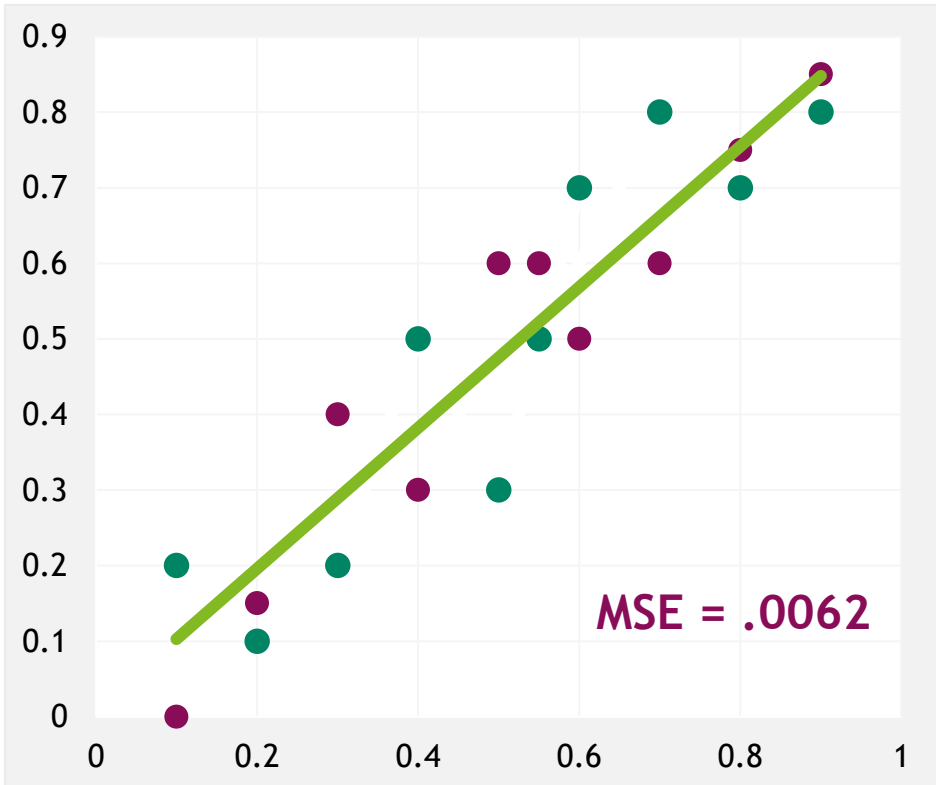




과적합 (OVERFITTING)

어떤 추세선이 더 나은가?





트레이닝 데이터와 검증 데이터 비교

암기 회피(Avoid memorization)

트레이닝 데이터 (Training data)

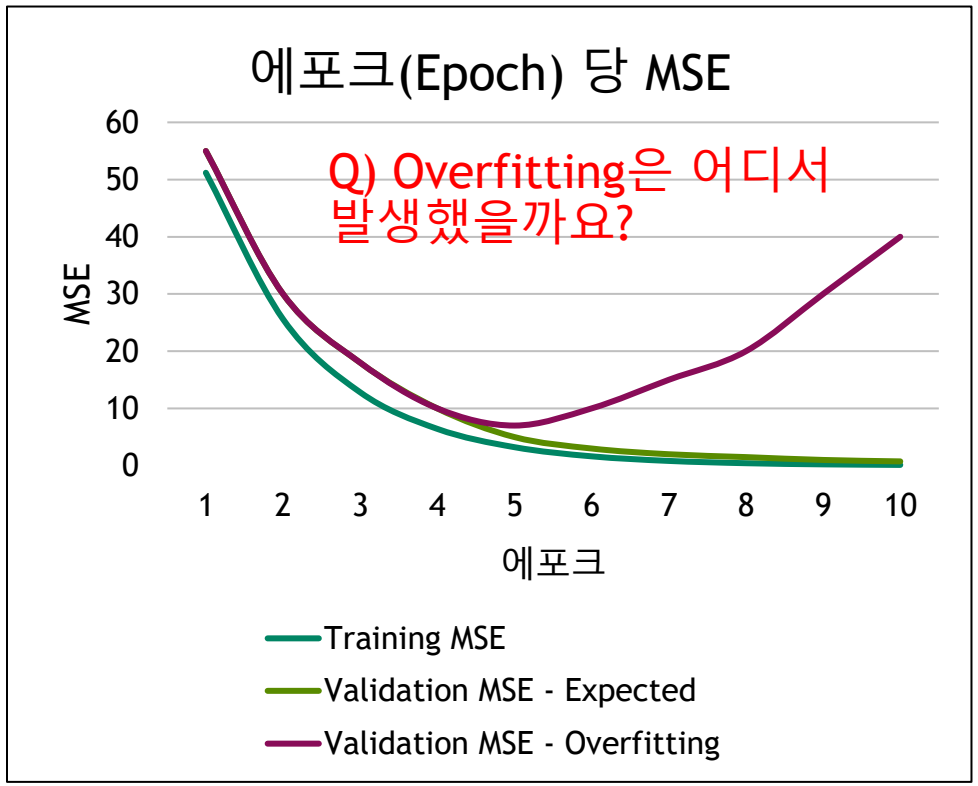
- 학습해야 하는 모델의 핵심 데이터세트

검증 데이터 (Validation data)

- 정말로 이해하는지(일반화 가능한지)를 확인하기 위한 모델의 새 데이터

과적합 (Overfitting) ↔ Underfitting

- 모델이 트레이닝 데이터에 대한 좋은 성능을 보이지만 검증 데이터는 그렇지 못함(암기의 증거)
- 정확도 및 손실이 두 데이터세트 간에 유사해야 이상적임



트레이닝 데이터와 검증 데이터 비교

암기 회피(Avoid memorization)

트레이닝 데이터 (Training data)

