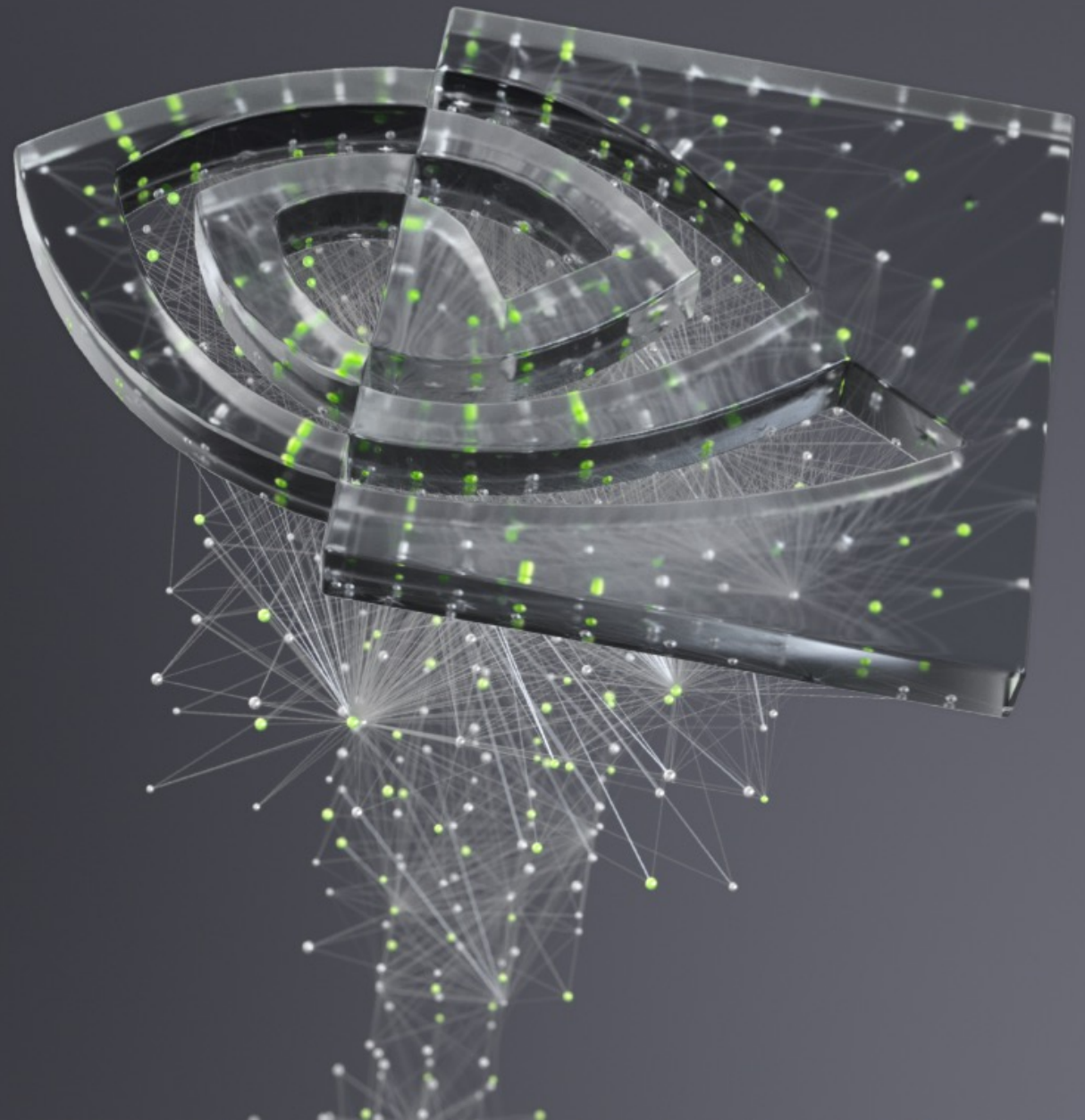


NVIDIA DEEP LEARNING INSTITUTE

딥러닝의 기초

2부: 뉴럴 네트워크의 트레이닝 방식

목차

목차 – 2부

- 요약
- 더 단순한 모델
- 뉴런에서 네트워크로
- 활성화 함수
- 과적합
- 뉴런에서 분류로

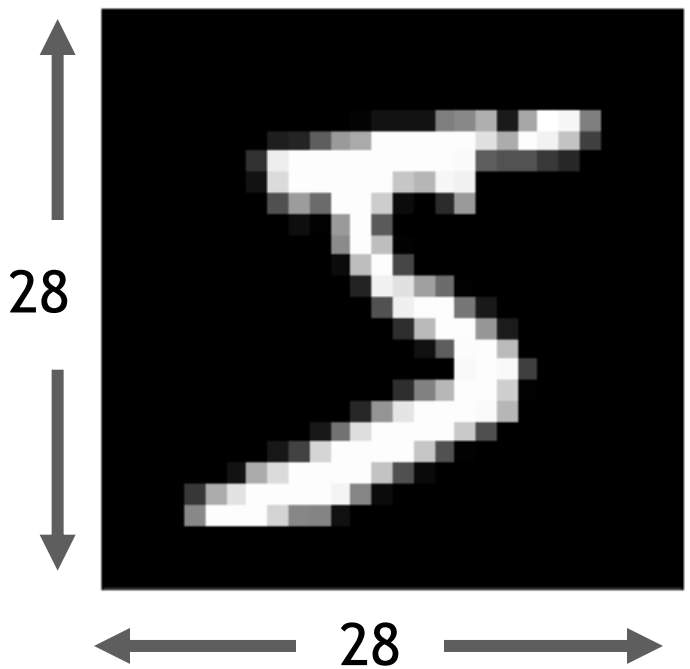
핸즈온 요약

방금 어떤 것들을 배웠나요?

- 데이터 로드 및 시각화
- 데이터 전처리(재구성, 정규화, 범주형)
- 모델 생성
- 모델 컴파일
- 데이터에 대해 모델 트레이닝

데이터 준비 (DATA PREPARATION)

배열을 통한 입력



28x28 = 784

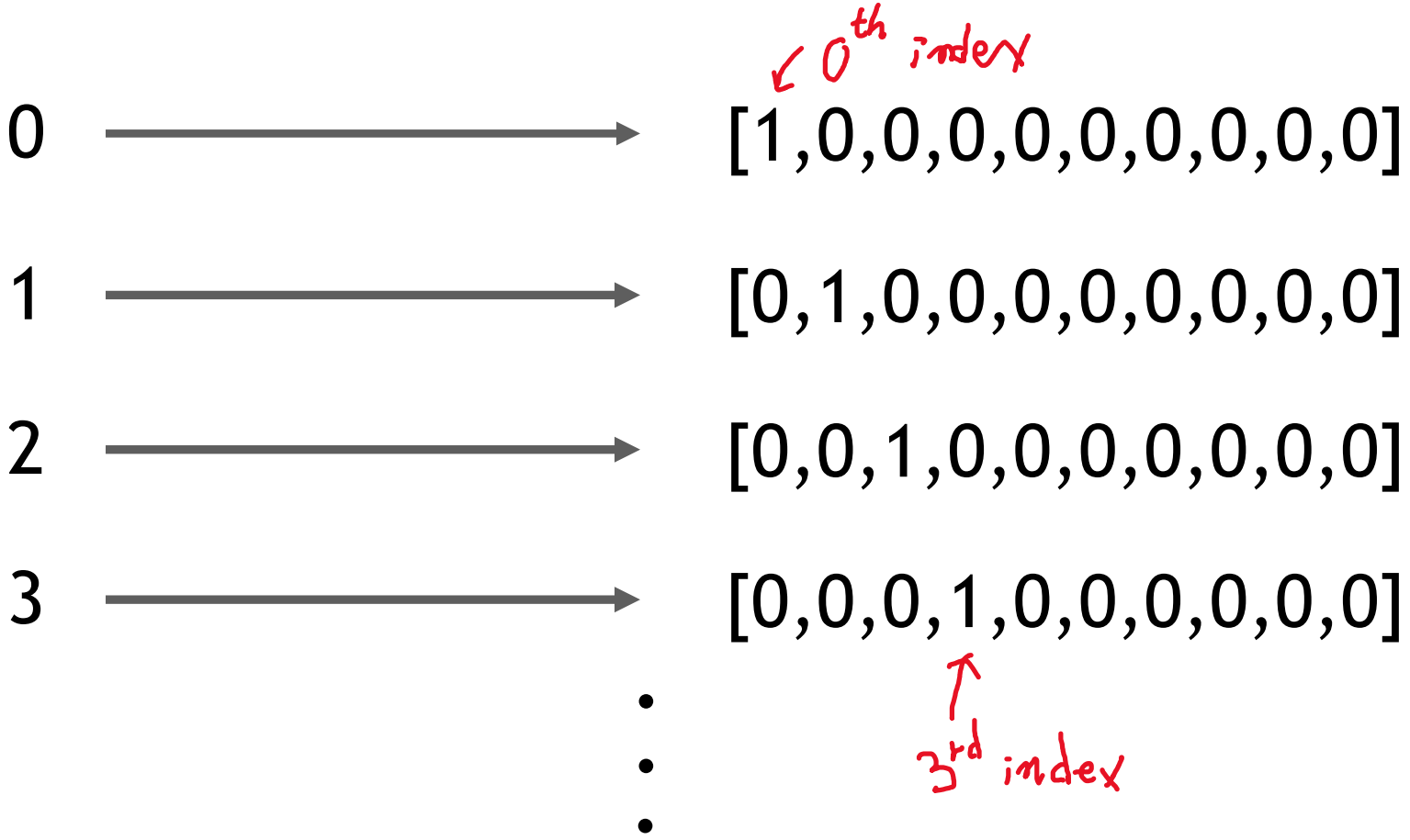
[0,0,0,24,75,184,185,78,32,55,0,0,0...]

÷ 255

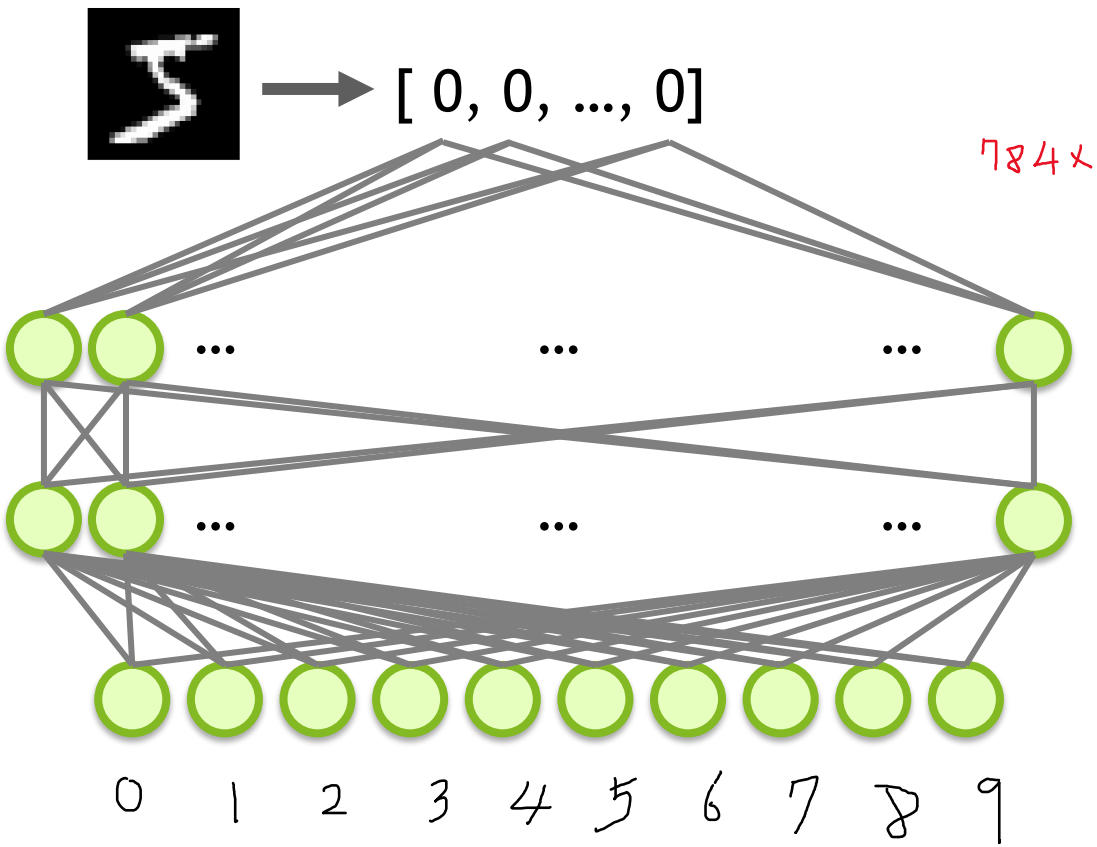
[0, 0, 0, 1, 0.3 ...]

데이터 준비 (DATA PREPARATION)

범주형(categorical)으로 타깃팅 : One-Hot-Encoding



트레이닝되지 않은 모델



* 파라미터(w, b)의 갯수는?

input (784,)

784 × 512 + 512
w b
+

(512,)

512 × 512 + 512
w b
+

(512,)

512 × 10 + 10
w b

(10,)

output

레이어 크기

hidden layers

11
668,672

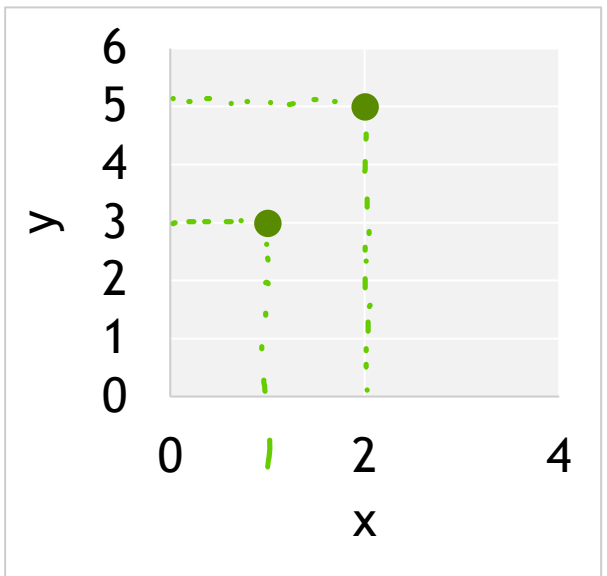

더욱 단순한 모델

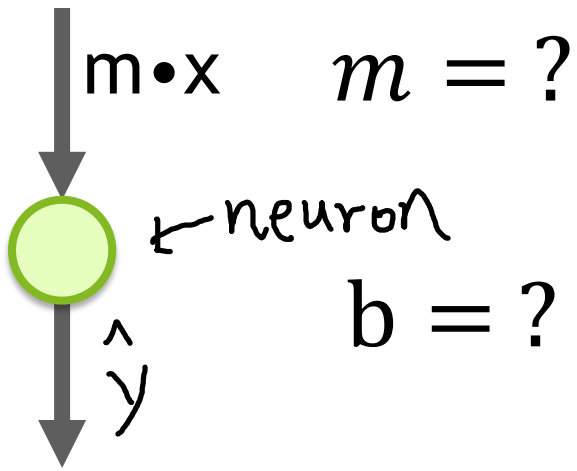
더욱 단순한 모델

$$y = mx + b$$

output ↑ , m = w

x	y
1	3
2	5





$m = ?$

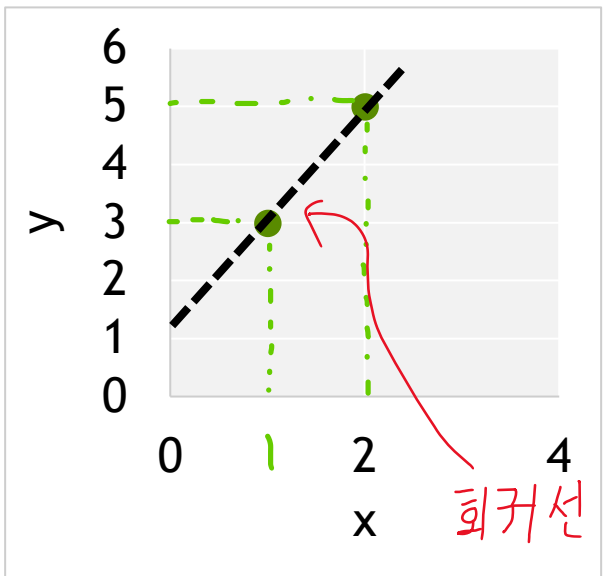
$b = ?$

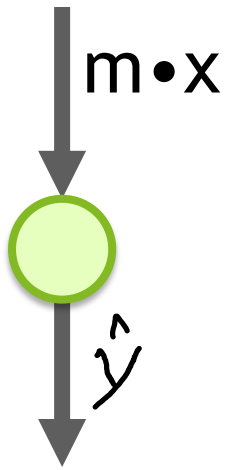
더욱 단순한 모델

$$y = mx + b$$

output

x	y
1	3
2	5





2

$m = ?$

$b = ?$

1

* y : 실제 값 (정답, label, 가설 함수)

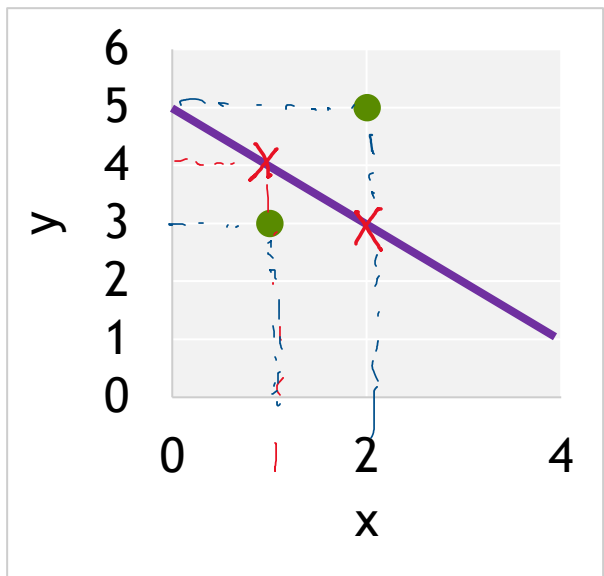
$\hat{y}$: 예측 값

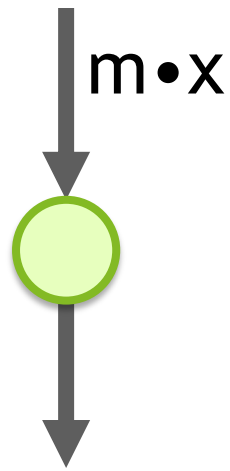
더욱 단순한 모델

$$y = mx + b$$

실제값(정답) 예측 값(가설)

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





무작위로 시작

$$m = -1$$

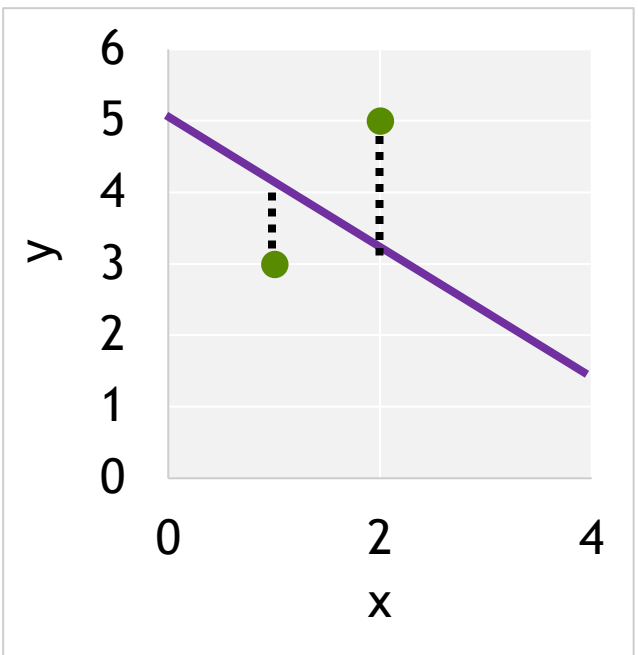
$$b = 5$$

더욱 단순한 모델

$$y = mx + b$$

-1 , 5

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



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

$(3-4)^2 + (5-3)^2 / 2 = 2.5$

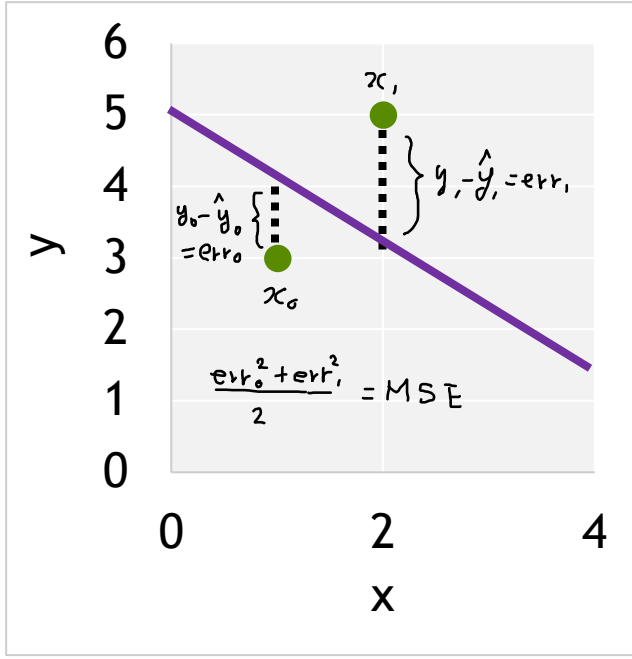
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2}$$

$\sqrt{2.5} = 1.6$

더욱 단순한 모델

$$y = mx + b$$

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



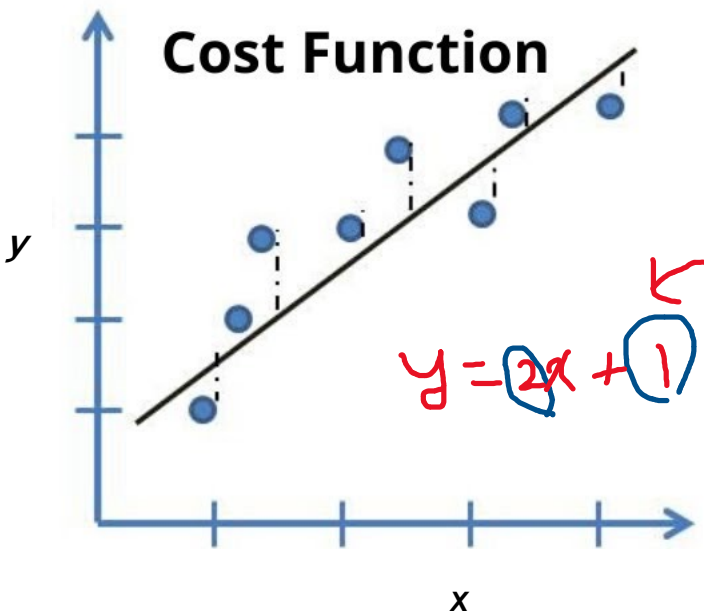

```python
data = [(1, 3), (2, 5)]
m = -1
b = 5

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
```

