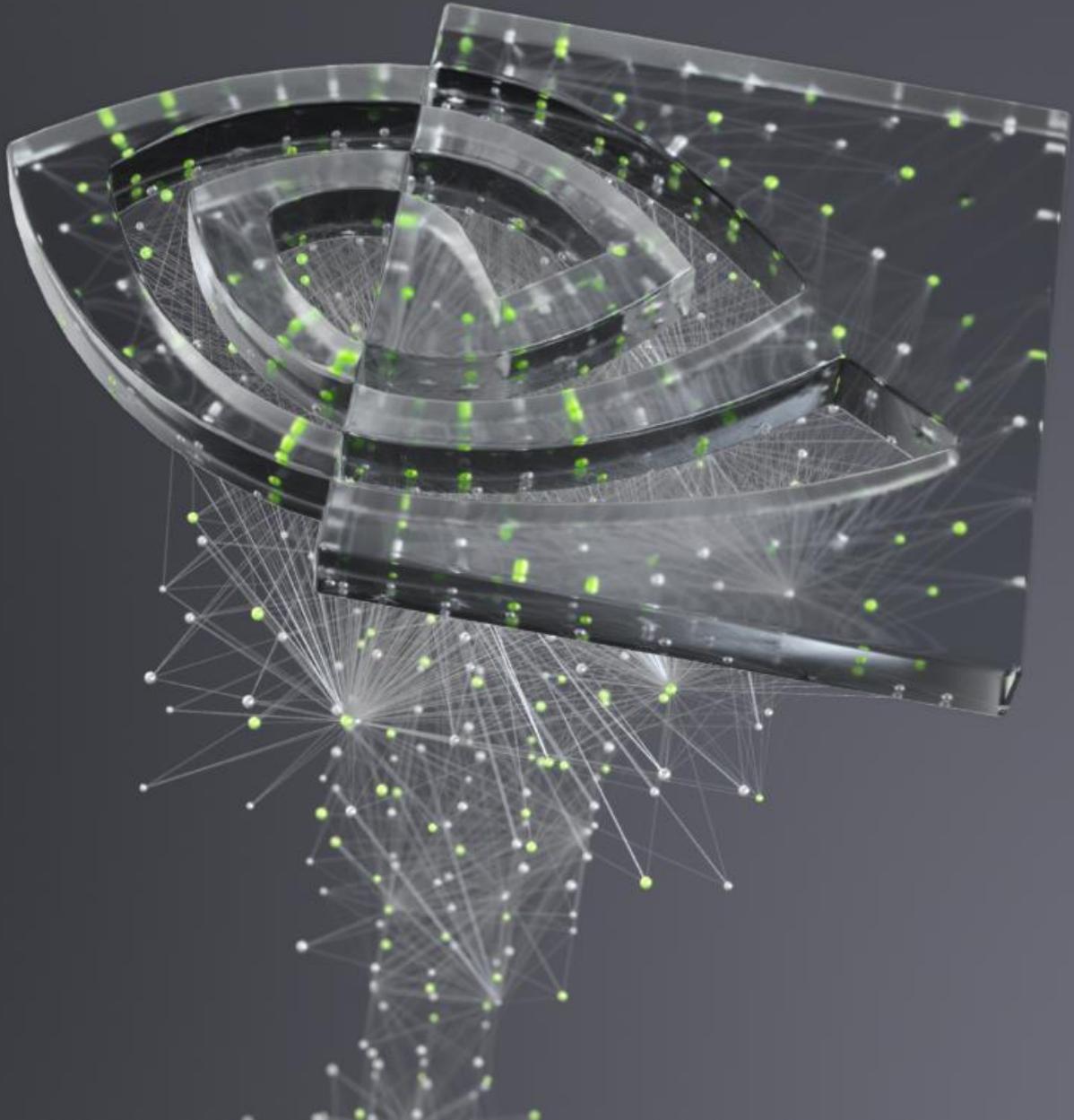




딥러닝의 기초

2부: 뉴럴 네트워크의 트레이닝 방식



목차

1부: 딥러닝 소개

2부: 뉴럴 네트워크의 트레이닝 방식

3부: CNN(Convolutional Neural Network)

4부: 데이터 증강 및 배포

5부: 사전 트레이닝된 모델

6부: 고급 아키텍처

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)

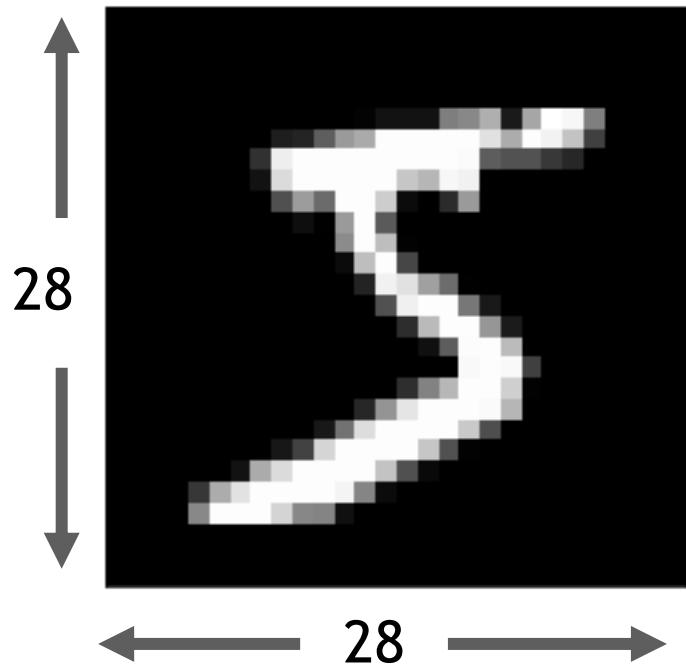
모델 생성

모델 컴파일

데이터에 대해 모델 트레이닝

데이터 준비 (DATA PREPARATION)

배열을 통한 입력



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

$28 \times 28 = 784$

$\div 255$ (정규화)

↓

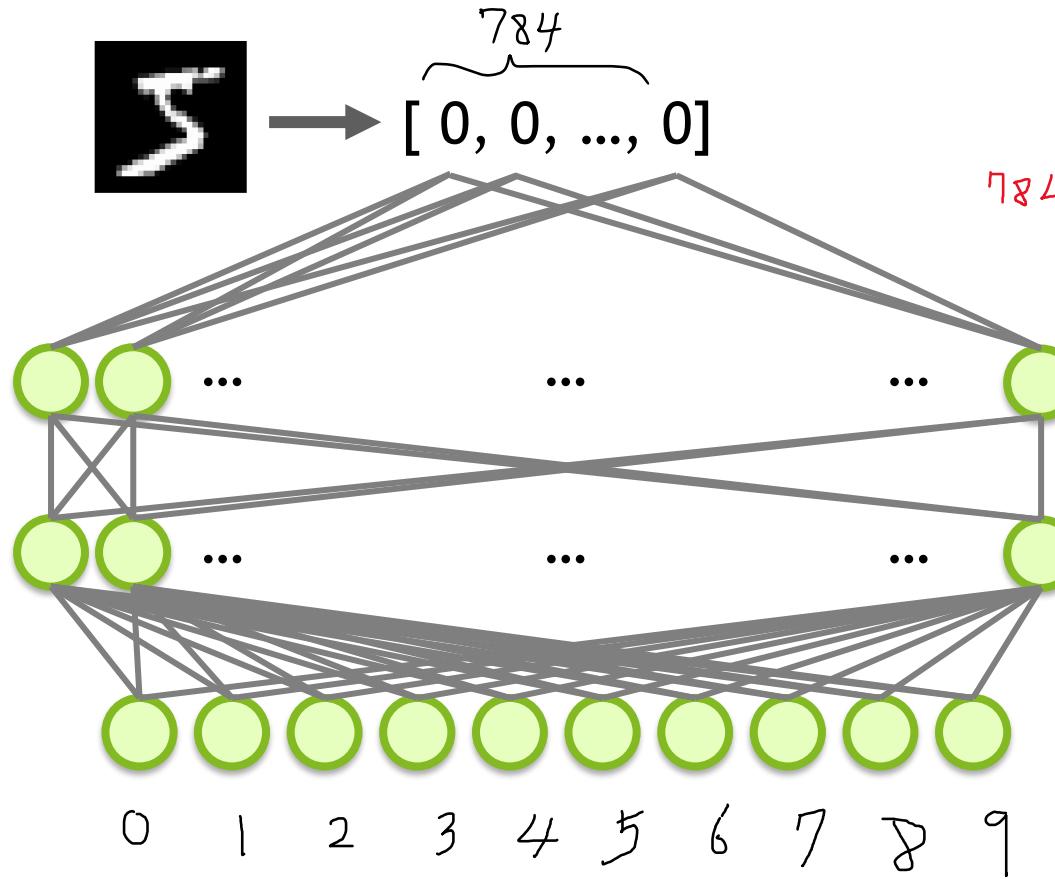
[0 , 0 , 0 , 1 , 0 . 3 ...]

데이터 준비 (DATA PREPARATION)

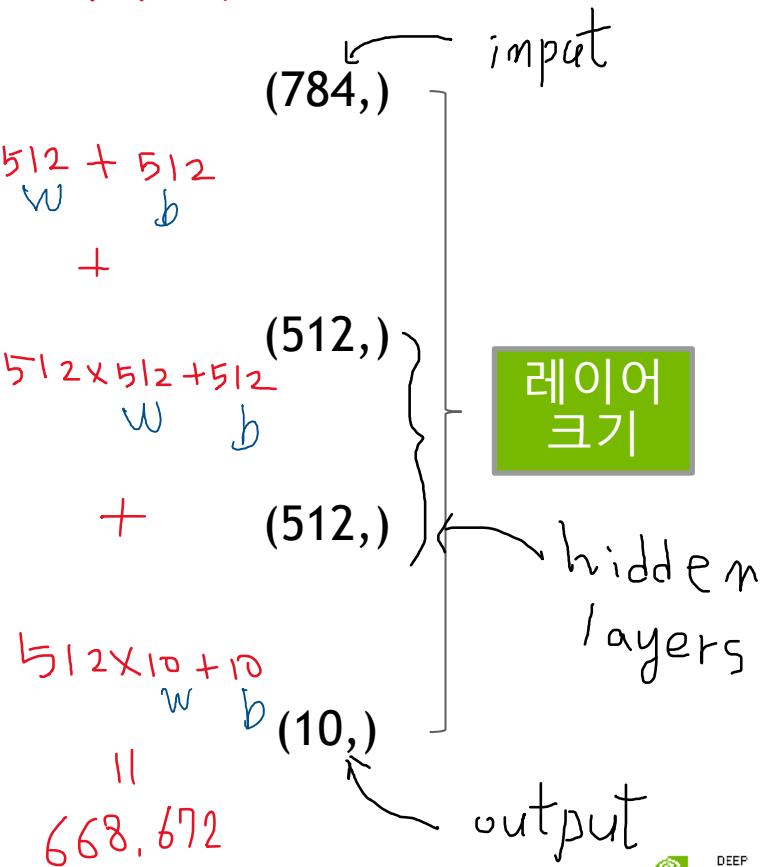
범주형(categorical)으로 타깃팅 : One-Hot-Encoding

| | | |
|---|---|-----------------------|
| 0 | → | [1,0,0,0,0,0,0,0,0,0] |
| 1 | → | [0,1,0,0,0,0,0,0,0,0] |
| 2 | → | [0,0,1,0,0,0,0,0,0,0] |
| 3 | → | [0,0,0,1,0,0,0,0,0,0] |
| ⋮ | ⋮ | ⋮ |

트레이닝되지 않은 모델



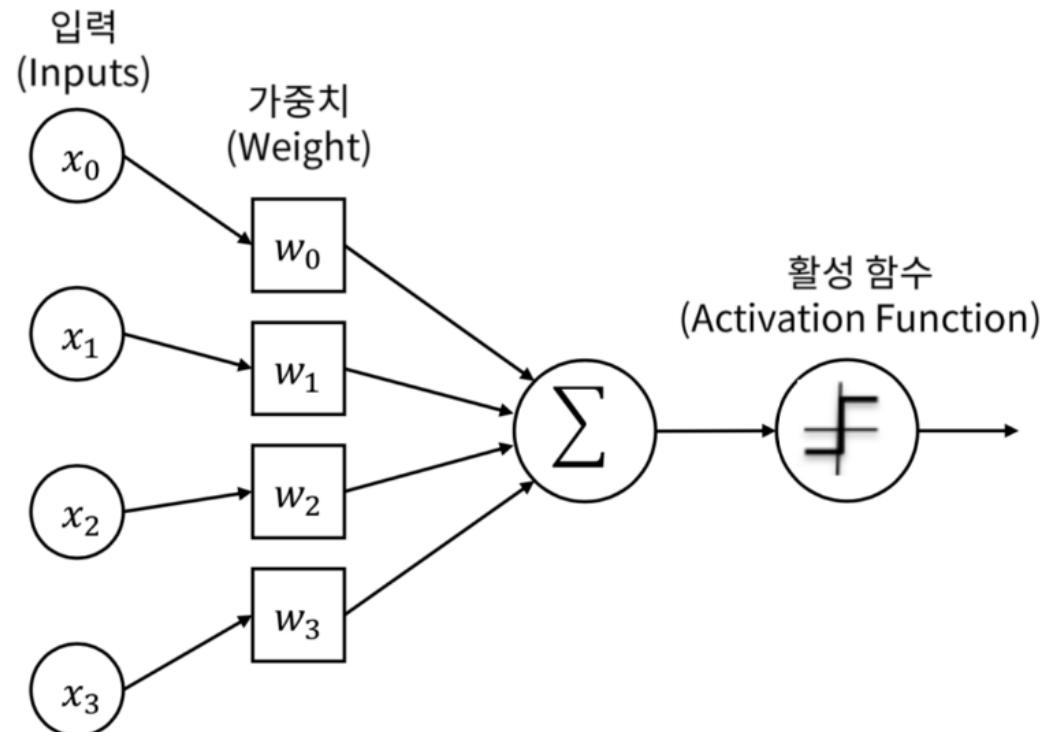
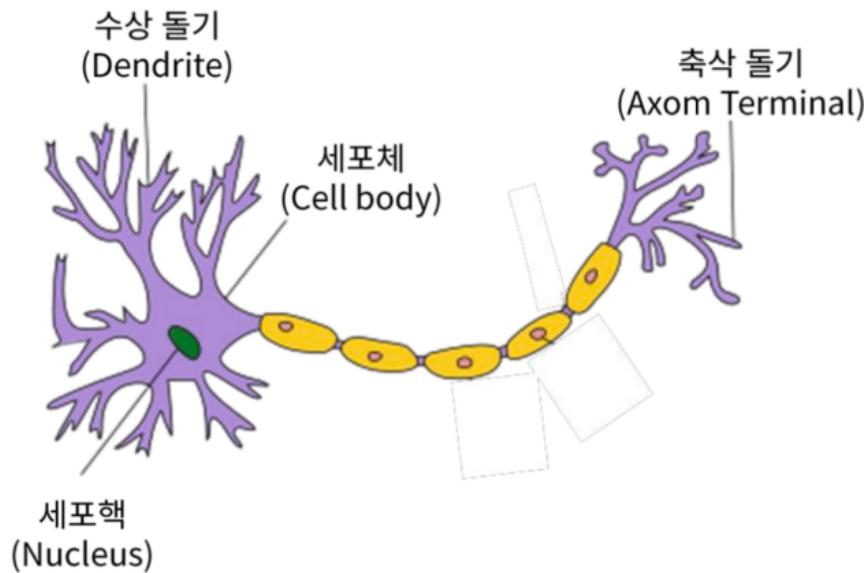
* 파라미터(w, b)의 갯수는?



A complex network graph is displayed against a dark gray background. The graph consists of numerous small, semi-transparent circular nodes scattered across the frame. Some nodes are white, while others are a bright lime green. These nodes are interconnected by a dense web of thin, light gray lines representing edges. The overall effect is one of a large, complex system or dataset.

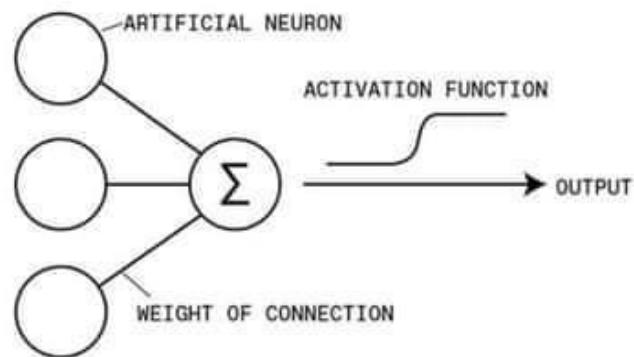
더욱 단순한 모델

신경망(NEURON) 구조



신경망(NEURON) 구조

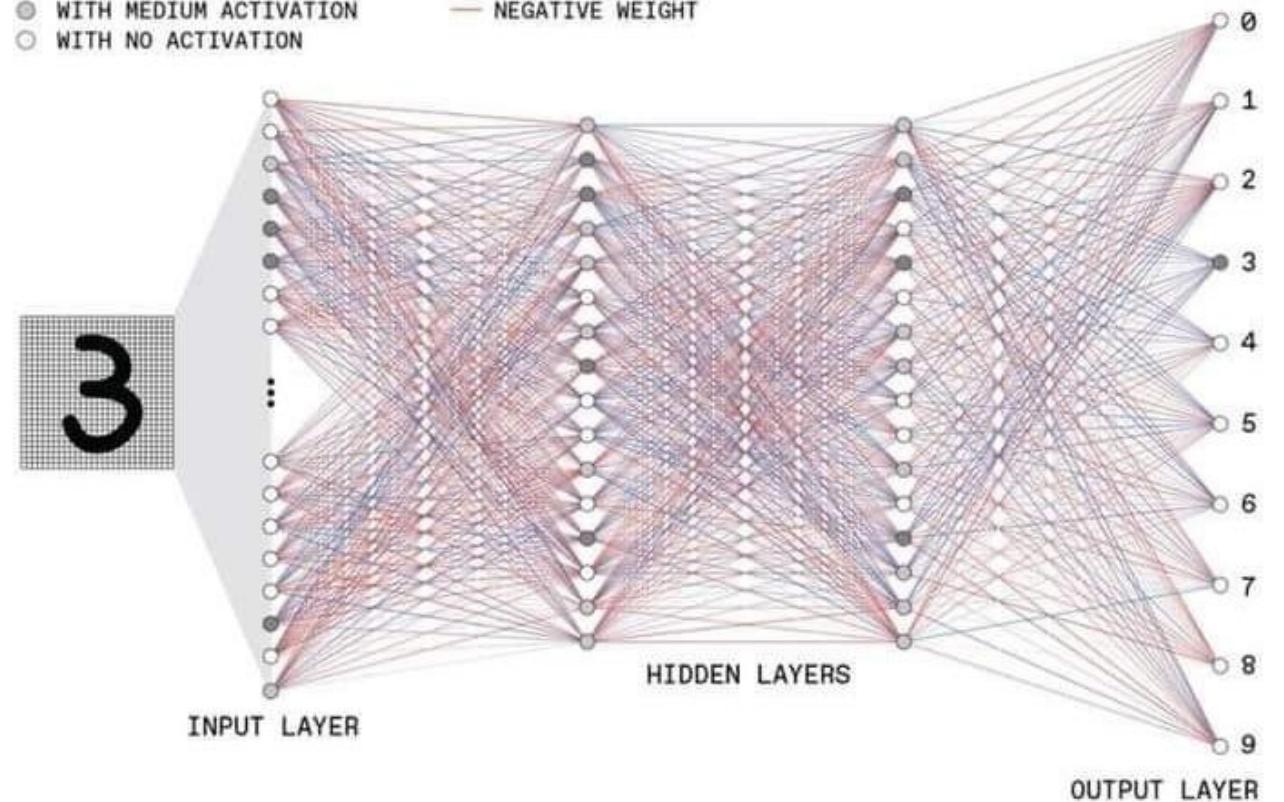
ARCHITECTURE



NEURONS:

- WITH HIGH ACTIVATION
- WITH MEDIUM ACTIVATION
- WITH NO ACTIVATION

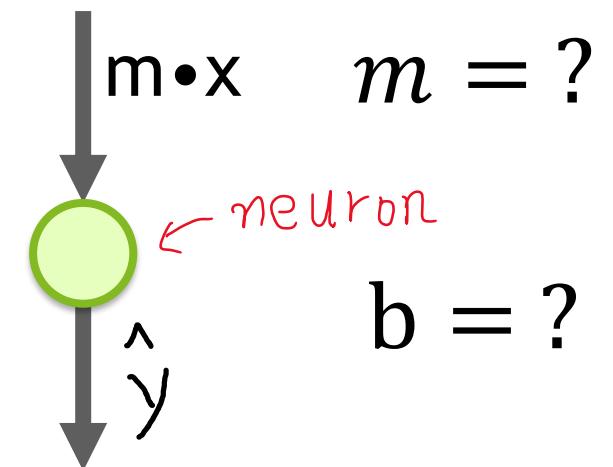
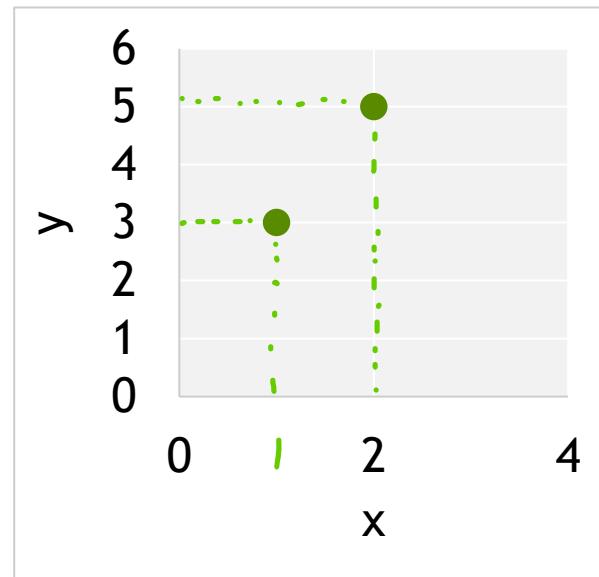
— POSITIVE WEIGHT
— NEGATIVE WEIGHT