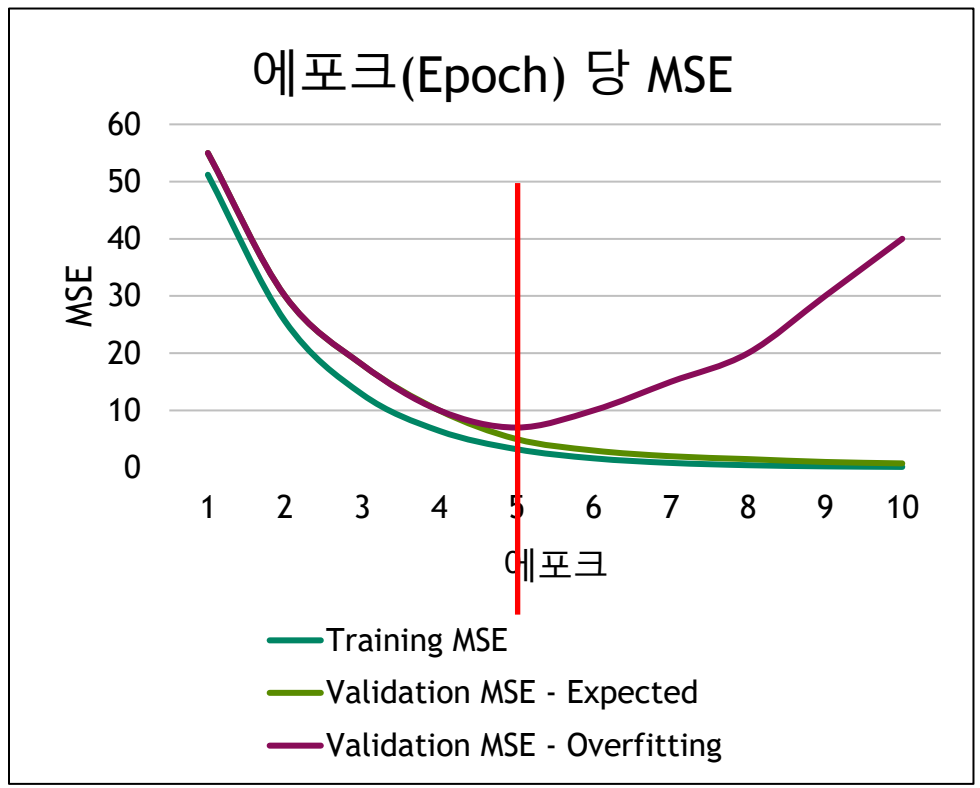
- 학습해야 하는 모델의 핵심 데이터세트

검증 데이터 (Validation data)

- 정말로 이해하는지(일반화 가능한지)를 확인하기 위한 모델의 새 데이터

과적합 (Overfitting) ↔ Underfitting

- 모델이 트레이닝 데이터에 대한 좋은 성능을 보이지만 검증 데이터는 그렇지 못함(암기의 증거)
- 정확도 및 손실이 두 데이터세트 간에 유사해야 이상적임

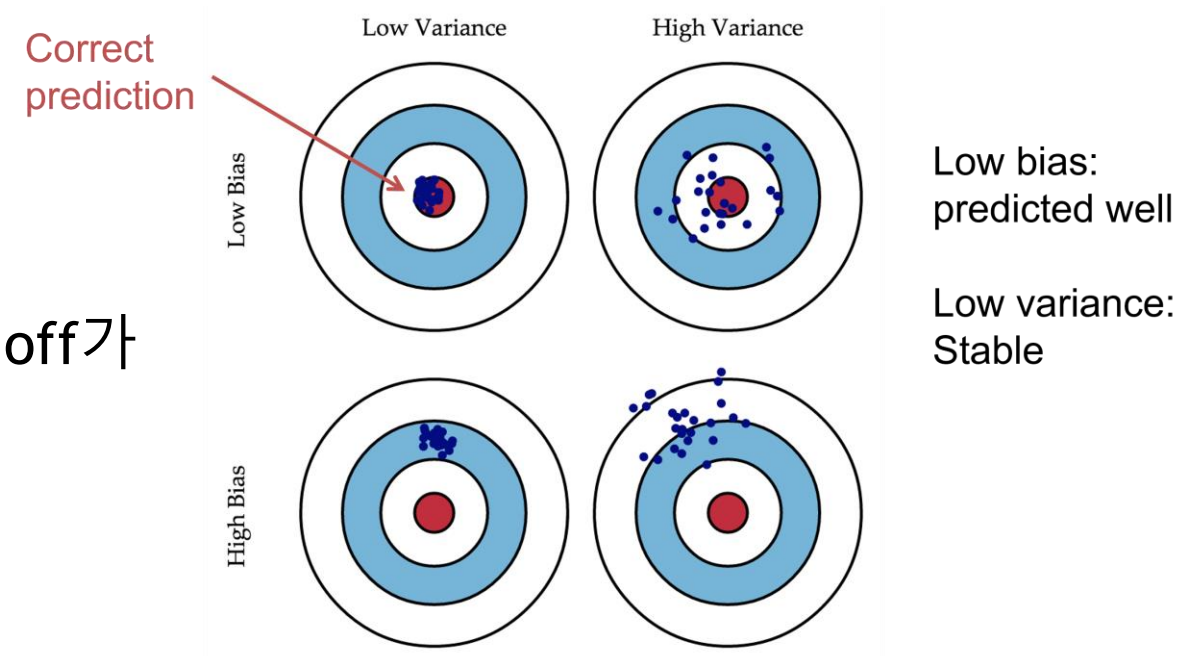


더 읽을거리

1) Bias-Variance Decomposition

2) Bias-Variance Trade-off

- Overfitting과 Bias-Variance trade-off가 어떤 관계가 있을까?



목차 – 2부

- 데이터 준비
- 뉴런의 학습 과정
- 뉴런에서 네트워크로
- 활성화 함수
- 과적합
- **회귀에서 분류로**

회귀에서 분류로
FROM REGRESSION TO CLASSIFICATION

MSE VS. CROSS-ENTROPY

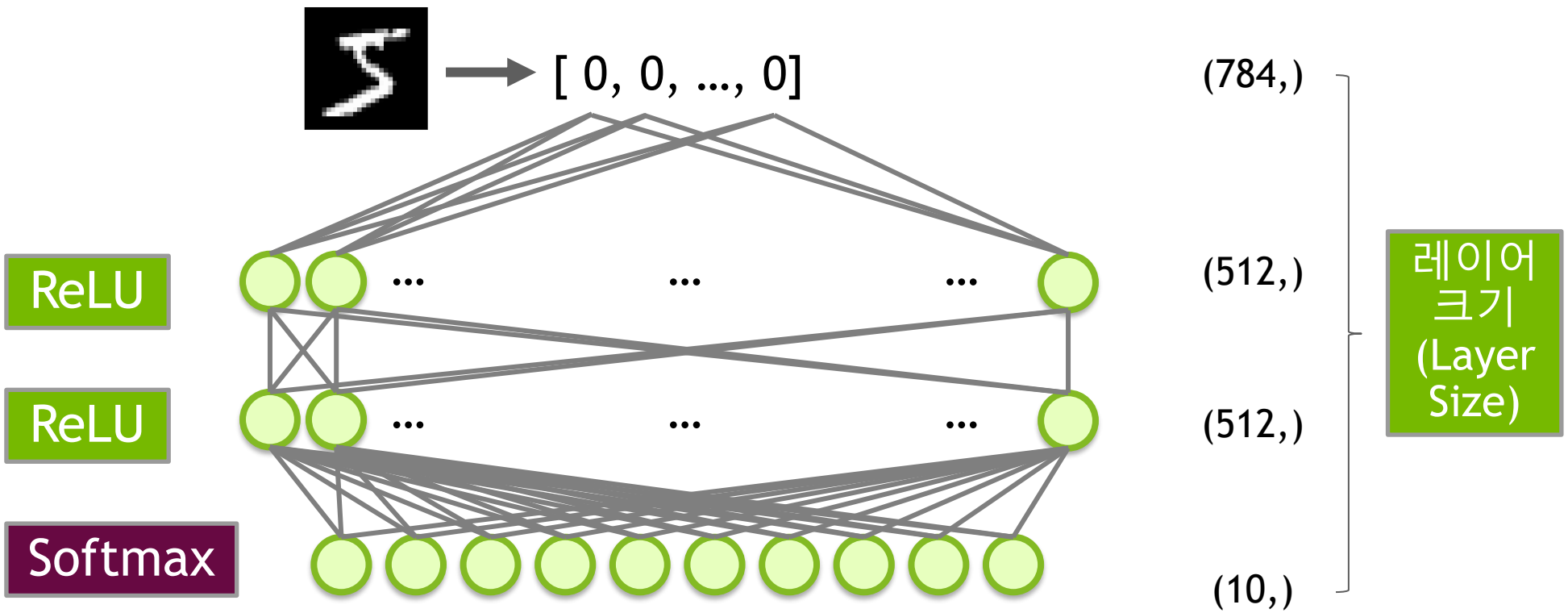
- Regression → MSE

- Classification → Cross-entropy

왜?????????? (수식적으로 설명할 수 있어야됨)