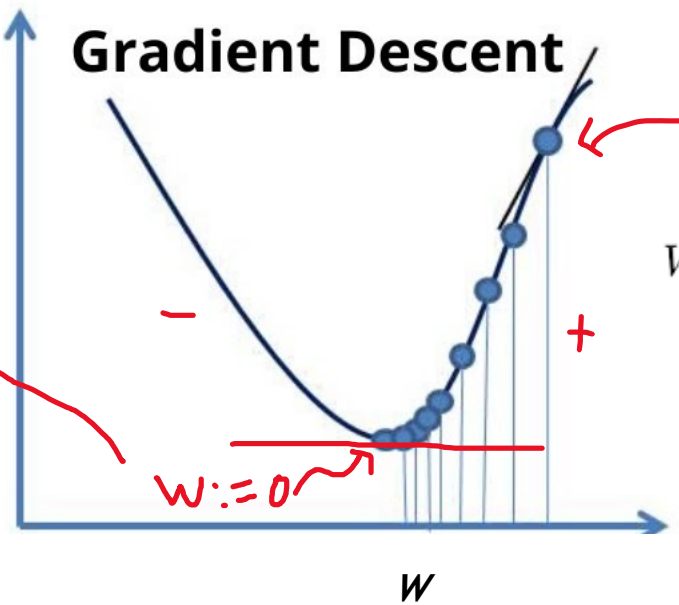
경사 하강법(GRADIENT DESCENT)

How to minimize cost(loss) $cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$

예측값($\hat{y}$) 정답(실제값(y))



cost(W)

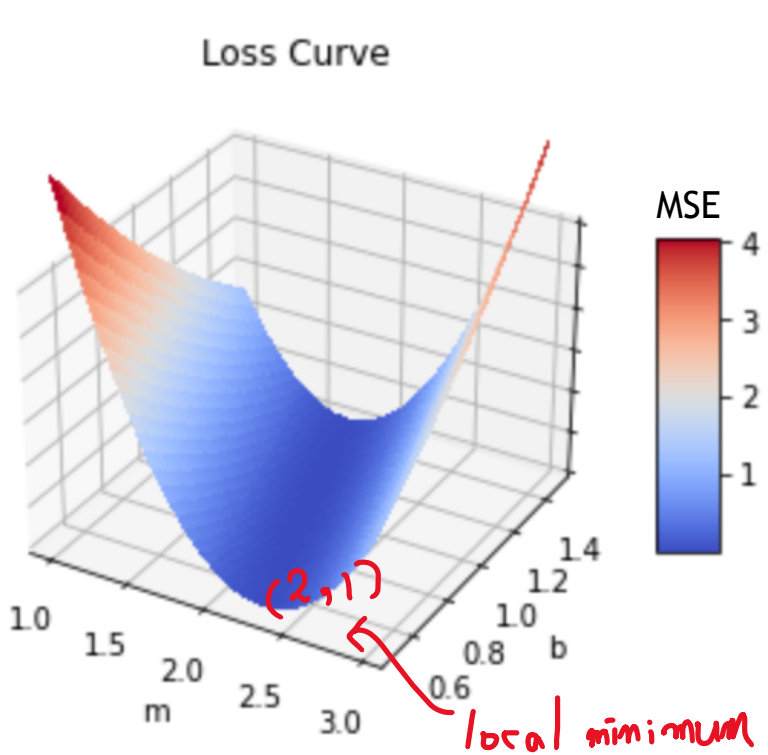


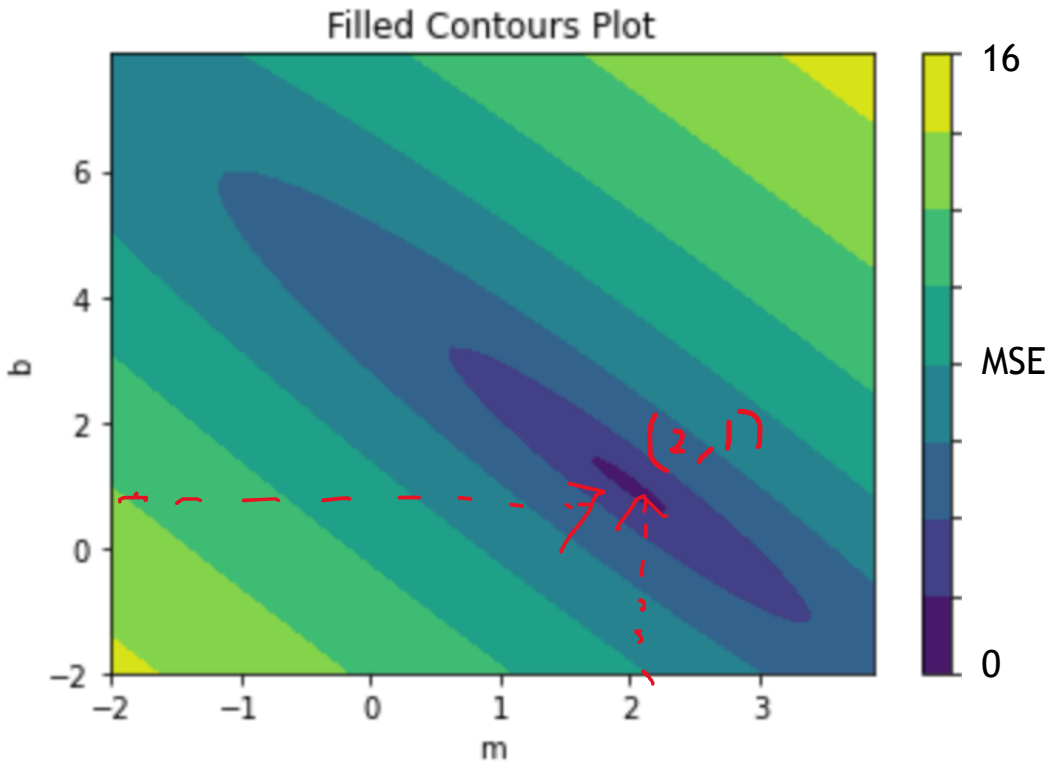
기울기 업데이트

$W := W - \alpha\frac{\partial}{\partial W}cost(W)$

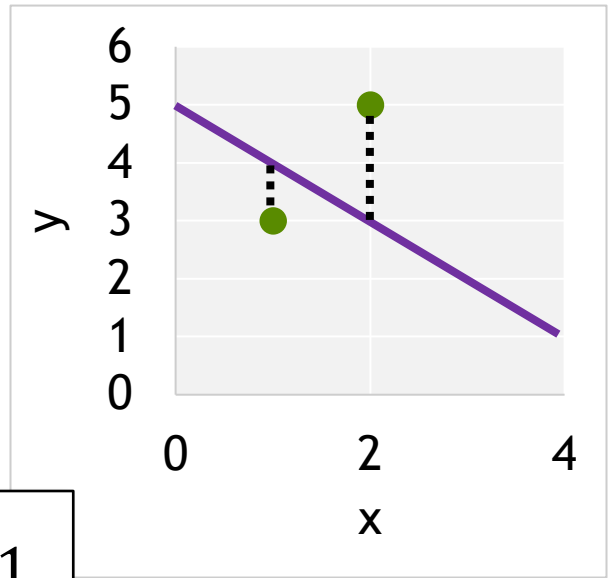
step(learning rate)

손실 곡선 (THE LOSS CURVE)



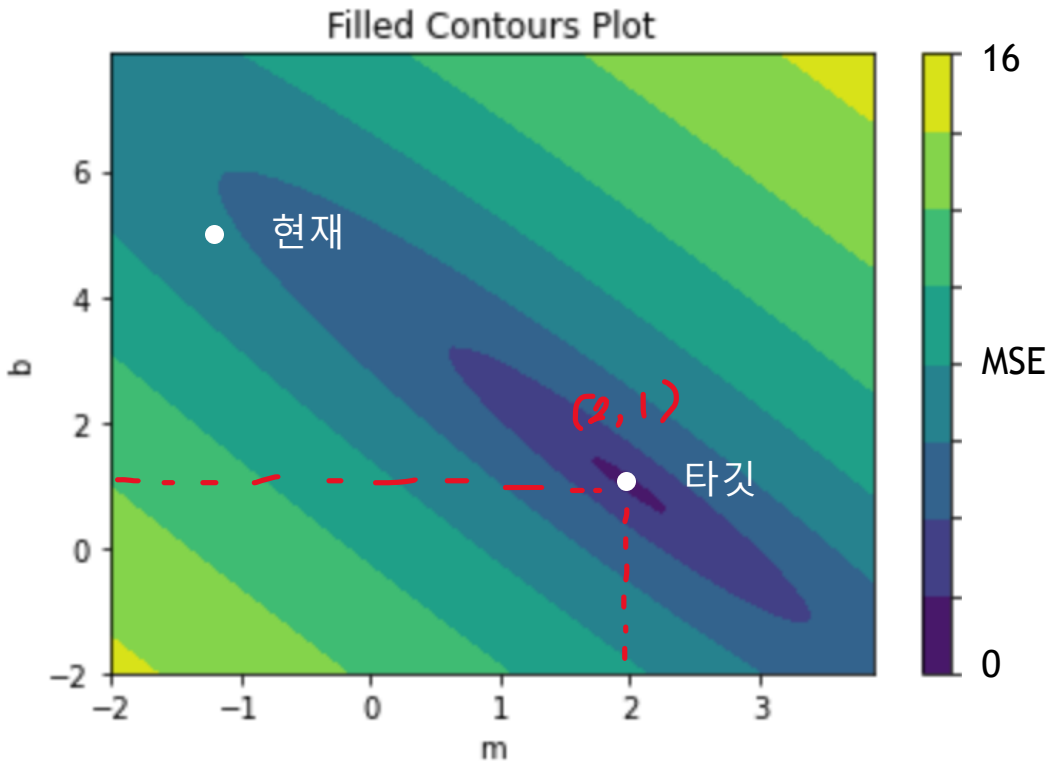


손실 곡선 (THE LOSS CURVE)

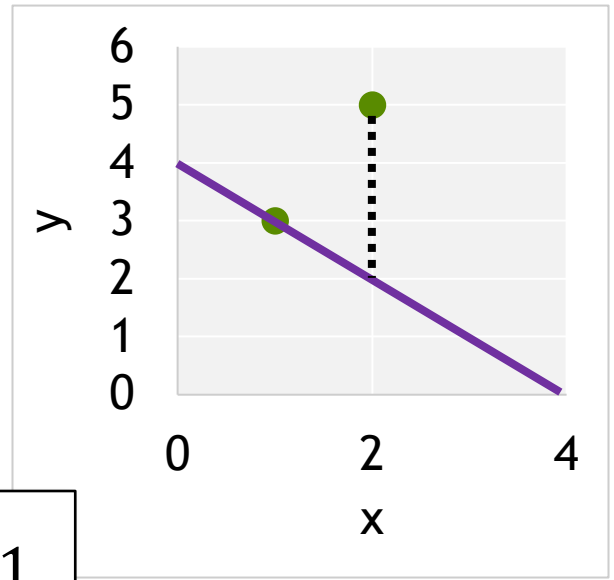


$m = -1$

$b = 5$

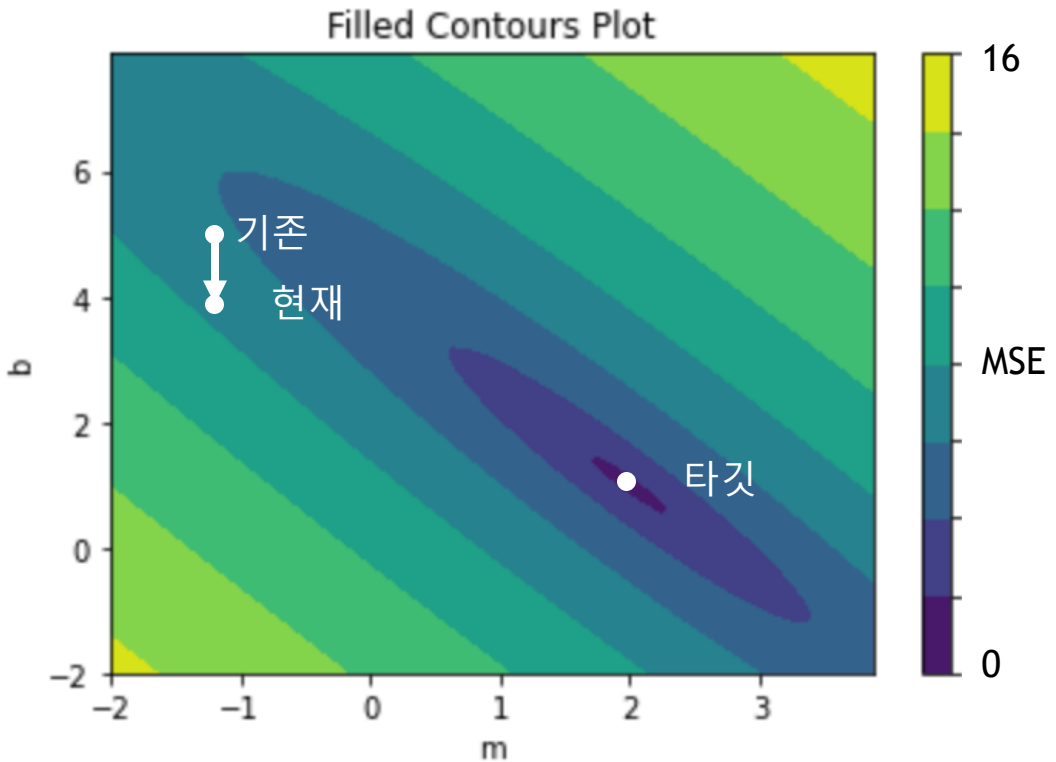


손실 곡선 (THE LOSS CURVE)

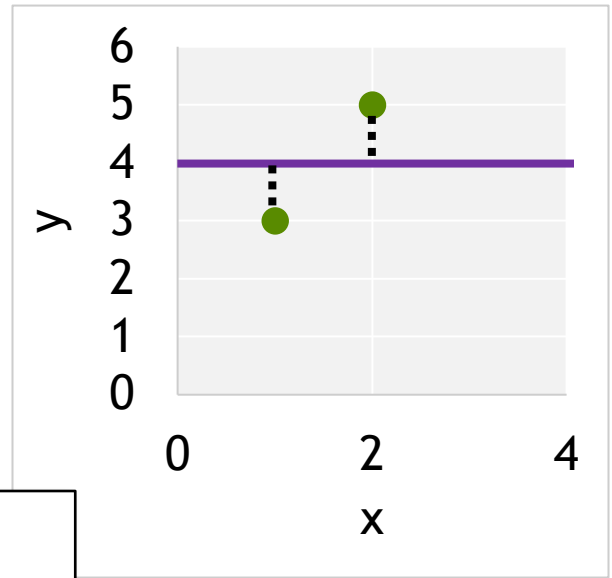


$m = -1$

$b = 4$

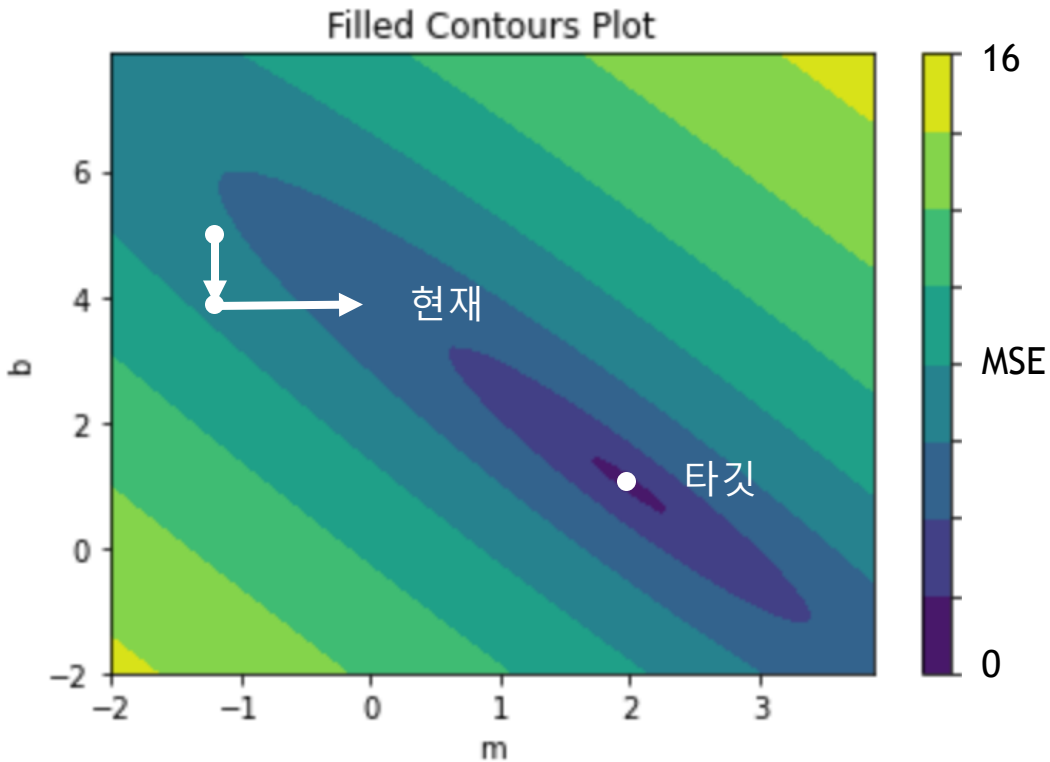


손실 곡선 (THE LOSS CURVE)



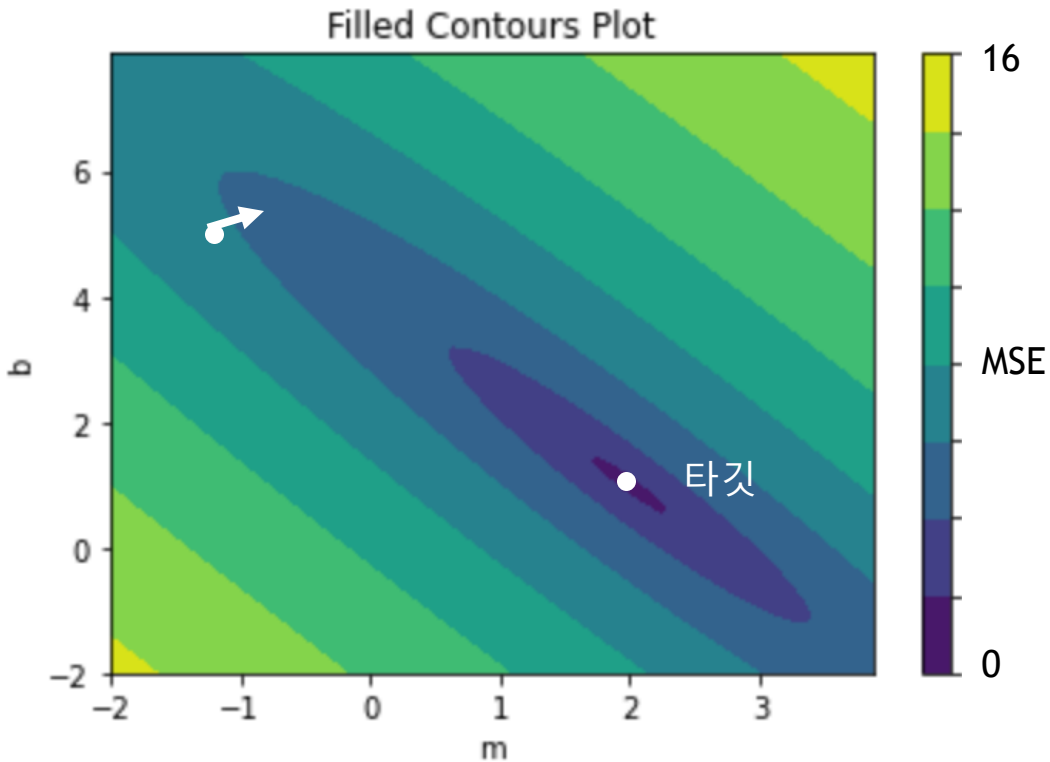
$m = 0$

$b = 4$



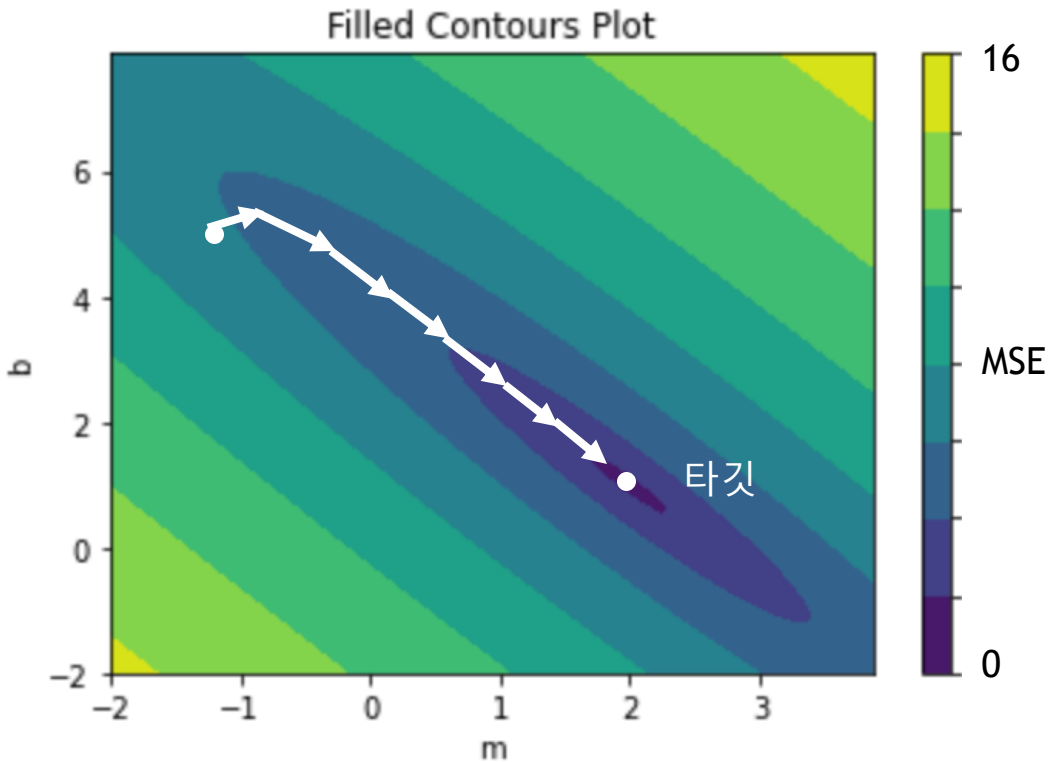
손실 곡선 (THE LOSS CURVE)