더욱 단순한 모델

$$y = mx + b$$

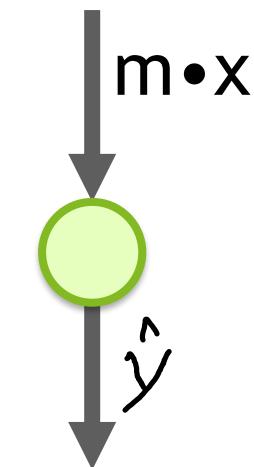
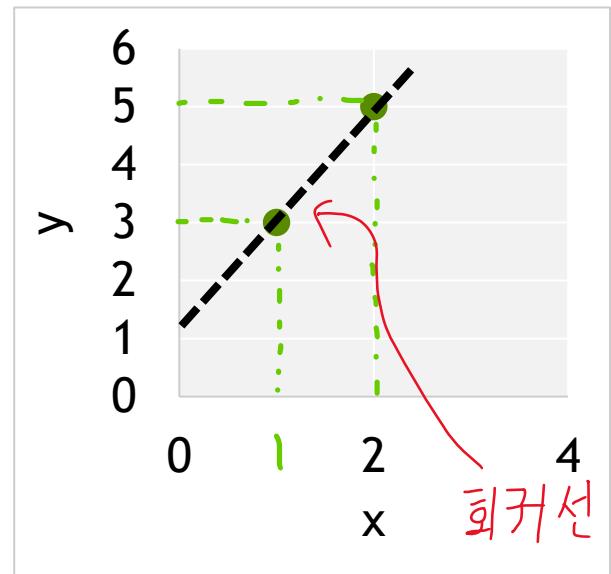
| x | y |
|---|---|
| 1 | 3 |
| 2 | 5 |



더욱 단순한 모델

$$y = mx + b$$

| x | y |
|---|---|
| 1 | 3 |
| 2 | 5 |



$$\begin{aligned} m &= ? \\ b &= ? \end{aligned}$$

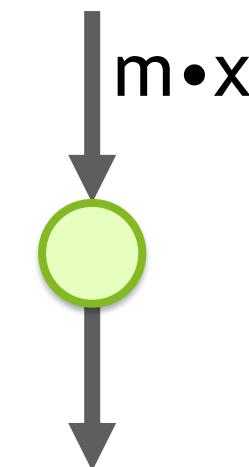
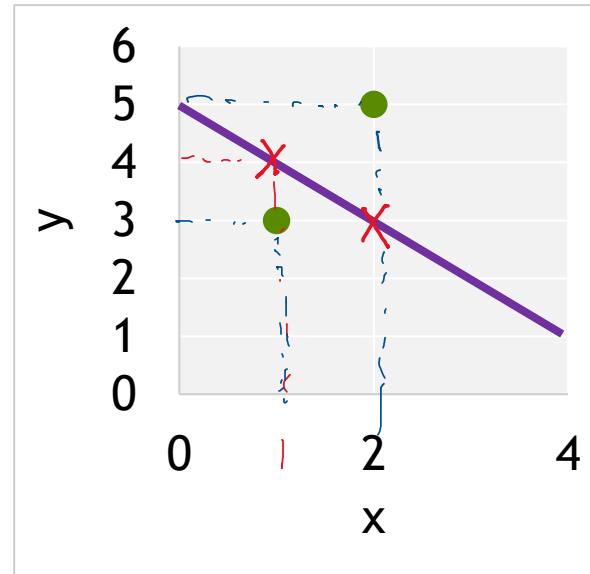
더욱 단순한 모델

$$y = mx + b$$

↑
- 1 ↗ 5

실제값(정답) 예측값(가설)

| x | y | \hat{y} |
|---|---|-----------|
| 1 | 3 | 4 |
| 2 | 5 | 3 |



무작위로 시작

$$m = -1$$

$$b = 5$$

더욱 단순한 모델

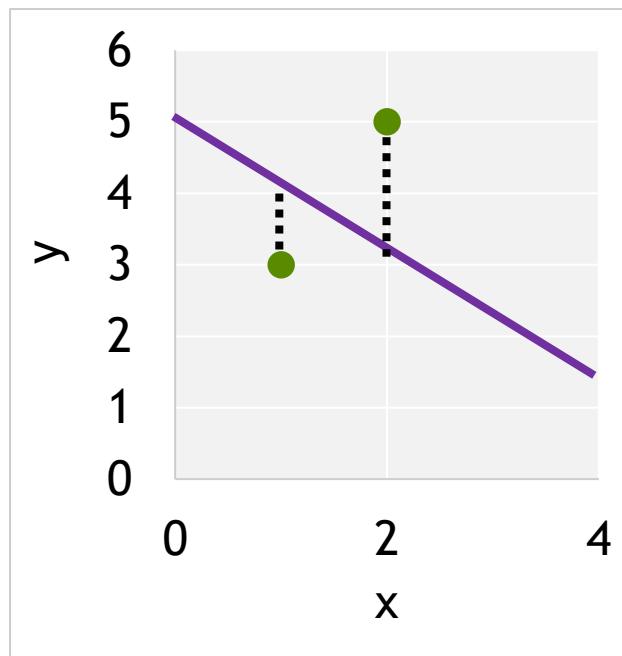
$$y = mx + b$$

↓ ↓
-1 5

| x | y | \hat{y} | err^2 |
|---|---|-----------|---------|
| 1 | 3 | 4 | 1 |
| 2 | 5 | 3 | 4 |

MSE = 2.5
RMSE = 1.6

오차(error)



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

$$(3 - 3.5)^2 + (5 - 1.5)^2 / 2 = 2.5$$

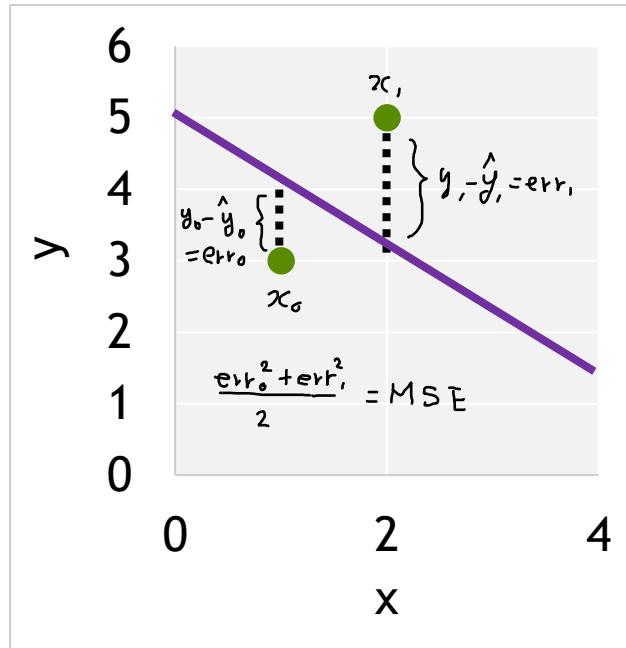
$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y - \hat{y})^2}$$

$$\sqrt{2.5} = 1.6$$

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 |



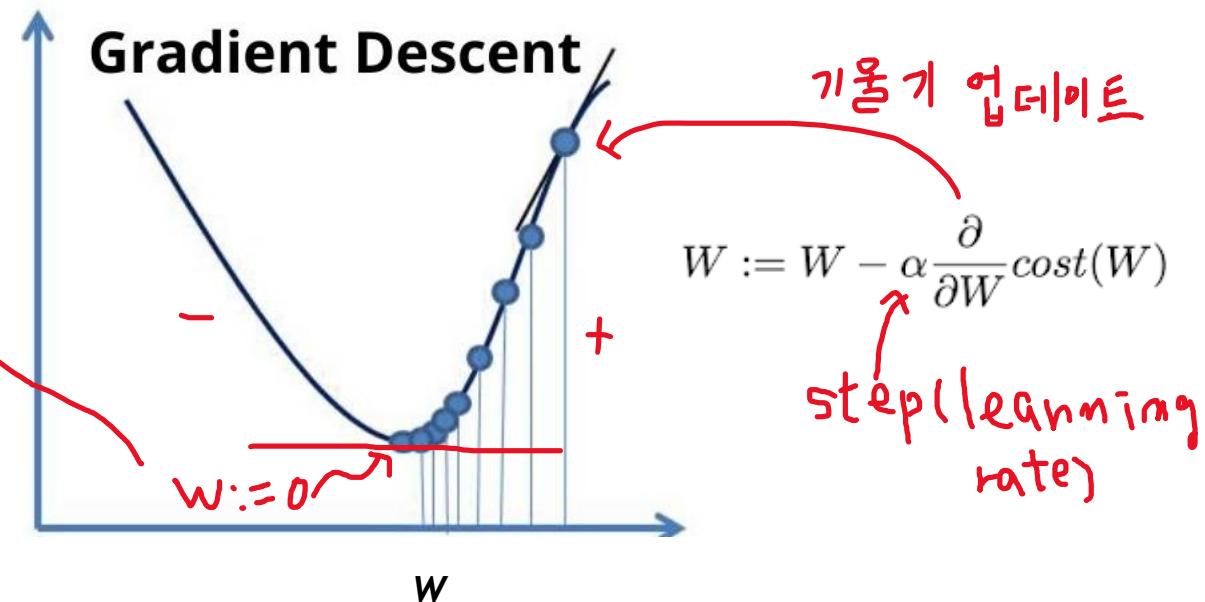
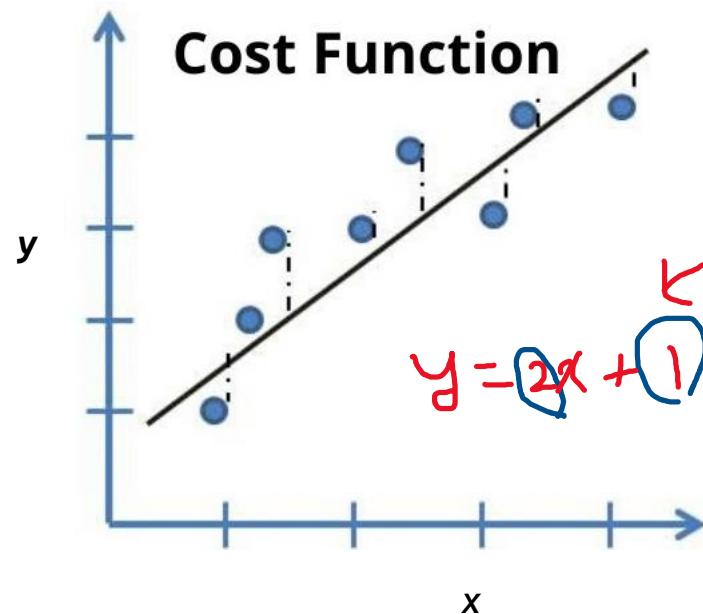
```
1  data = [(1, 3), (2, 5)]
2  m = -1
3  b = 5
4
5  def get_mse(data, m, b):
6      """Calculates Mean Square Error"""
7      n = len(data)
8      squared_error = 0
9
10     for x, y in data:
11         # Find predicted y
12         y_hat = m*x+b
13
14         # Square difference between
15         # prediction and true value
16         squared_error += (
17             y - y_hat)**2
18
19     # Get average squared difference
20     mse = squared_error / n
21
22     return mse
```

경사 하강법(GRADIENT DESCENT)

How to minimize cost(loss)

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

$\hat{y} = mx + b$
예측값(\hat{y}) 정답(실제값(y))



- 경사하강법 쉽게 이해하기 : <https://youtu.be/GEdLNvPlbiM>

경사 하강법(GRADIENT DESCENT)

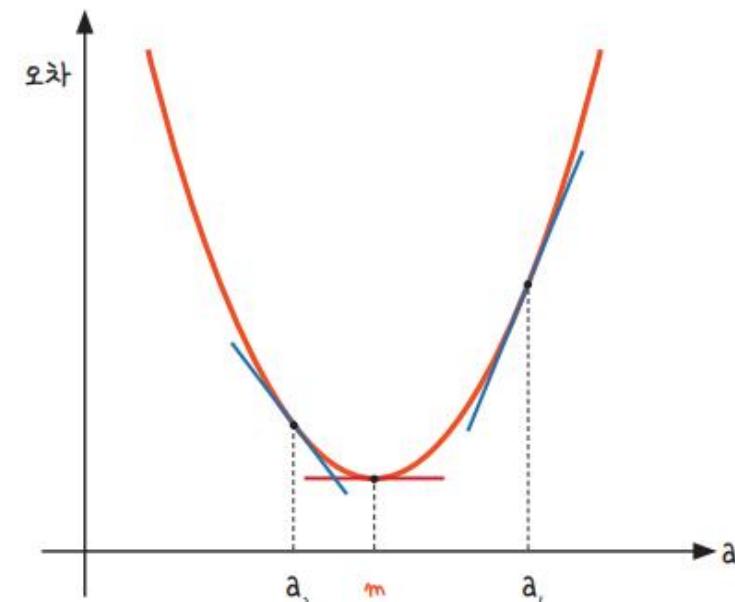
How to minimize cost(loss)
$$cost(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

$\hat{y} = ax + b$
예측값(\hat{y}) 정답(실제값(y))

- (편)미분은 한 점에서의 순간 기울기 임
- $y = x^2$ 그래프에서 x 에 다음과 같이 a_1, a_2 그리고 m 을 대입해 그 자리에서 미분하면 각 점에서의 순간 기울기가 그려짐

$$\frac{\partial W}{\partial a} : a \text{로 편미분한 결과} = \frac{2}{m} \sum -x_i(y_i - (ax_i + b))$$

$$\frac{\partial W}{\partial b} : b \text{로 편미분한 결과} = \frac{2}{m} \sum -(y_i - (ax_i + b))$$

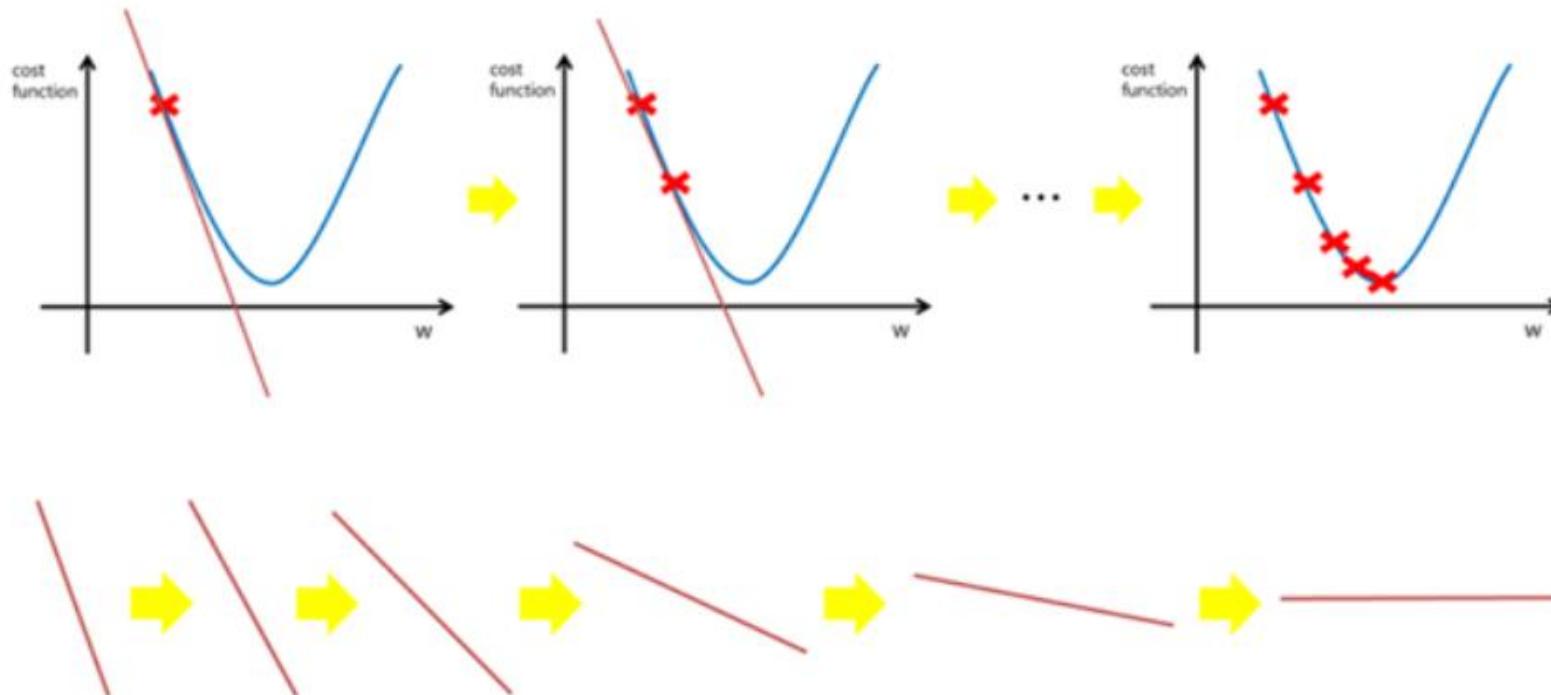


경사 하강법(GRADIENT DESCENT)

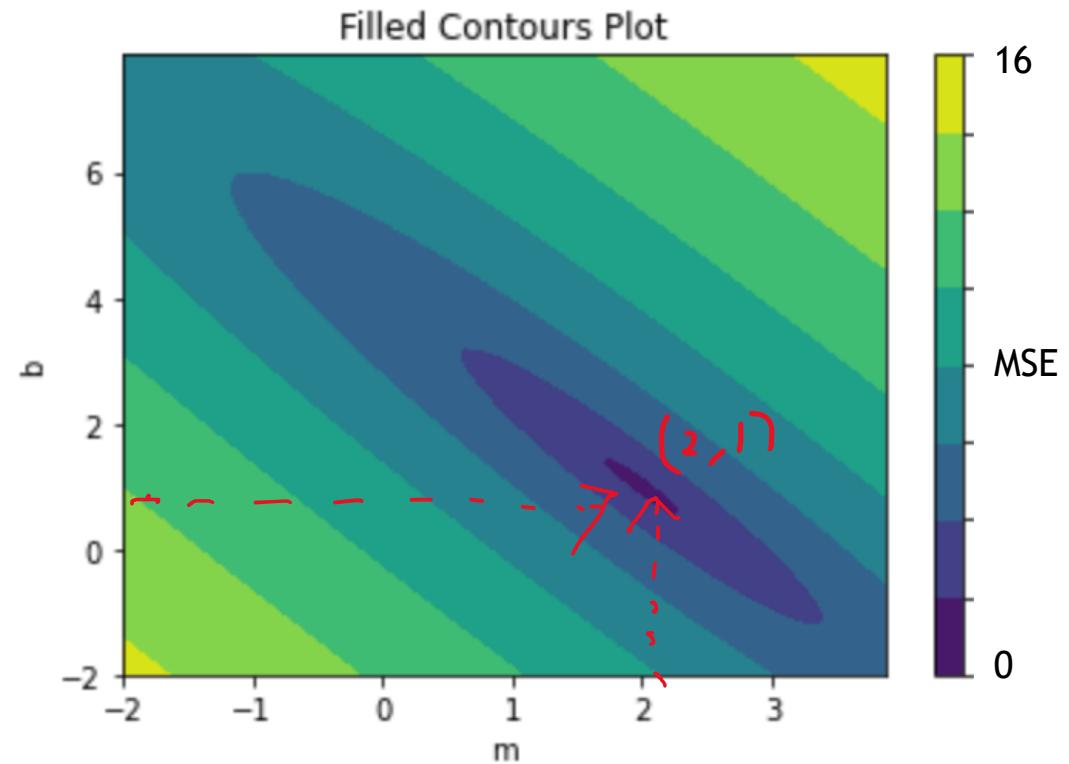
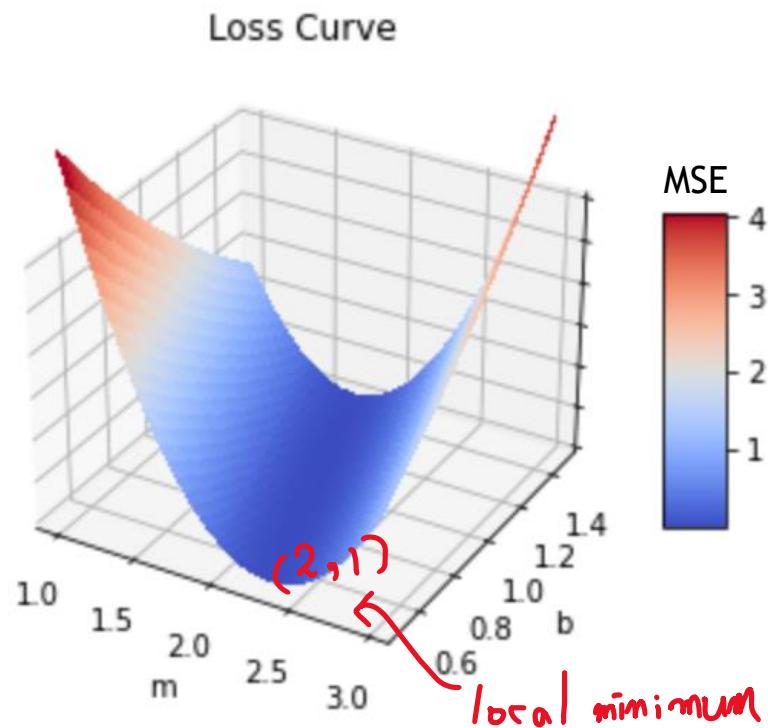
How to minimize cost(loss)

$$cost(W) = \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - \hat{y}^{(i)})^2$$