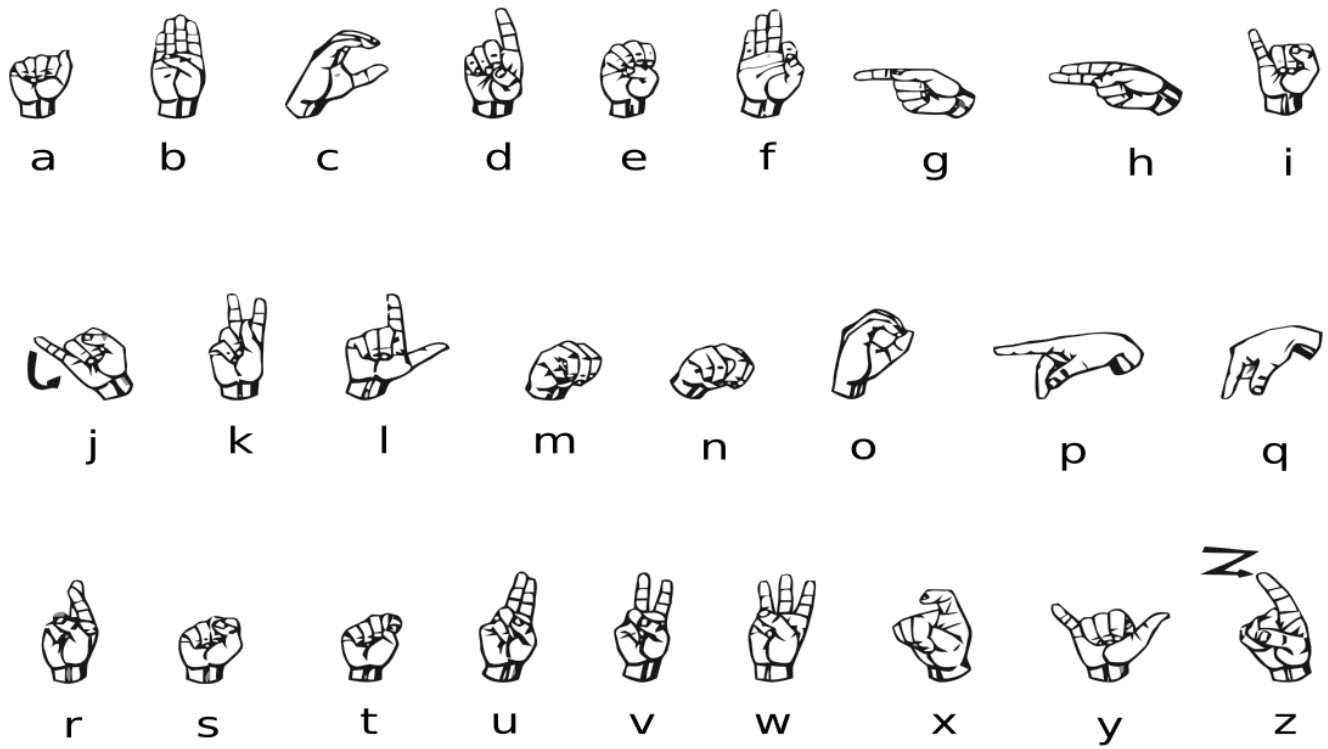
연습문제

MNIST 모델



연습 문제

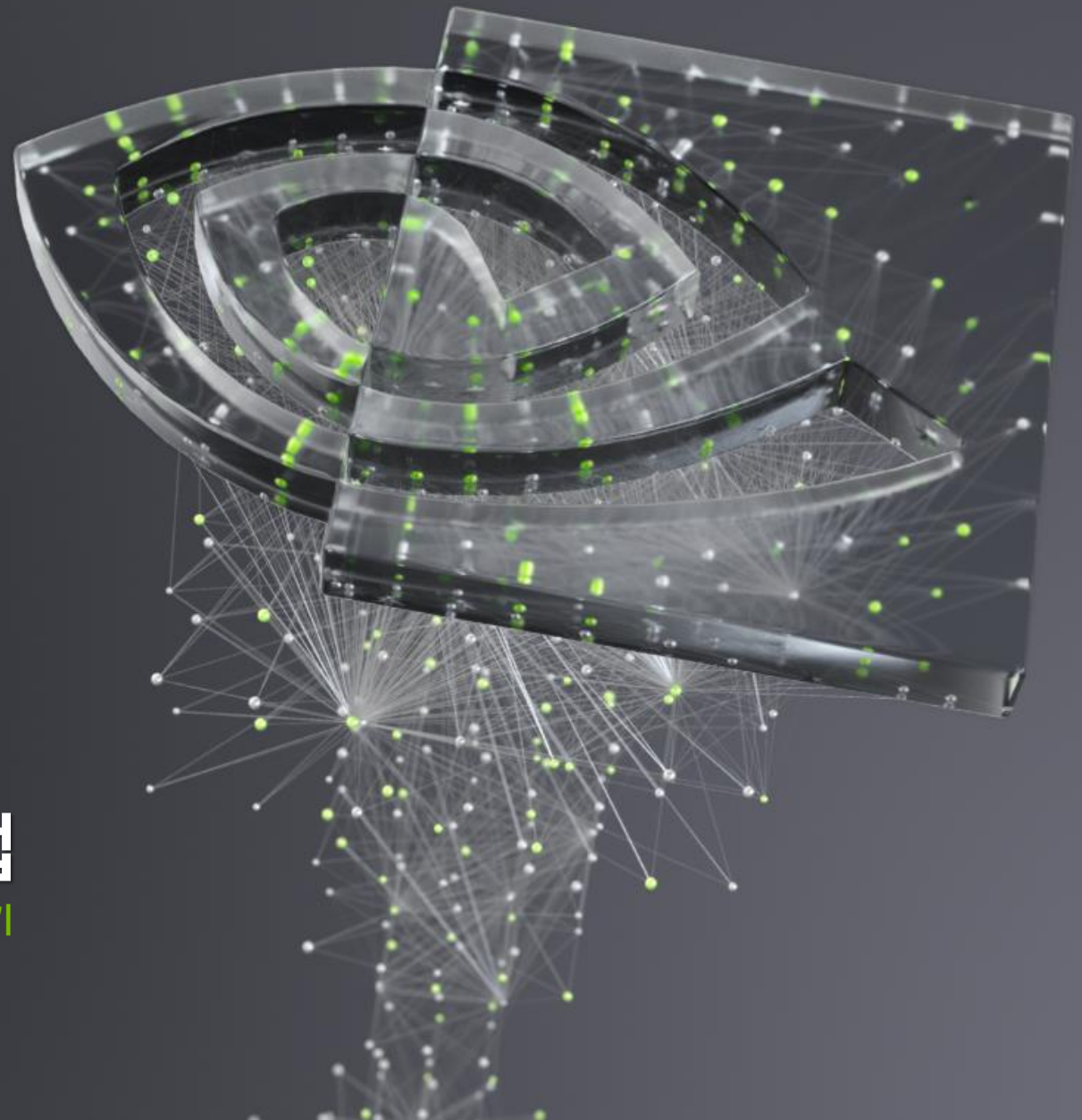
미국 수화 알파벳



시작하겠습니다!