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

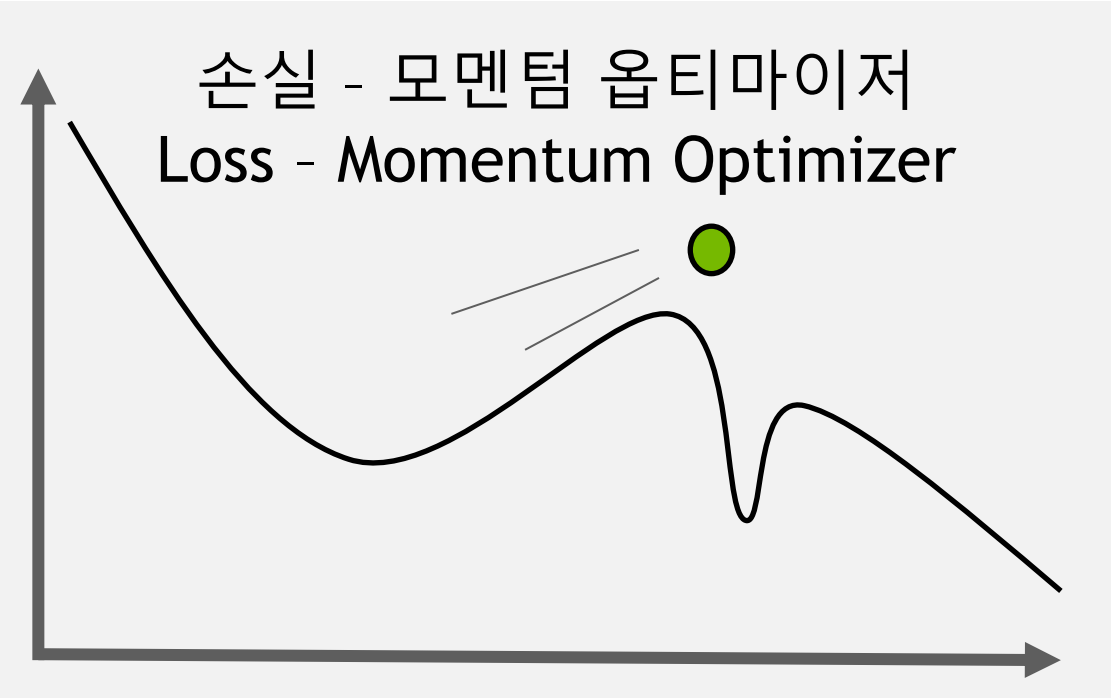


손실 곡선 (THE LOSS CURVE)

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

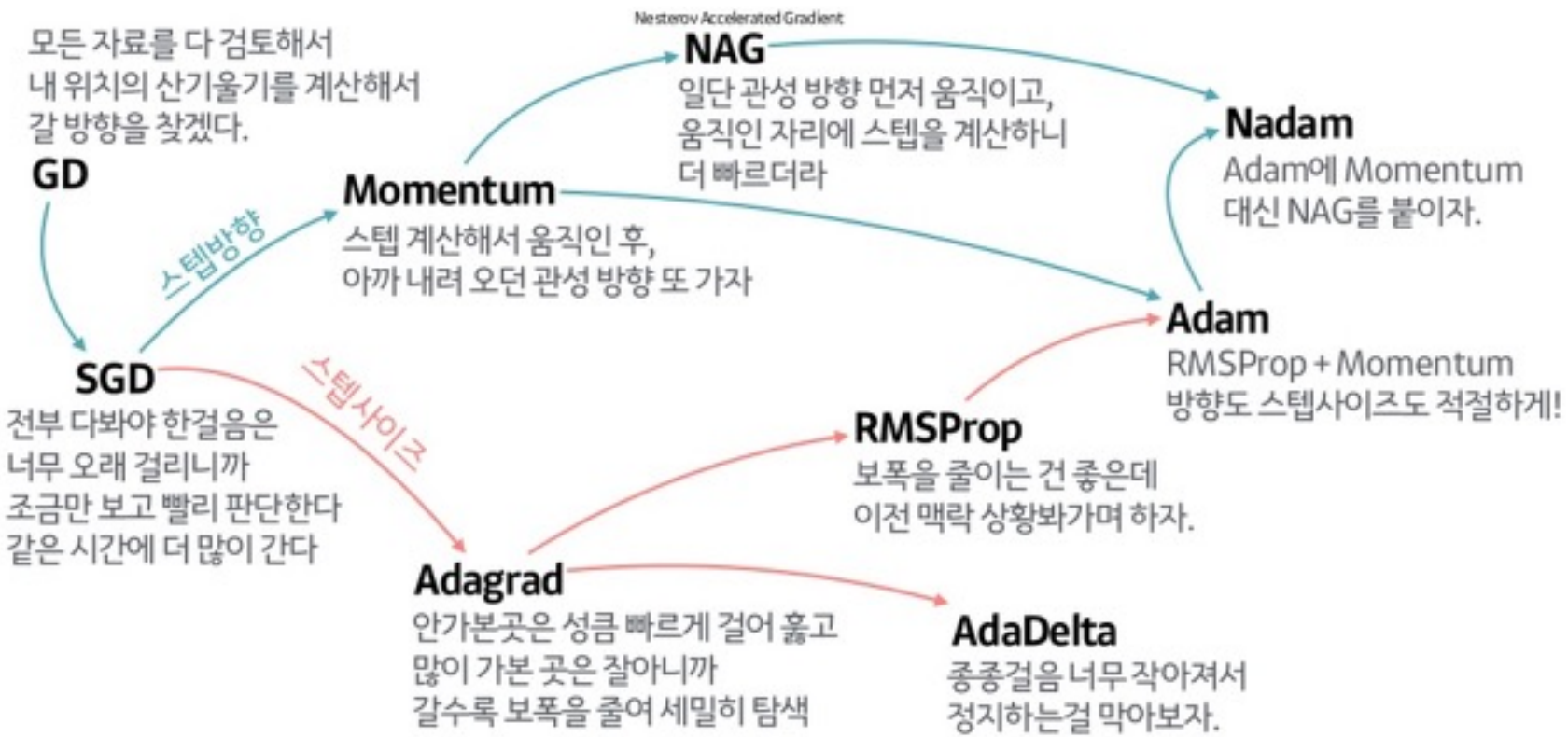


옵티마이저 (OPTIMIZERS)

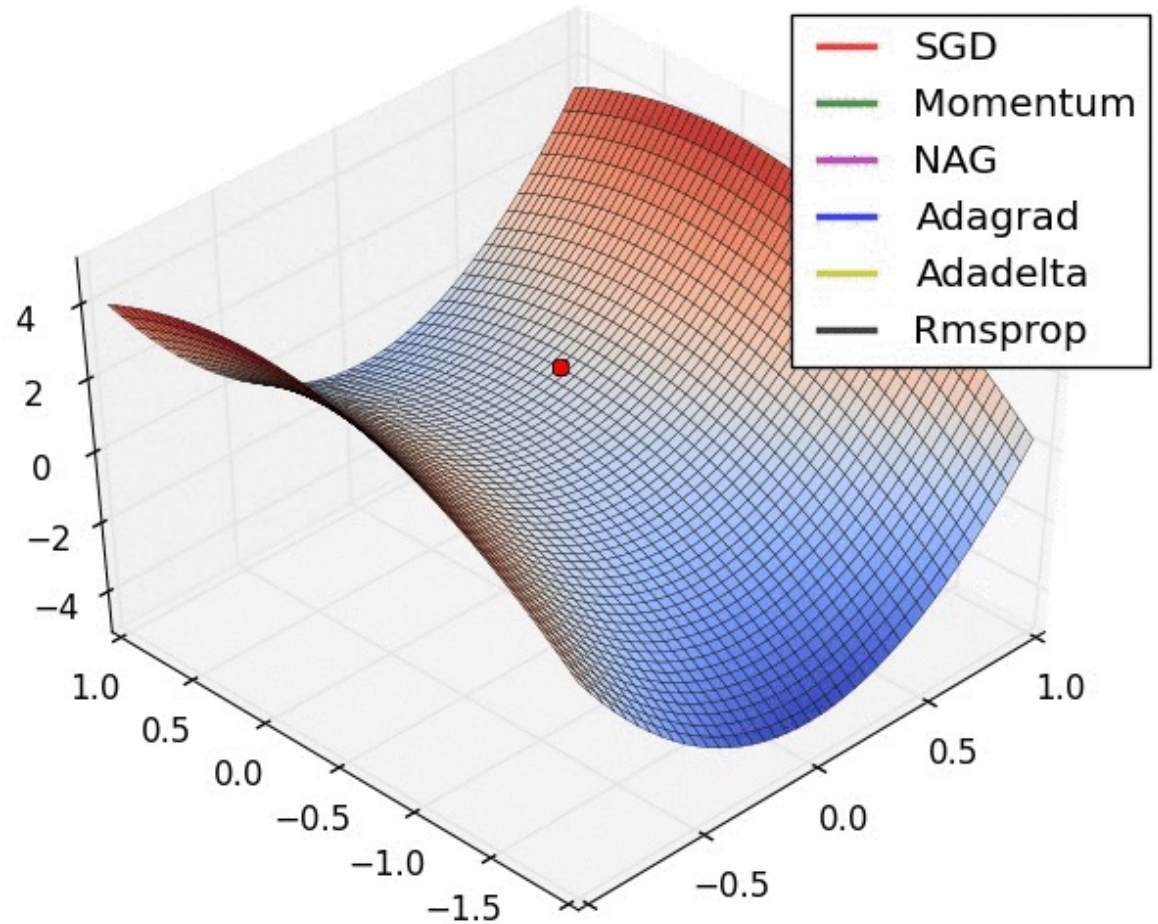


- Adam
- Adagrad
- RMSprop
- SGD

옵티마이저 (OPTIMIZERS)

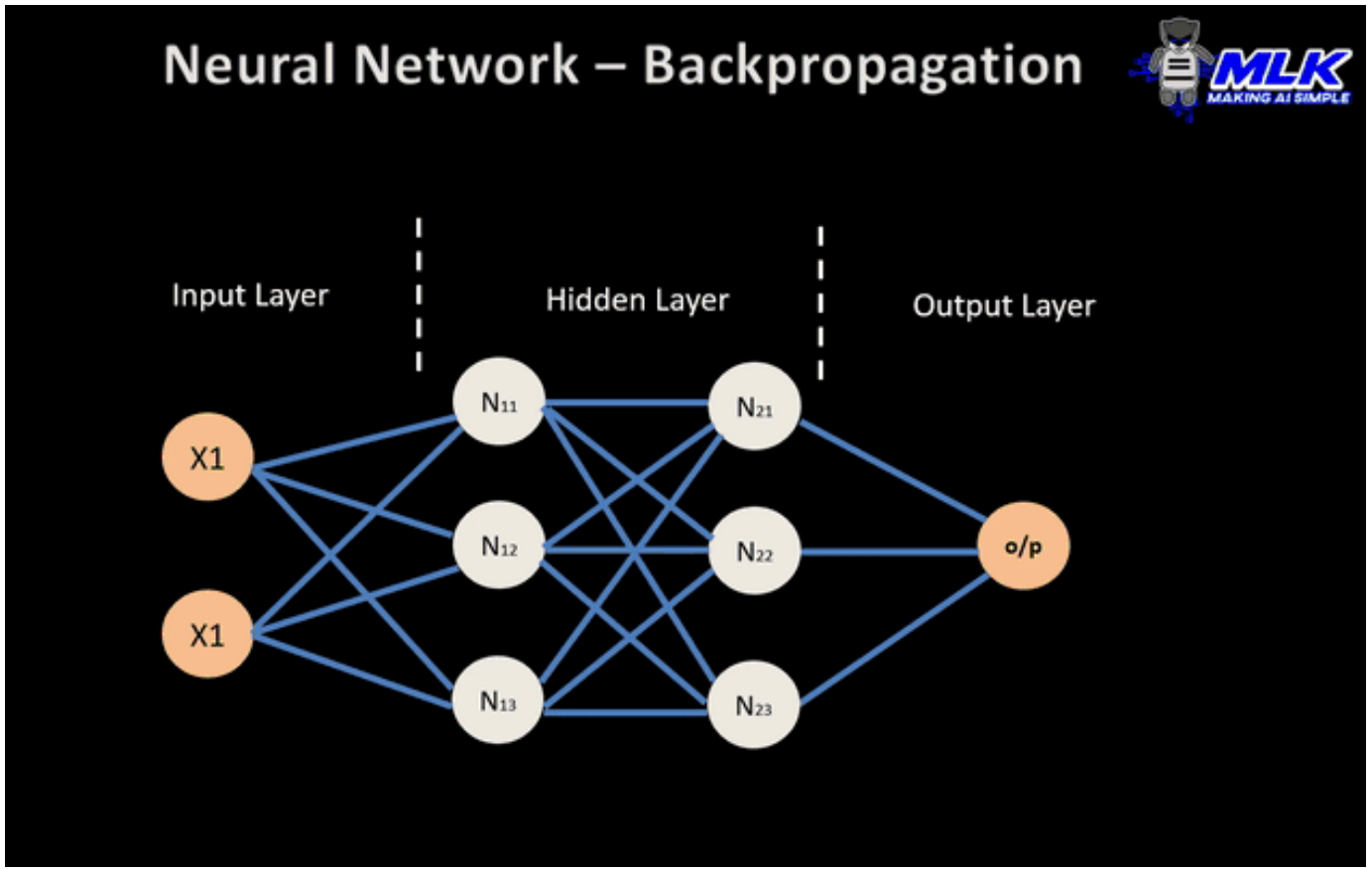


옵티마이저 (OPTIMIZERS) 성능 비교



오차 역전파 (ERROR BACKPROPAGATION)

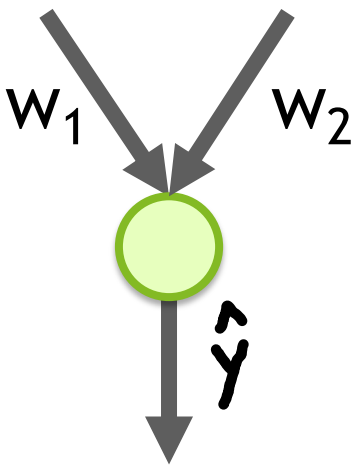
체인 룰(chain rule)사용



참조 : machinelearningknowledge.ai

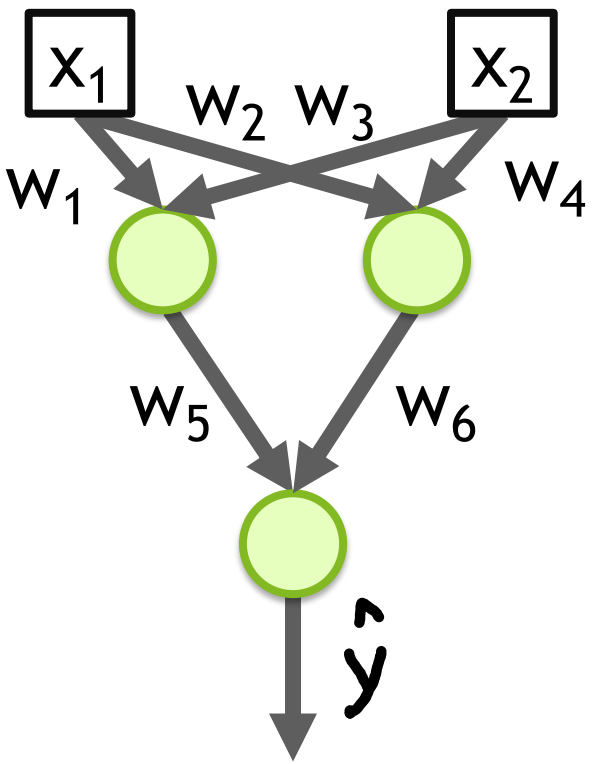

뉴런에서 네트워크로

네트워크 구축



- 더 많은 변수로의 확장

네트워크 구축

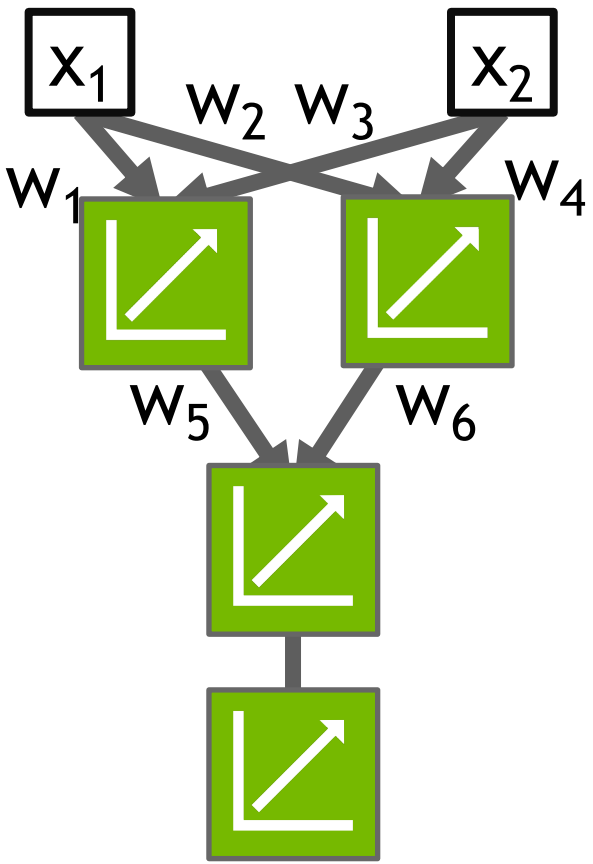


- 더 많은 변수로의 확장
- 뉴런들간의 연결

Backpropagation algorithm :

https://developers-dot-devsite-v2-prod.appspot.com/machine-learning/crash-course/backprop-scroll

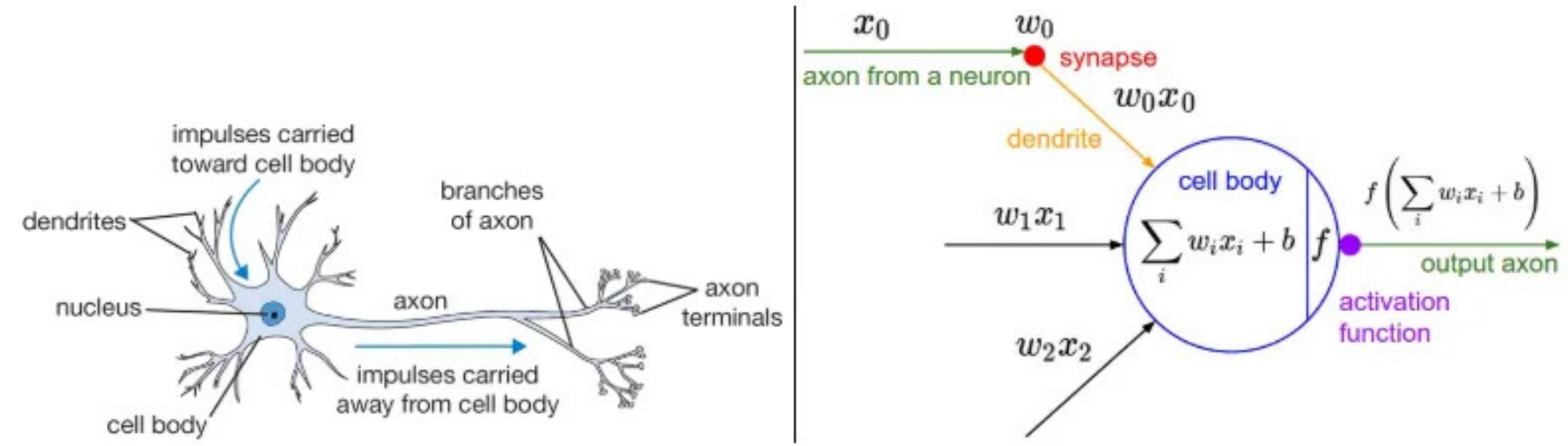
네트워크 구축



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

활성화 함수
ACTIVATION FUNCTIONS

활성화 함수 (ACTIVATION FUNCTIONS)



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

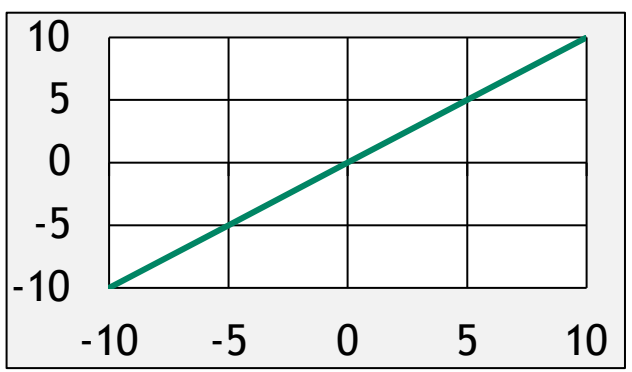
참조 : Standford University cs231n

활성화 함수 (ACTIVATION FUNCTIONS)

Linear(선형)

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

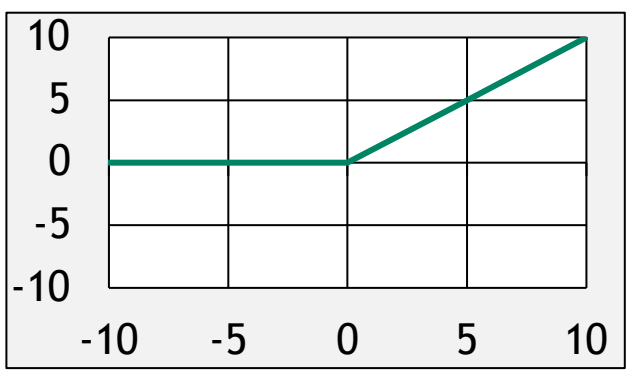
```
# Multiply each input
# with a weight (w) and
# add intercept (b)
y_hat = wx+b
```



ReLU

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

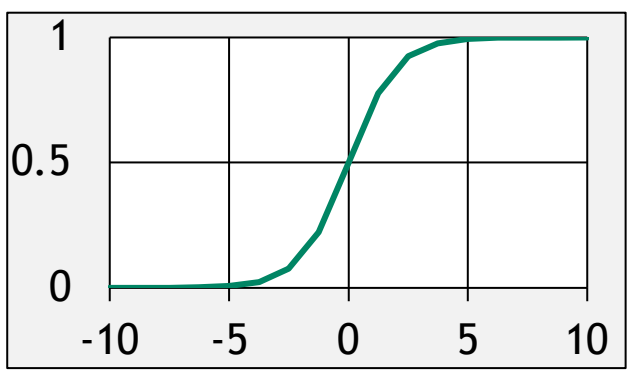
```
# Only return result
# if total is positive
linear = wx+b
y_hat = linear * (linear > 0)
```



Sigmoid(시그모이드)