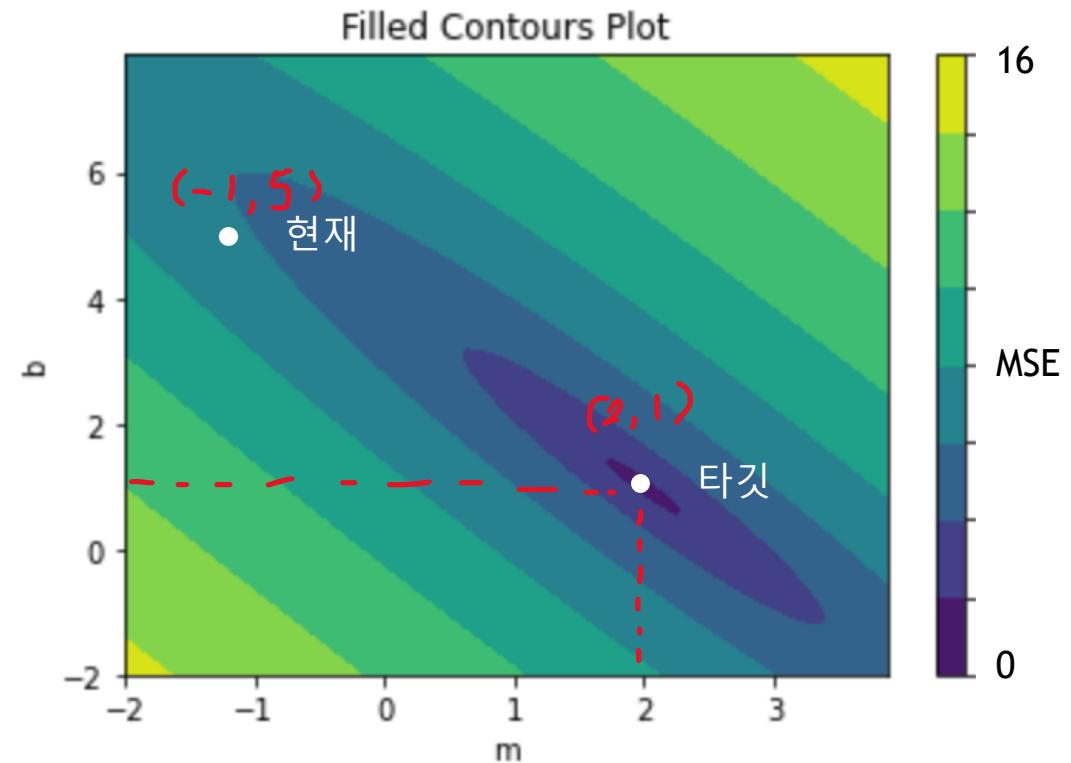
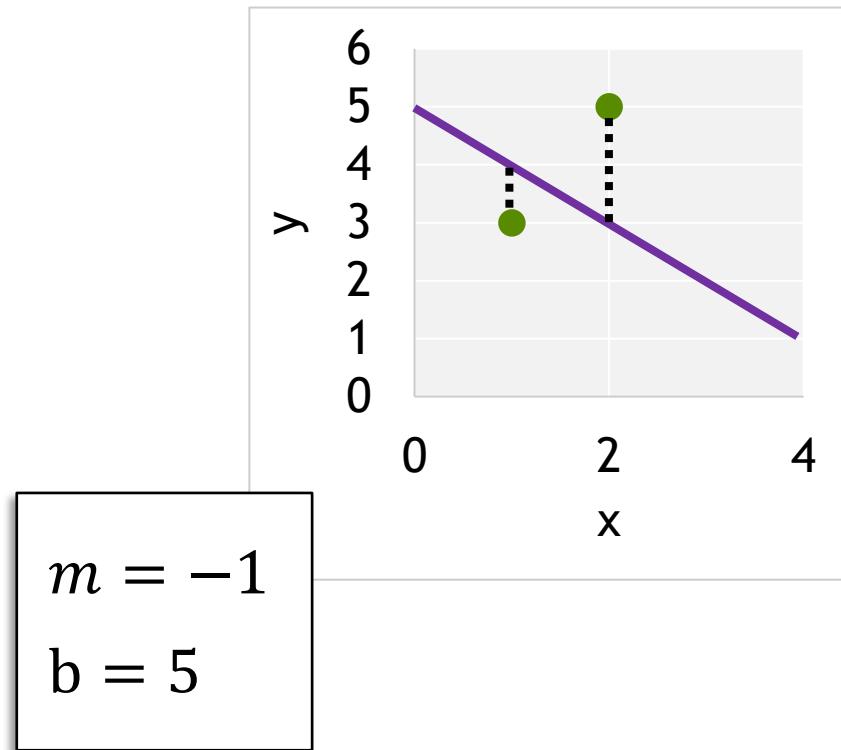
예측값(\hat{y}) 정답(실제값(y))



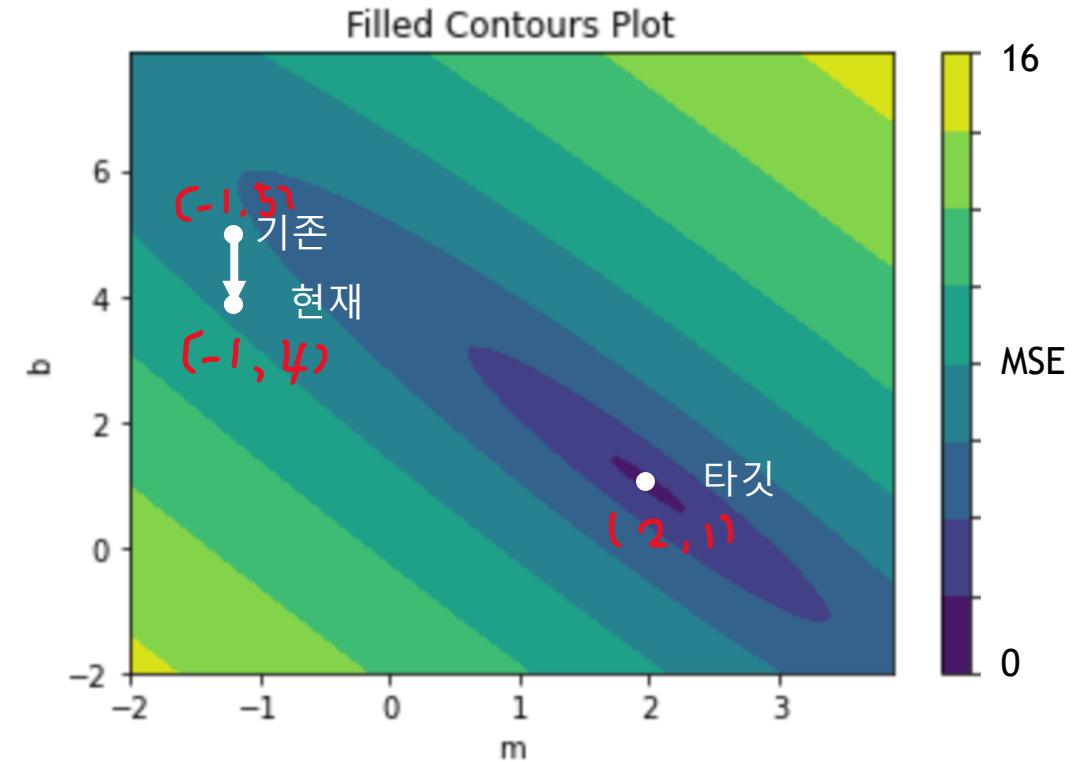
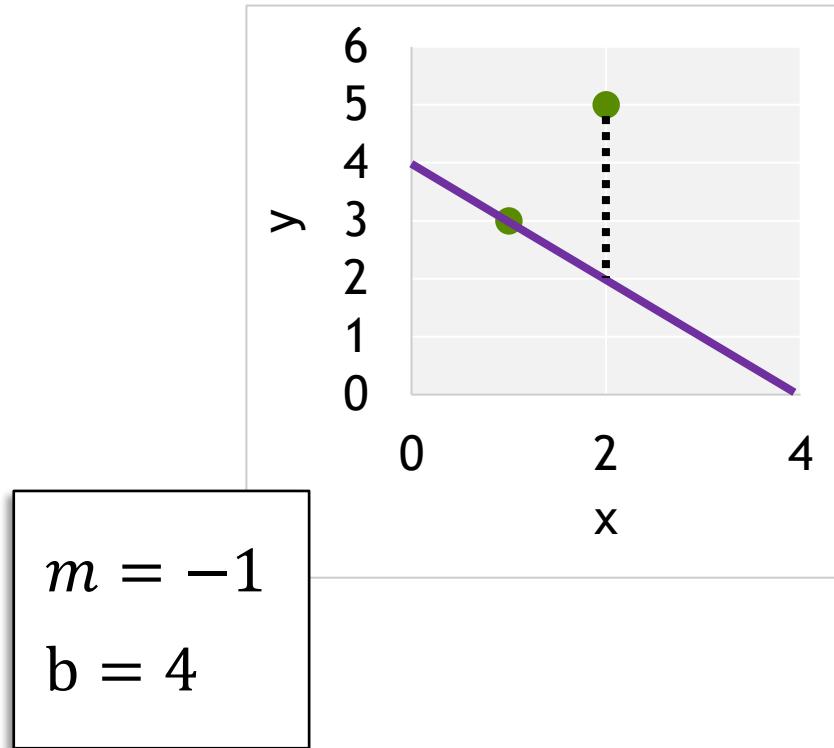
손실 곡선 (THE LOSS CURVE)



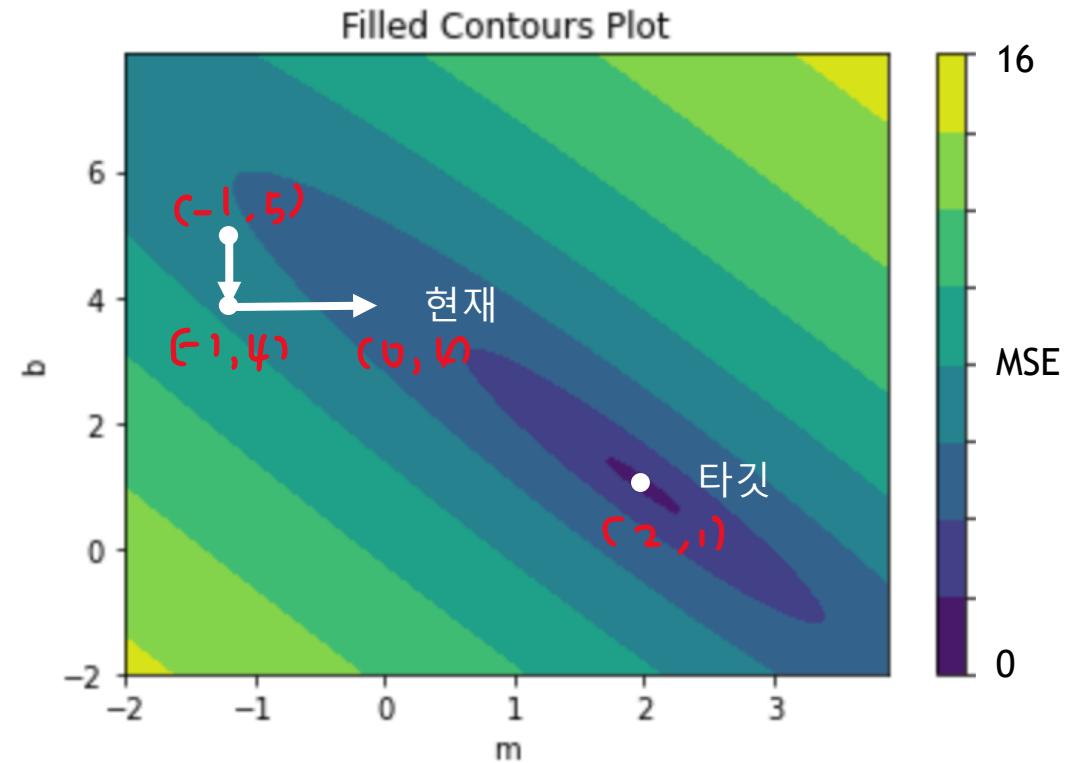
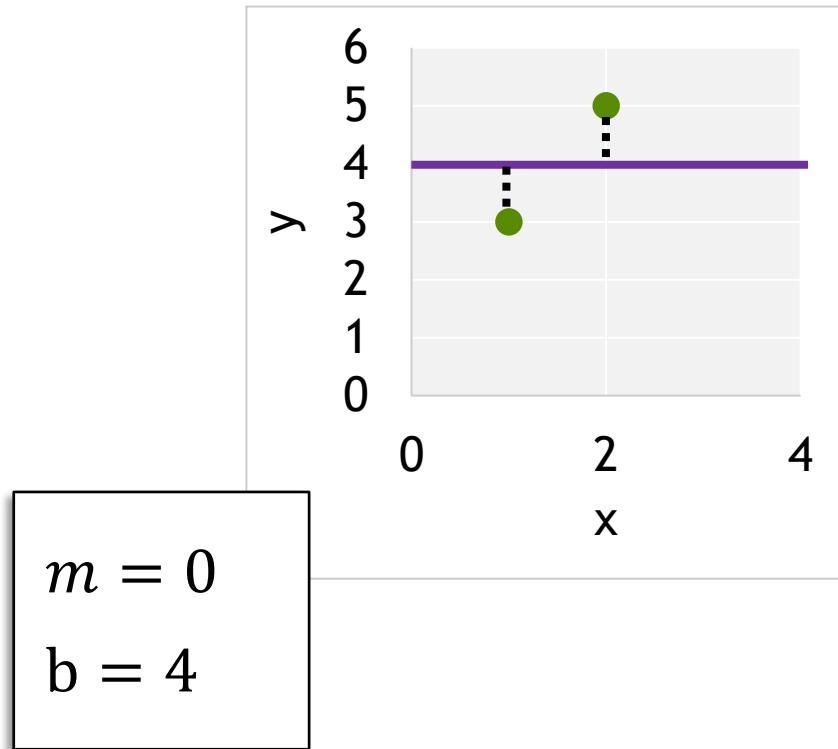
손실 곡선 (THE LOSS CURVE)



손실 곡선 (THE LOSS CURVE)



손실 곡선 (THE LOSS CURVE)



손실 곡선 (THE LOSS CURVE)

그래디언트
(Gradient)

어떤 방향으로 손실이 가장 많이
감소하는가

λ : 학습률
(Learning rate)

이동 거리

에포크
(Epoch)

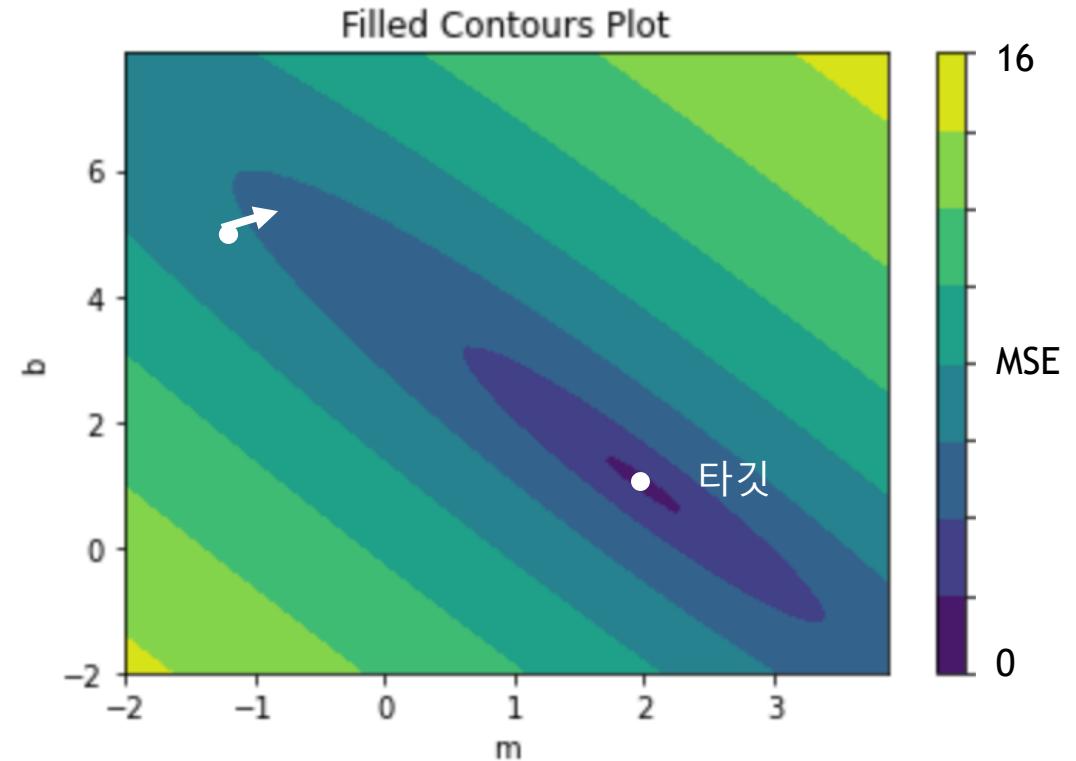
전체 데이터세트로 모델 업데이트

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)

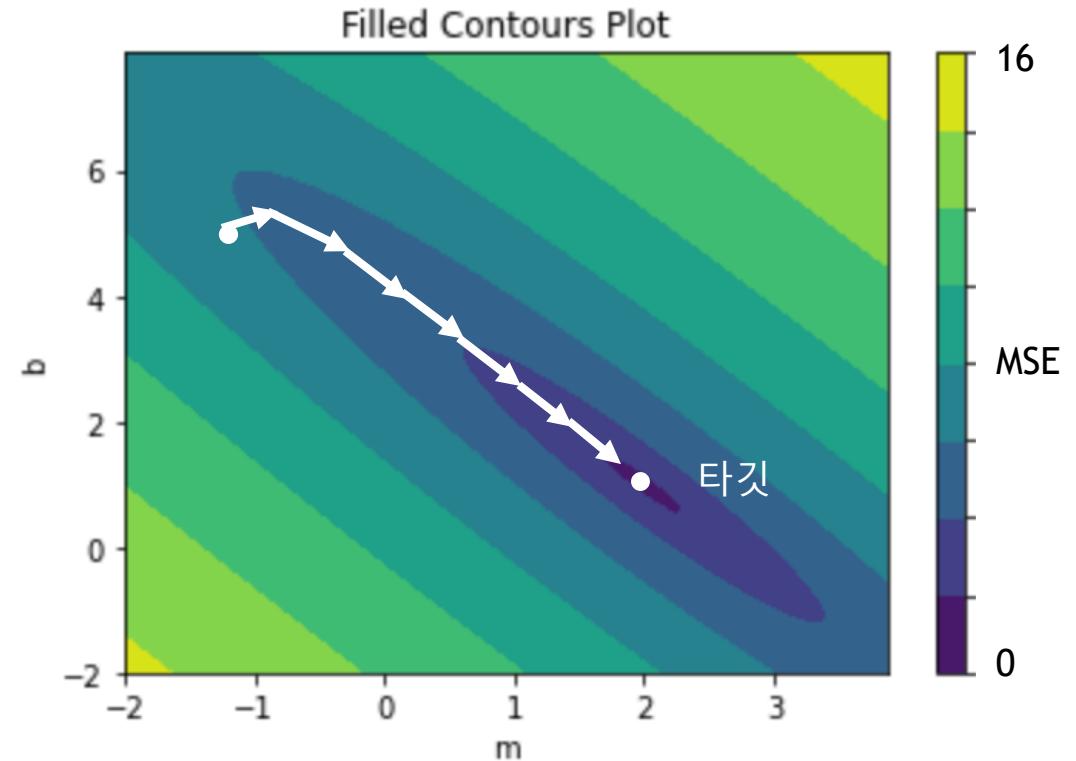
전체 데이터세트로 모델 업데이트

배치 (Batch)

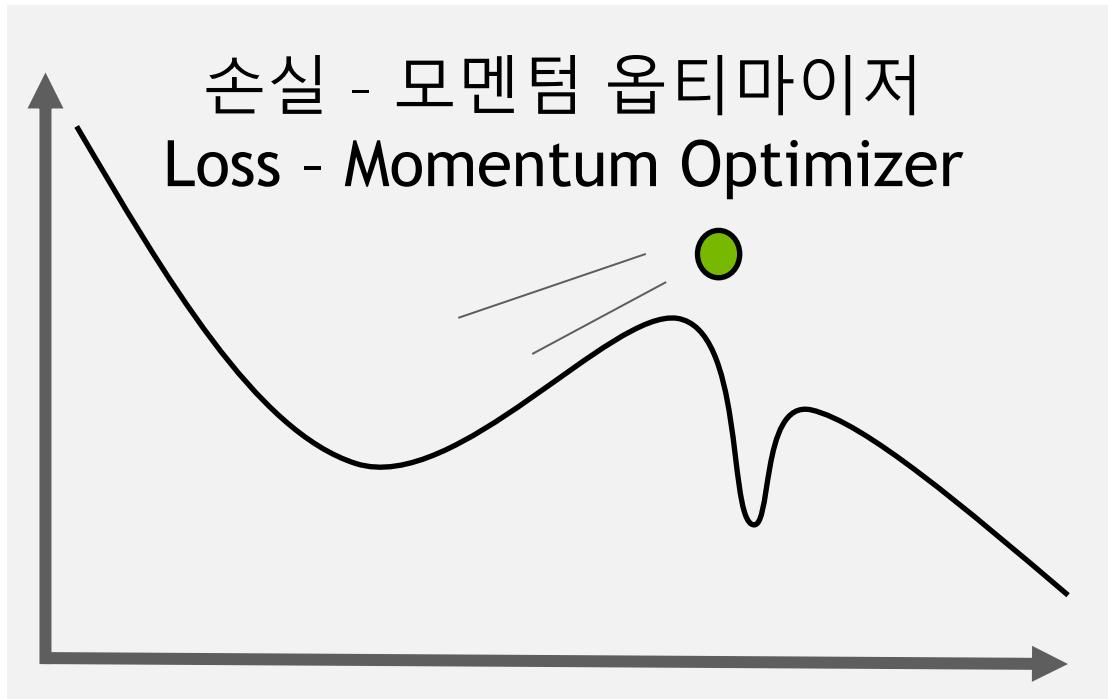
전체 데이터세트 샘플

단계 (Step)