2) 02_asl.ipynb


감사합니다
https://jeiyoon.github.io/
NVIDIA
DEEP
LEARNING
INSTITUTE



NVIDIA | DEEP LEARNING INSTITUTE

부록: 그래디언트 하강법

컴퓨터가 미적분에 대한 편법을 사용할 수 있도록 돕기

에러를 통한 학습

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 = \frac{1}{n}\sum_{i=1}^{n}(y - (mx + b))^2$$

$$MSE = \frac{1}{2}((3 - (m(1) + b))^2 + (5 - (m(2) + b))^2)$$

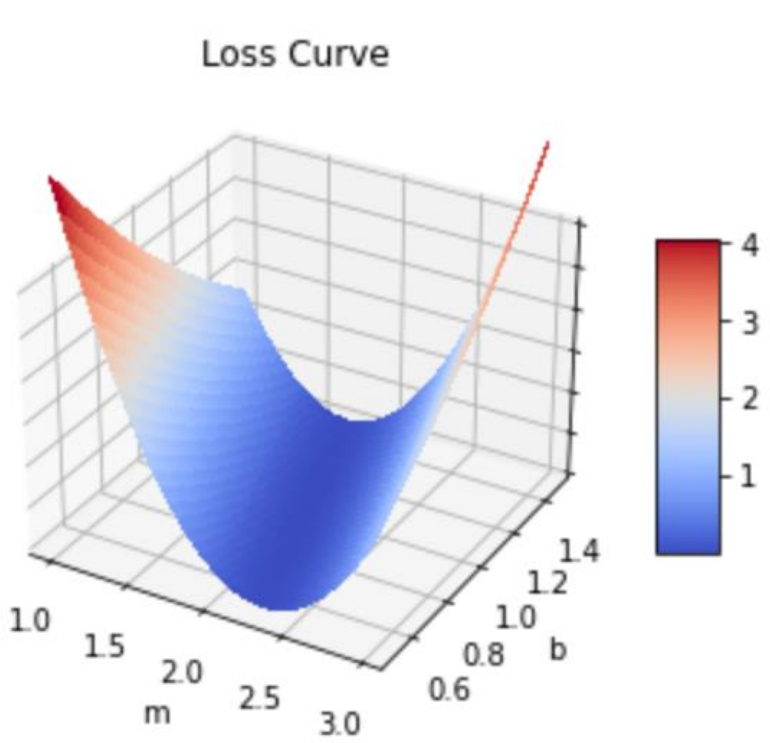
$$\frac{\partial MSE}{\partial m} = 9m + 5b - 23 \qquad \frac{\partial MSE}{\partial b} = 5m + 3b - 13$$

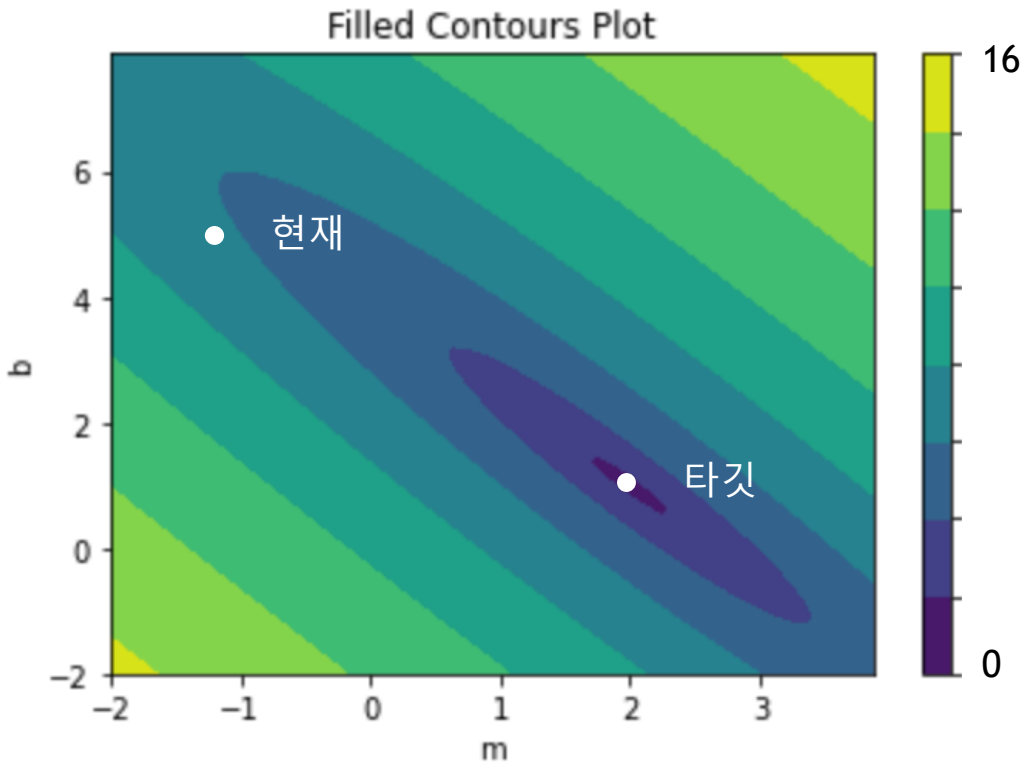
$$\frac{\partial MSE}{\partial m} = -7 \qquad \frac{\partial MSE}{\partial b} = -3$$