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

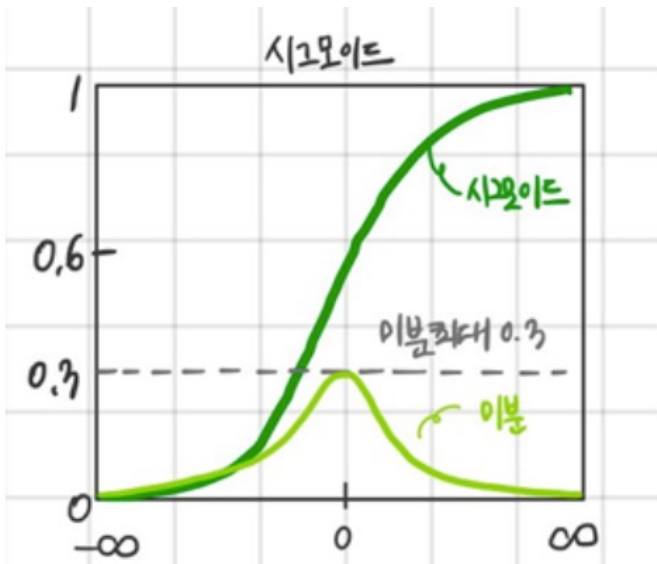
```
# Start with line
linear = wx + b
# Warp to - inf to 0
inf_to_zero = np.exp(-1 * linear)
# Squish to -1 to 1
y_hat = 1 / (1 + inf_to_zero)
```



가중치 소실 문제 (VANISHING GRADIENT)

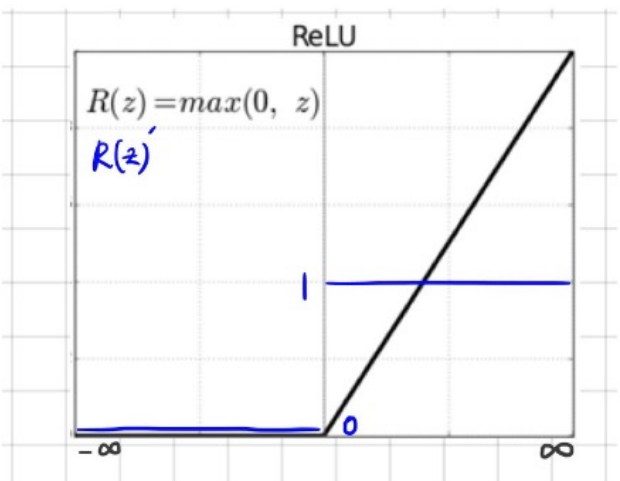
Sigmoid(시그모이드)

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



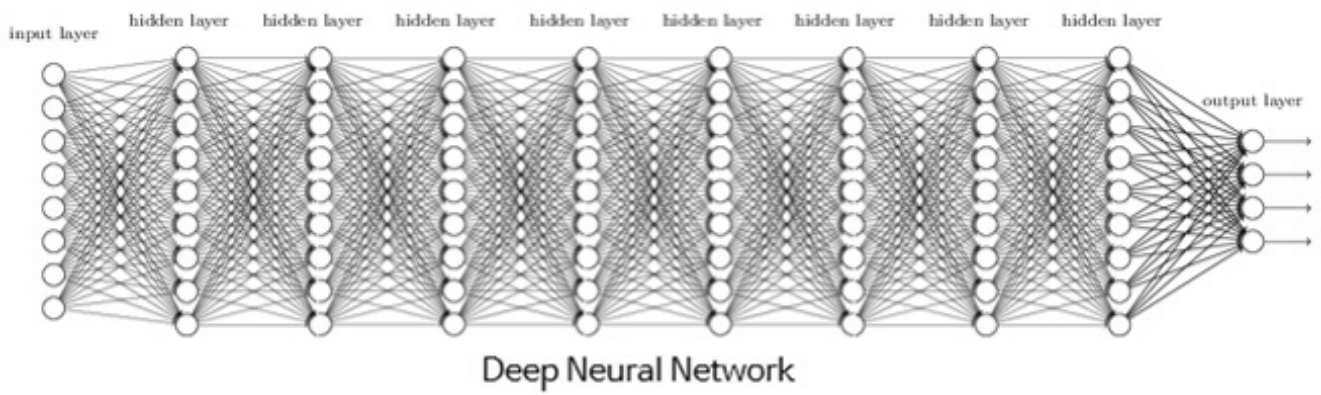
ReLU

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



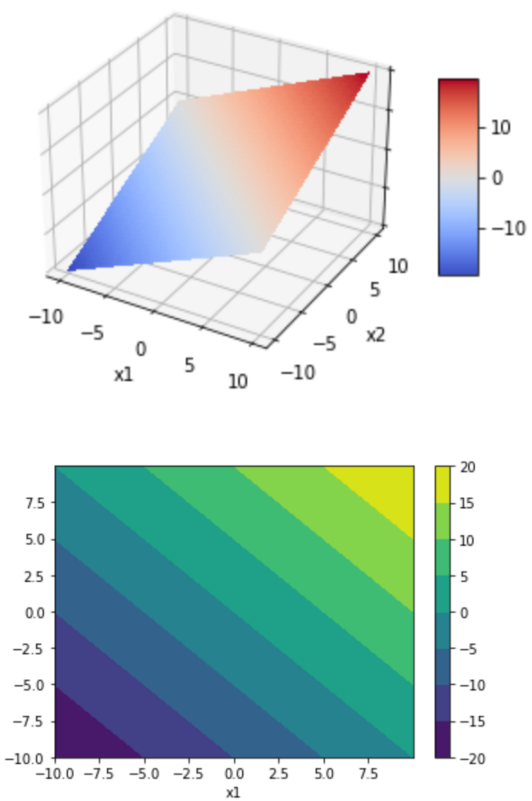
$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \qquad f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$$

가중치 소실 문제 (VANISHING GRADIENT)

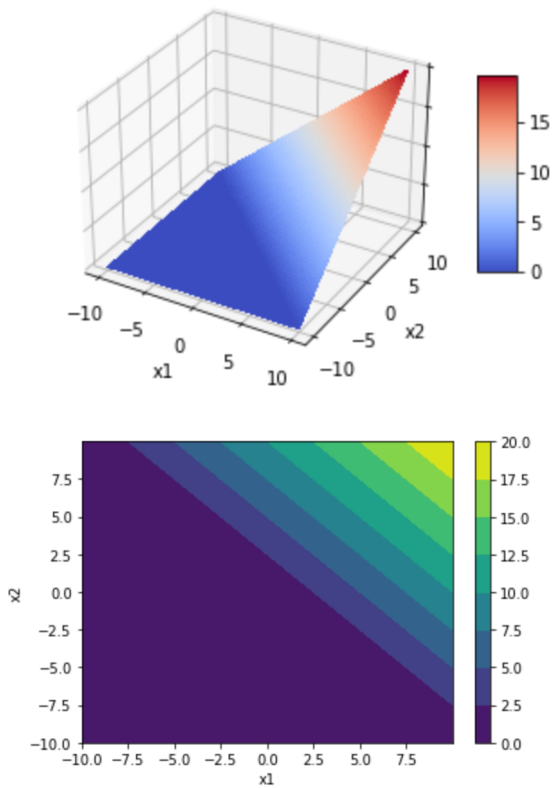


활성화 함수 (ACTIVATION FUNCTIONS)

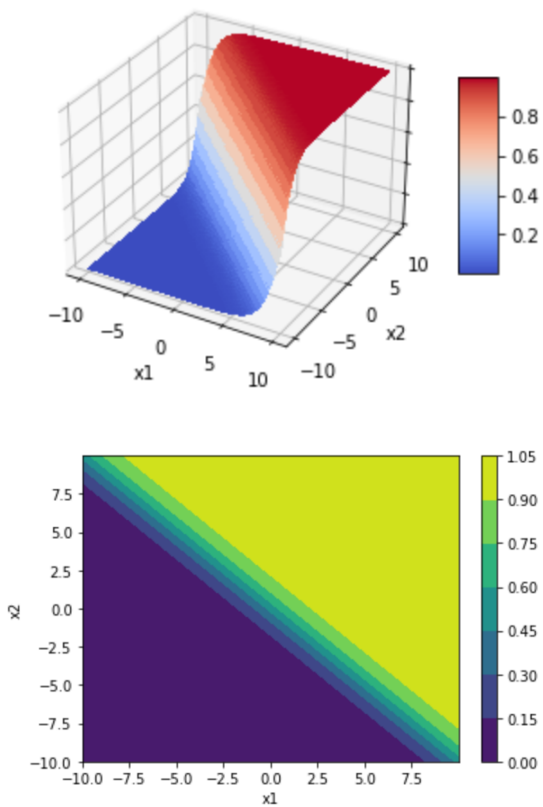
Linear(선형)



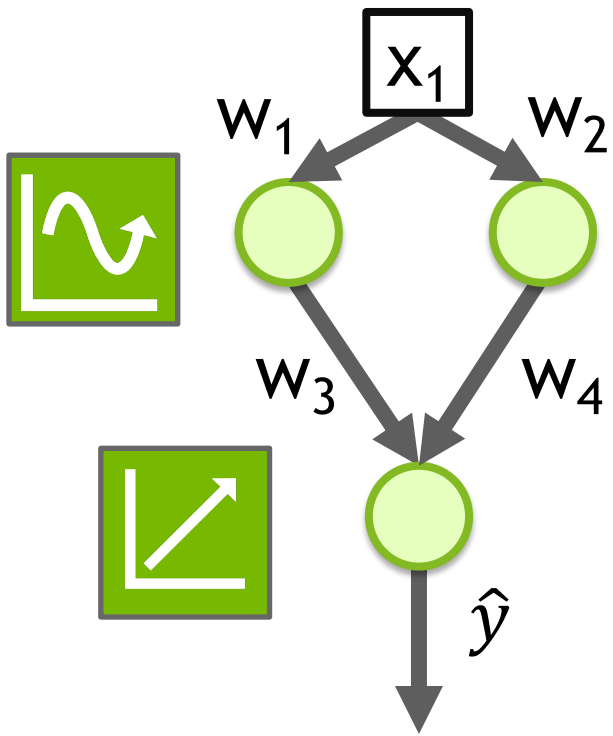
ReLU

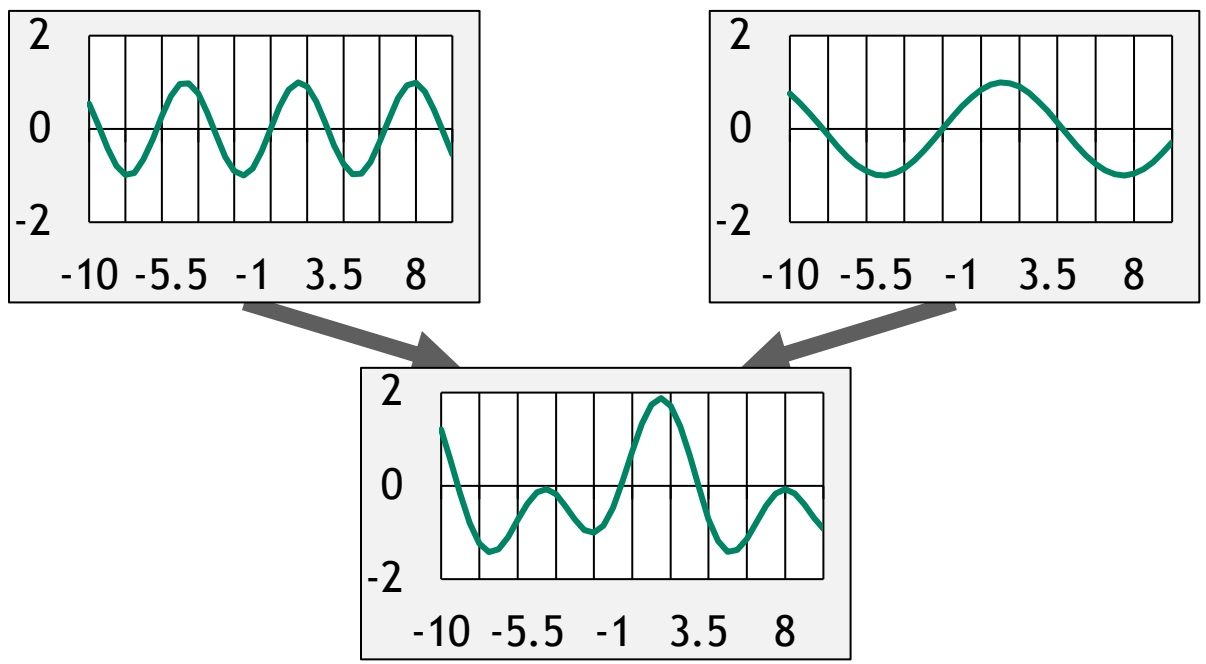


Sigmoid(시그모이드)



활성화 함수 (ACTIVATION FUNCTIONS)

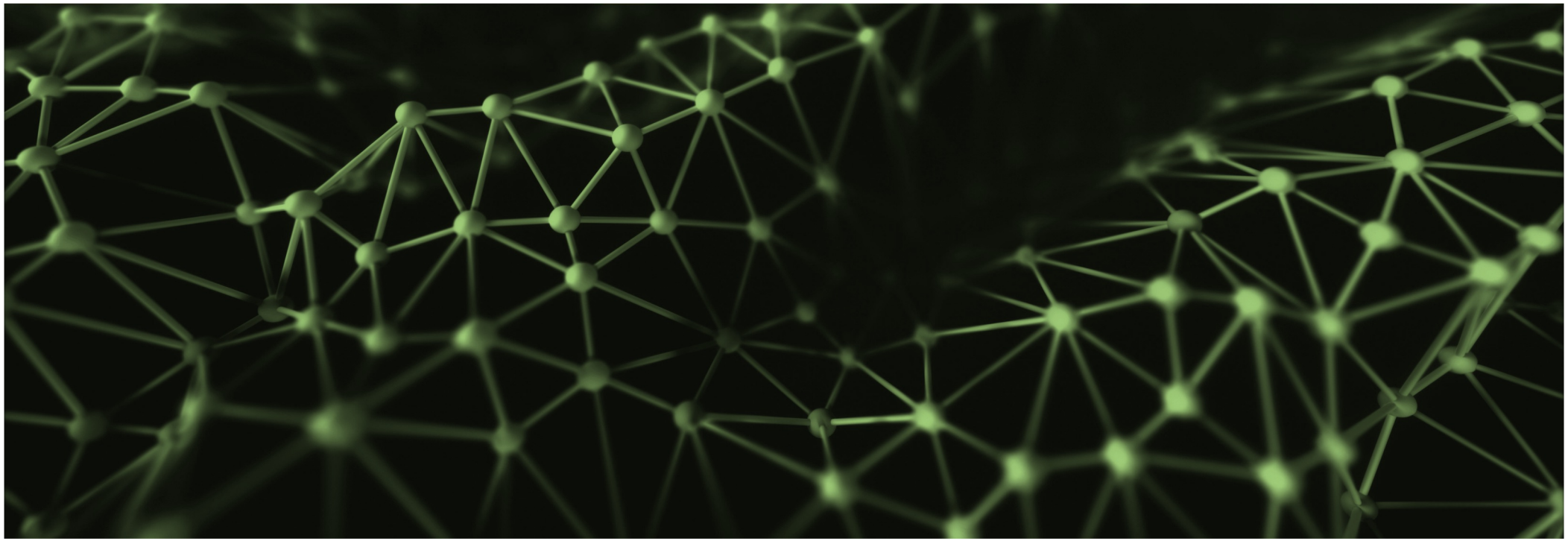




과적합
OVERFITTING

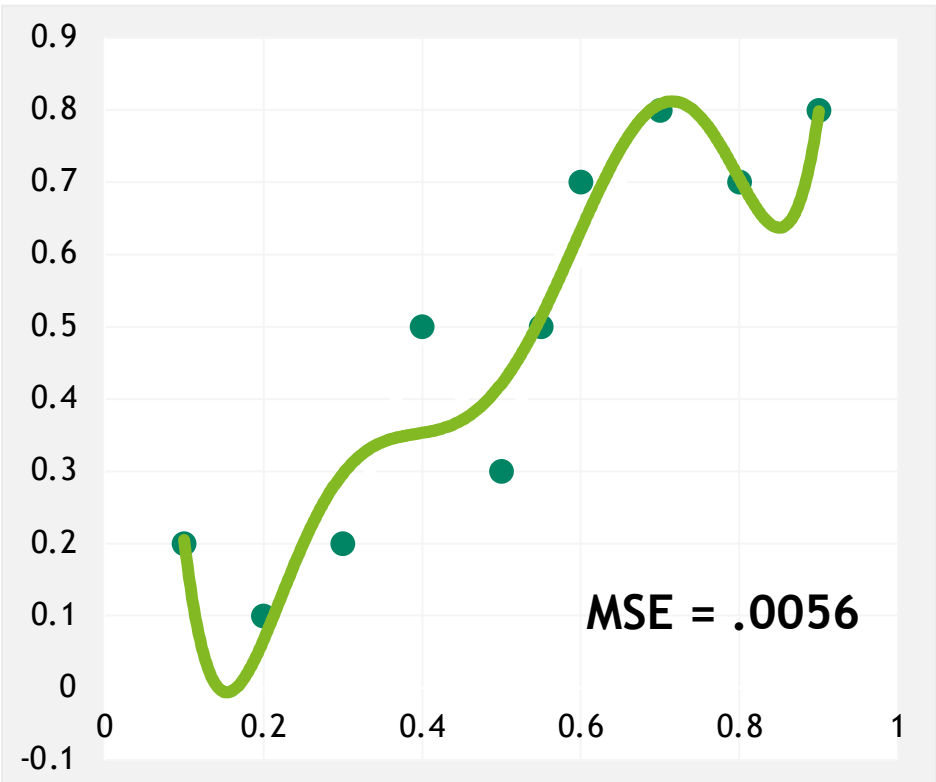
과적합 (OVERFITTING)

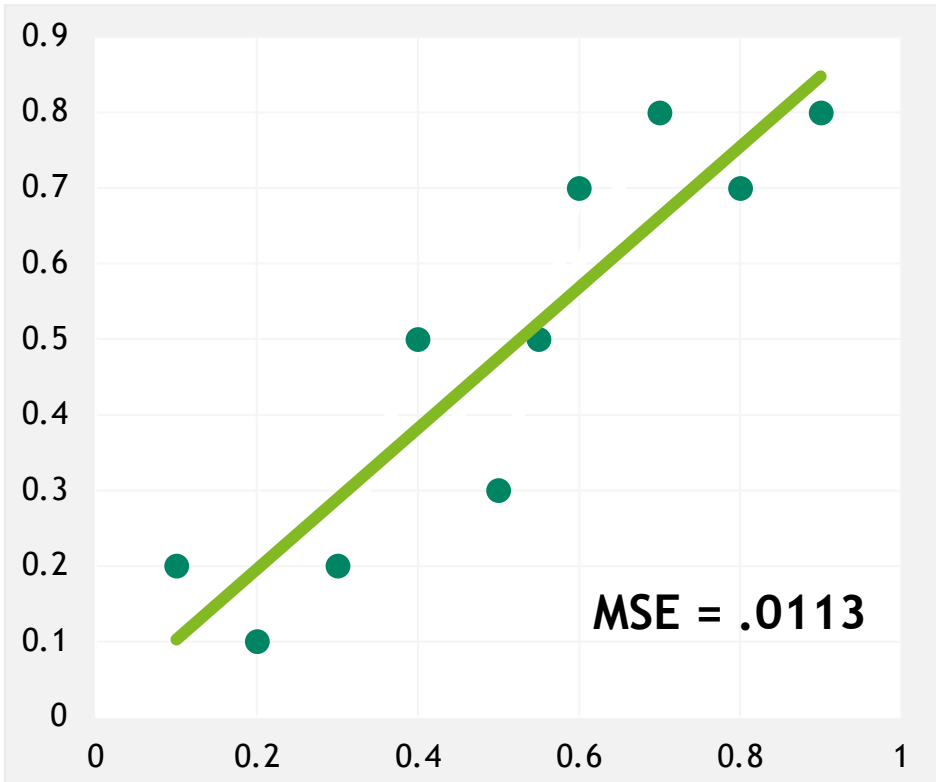
거대한 뉴럴 네트워크를 구축하면 되지 않을까?



과적합 (OVERFITTING)

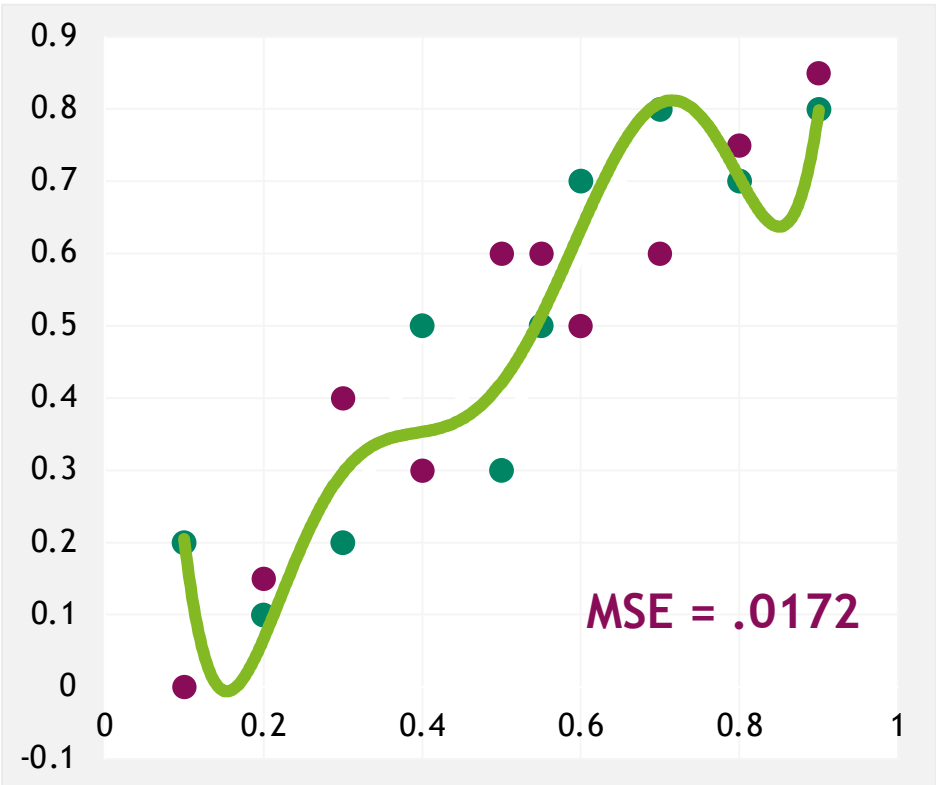
어떤 추세선이 더 나은가?