가중치 매개변수에 대한 업데이트

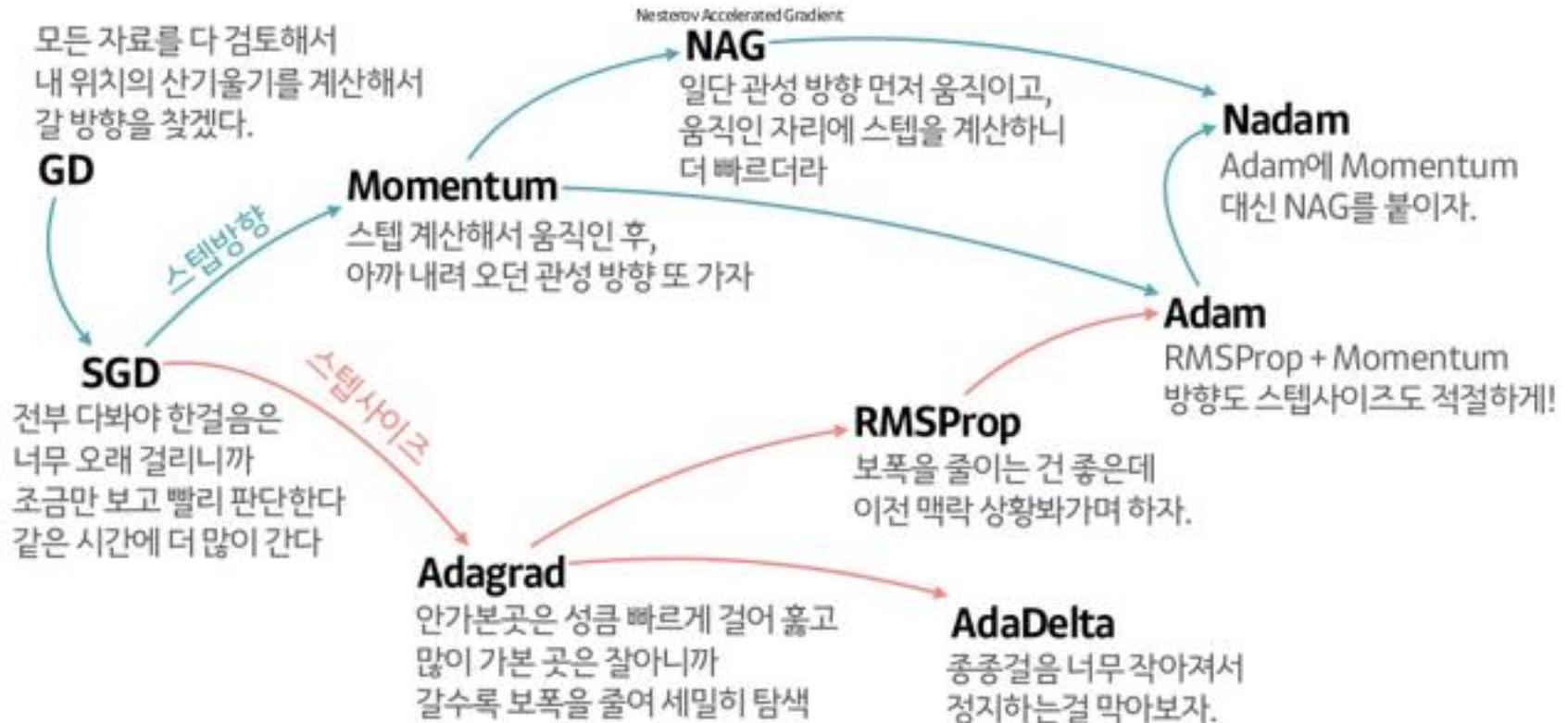


옵티마이저 (OPTIMIZERS)

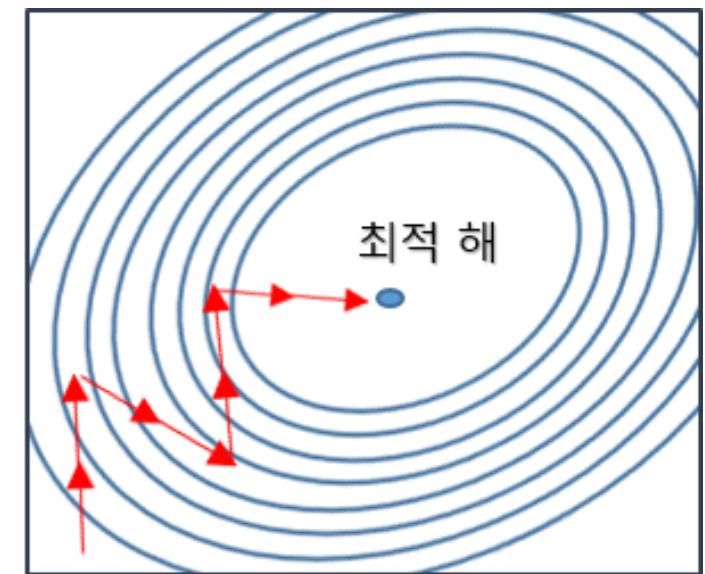
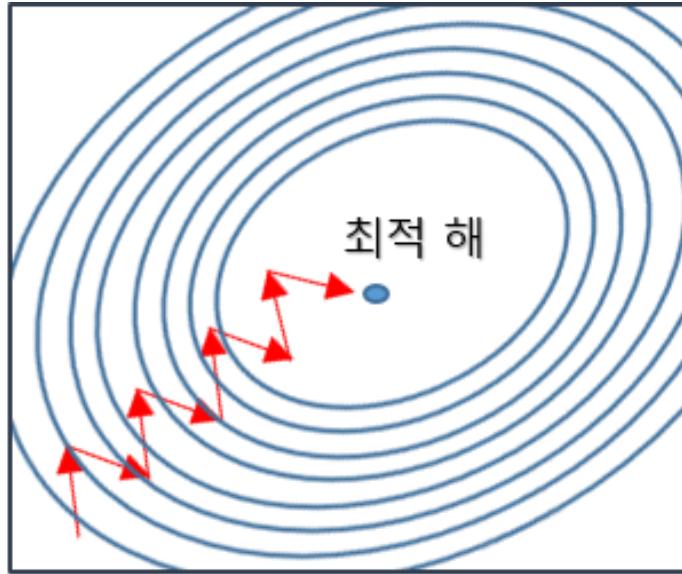
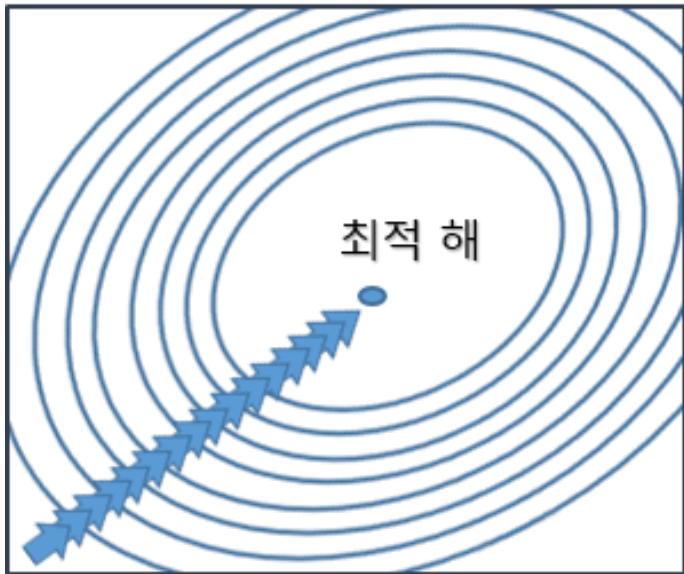


- Adam
- Adagrad
- RMSprop
- SGD

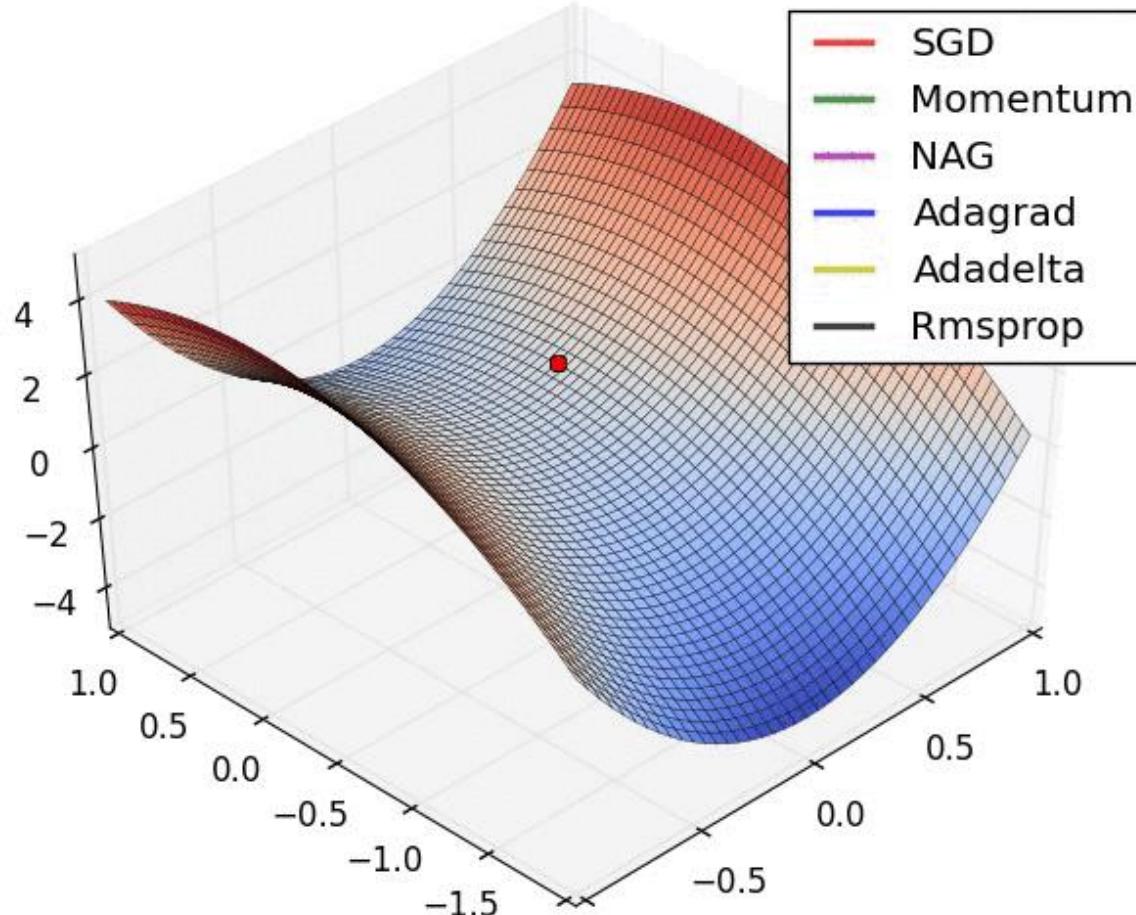
옵티마이저 (OPTIMIZERS)



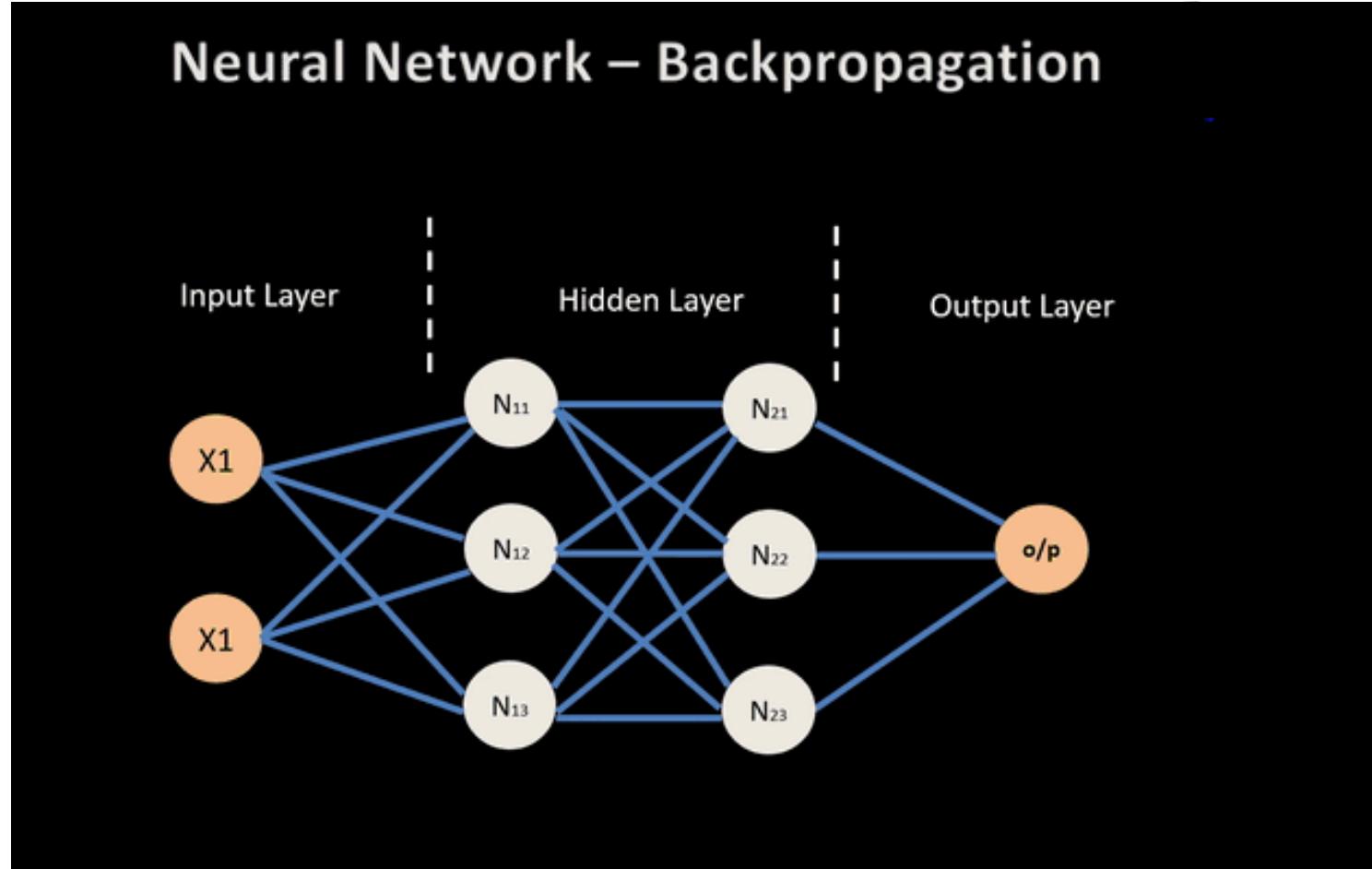
옵티마이저 (OPTIMIZERS)



옵티마이저 (OPTIMIZERS) 성능 비교



오차 역전파 (ERROR BACKPROPAGATION)

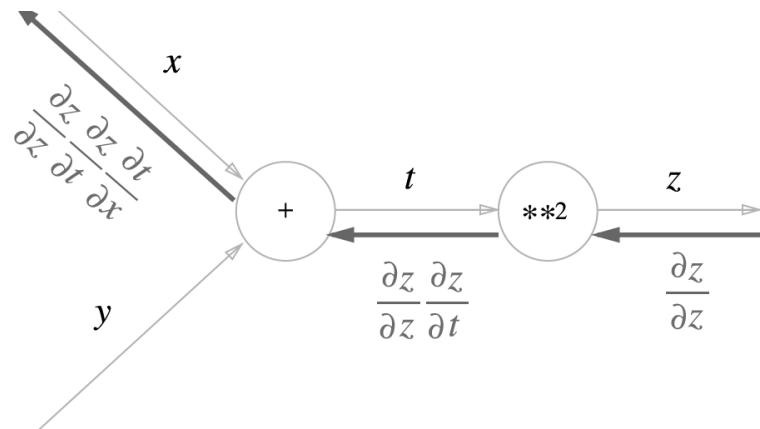


참조 : machinelearningknowledge.ai

1 오차 역전파, 체인 룰(CHAIN RULE)

$$\begin{array}{l} z = g(y), y = f(x) \\ \text{미분} \\ \rightarrow (g \circ f)'(x) = g'(y)f'(x) \\ z = (g \circ f)(x) = g(f(x)) \\ \downarrow \text{Chain rule} \end{array}$$

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



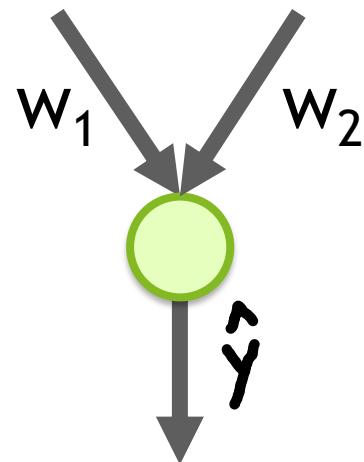
Backpropagation algorithm :

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

A complex network graph is displayed against a dark gray background. The graph consists of numerous small, semi-transparent white and light green circular nodes, connected by a dense web of thin, light gray lines representing edges. The nodes are scattered across the frame, with a higher density in the upper right quadrant and more isolated nodes towards the bottom left.

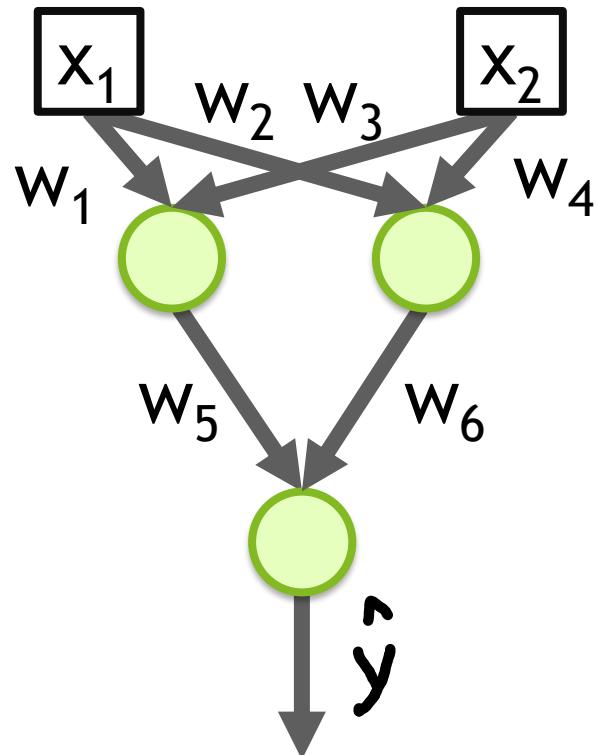
뉴런에서 네트워크로

네트워크 구축



- 더 많은 변수로의 확장

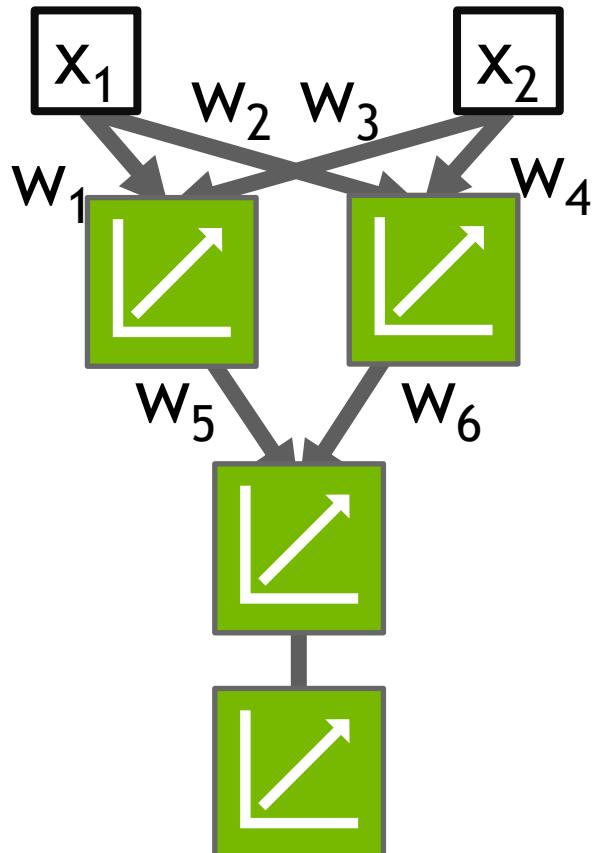
네트워크 구축



- 더 많은 변수로의 확장
- 뉴런들간의 연결

네트워크 구축

- 딥뉴럴 네트워크에서 임의의 함수를 통해 복잡한 관계를 찾기 위해 비선형성이 필요



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

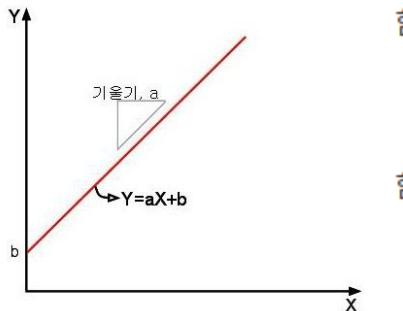


활성화 함수 ACTIVATION FUNCTIONS

선형(LINEAR), 비선형(NON-LINEAR)

● 선형(linear)

- 그래프의 형태가 1개의 직선으로 표현된다는 뜻



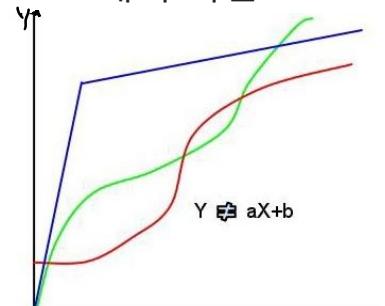
함수 f 에 대해,

- 가산성(Additivity), 즉, 임의의 수 x, y 에 대해 $f(x + y) = f(x) + f(y)$ 가 항상 성립하고
- 동차성(Homogeneity), 즉, 임의의 수 x 와 α 에 대해 $f(\alpha x) = \alpha f(x)$ 가 항상 성립할 때

함수 f 는 선형이라고 한다.