$m = -1$

$b = 5$

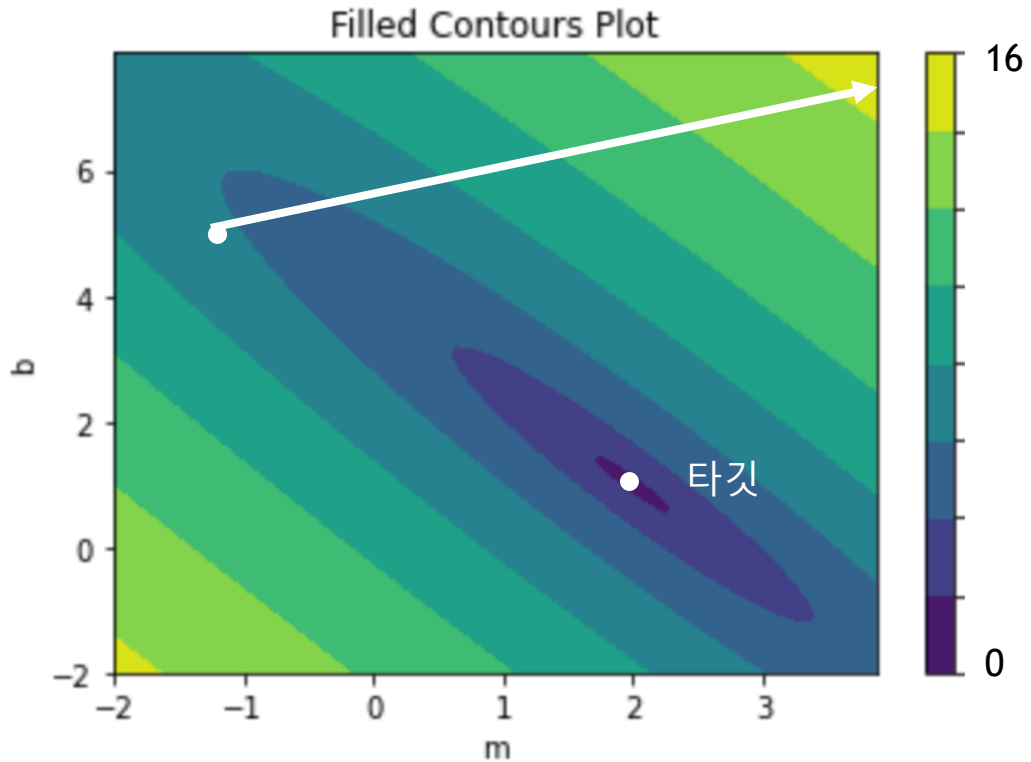
손실 곡선 (THE LOSS CURVE)





손실 곡선 (THE LOSS CURVE)

$$\frac{\partial MSE}{\partial m} = -7 \qquad \frac{\partial MSE}{\partial b} = -3$$

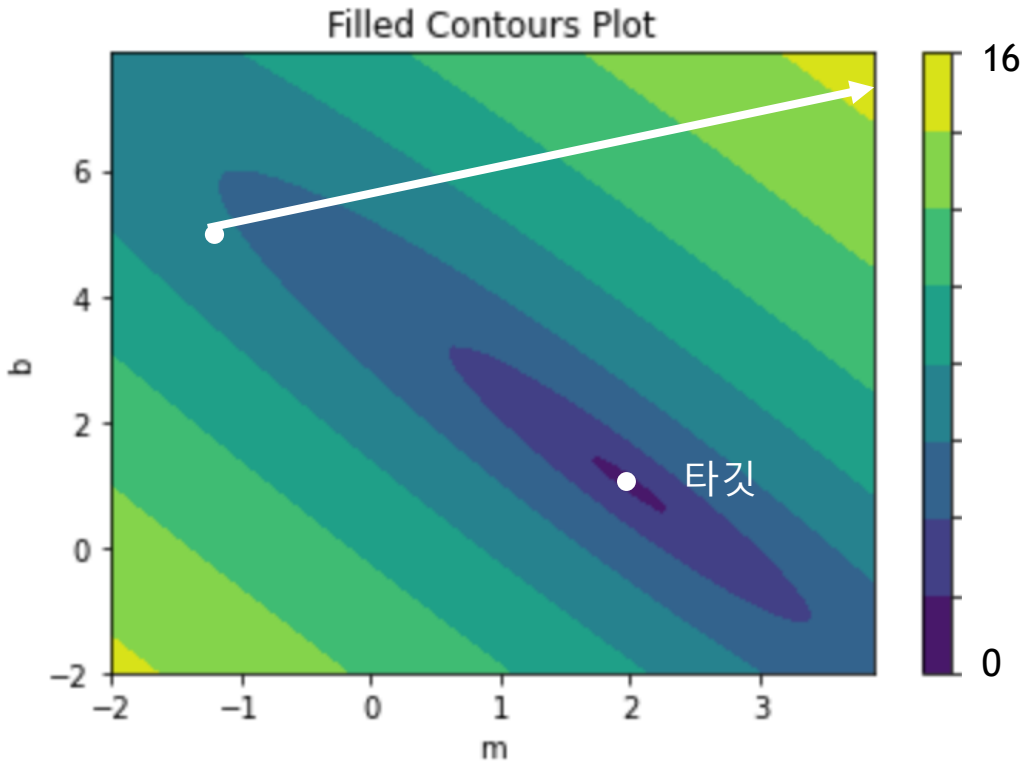


손실 곡선 (THE LOSS CURVE)

$$\frac{\partial MSE}{\partial m} = -7 \qquad \frac{\partial MSE}{\partial b} = -3$$

$$\mathrm{m} := \mathrm{m} - \lambda \frac{\partial MSE}{\partial m}$$

$$\mathrm{b} := b - \lambda \frac{\partial MSE}{\partial b}$$



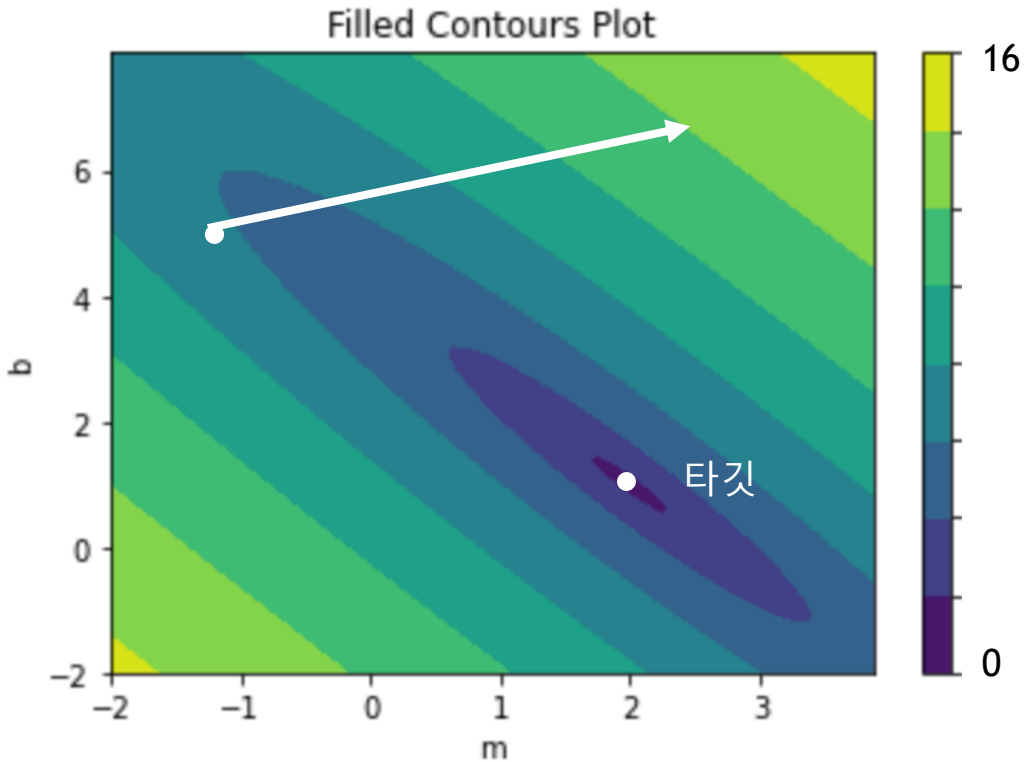
손실 곡선 (THE LOSS CURVE)

$$\frac{\partial MSE}{\partial m} = -7 \qquad \frac{\partial MSE}{\partial b} = -3$$