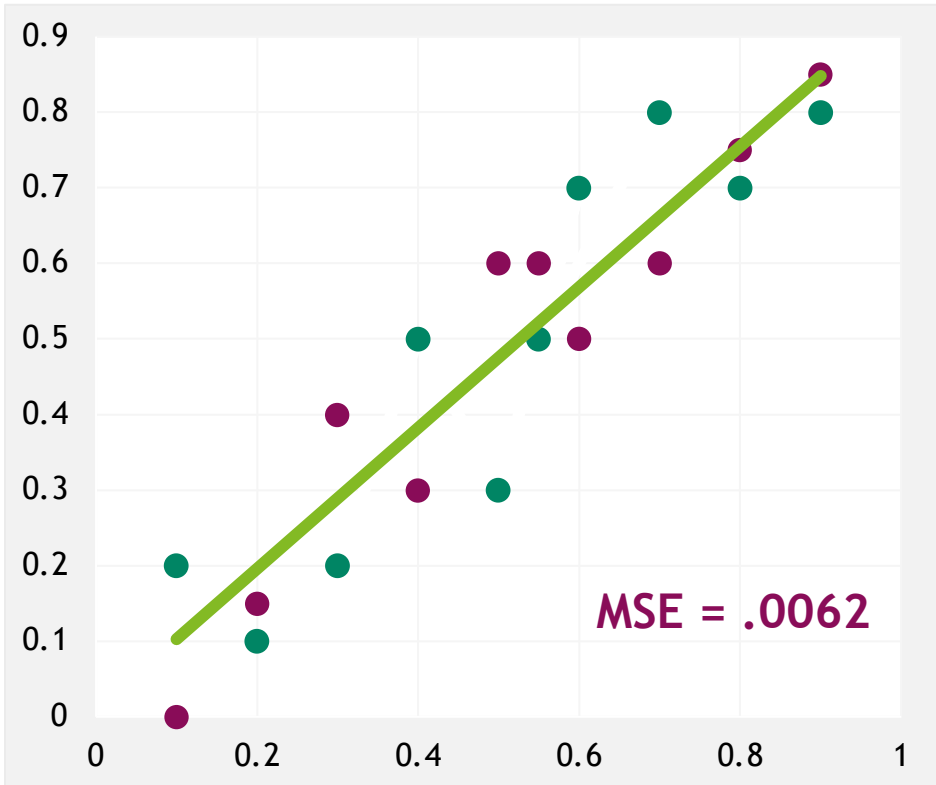




과적합 (OVERFITTING)

어떤 추세선이 더 나은가?





트레이닝 데이터와 검증 데이터 비교

암기 회피(Avoid memorization)

트레이닝 데이터 (Training data)

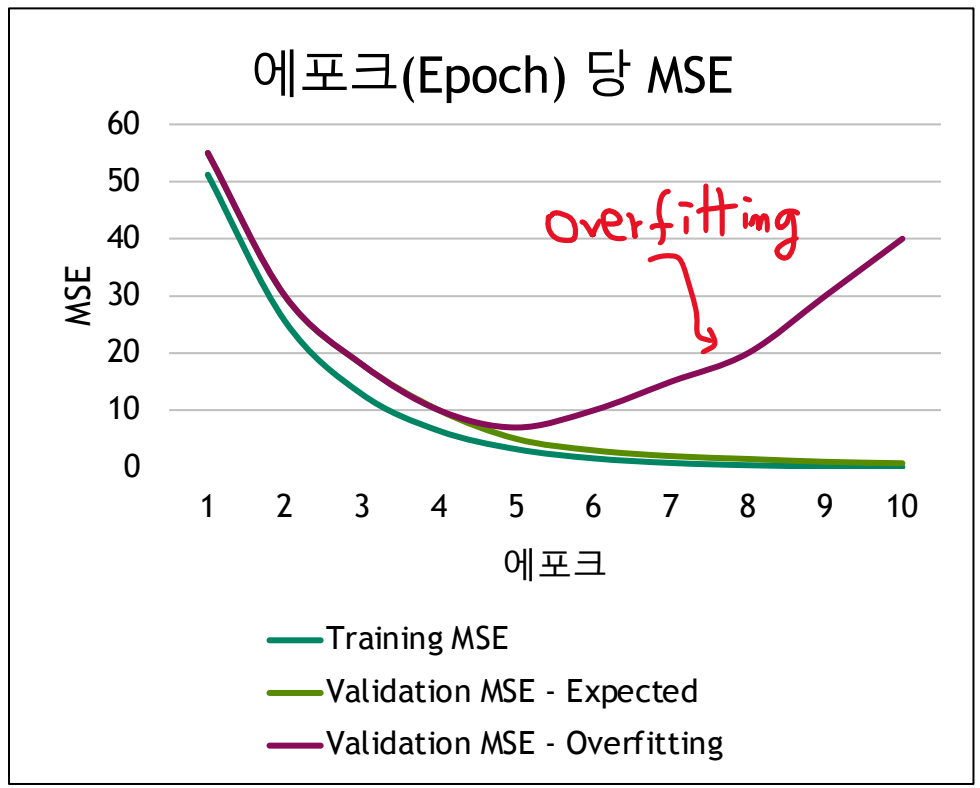
- 학습해야 하는 모델의 핵심 데이터세트

검증 데이터 (Validation data)

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

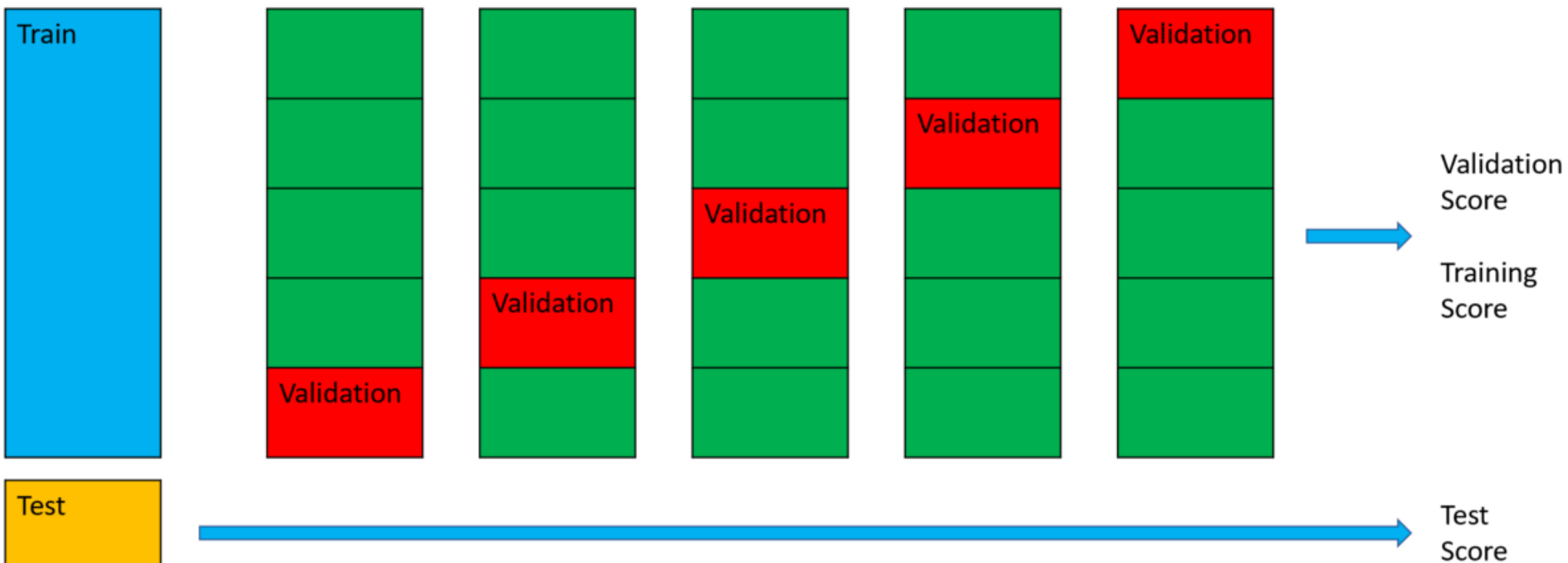
과적합 (Overfitting)

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

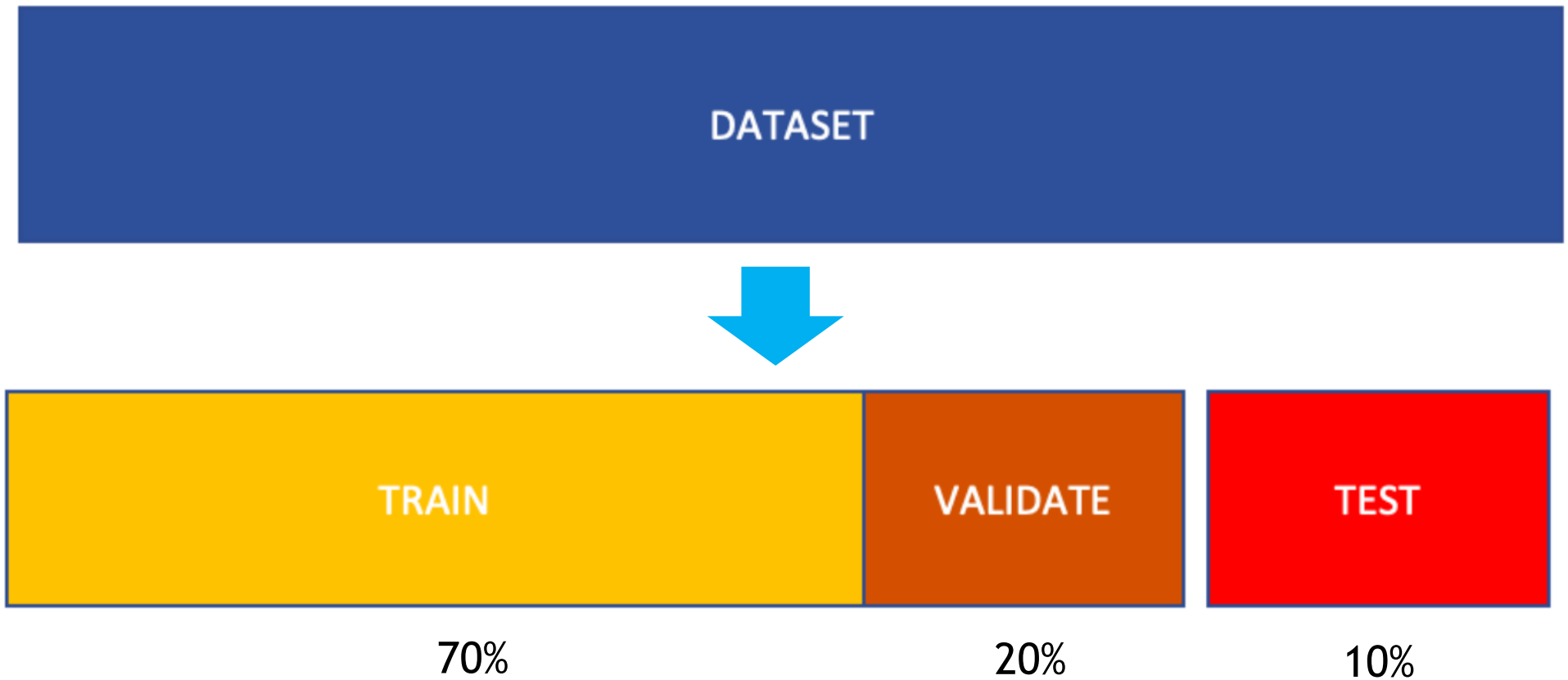



```
model.fit(x_train, y_train, epochs=20, verbose=1, validation_data=(x_valid, y_valid))
```