● 비선형(non-linear)

- 1개의 직선으로 표현되지 않는 모든 형태로 1차 방정식으로 표현되지 않는 모든

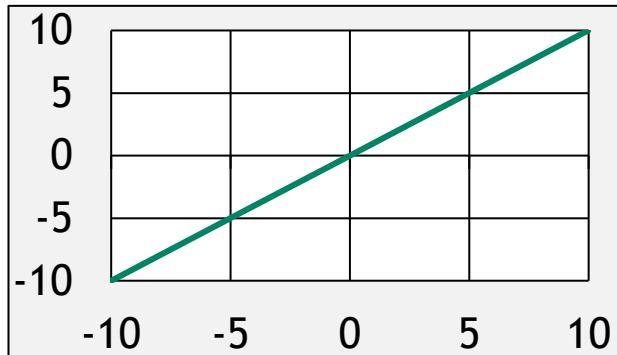


활성화 함수 (ACTIVATION FUNCTIONS)

Linear(선형)

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

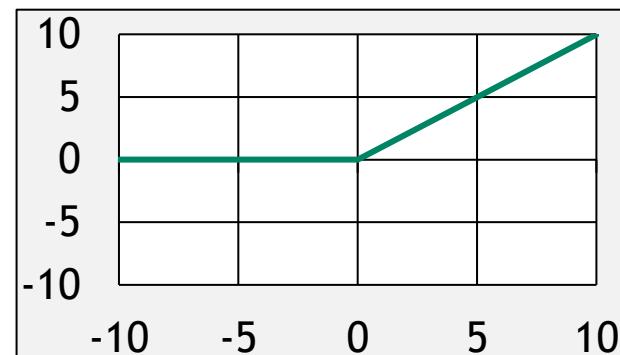
```
1 # Multiply each input  
2 # with a weight (w) and  
3 # add intercept (b)  
4 y_hat = wx+b
```



ReLU

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

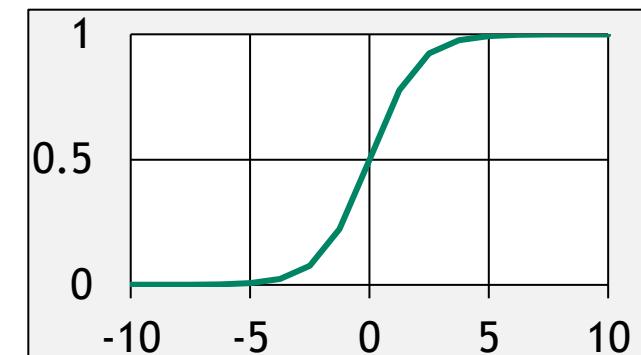
```
1 # Only return result  
2 # if total is positive  
3 linear = wx+b  
4 y_hat = linear * (linear > 0)
```



Sigmoid(시그모이드)

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

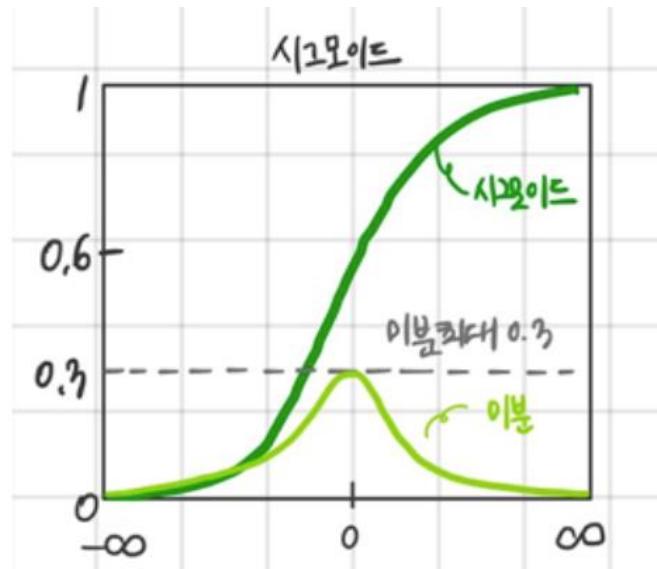
```
1 # Start with line  
2 linear = wx + b  
3 # Warp to - inf to 0  
4 inf_to_zero = np.exp(-1 * linear)  
5 # Squish to -1 to 1  
6 y_hat = 1 / (1 + inf_to_zero)
```



가중치 소실 문제 (VANISHING GRADIENT)

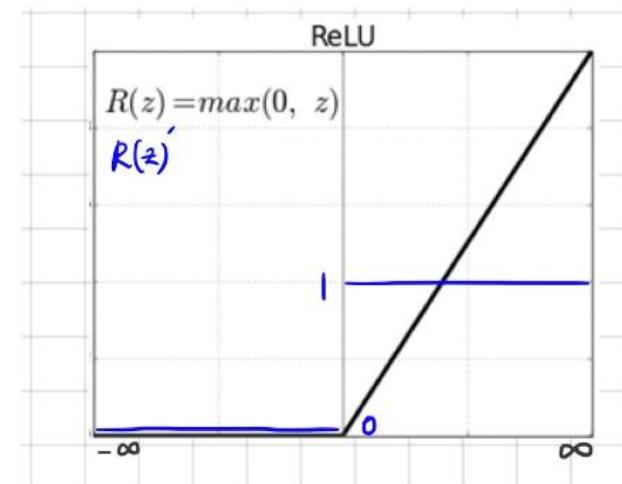
Sigmoid(시그모이드)

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



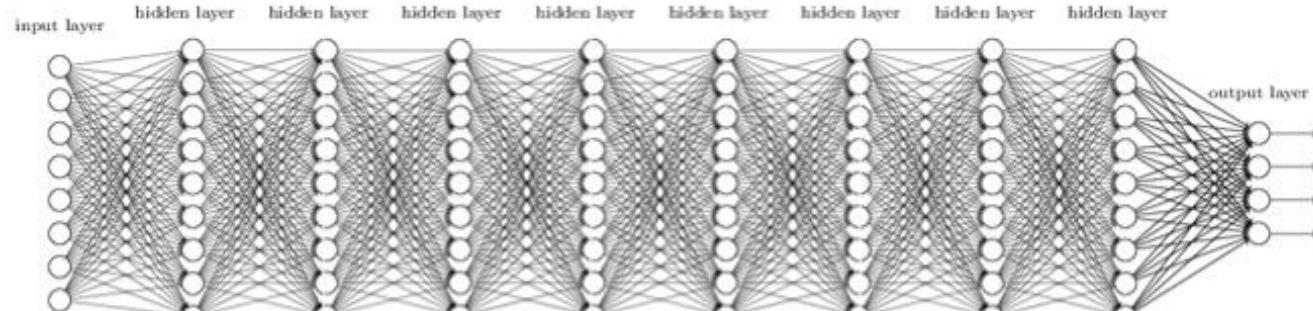
ReLU

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

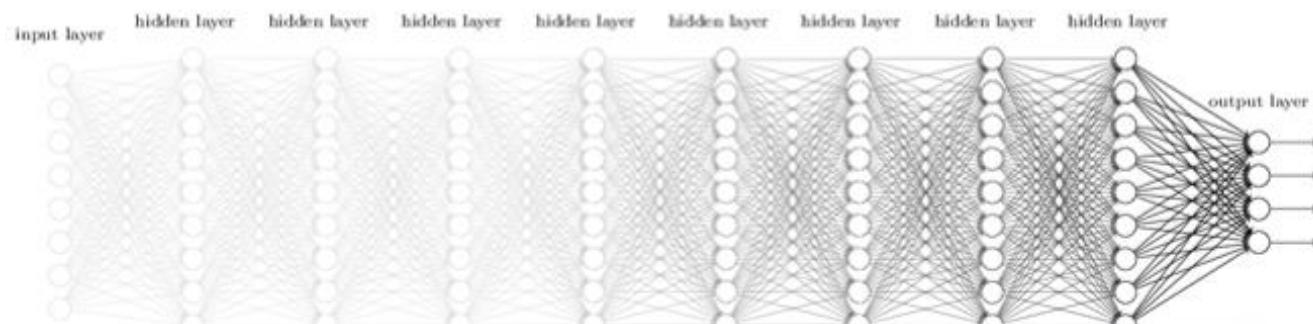


$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

가중치 소실 문제 (VANISHING GRADIENT)



Deep Neural Network

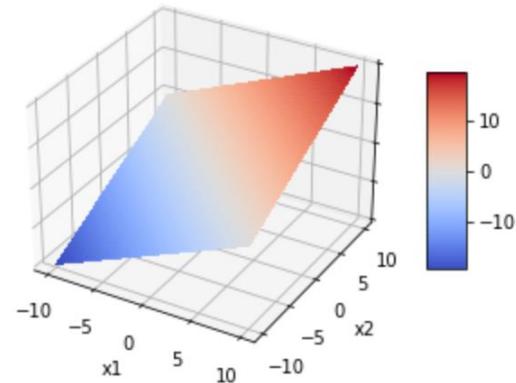


Vanishing Gradient

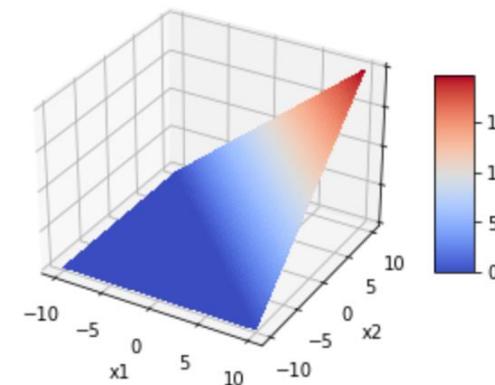
Backpropagation

활성화 함수 (ACTIVATION FUNCTIONS)

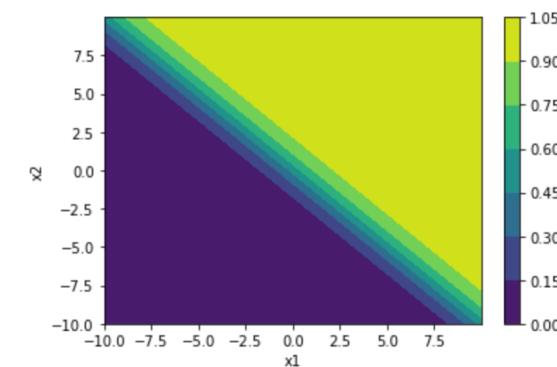
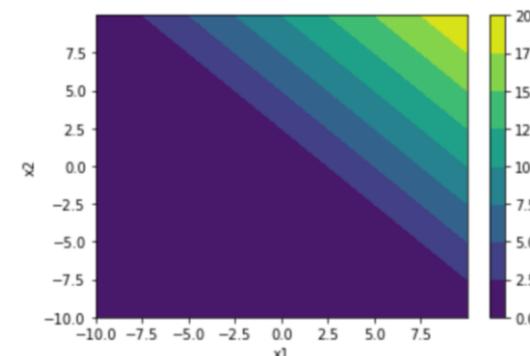
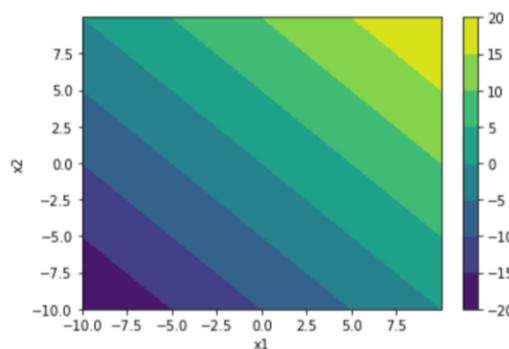
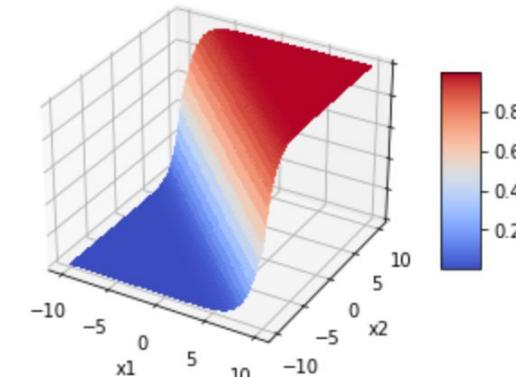
Linear(선형)



ReLU

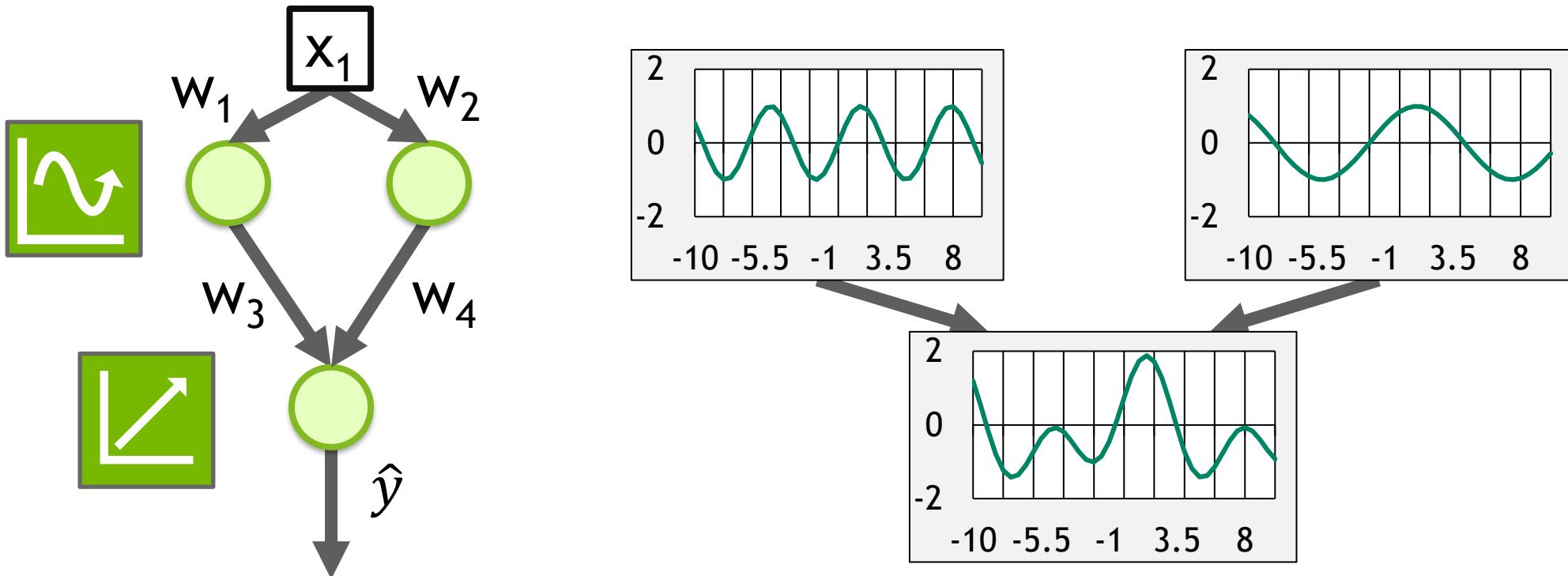


Sigmoid(시그모이드)



활성화 함수 (ACTIVATION FUNCTIONS)

- 딥뉴럴 네트워크에서 임의의 함수를 통해 복잡한 관계를 찾기 위해 비선형성이 필요





과적합 OVERFITTING

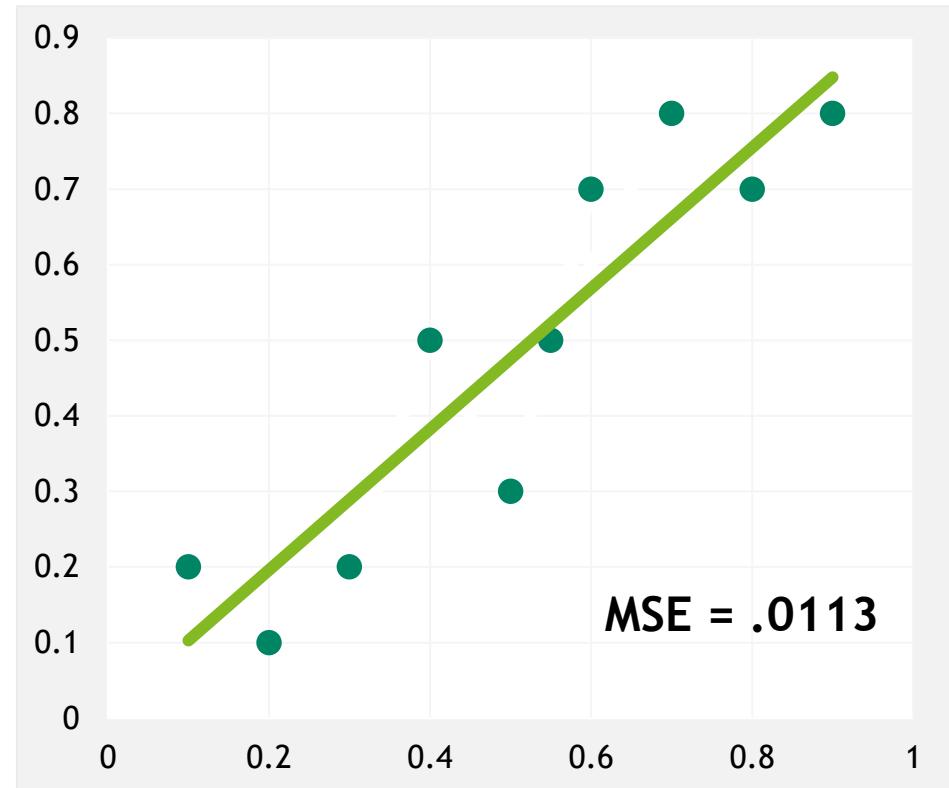
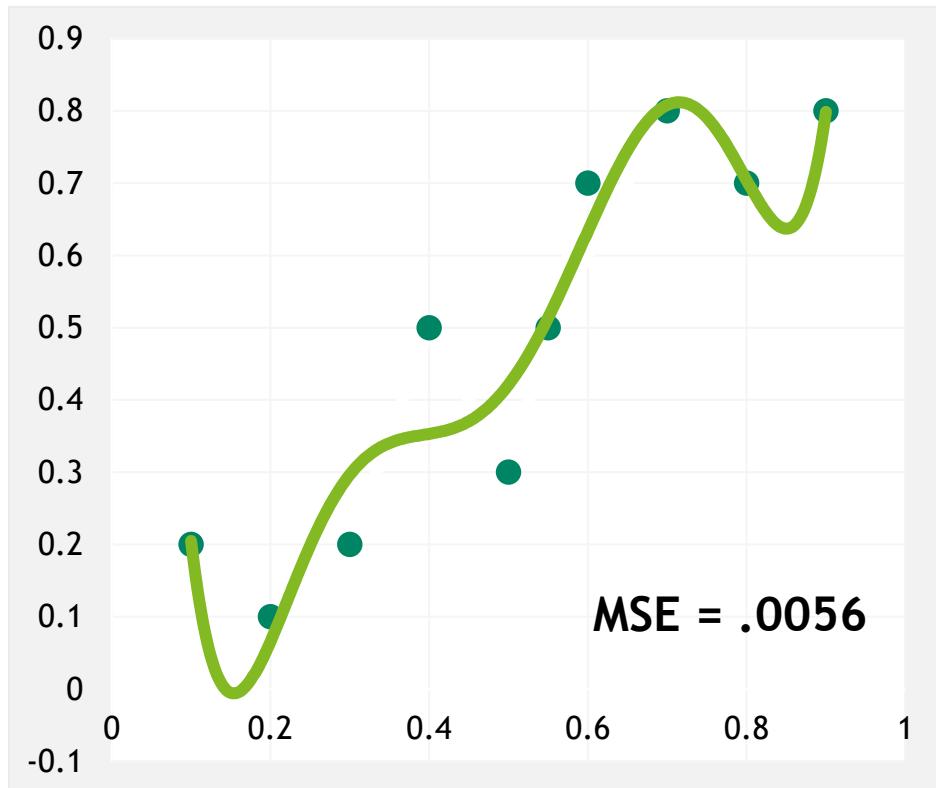
과적합 (OVERFITTING)

거대한 뉴럴 네트워크를 구축하면 되지 않을까?