$$\mathrm{m} := \mathrm{m} - \lambda \frac{\partial MSE}{\partial m}$$

$$\lambda = .5$$

$$\mathrm{b} := b - \lambda \frac{\partial MSE}{\partial b}$$



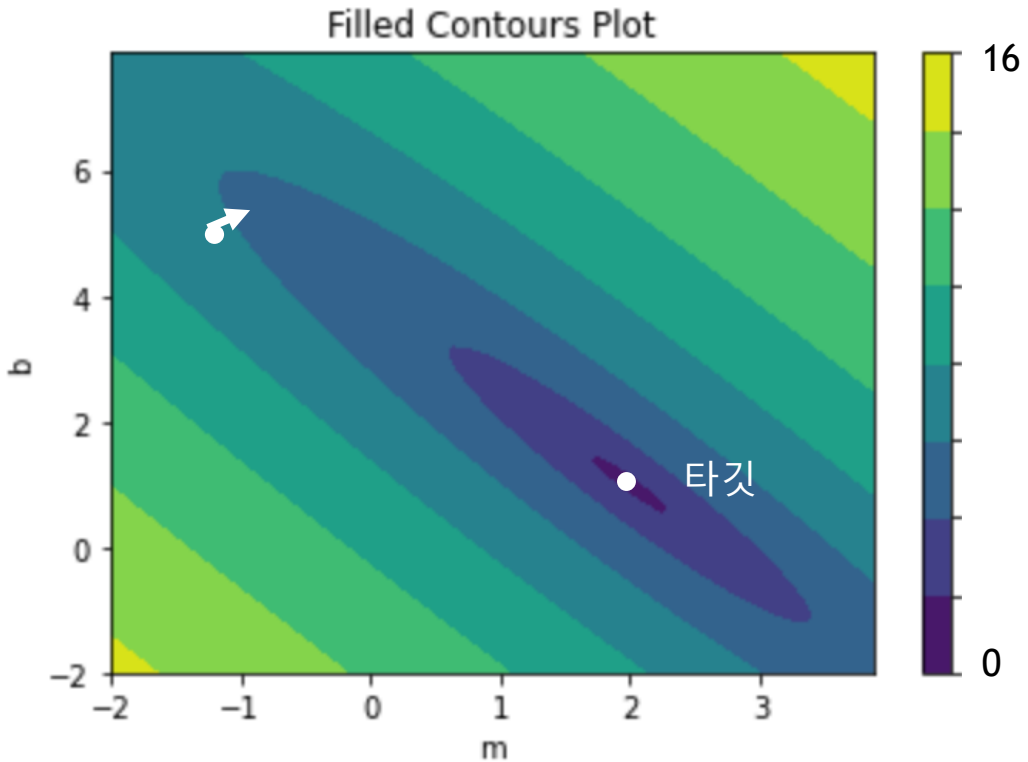
손실 곡선 (THE LOSS CURVE)

$$\frac{\partial MSE}{\partial m} = -7 \qquad \frac{\partial MSE}{\partial b} = -3$$

$$\mathrm{m} := \mathrm{m} - \lambda \frac{\partial MSE}{\partial m}$$

$$\lambda = .005$$

$$\mathrm{b} := b - \lambda \frac{\partial MSE}{\partial b}$$

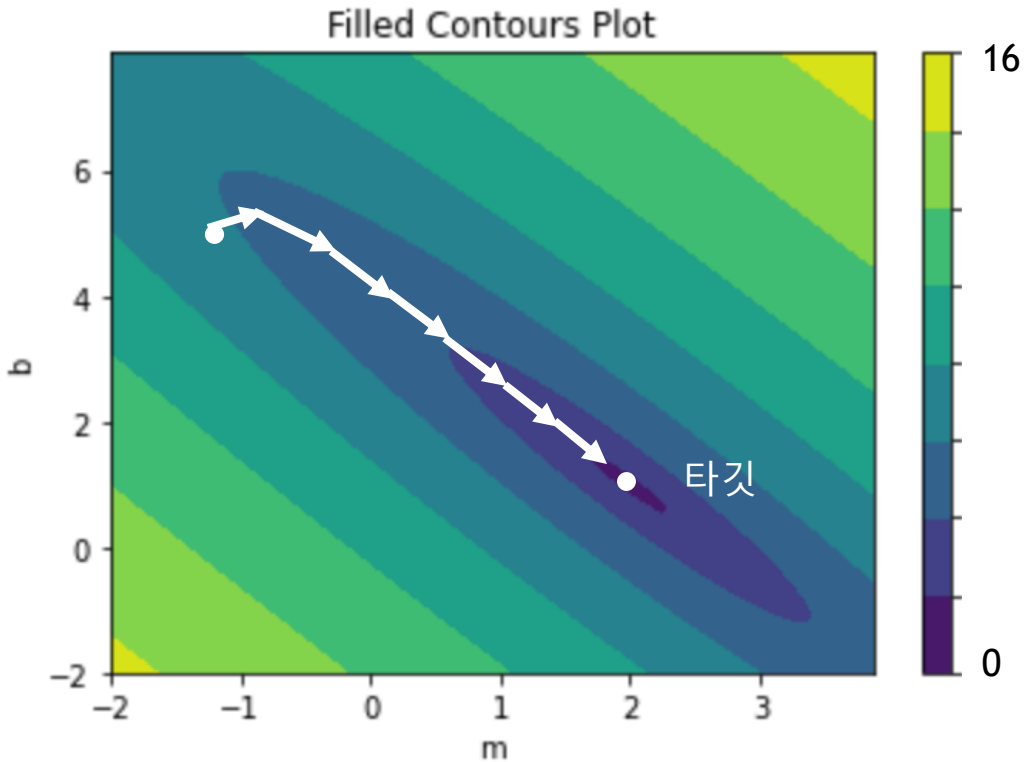


손실 곡선 (THE LOSS CURVE)