TRAIN DATA, VALIDATION DATA, TEST DATA

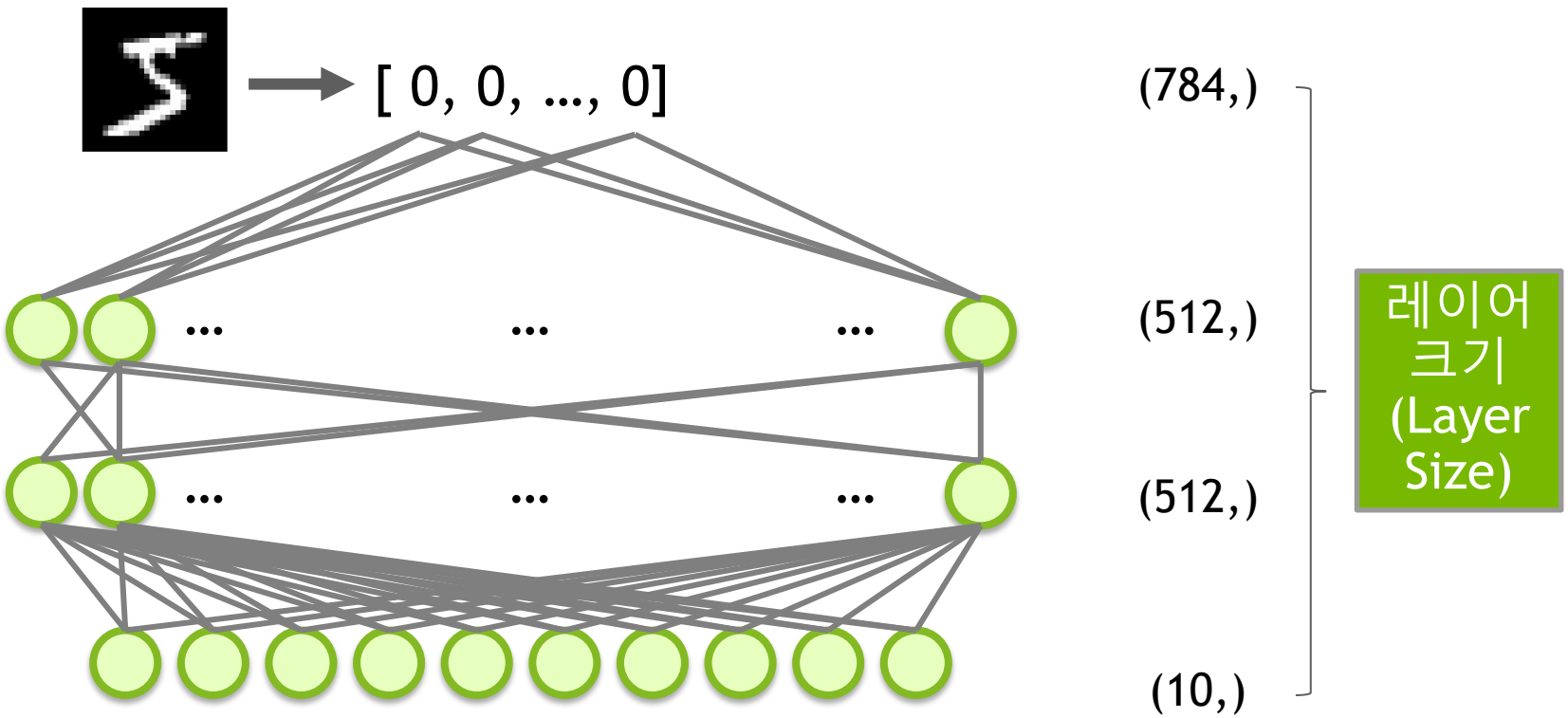


TRAIN DATA, VALIDATION DATA, TEST DATA

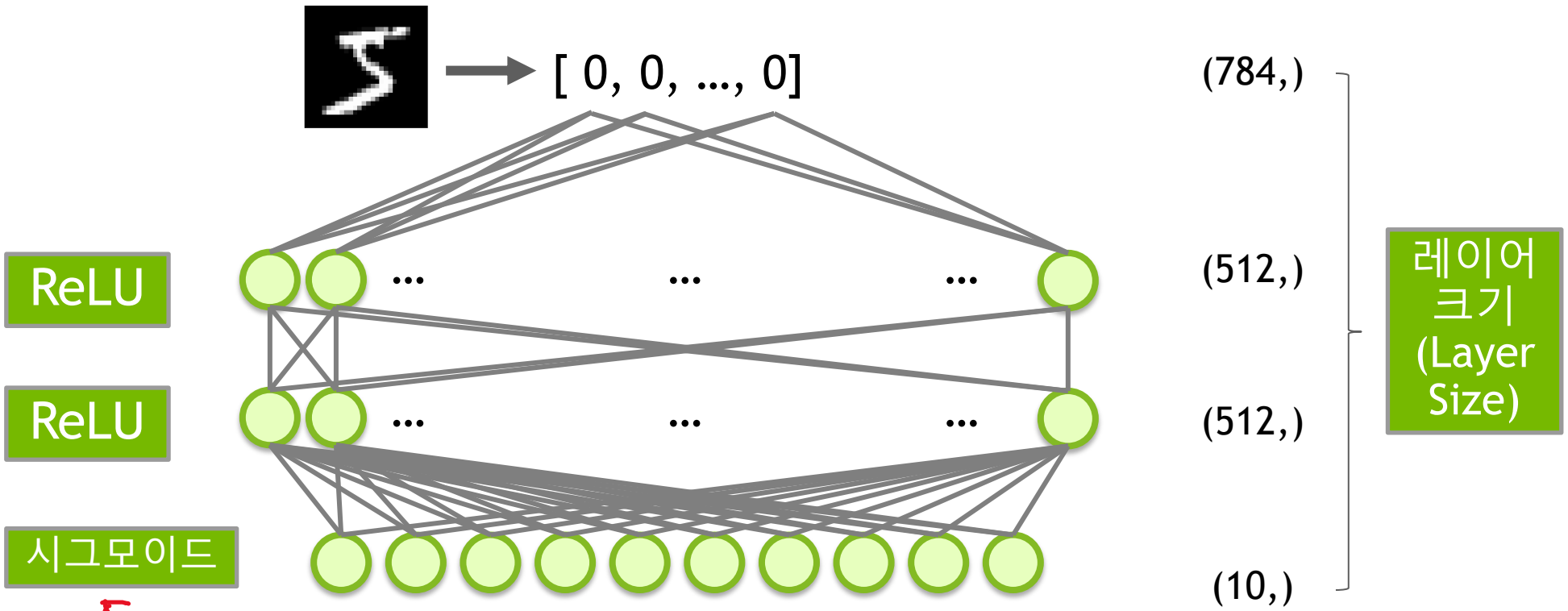


회귀에서 분류로
FROM REGRESSION TO CLASSIFICATION

MNIST 모델



MNIST 모델



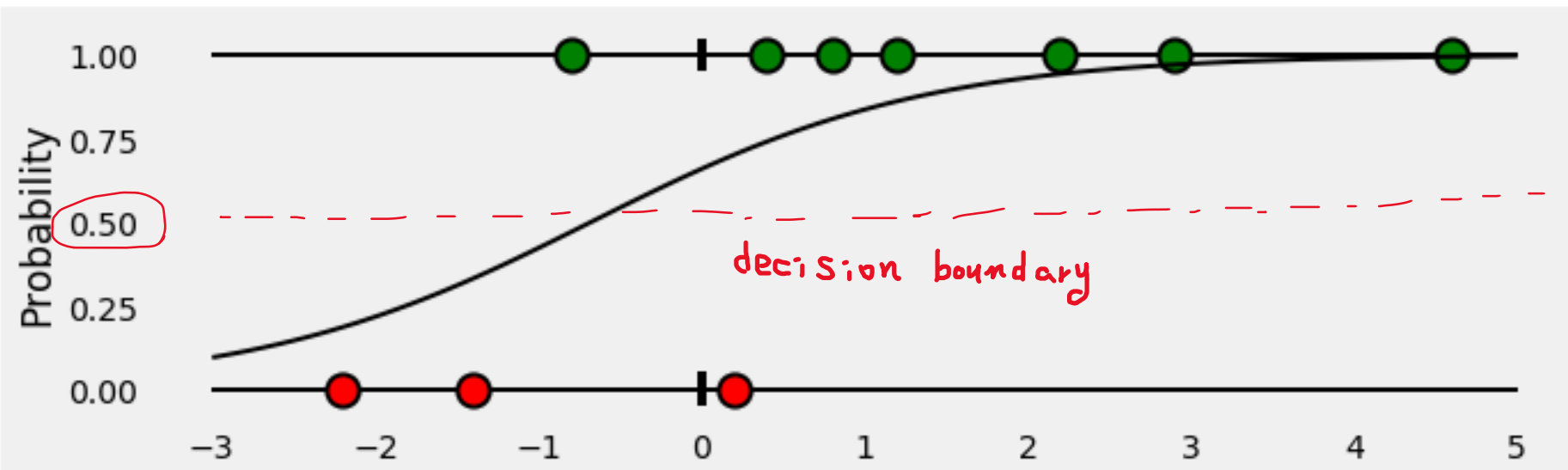
확률값(0~1) : 10개의 이진분류기 필요 → Softmax분류기

SIGMOID(LOGISTIC) 활성화 함수

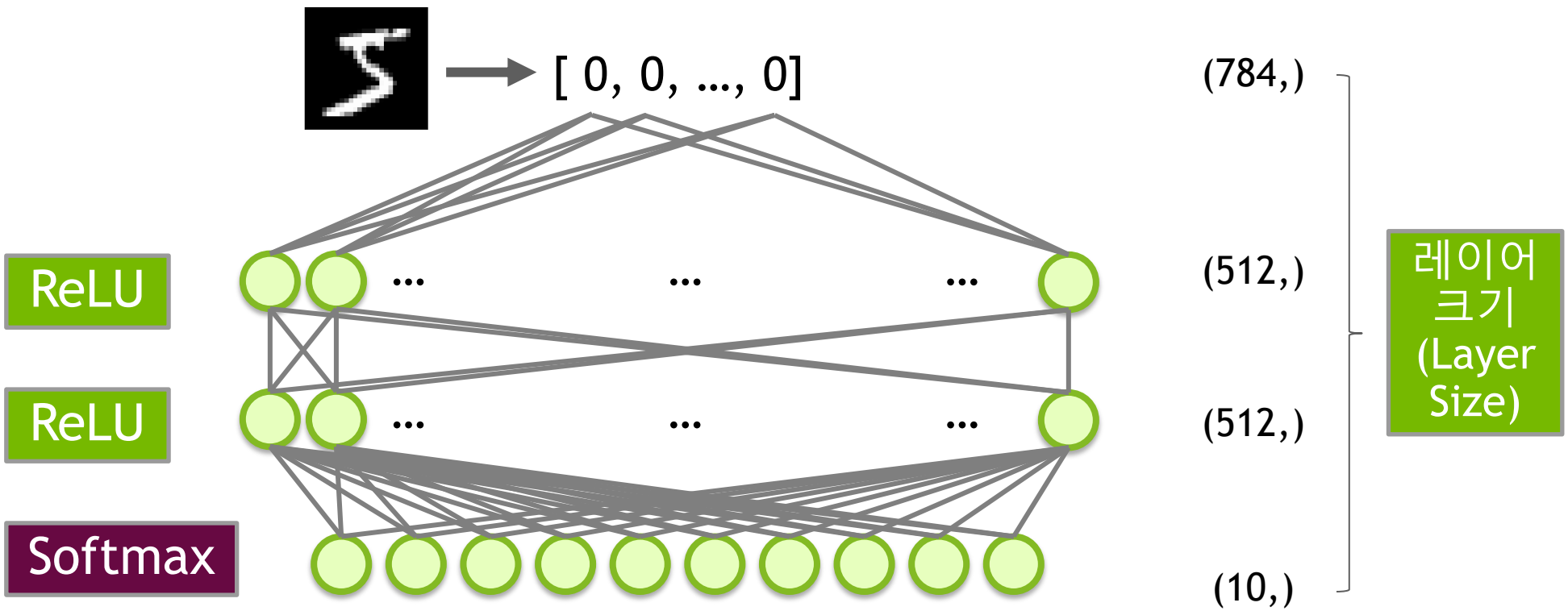
$$f(x) = \frac{1}{1 + e^{-(x)}}$$

wx+b

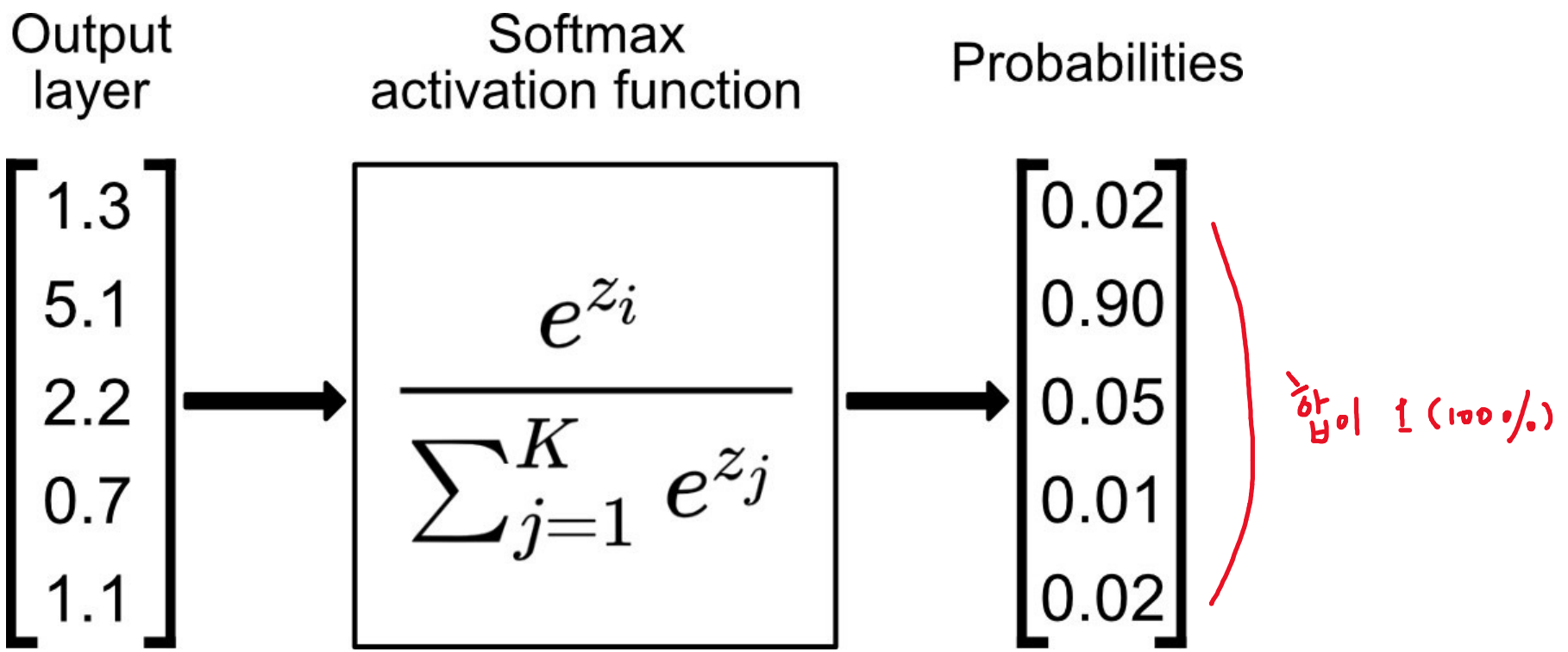
- 이진 분류
  - 시험 합격 여부 : Pass or Fail
  - 스팸 필터 : Spam or Ham
  - SNS 피드 노출 : Show or Hide



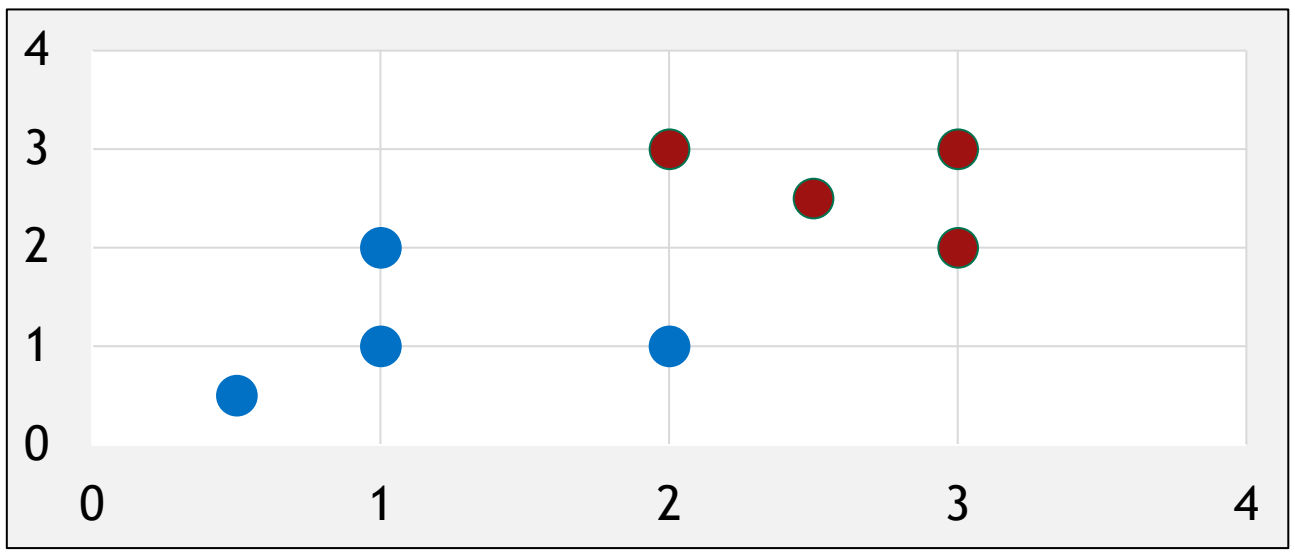
MNIST 모델



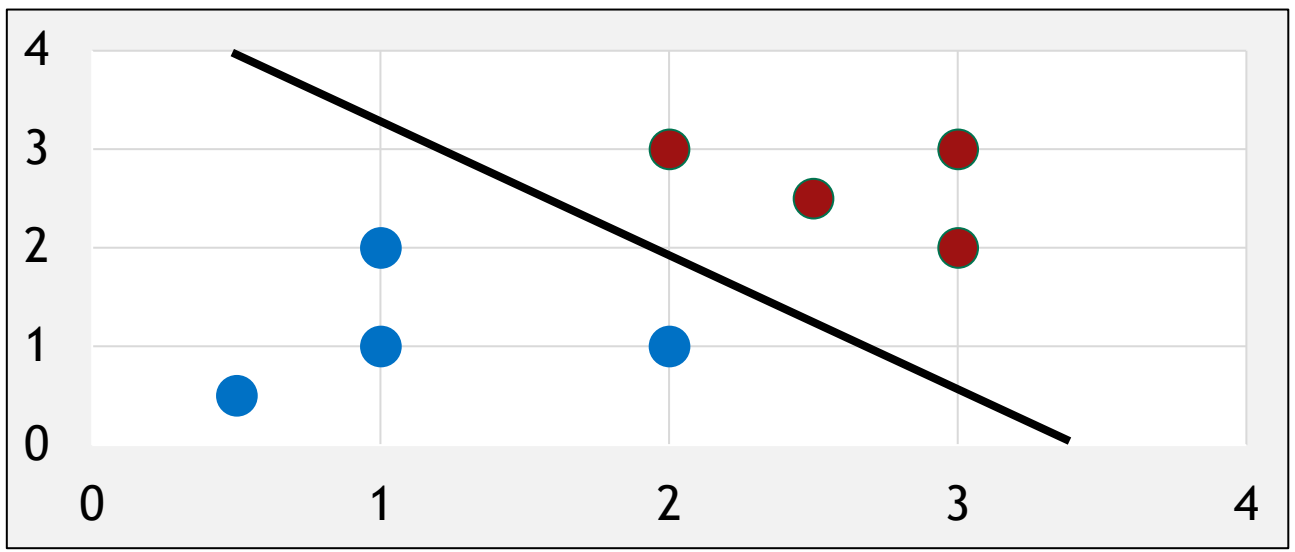
SOFTMAX 활성화 함수



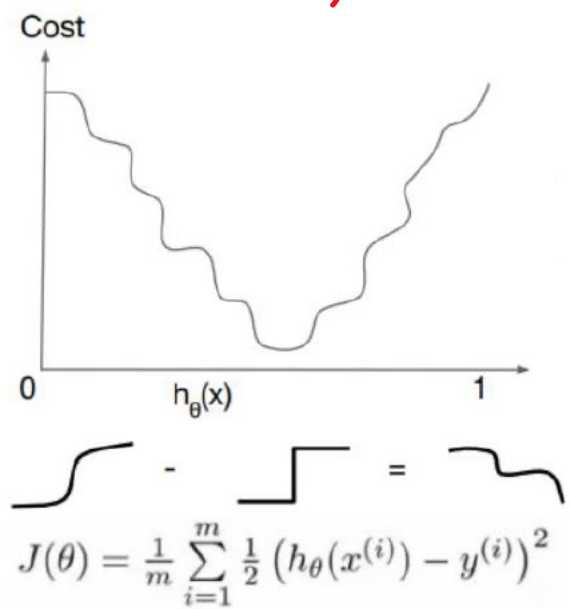
RMSE로 확률 예측?



RMSE로 확률 예측?

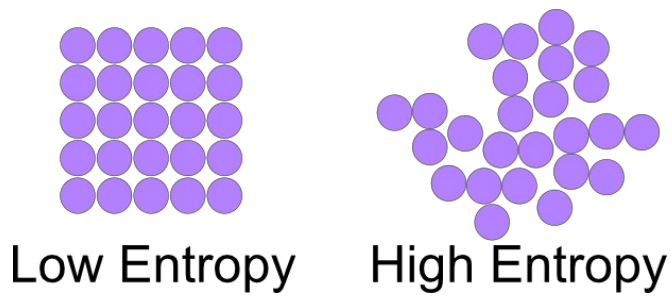


non convex function



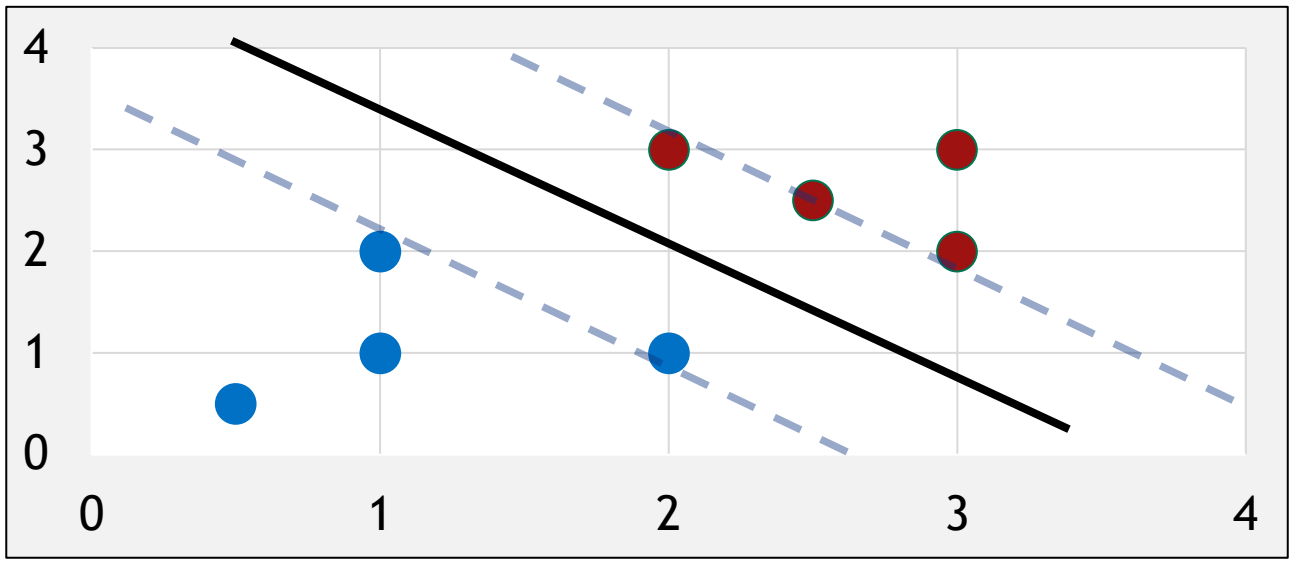
크로스 엔트로피 (CROSS ENTROPY)

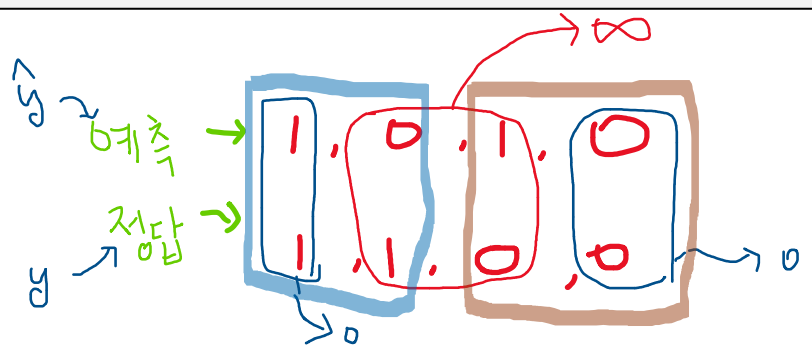
- **엔트로피(ENTROPY)**
  - 무질서 측도
  - 값이 클수록 데이터가 혼재, 분류가 잘 안된 상태
  - 값이 작을 수록 데이터가 잘 분류된 상태

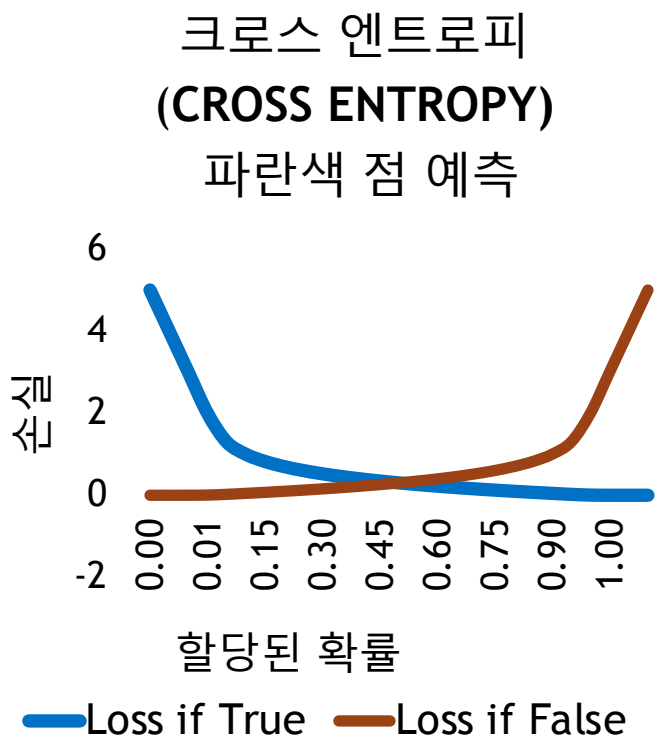


- **크로스 엔트로피(CROSS ENTROPY)**
  - 모델에서 예측한 확률 값이 실제 값과 비교했을 때 틀릴 수 있는 정보량
    - 그 값이 적을 수록 모델이 데이터를 더 잘 예측하는 모델임을 의미
  - 딥러닝에서 예측 값과 정답 값의 크로스 엔트로피 값을 줄이기 위해 가중치(w)와 편향(b)을 업데이트하며 학습(오차 역전파)

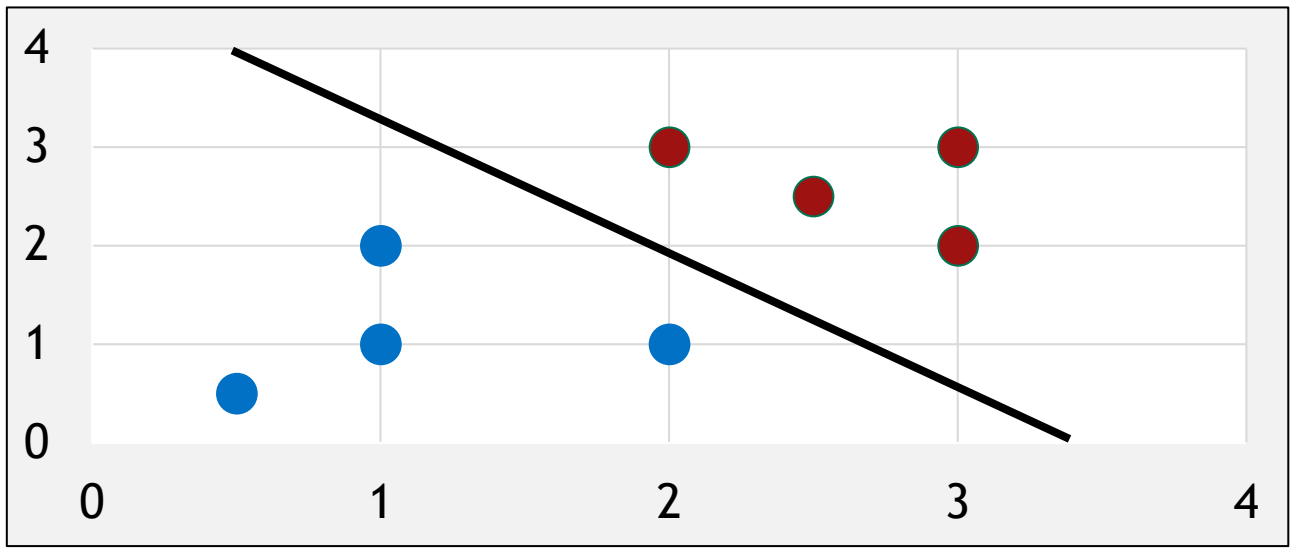
크로스 엔트로피 (CROSS ENTROPY)







크로스 엔트로피 (CROSS ENTROPY)

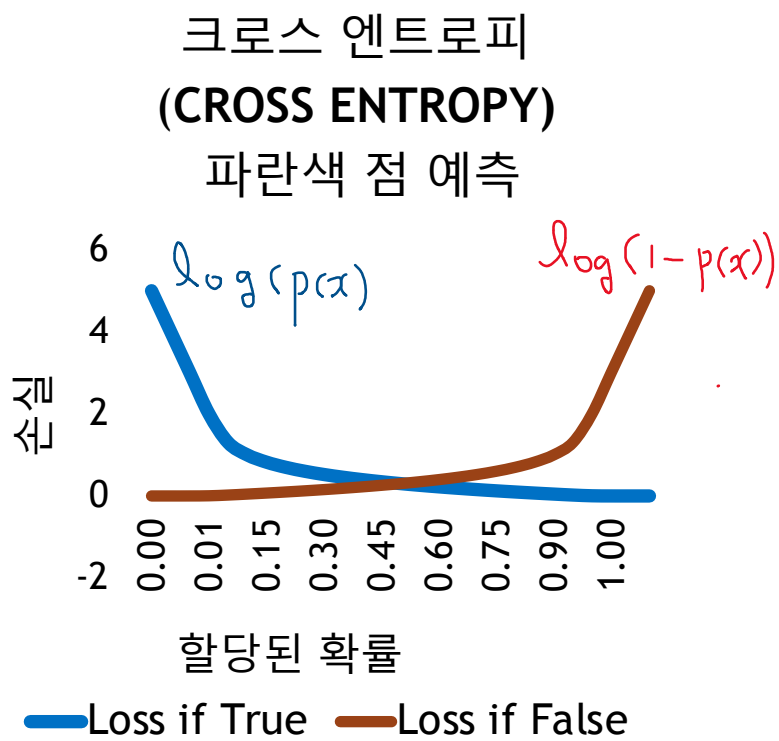


$$Loss = -(t(x) \cdot \log(p(x)) + (1 - t(x)) \cdot \log(1 - p(x)))$$

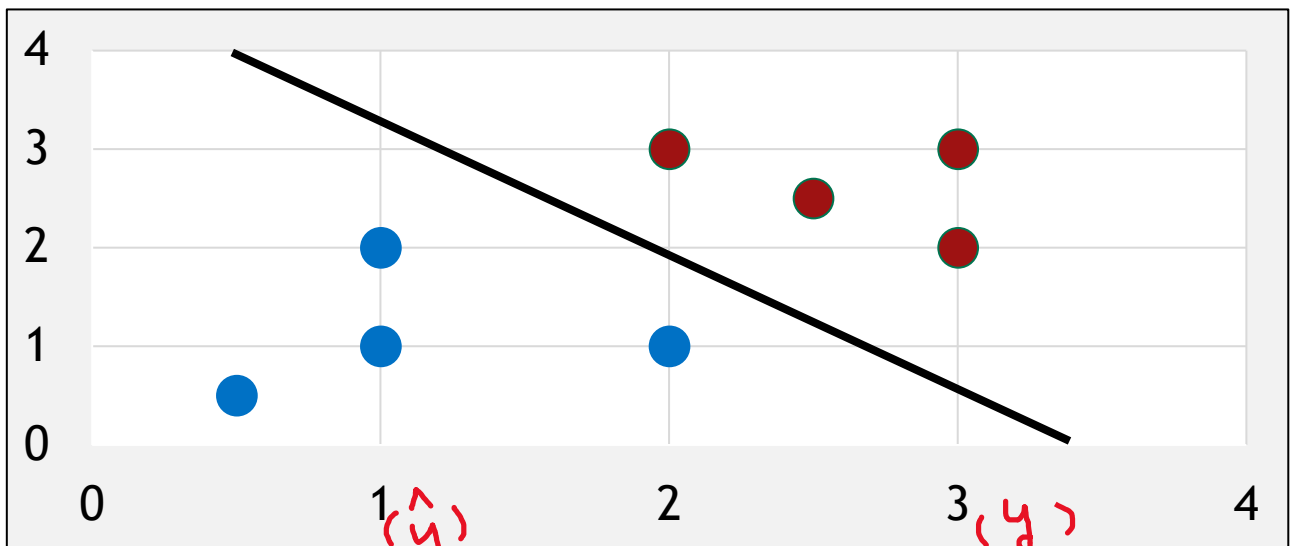
$t(x) = target\ (1\ if\ True, 0\ if\ False)$