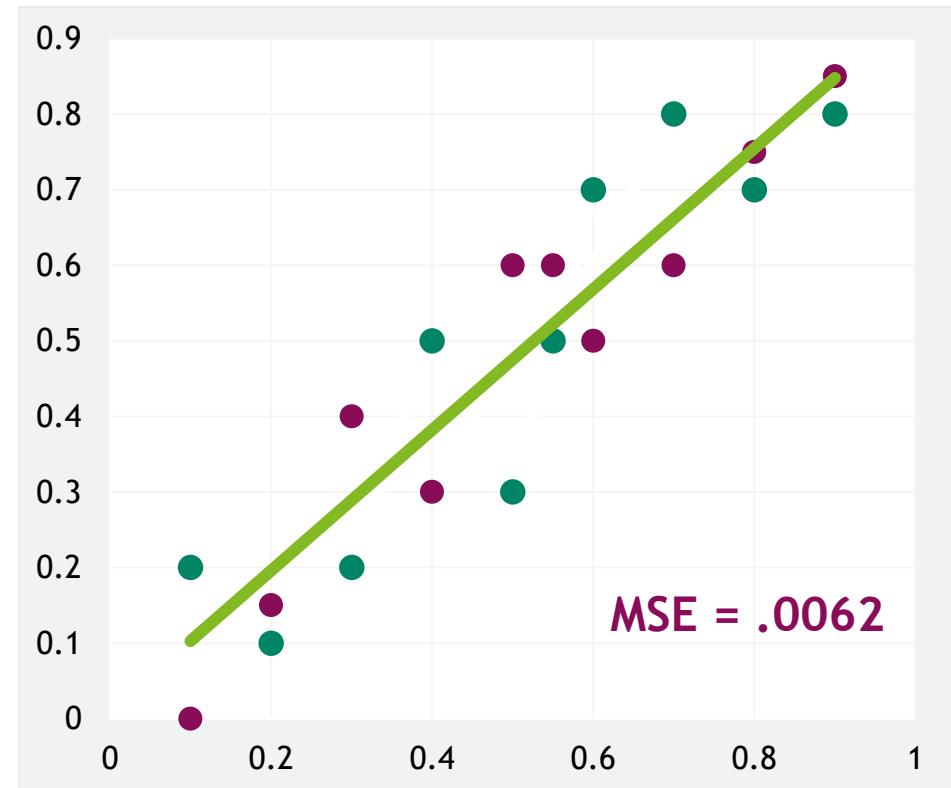
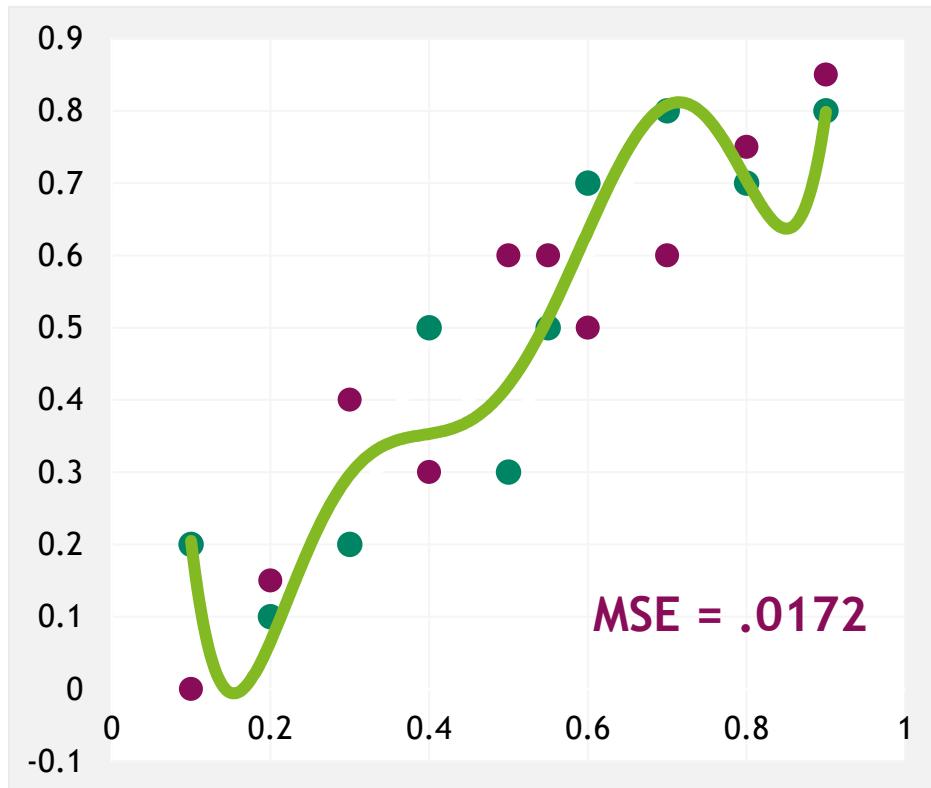
과적합 (OVERFITTING)

어떤 추세선이 더 나은가?



과적합 (OVERFITTING)

어떤 추세선이 더 나은가?



트레이닝 데이터와 검증 데이터 비교

암기 회피(Avoid memorization)

트레이닝 데이터 (Training data)

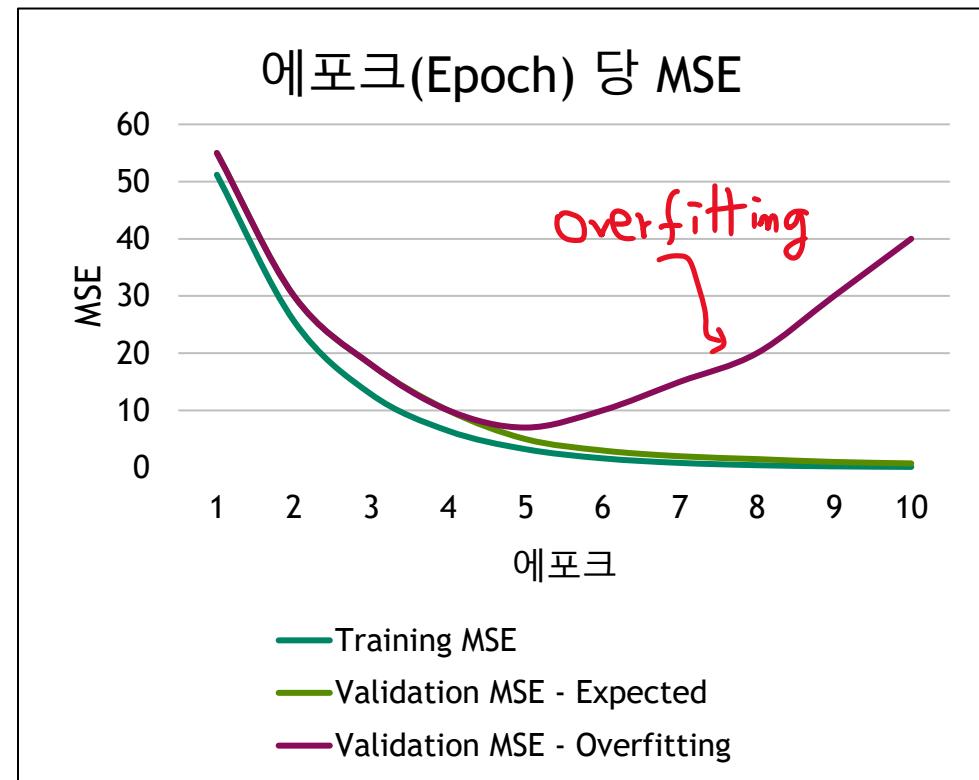
- 학습해야 하는 모델의 핵심 데이터세트

검증 데이터 (Validation data)

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

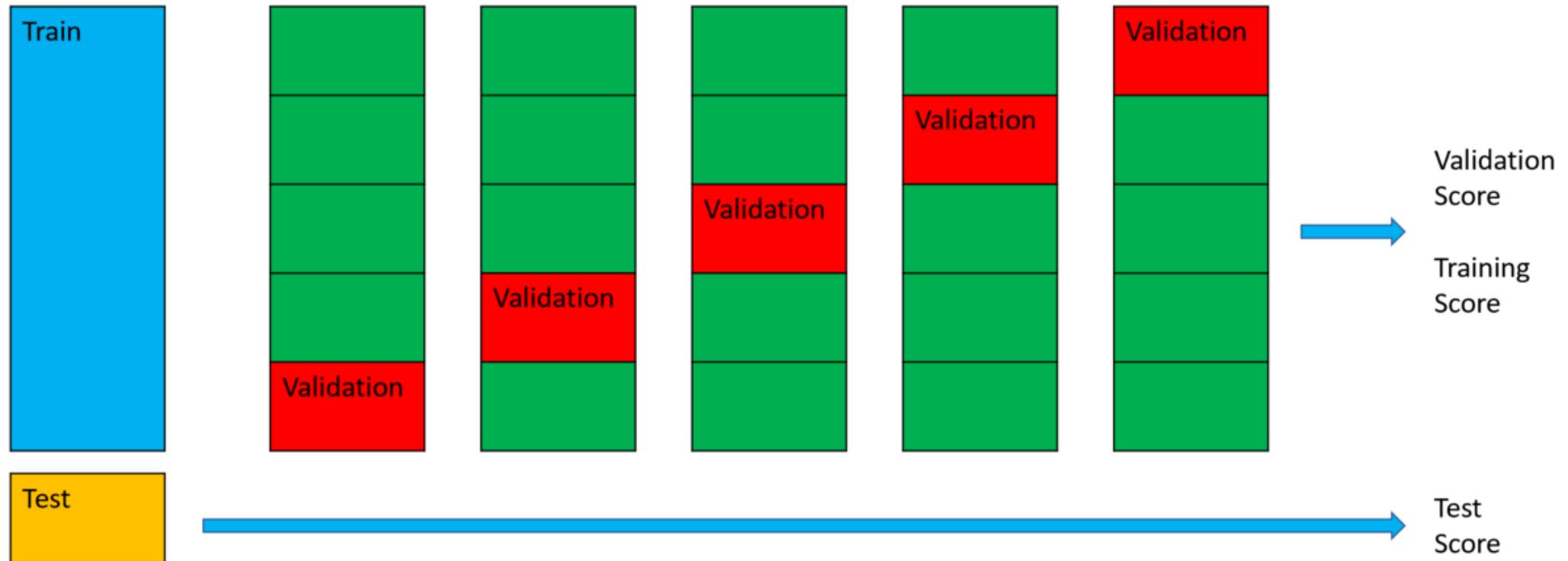
과적합 (Overfitting)

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

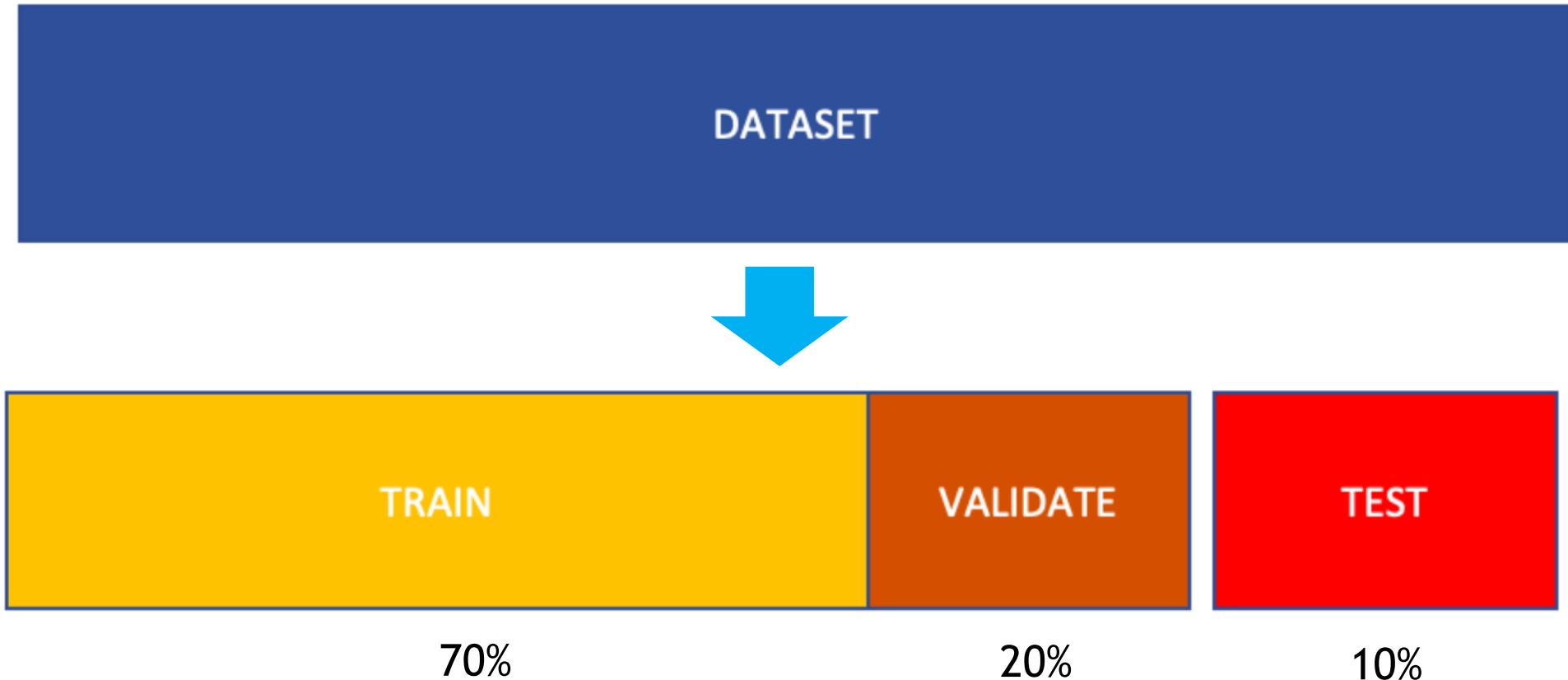


```
model.fit(x_train, y_train, epochs=20, verbose=1, validation_data=(x_valid, y_valid))
```

TRAIN DATA, VALIDATION DATA, TEST DATA



TRAIN DATA, VALIDATION DATA, TEST DATA



실습 01 : 선형 회귀(LINEAR REGRESSION)

- 단순 선형회귀 : 공부한 시간에 따른 성적 예측

| | | | | |
|-----------------|----|----|----|----|
| 공부한 시간(x_1) | 2 | 4 | 6 | 8 |
| 성적(y) | 81 | 93 | 91 | 97 |

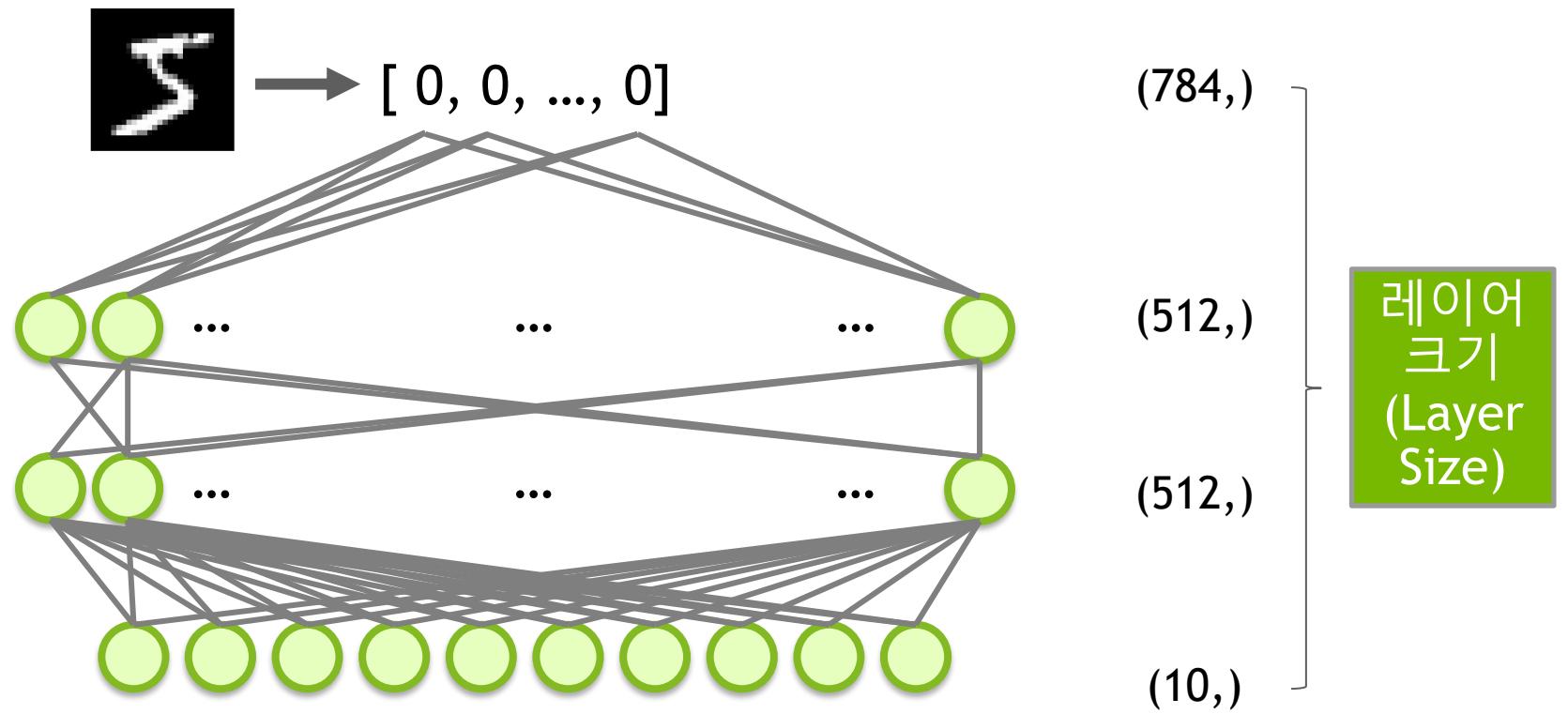
- 다중 선형회귀 : 공부한 시간, 과외수업 횟수에 따른 성적 예측

| | | | | |
|-------------------|----|----|----|----|
| 공부한 시간(x_1) | 2 | 4 | 6 | 8 |
| 과외 수업 횟수(x_2) | 0 | 4 | 2 | 3 |
| 성적(y) | 81 | 93 | 91 | 97 |

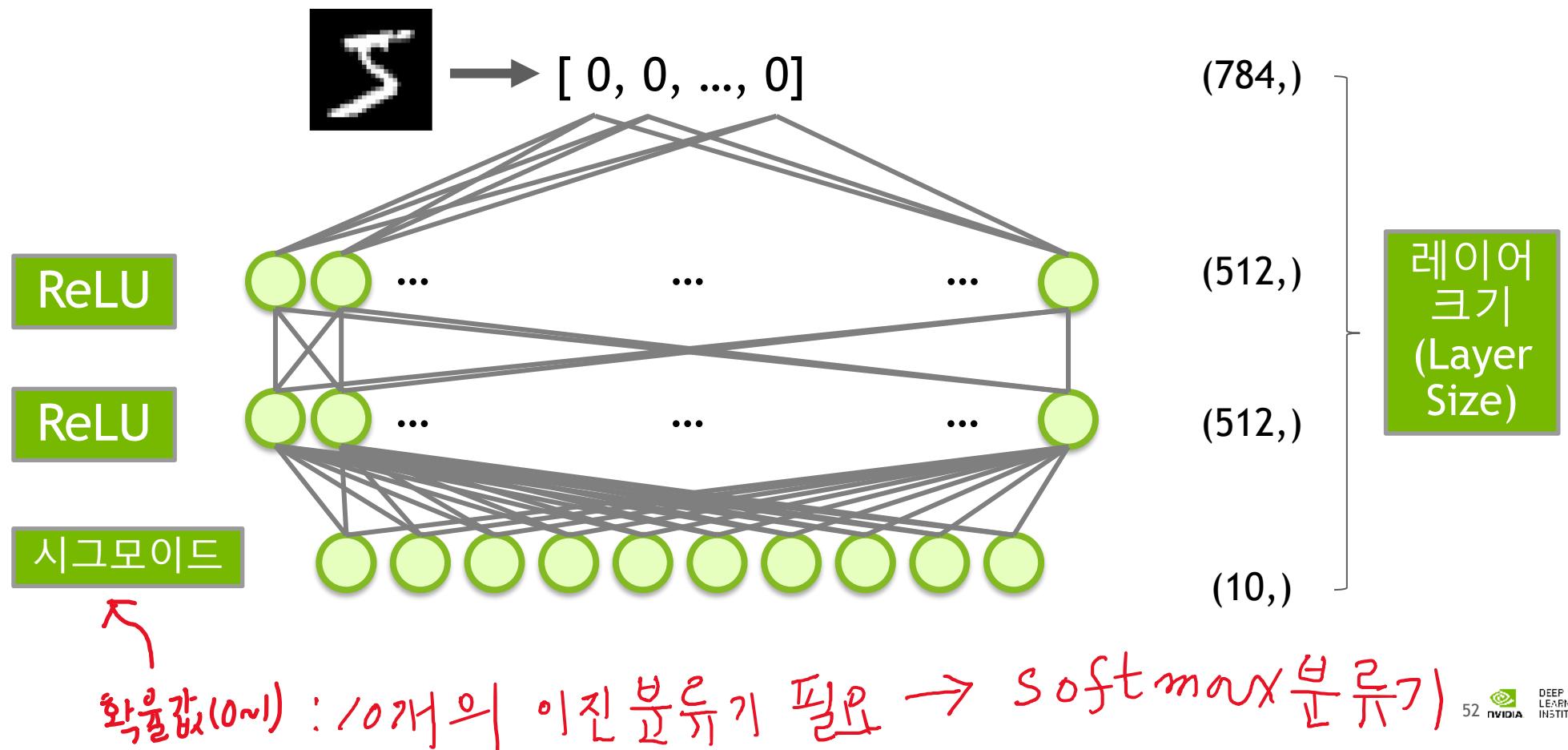


회귀에서 분류로
FROM REGRESSION TO
CLASSIFICATION

MNIST 모델



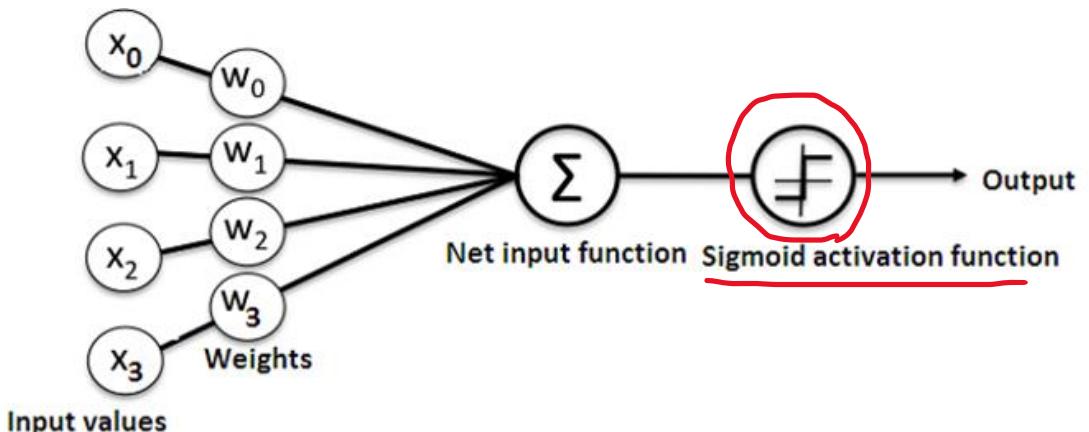
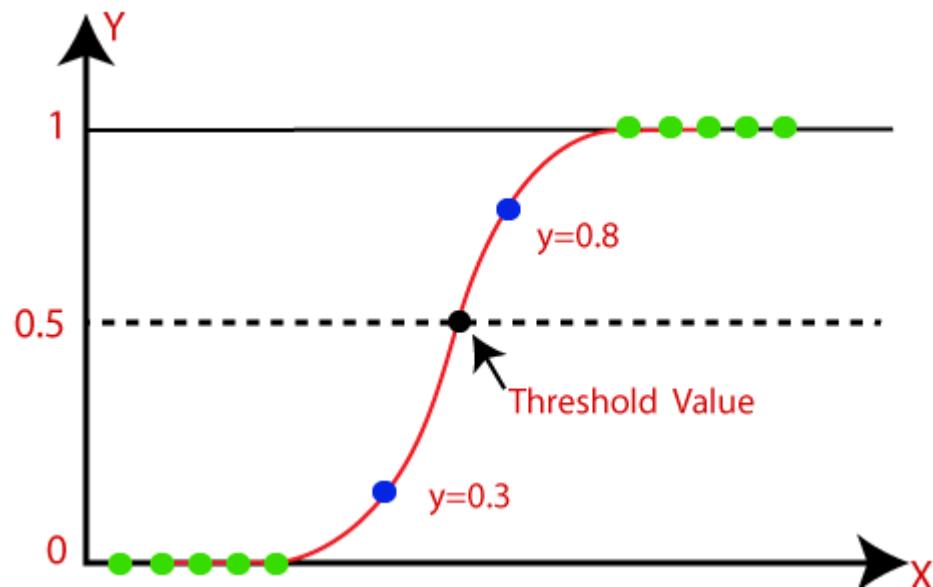
MNIST 모델