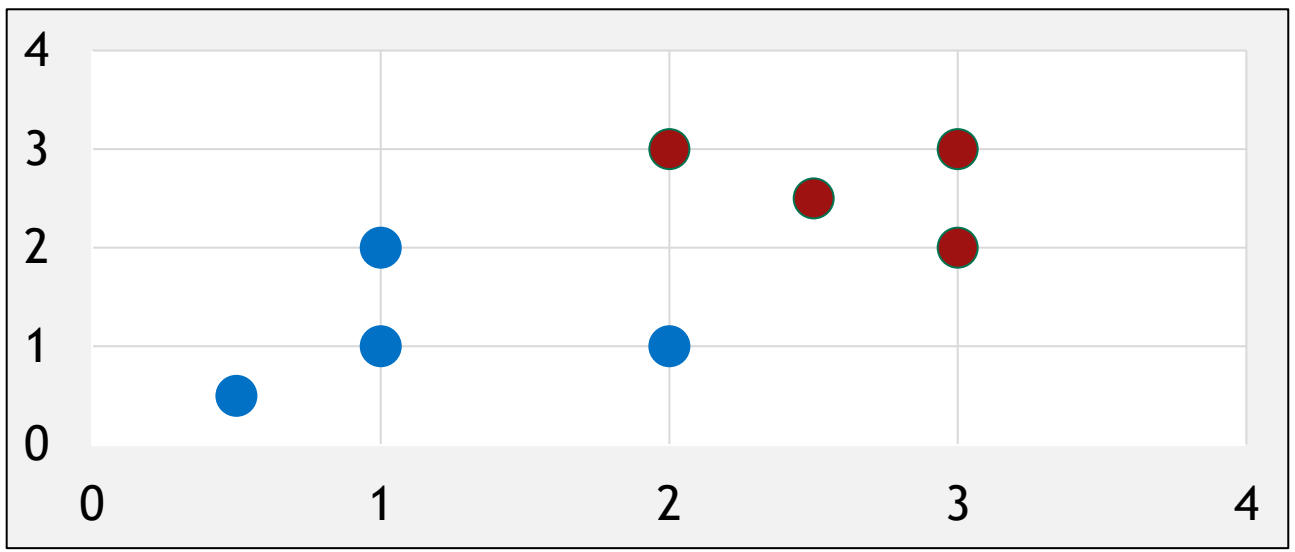
$\lambda = .1$

$m := -1 - 7\lambda = -1.7$

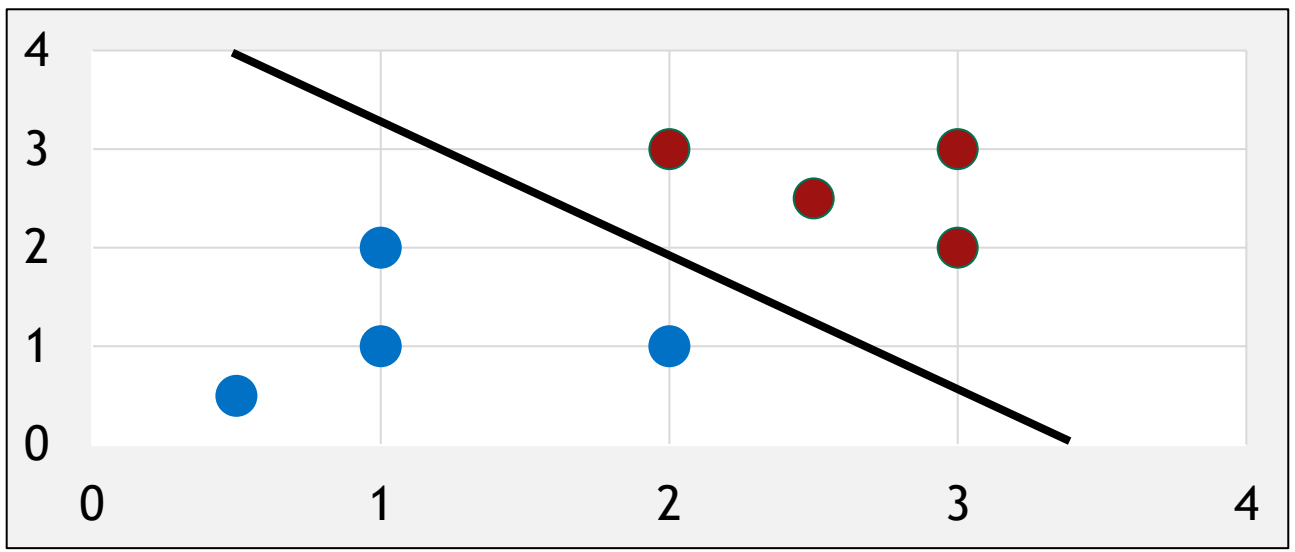
$b := 5 - 3\lambda = 5.3$



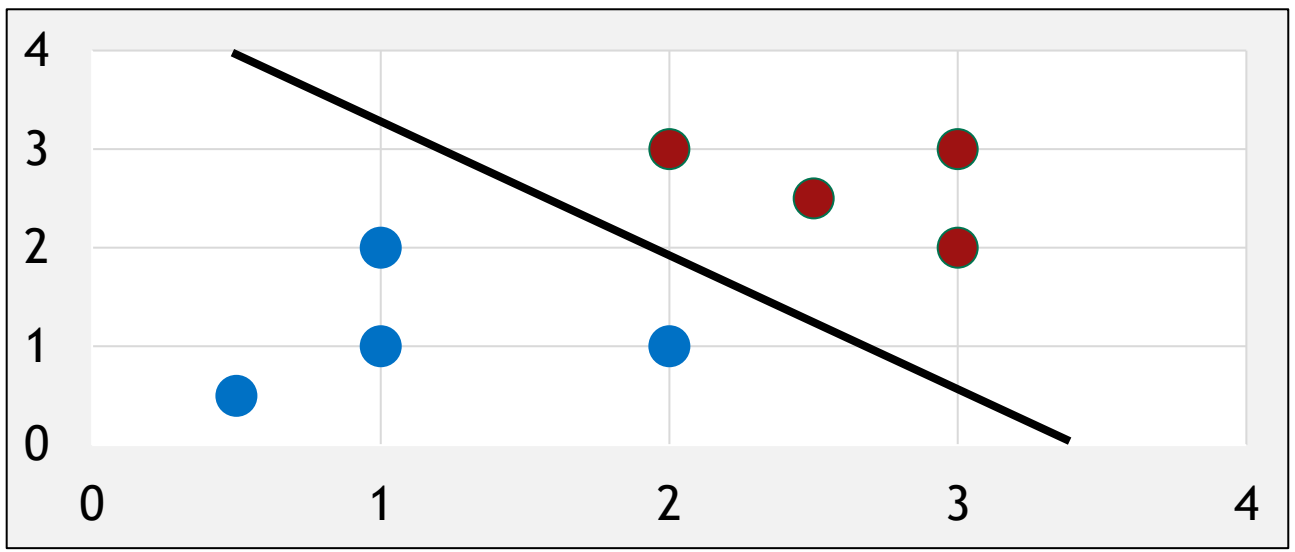
RMSE로 확률 예측?

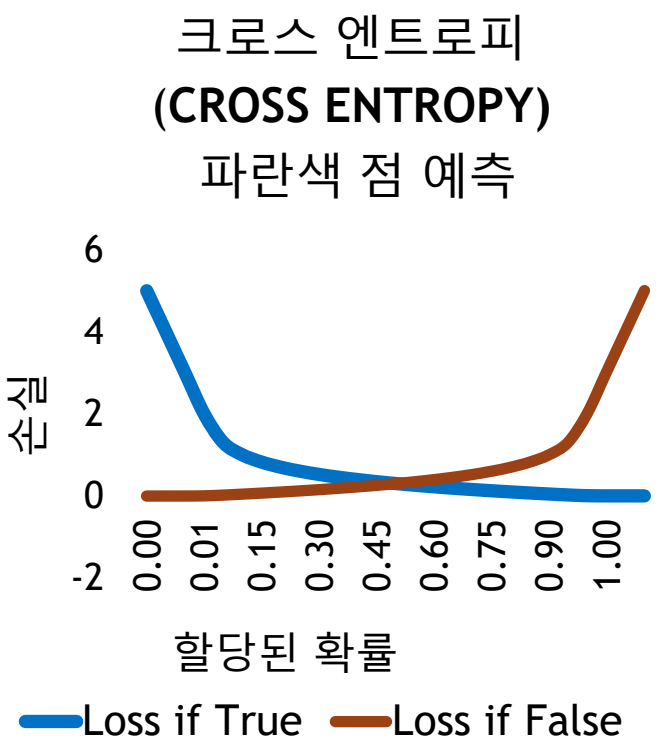


RMSE로 확률 예측?