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

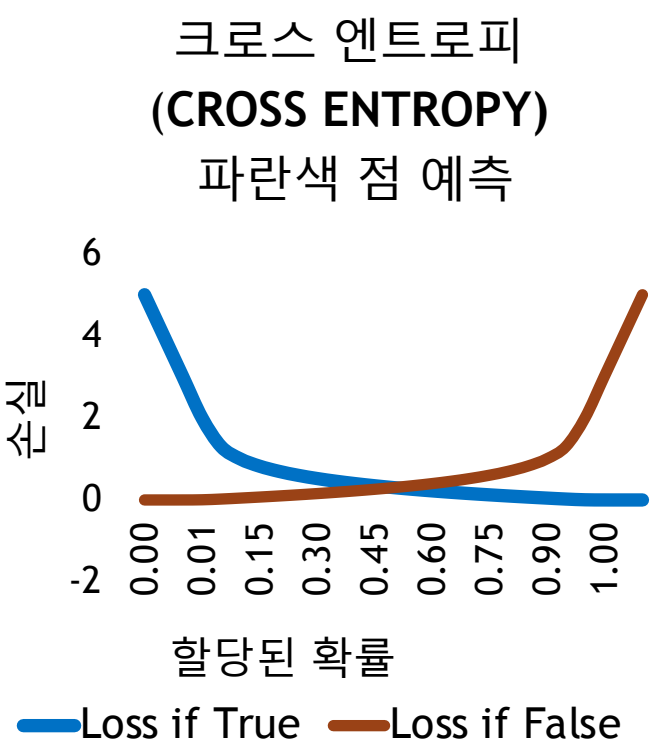
* $\log(1) = 0$
  $\log(0) = \infty$



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
```



크로스 엔트로피 (BINARY) CROSS ENTROPY

Linear Regression

$$\text{cost} = (H(x) - y)^2$$

가설 예측 값 $\hat{y}(y_{pred})$ 정답 값

(Binary) Classification

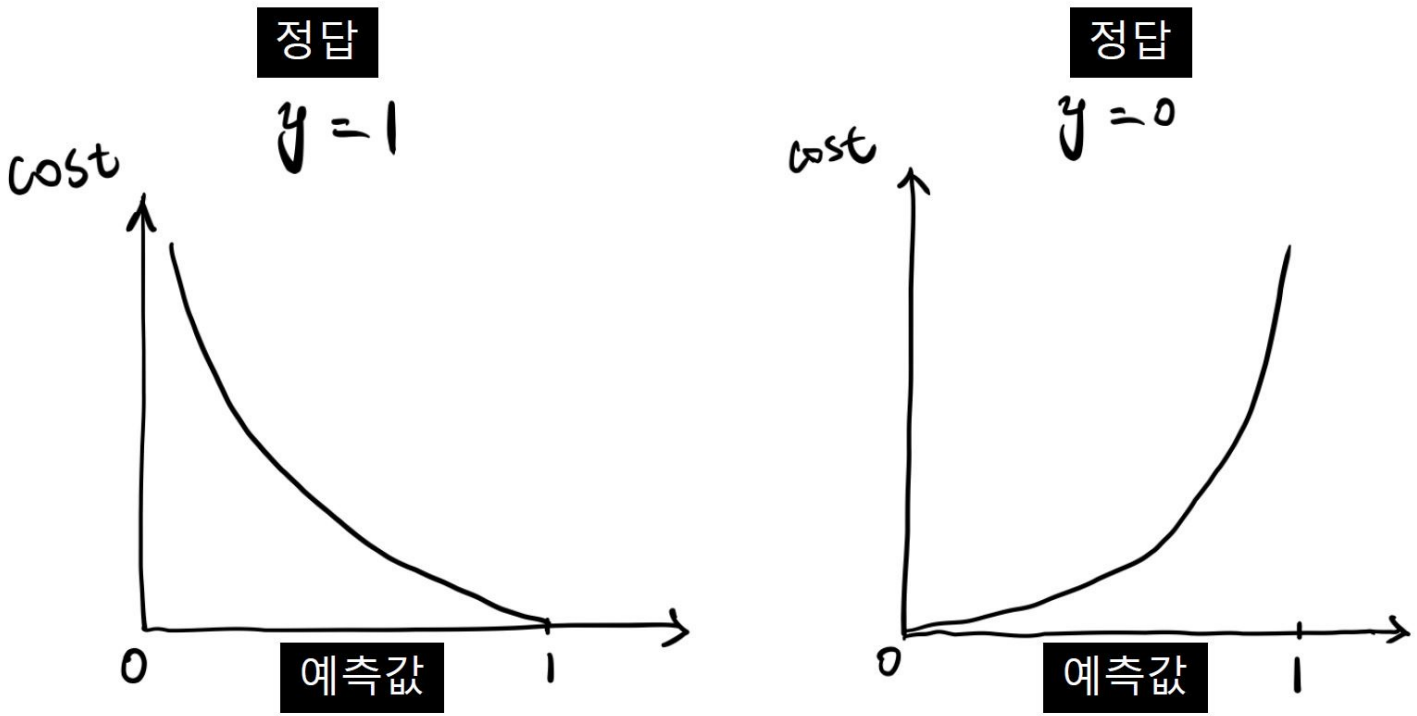
가설 예측 값

$$\text{cost} = -y \log(H(x)) - (1-y) \log(1 - H(x))$$

정답 값

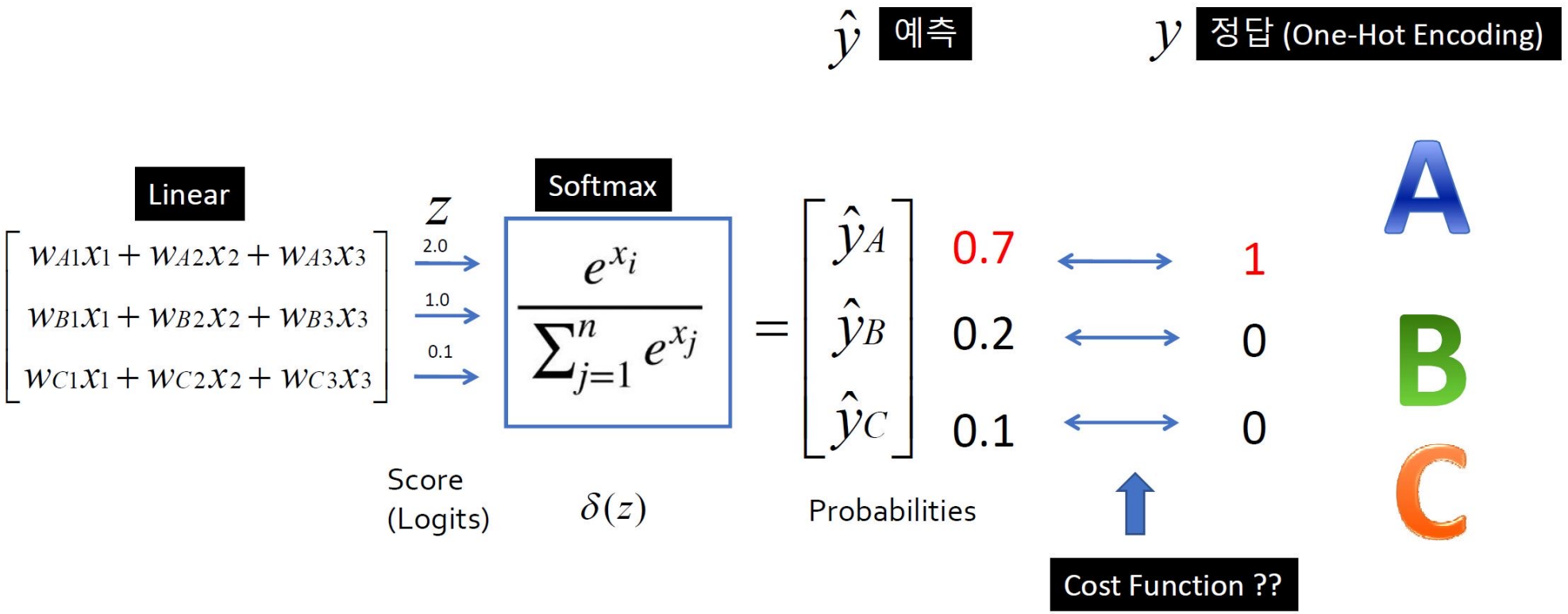
$Y_{actual/\ target/\ label}$	y_{pred}	Loss(cost)	Loss(cost)
1	1	log(1)	0
0	0	log(1)	0
1	0	log(0)	∞
0	1	log(0)	∞

크로스 엔트로피 (BINARY) CROSS ENTROPY

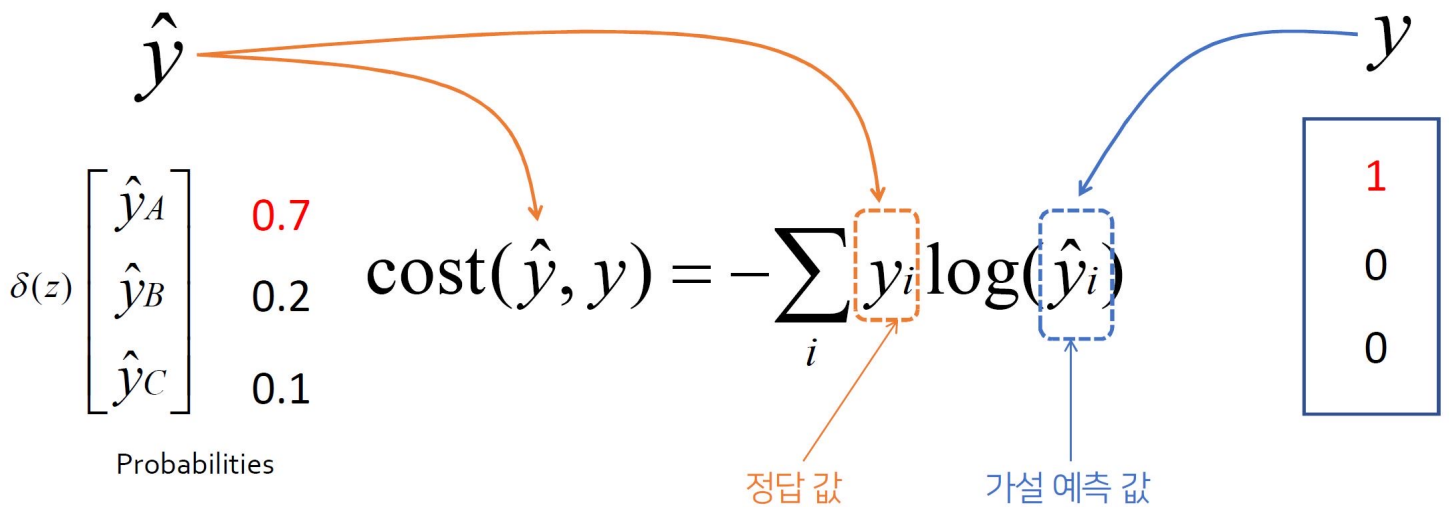


$$\text{cost}(H(x), y) = \begin{cases} -\log(H(x)) & \text{if } y = 1 \\ -\log(1 - H(x)) & \text{if } y = 0 \end{cases}$$

크로스 엔트로피 (CROSS ENTROPY)



크로스 엔트로피 (CROSS ENTROPY)



* 이진분류 일 때 loss(cost) : binary cross entropy

$$-\sum_{i=1}^{2} y_i \log(\hat{y}_i) = -(y_1 \log(\hat{y}_1) + (y_2 \log(\hat{y}_2))$$

$$= -(y \log(\hat{y}) + (1-y) \log(1-\hat{y}))$$

https://youtu.be/jMU9G5WEtBc

정리

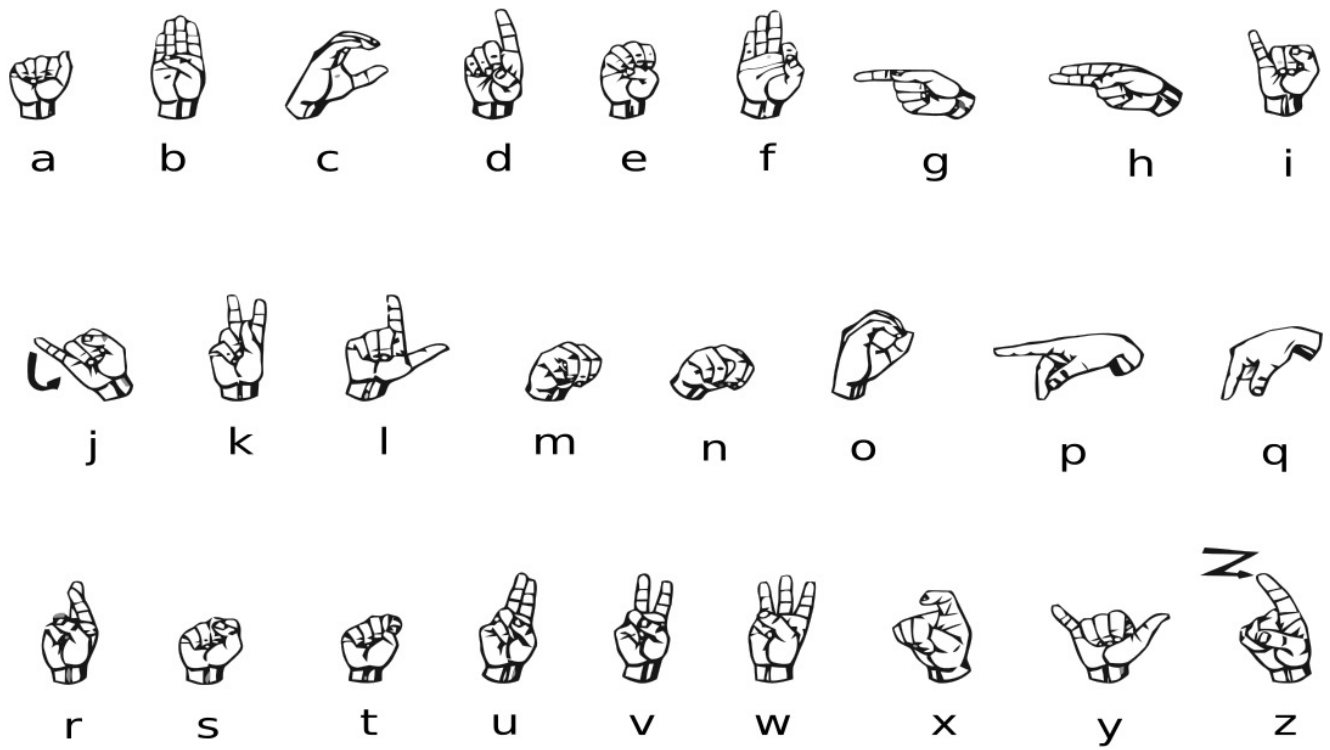
Tensorflow 2.x

	활성화 함수 activation	손실 함수 loss	최적화 방법 optimizer	레이블 형식
선형 회귀	linear	MSE, RMSE	Gradient Descent	-
이진 분류	sigmoid	binary_crossentropy	Gradient Descent	-
다중 분류	softmax	categorical_crossentorpy	Gradient Descent	범주형 one-hot-encoding
다중 분류	softmax	sparse_categorical_crossentropy	Gradient Descent	정수형 label

종합 정리

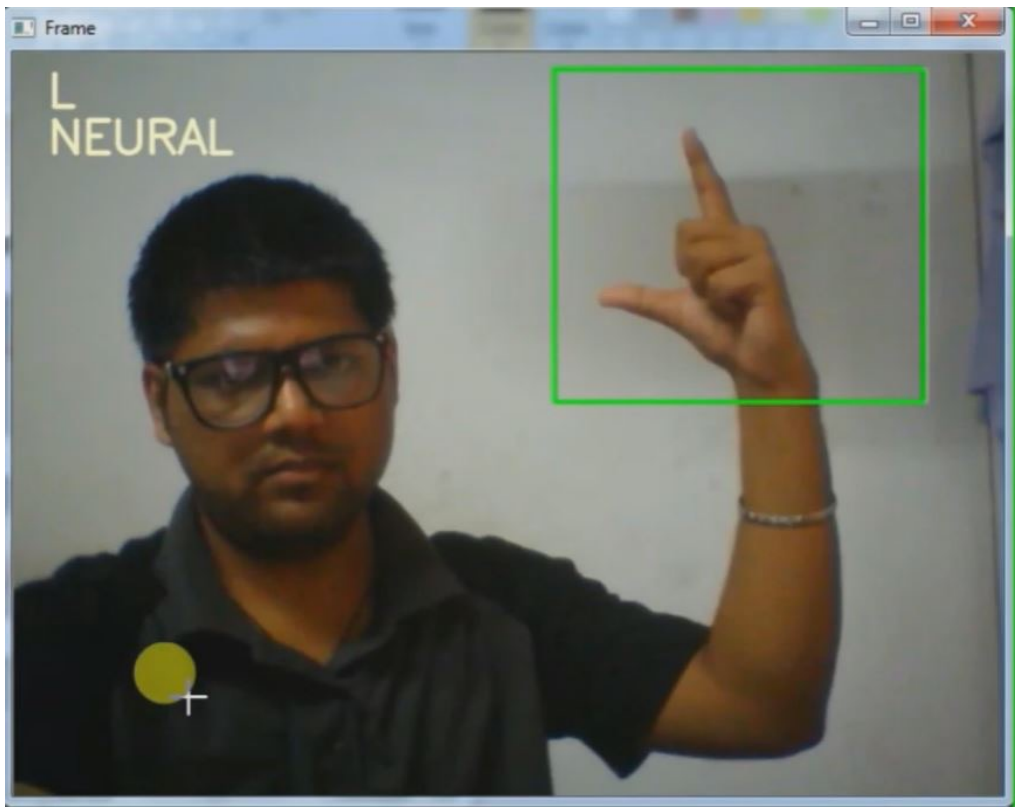
연습 문제

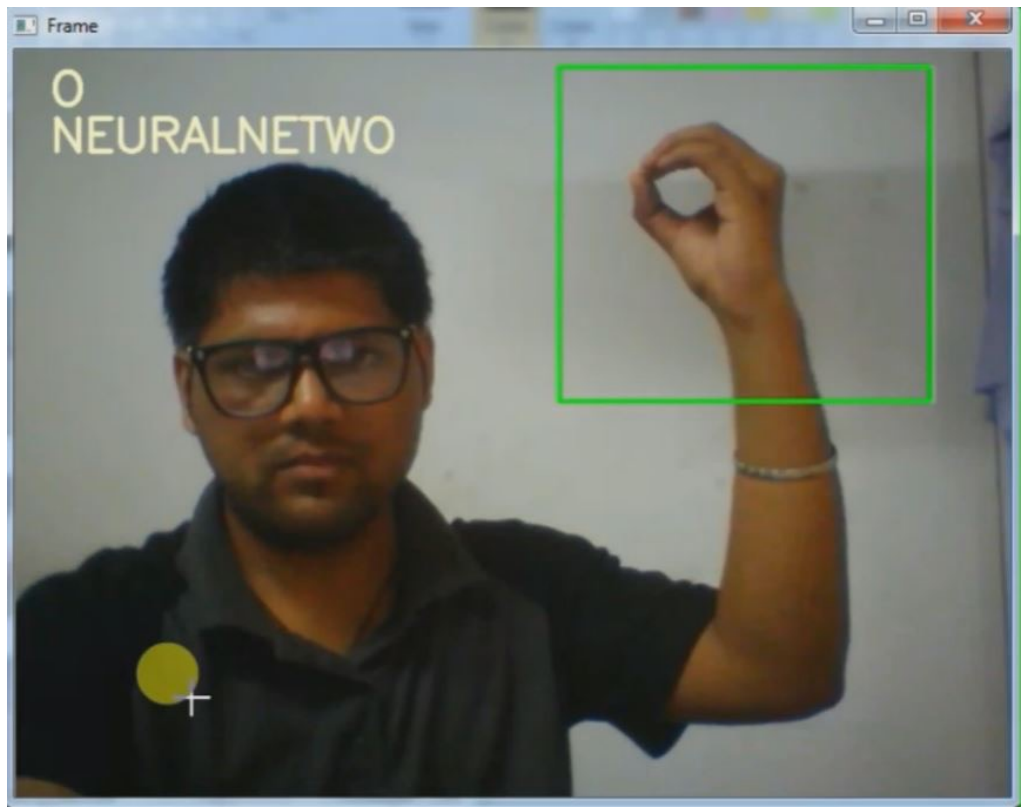
미국 수화 알파벳



연습 문제

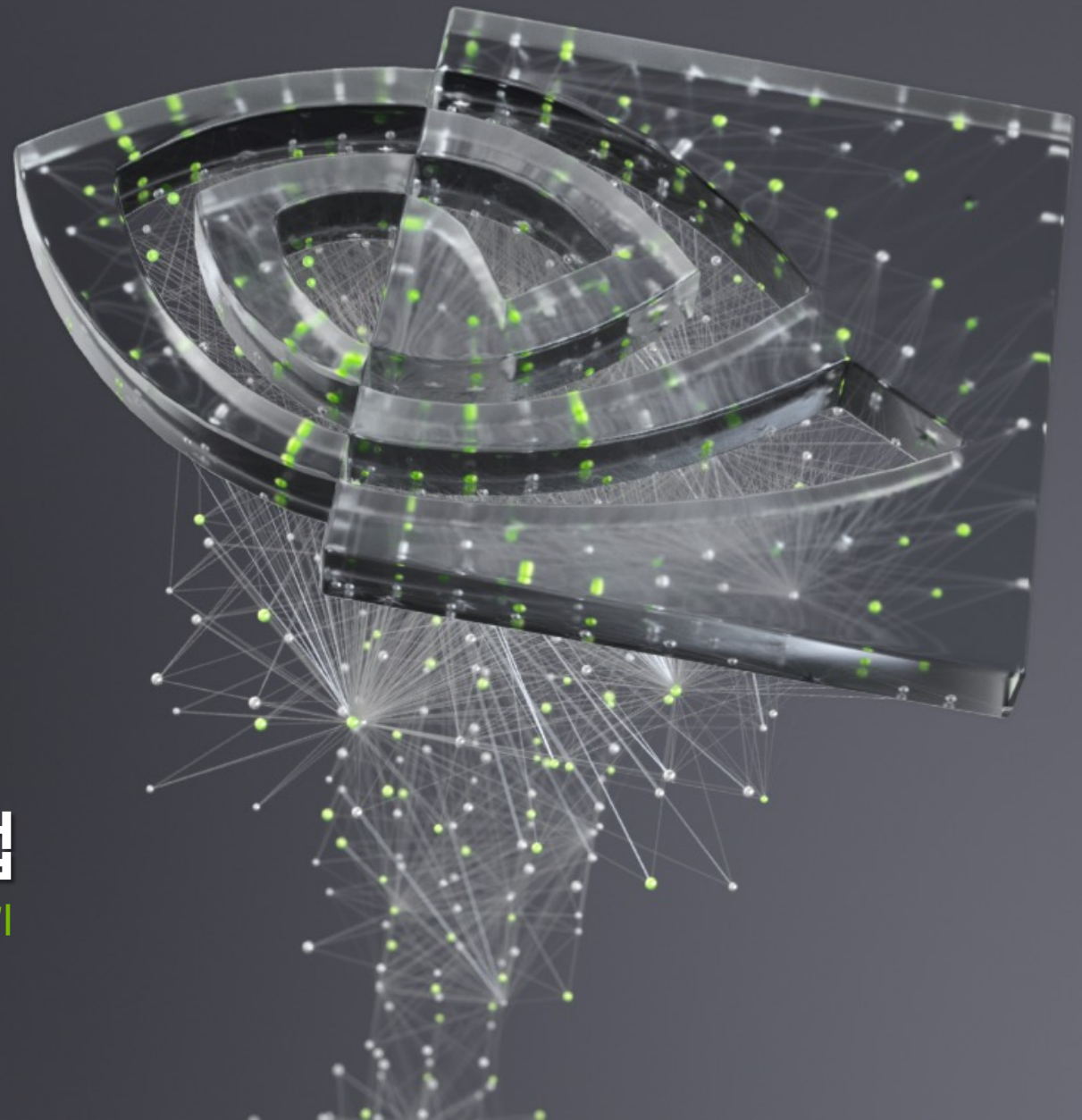
미국 수화 알파벳





https://youtu.be/NBzqY9tJd7M


시작하겠습니다!