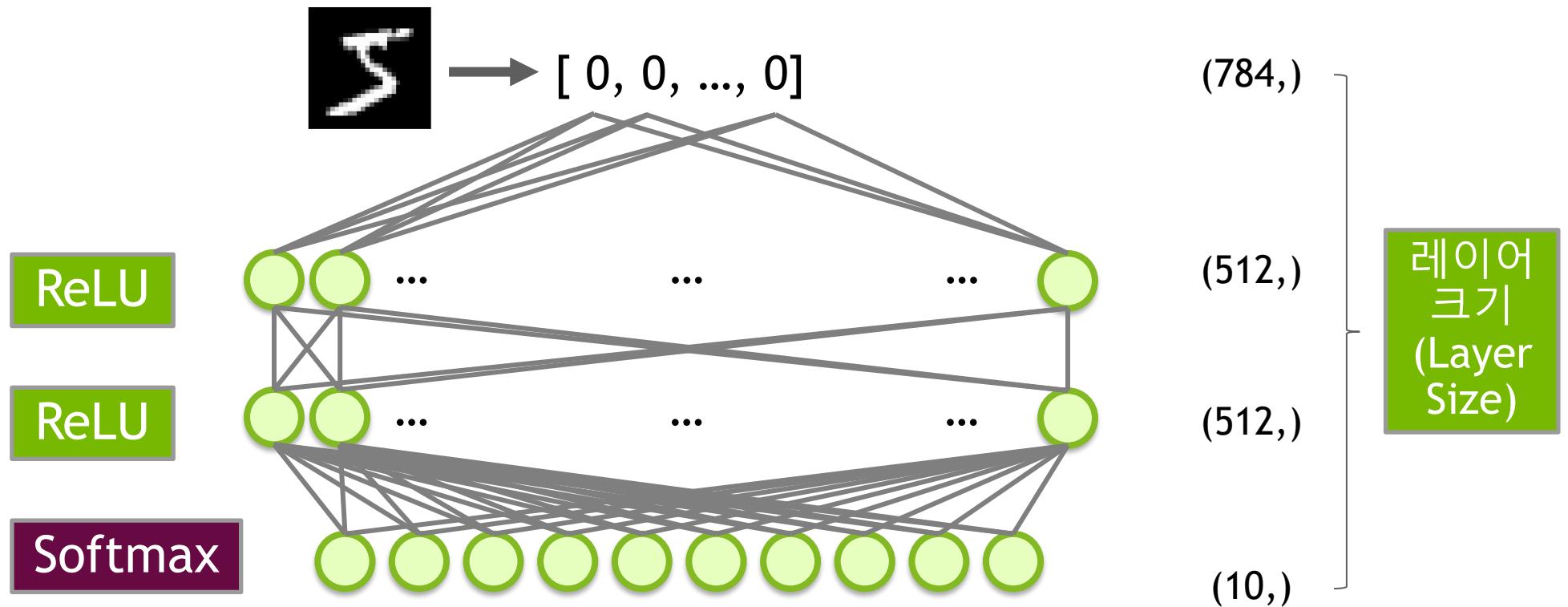
SIGMOID(LOGISTIC) 활성화 함수

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

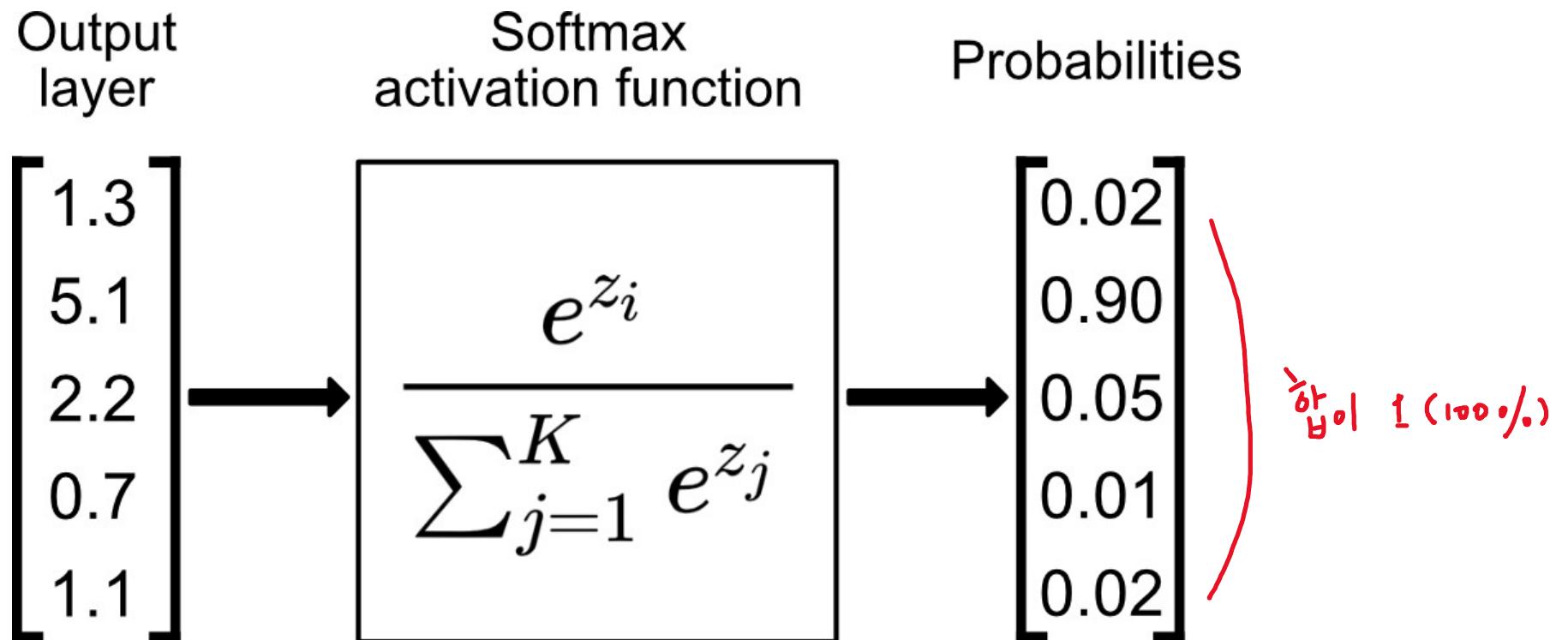
- 이진 분류
 - 시험 합격 여부 : Pass or Fail
 - 스팸 필터 : Spam or Ham
 - SNS 피드 노출 : Show or Hide



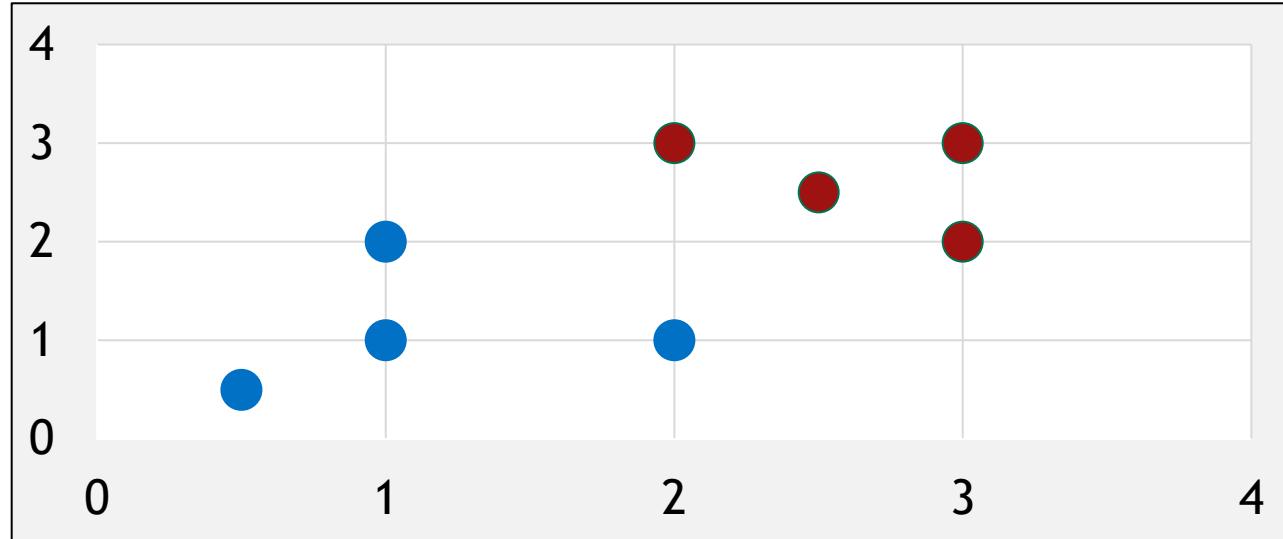
MNIST 모델 : 다중 분류



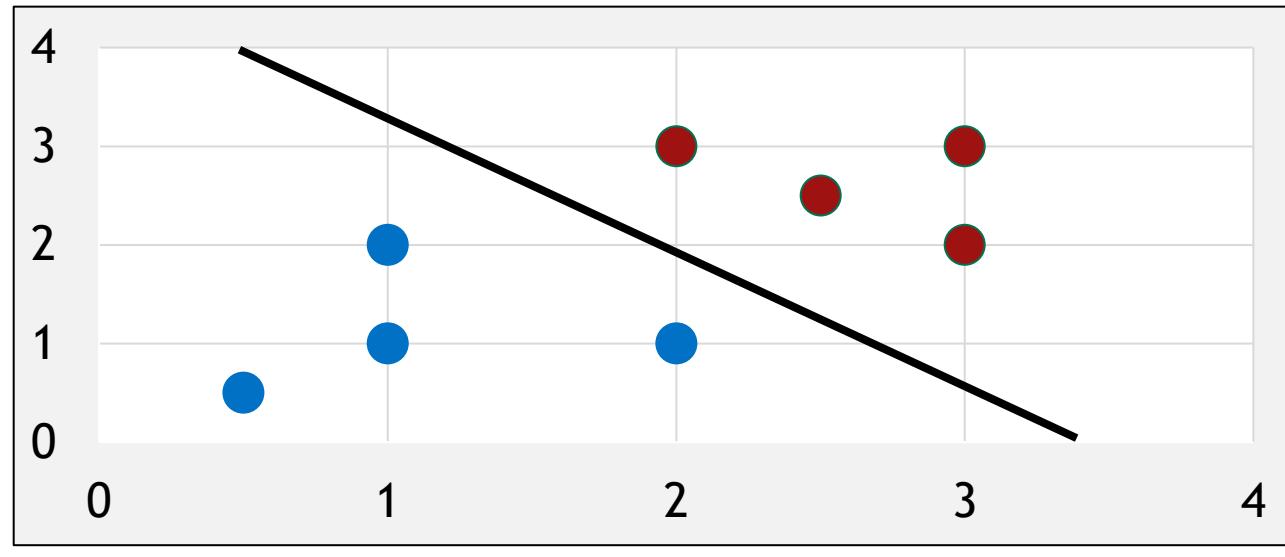
SOFTMAX 활성화 함수 : 다중 분류



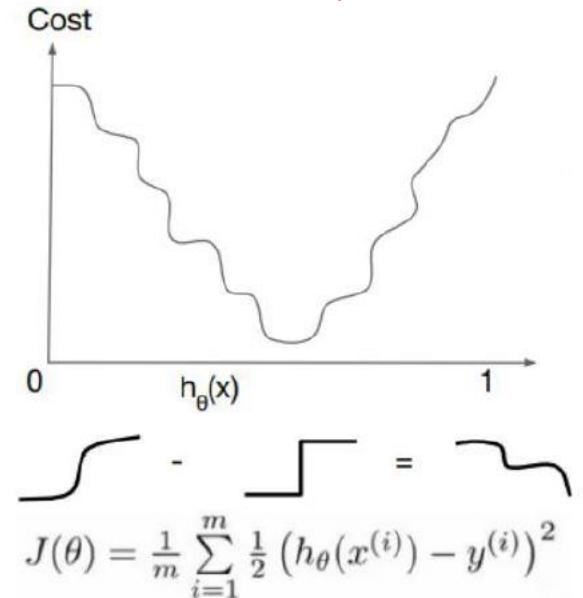
RMSE로 확률 예측?



RMSE로 확률 예측?



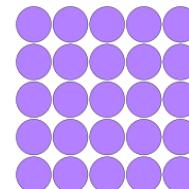
non convex function



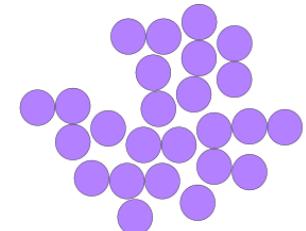
크로스 엔트로피 (CROSS ENTROPY)

- **엔트로피(ENTROPY)**

- 무질서 측도
- 값이 클수록 데이터가 혼재, 분류가 잘 안된 상태
- 값이 작을 수록 데이터가 잘 분류된 상태



Low Entropy

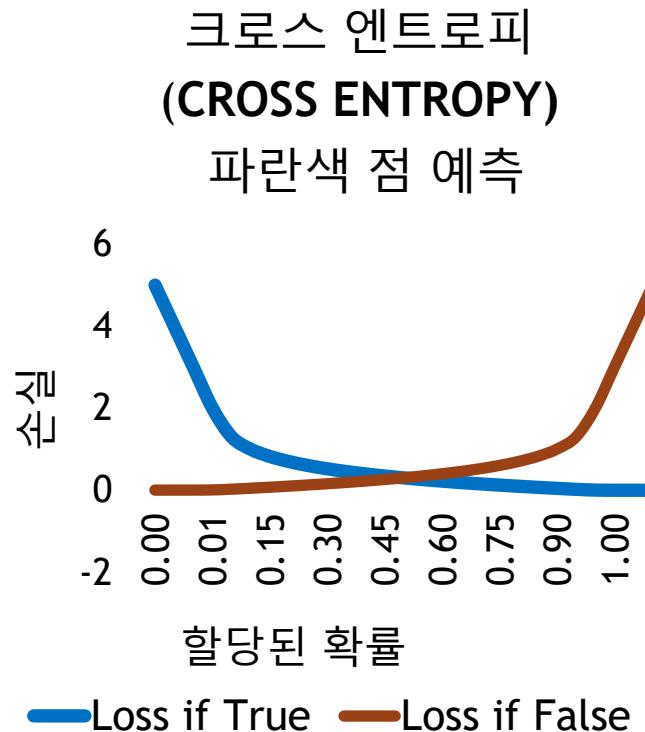
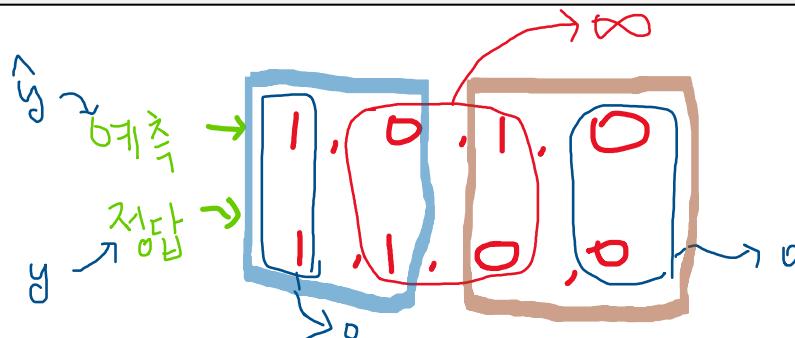
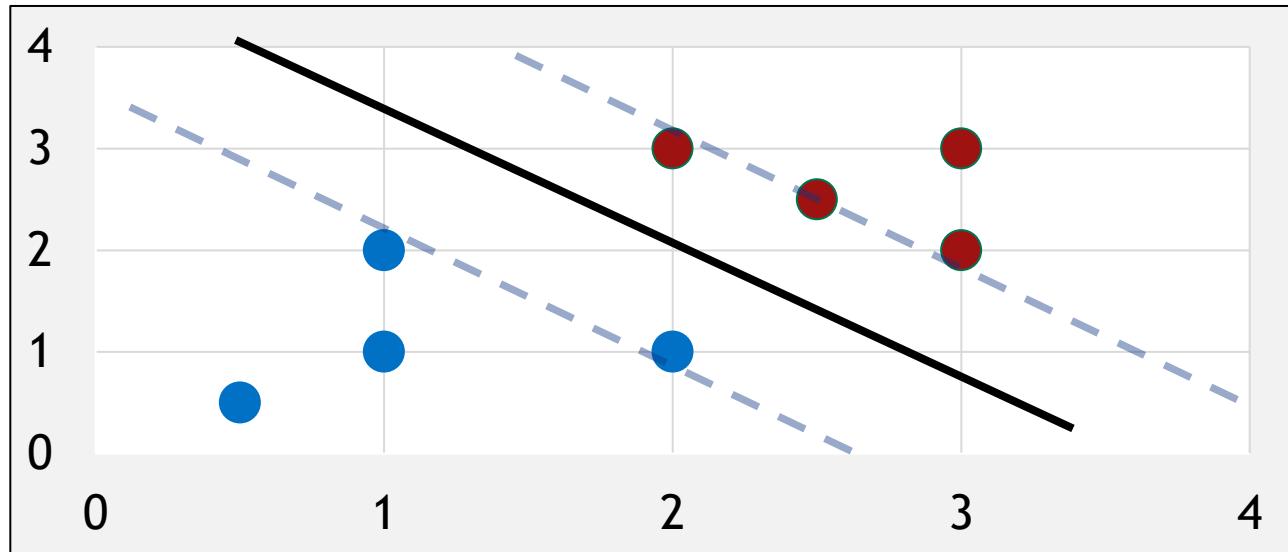


High Entropy

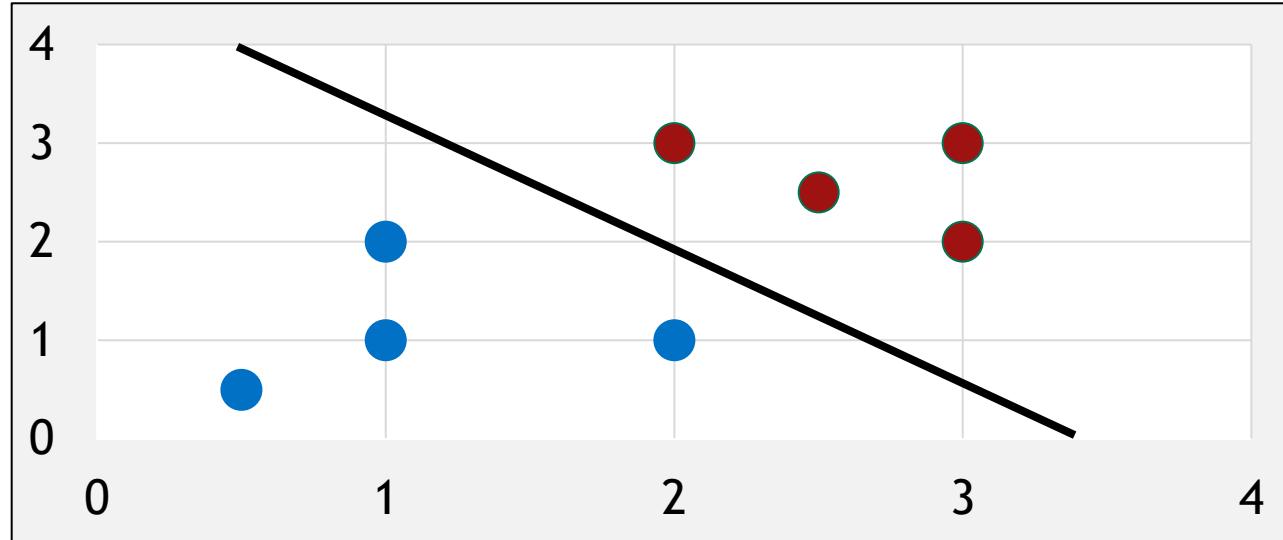
- **크로스 엔트로피(CROSS ENTROPY)**

- 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)