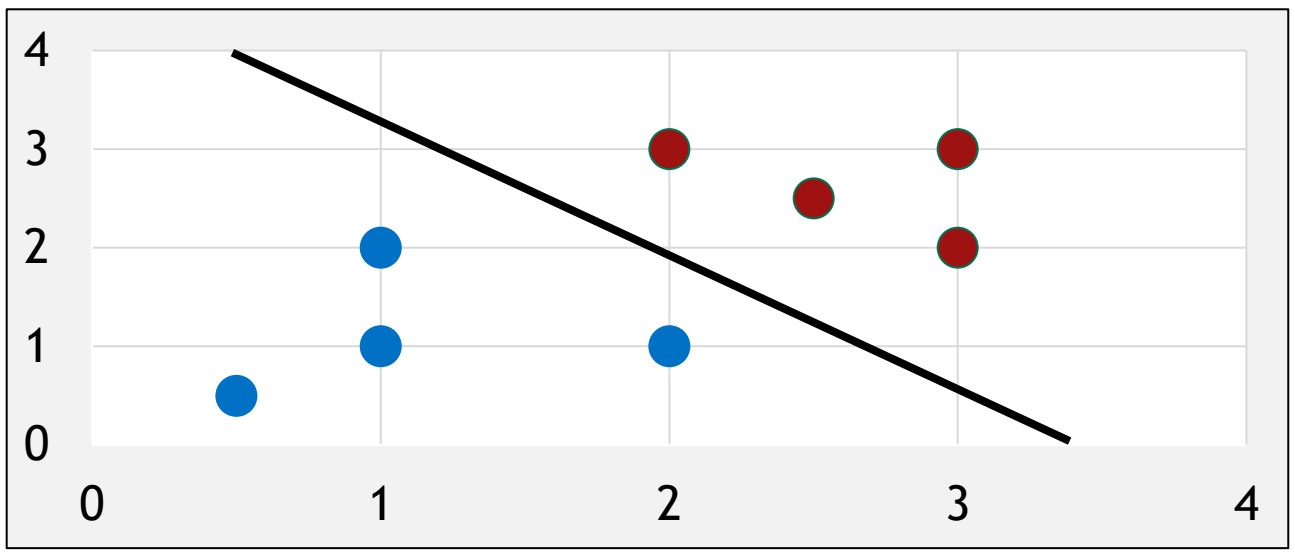


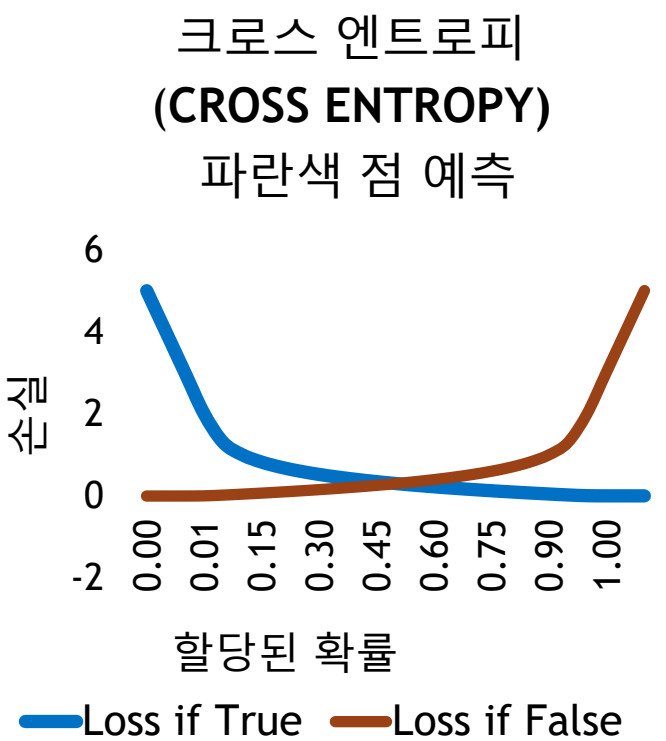
크로스 엔트로피 (CROSS ENTROPY)





크로스 엔트로피 (CROSS ENTROPY)



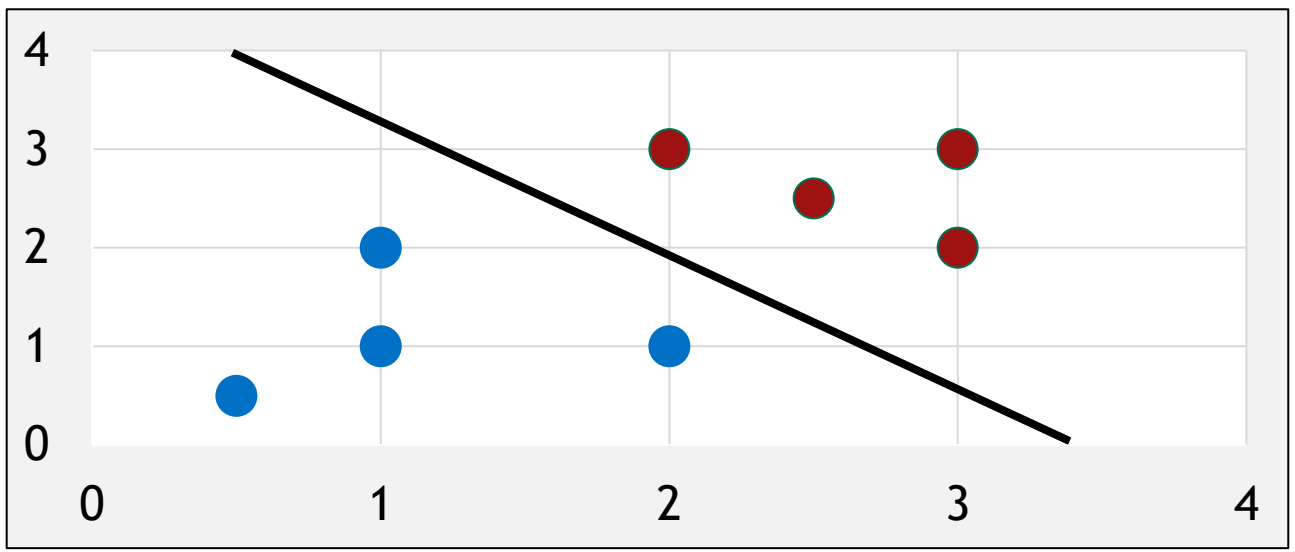


$$Loss = -(t(x) \cdot \log(p(x) + \big(1 - t(x)\big) \cdot \log(1 - p(x)))$$

$$t(x) = target\ (1\ if\ True, 0\ if\ False)$$

$$p(x) = probability\ prediction\ of\ point\ x$$

크로스 엔트로피 (CROSS ENTROPY)




```python
def cross_entropy(y_hat, y_actual):
    """Infinite error for misplaced confidence."""
    loss = log(y_hat) if y_actual else log(1-y_hat)
    return -1*loss
```