NVIDIA | DEEP LEARNING INSTITUTE

부록: 그래디언트 하강법

컴퓨터가 미적분에 대한 편법을 사용할 수 있도록 돕기

에러를 통한 학습

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

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

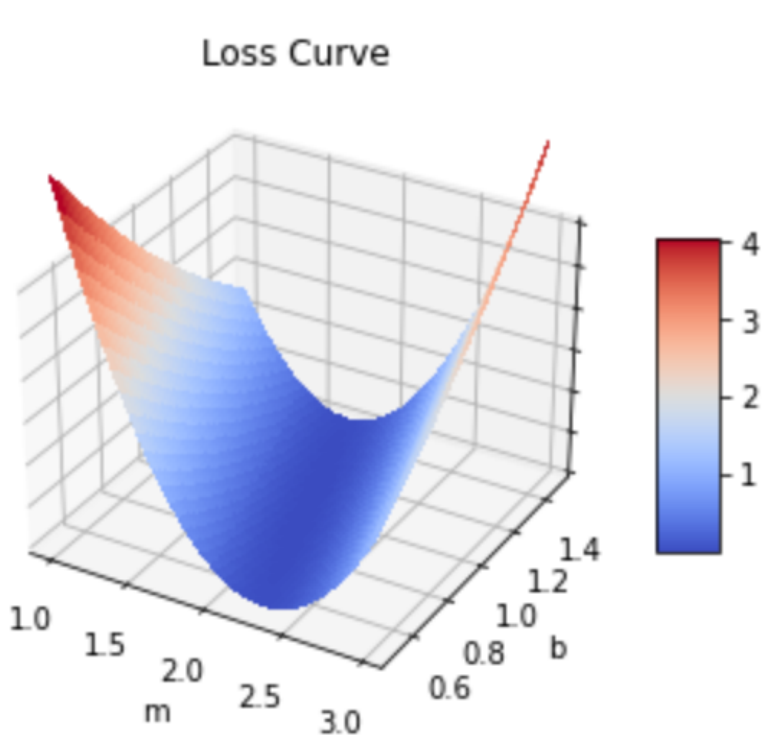
$$\frac{\partial MSE}{\partial m} = 9m + 5b - 23 \qquad \frac{\partial MSE}{\partial b} = 5m + 3b - 13$$

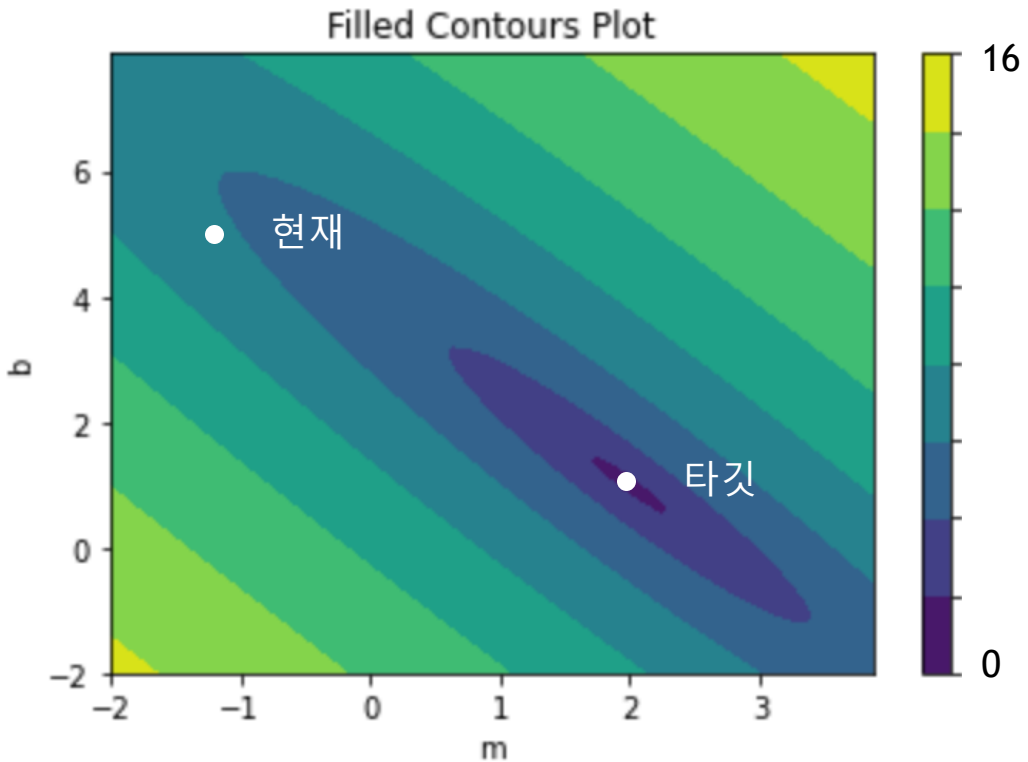
$$\frac{\partial MSE}{\partial m} = -7 \qquad \frac{\partial MSE}{\partial b} = -3$$

$m = -1$

$b = 5$

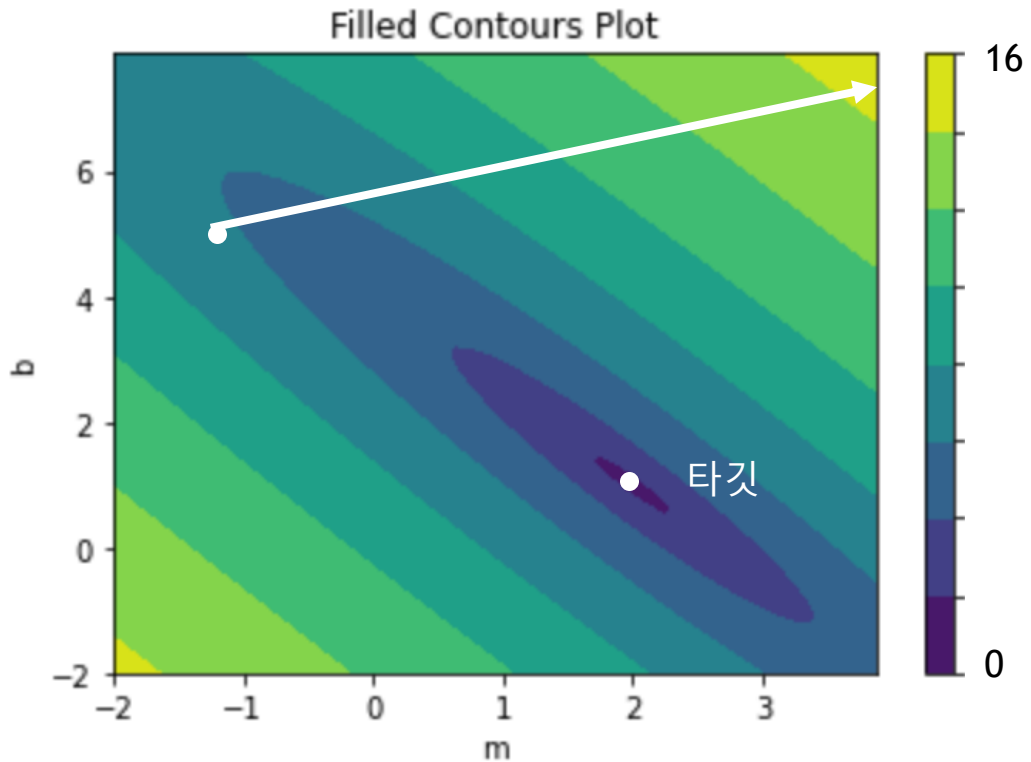
손실 곡선 (THE LOSS CURVE)





손실 곡선 (THE LOSS CURVE)

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

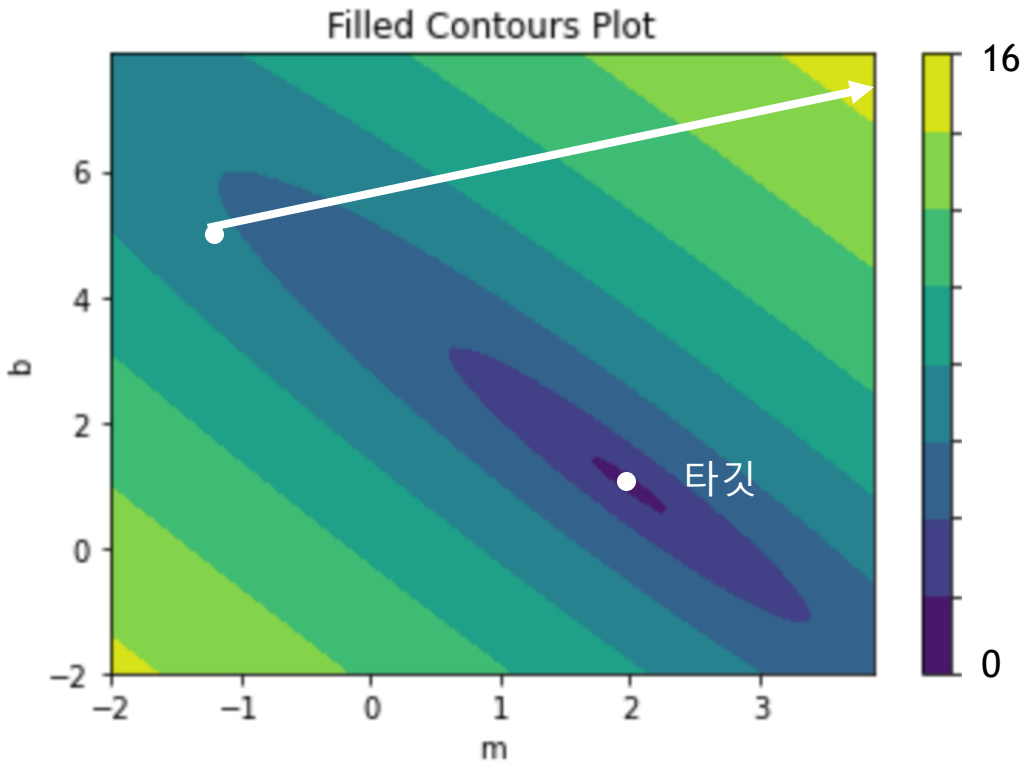


손실 곡선 (THE LOSS CURVE)

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

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

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



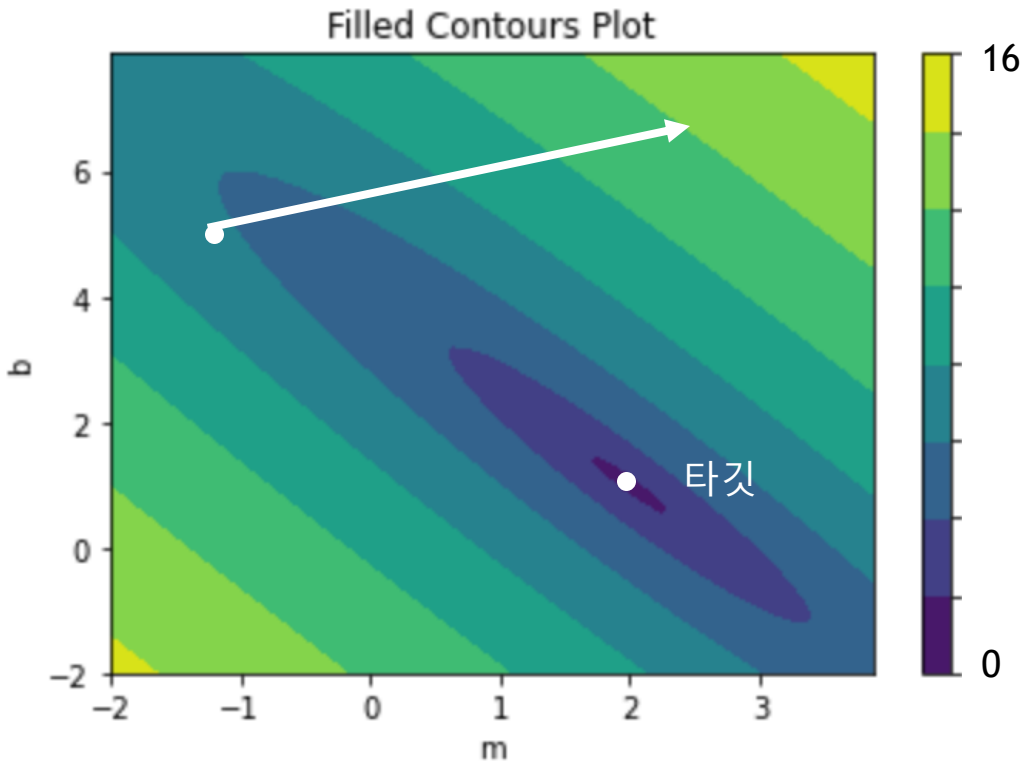
손실 곡선 (THE LOSS CURVE)

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

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

$$\lambda = .5$$

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



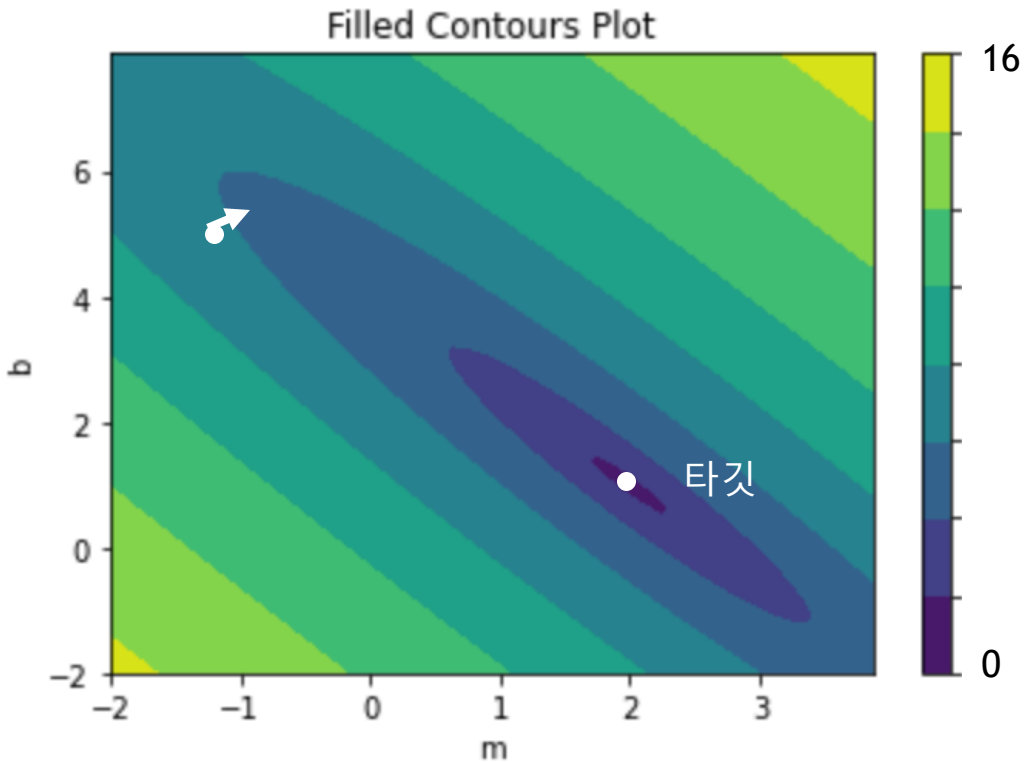
손실 곡선 (THE LOSS CURVE)

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

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

$$\lambda = .005$$

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

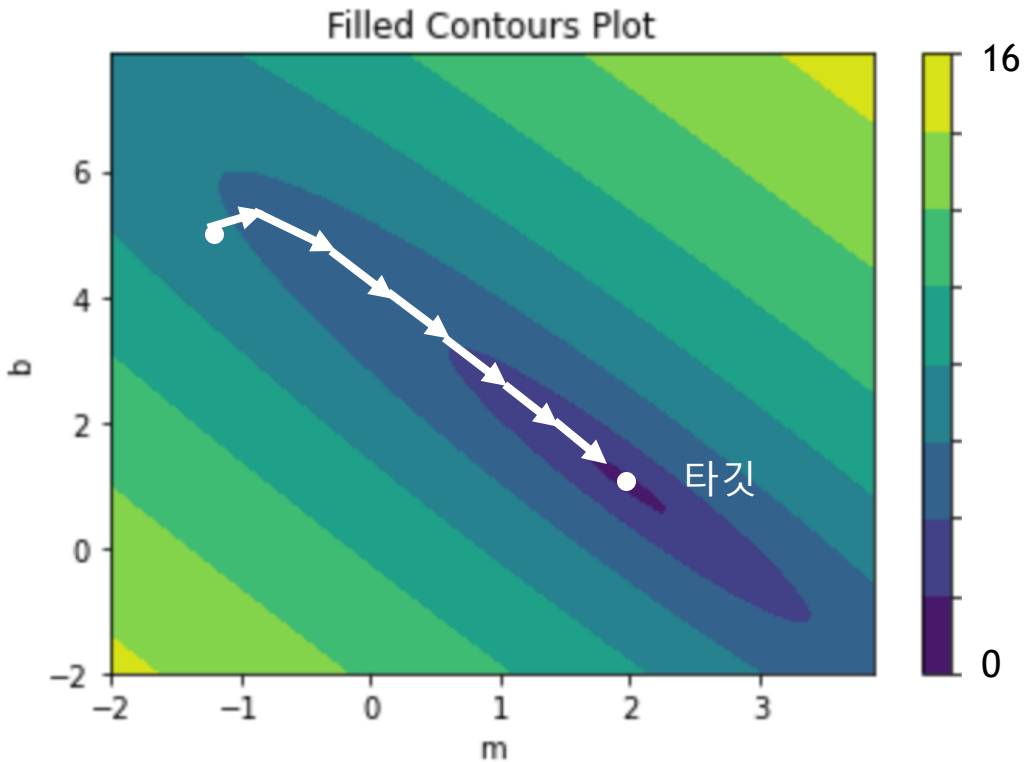


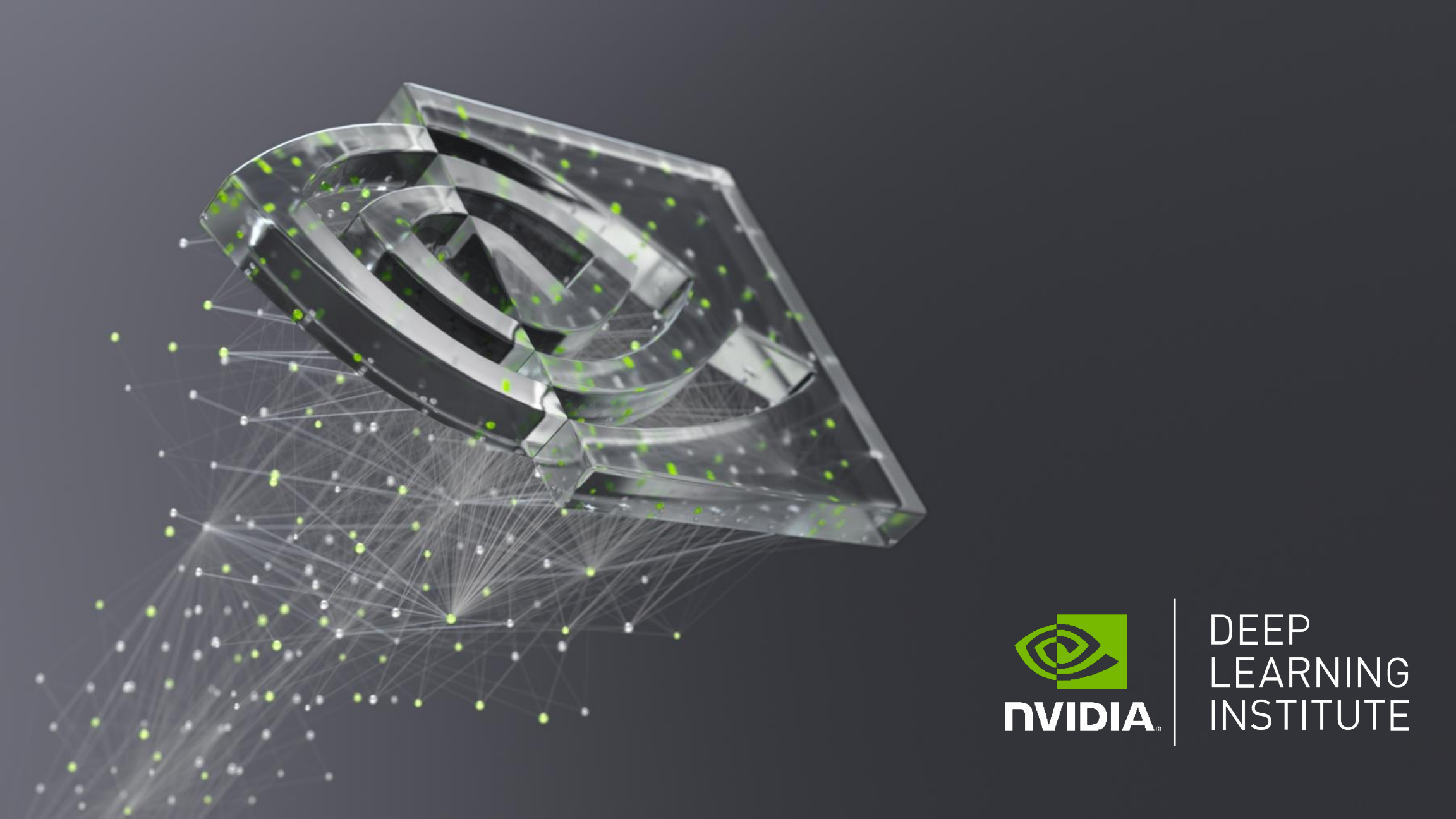
손실 곡선 (THE LOSS CURVE)

$$\lambda = .1$$

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

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




NVIDIA
DEEP
LEARNING
INSTITUTE