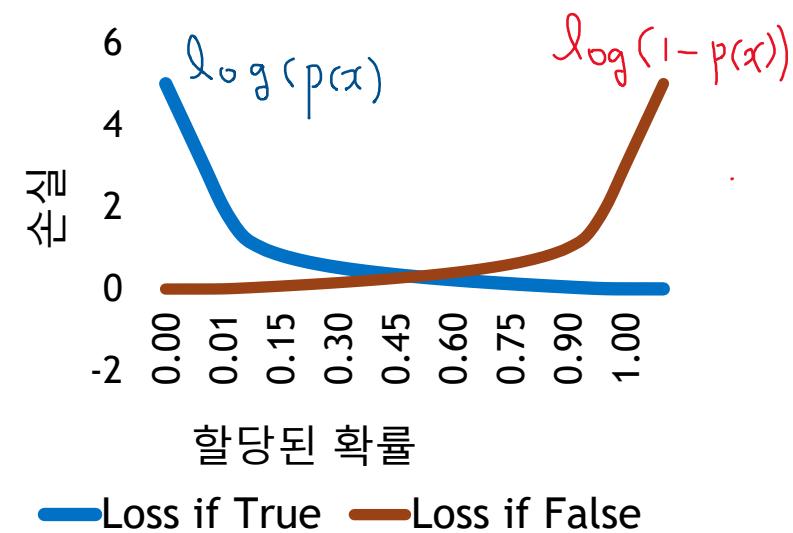
$$Loss = -(t(x) \cdot \log(p(x)) + (1 - t(x)) \cdot \log(1 - p(x)))$$

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

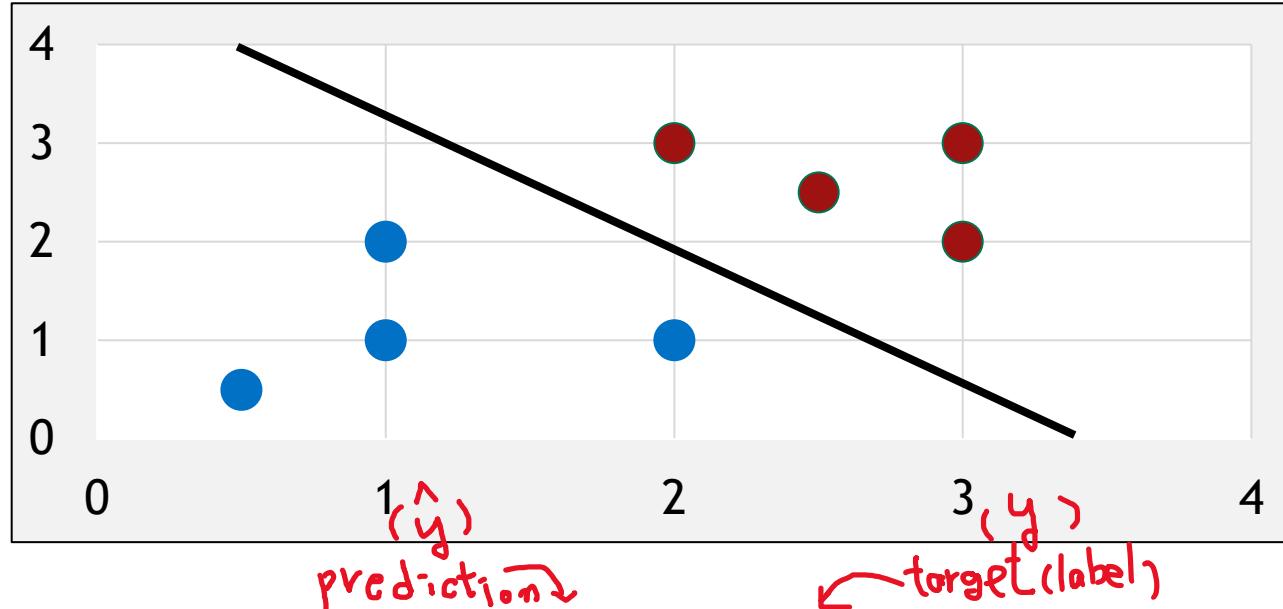
$p(x)$ = probability prediction of point x

$$\begin{aligned} * \log(1) &= 0 \\ \log(0) &= \infty \end{aligned}$$

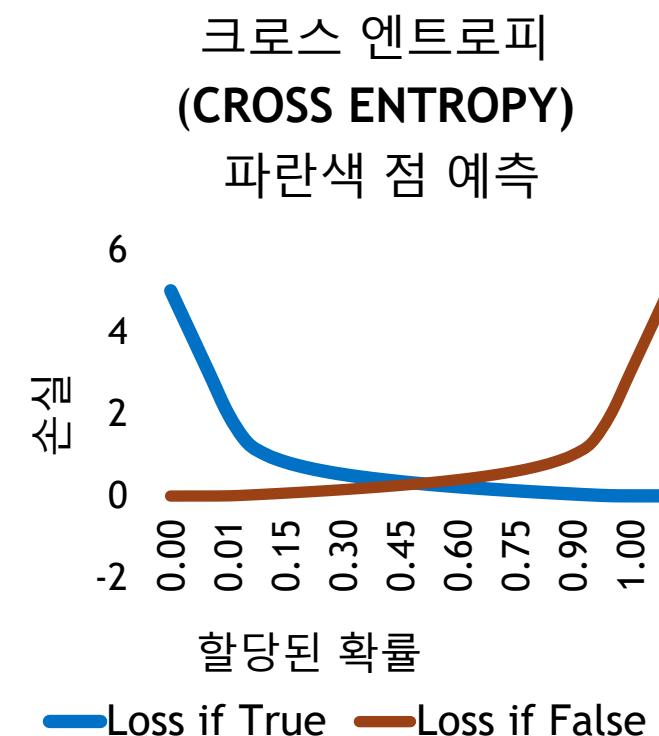
크로스 엔트로피
(CROSS ENTROPY)
파란색 점 예측



크로스 엔트로피 (CROSS ENTROPY)



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



크로스 엔트로피 (BINARY) CROSS ENTROPY

Linear Regression

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

가설 예측 값
 $\hat{y}(y_{\text{pred}})$

정답 값

(Binary) Classification

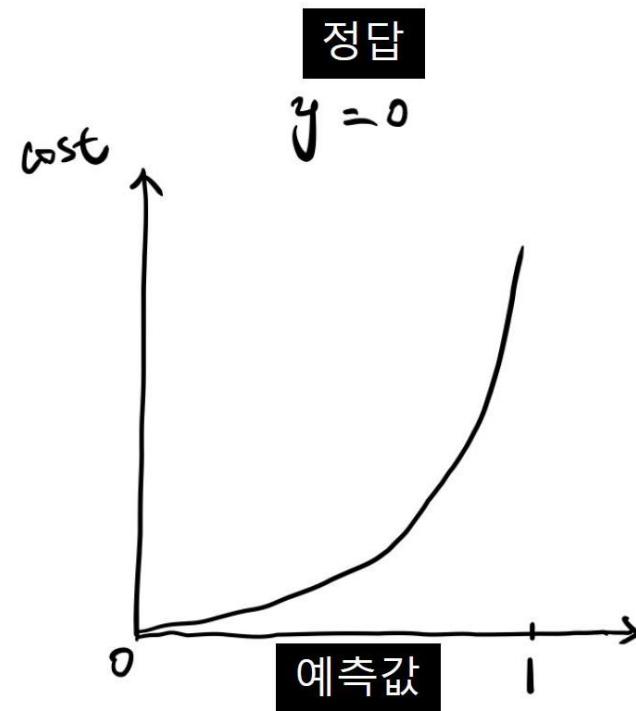
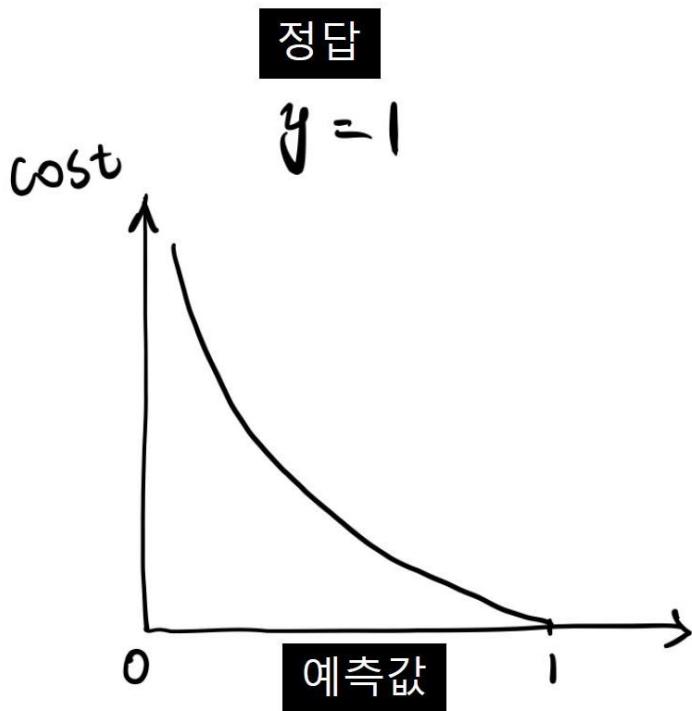
$$\text{cost} = -y \log(H(x)) - (1-y) \log(1-H(x))$$

가설 예측 값

정답 값

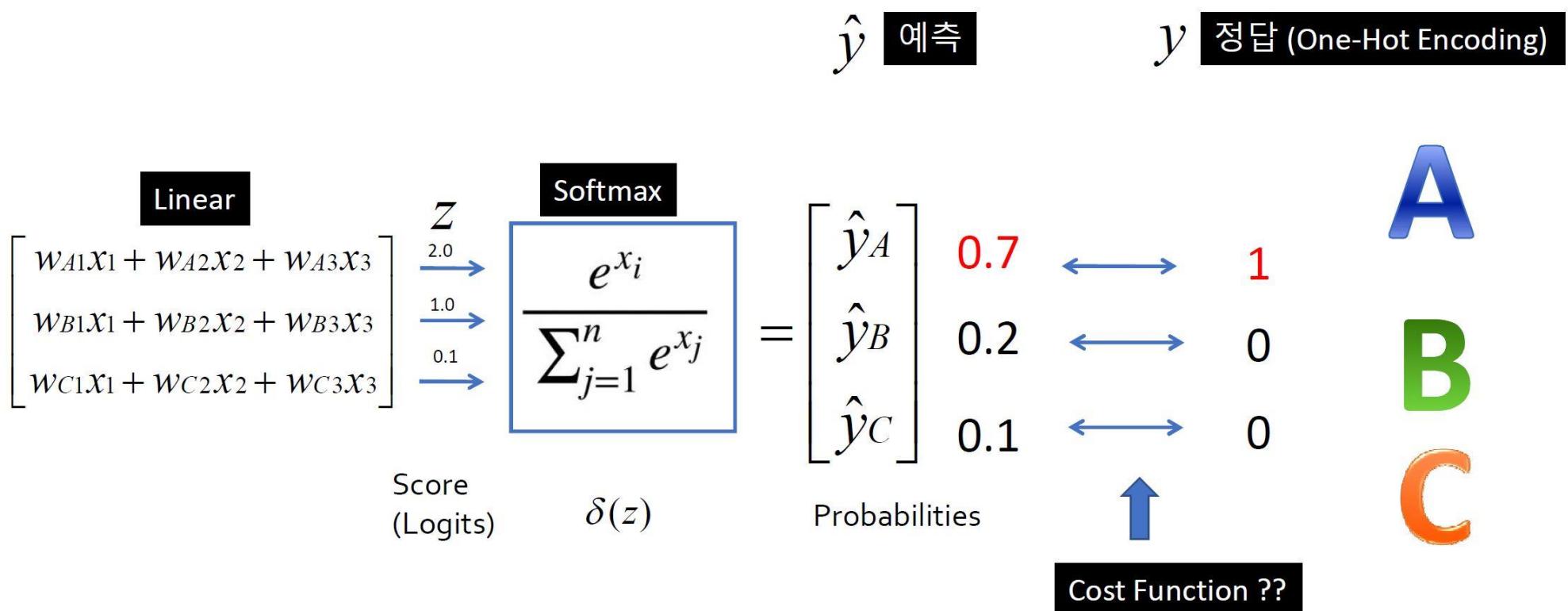
| $Y_{\text{actual/ target/ label}}$ | y_{pred} | Loss(cost) | Loss(cost) |
|------------------------------------|-------------------|------------|------------|
| 1 | 1 | $\log(1)$ | 0 |
| 0 | 0 | $\log(1)$ | 0 |
| 1 | 0 | $\log(0)$ | ∞ |
| 0 | 1 | $\log(0)$ | ∞ |

크로스 엔트로피 (BINARY) CROSS ENTROPY

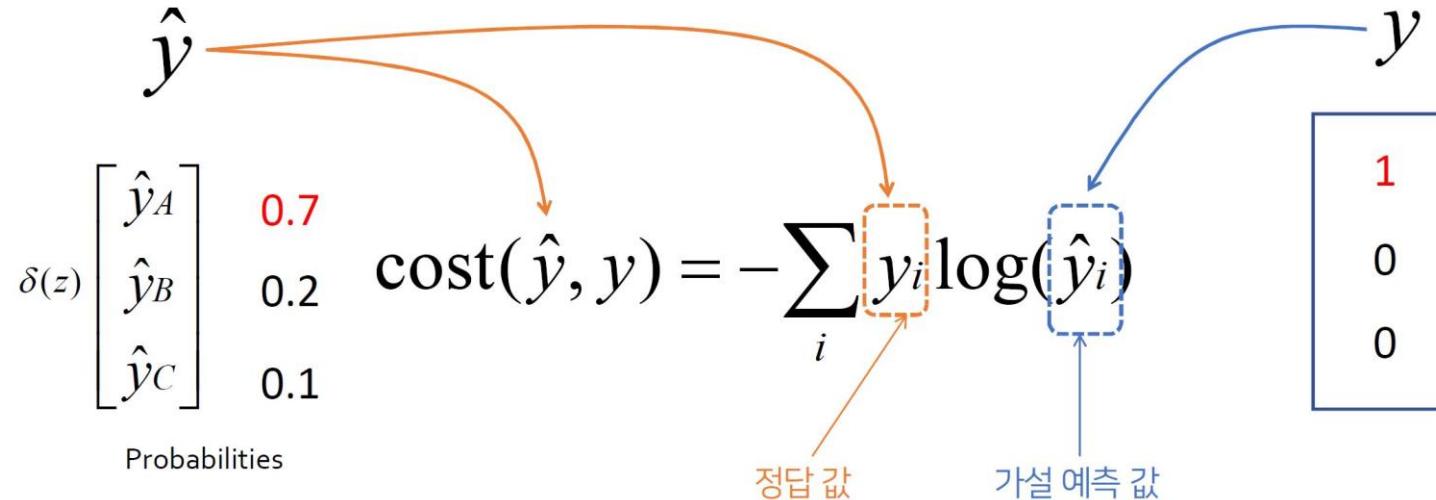


$$\text{cost}(H(x), y) = \begin{cases} -\log(H(x)) & \text{if } y = 1 \\ -\log(1 - H(x)) & \text{if } y = 0 \end{cases}$$

크로스 엔트로피 (CROSS ENTROPY)



크로스 엔트로피 (CROSS ENTROPY)



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

$$\begin{aligned} -\sum_{i=1}^2 y_i \log(\hat{y}_i) &= -(y_1 \log(\hat{y}_1) + (1-y_1) \log(1-\hat{y}_1)) \\ &= -(y \log(\hat{y}) + (1-y) \log(1-\hat{y})) \end{aligned}$$

<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 (SGD, RMSProp, Adam...) | - |
| 이진 분류 | sigmoid | binary_crossentropy | Gradient Descent (SGD, RMSProp, Adam...) | - |
| 다중 분류 | softmax | categorical_crossentropy | Gradient Descent (SGD, RMSProp, Adam...) | 범주형 one-hot-encoding |
| 다중 분류 | softmax | sparse_categorical_crossentropy | Gradient Descent (SGD, RMSProp, Adam...) | 정수형 |

실습 02 : 이진 분류

- 공부한 시간에 따른 합격(x), 불합격(y) 예측

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

x = np.array([2, 4, 6, 8, 10, 12, 14])
y = np.array([0, 0, 0, 1, 1, 1, 1])

model = Sequential()
model.add(Dense(1, input_dim=1, activation='sigmoid'))
```

```
# 교차 엔트로피 오차 함수를 이용하기 위해 'binary_crossentropy'로 설정합니다.

model.compile(optimizer='sgd', loss='binary_crossentropy')
model.fit(x, y, epochs=5000)

# 그래프로 확인해 봅니다.

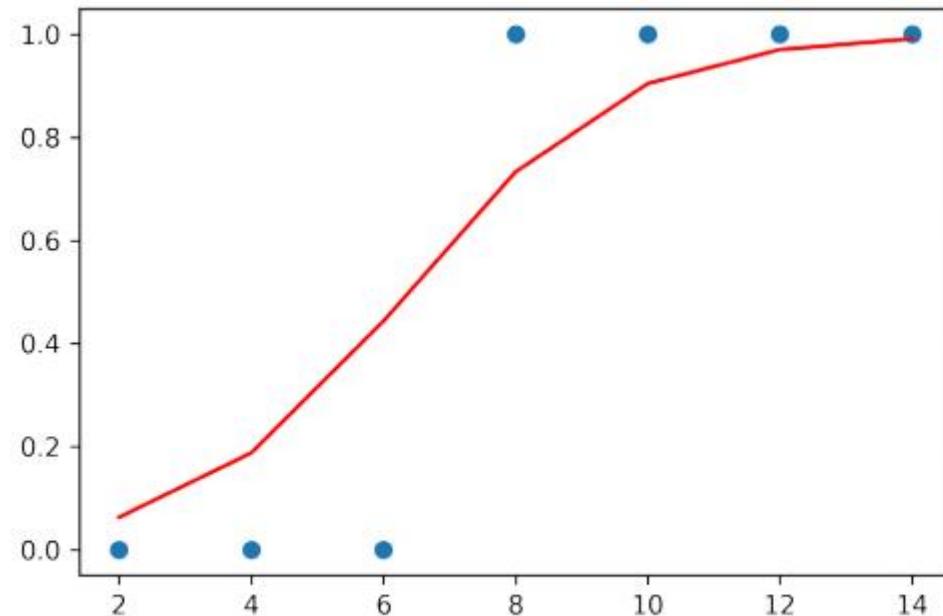
plt.scatter(x, y)
plt.plot(x, model.predict(x), 'r')
plt.show()

# 임의의 학습 시간을 집어넣어 합격 예상 확률을 예측해 보겠습니다.

hour = 7
prediction = model.predict([hour])
print("%.f시간을 공부할 경우, 합격 예상 확률은 %.01f%%입니다." % (hour,
prediction * 100))
```

실습 02 : 이진 분류

- 공부한 시간에 따른 합격(x), 불합격(y) 예측



7시간을 공부할 경우, 합격 예상 확률은 59.7%입니다.

실습 03 : 이진 분류

- 공부한 시간, 과외 횟수([x1, x2]...)에 따른 합격, 불합격(y) 예측

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# [공부시간, 과외횟수]
x = np.array([[2, 0], [4, 2], [6, 1], [8, 2], [10, 1], [12, 2], [14, 0]])
y = np.array([0, 0, 1, 1, 0, 1, 0])

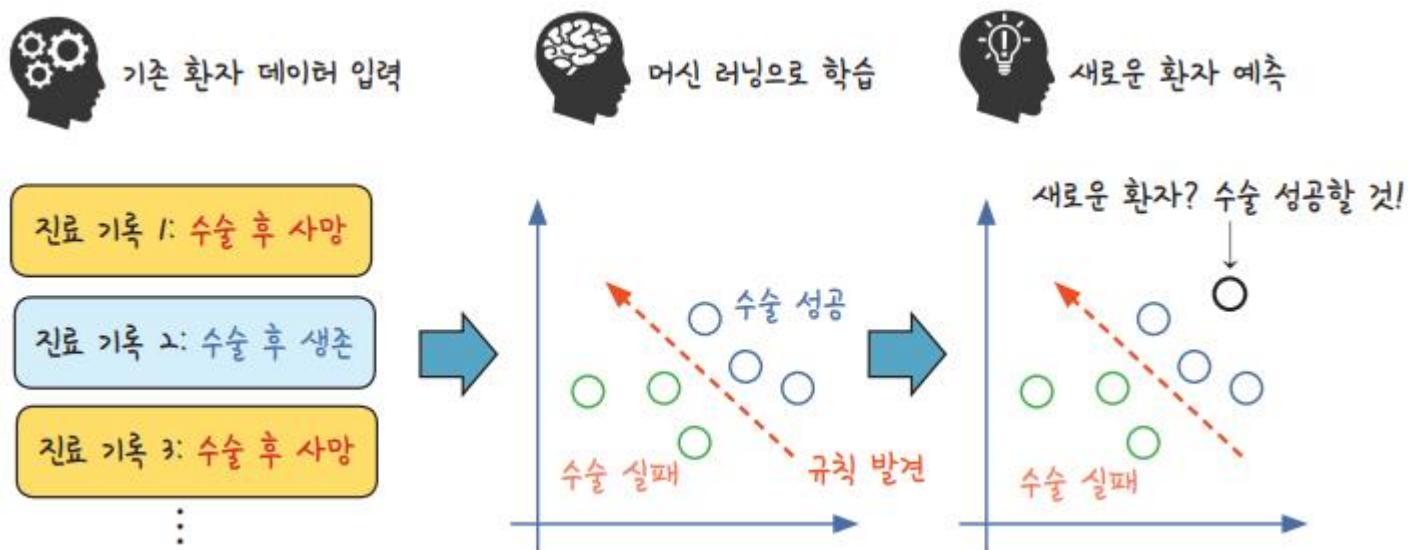
model = Sequential()
model.add(Dense(1, input_dim=2, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy')

model.fit(x, y, epochs=2000)
```

실습 04 : 이진 분류

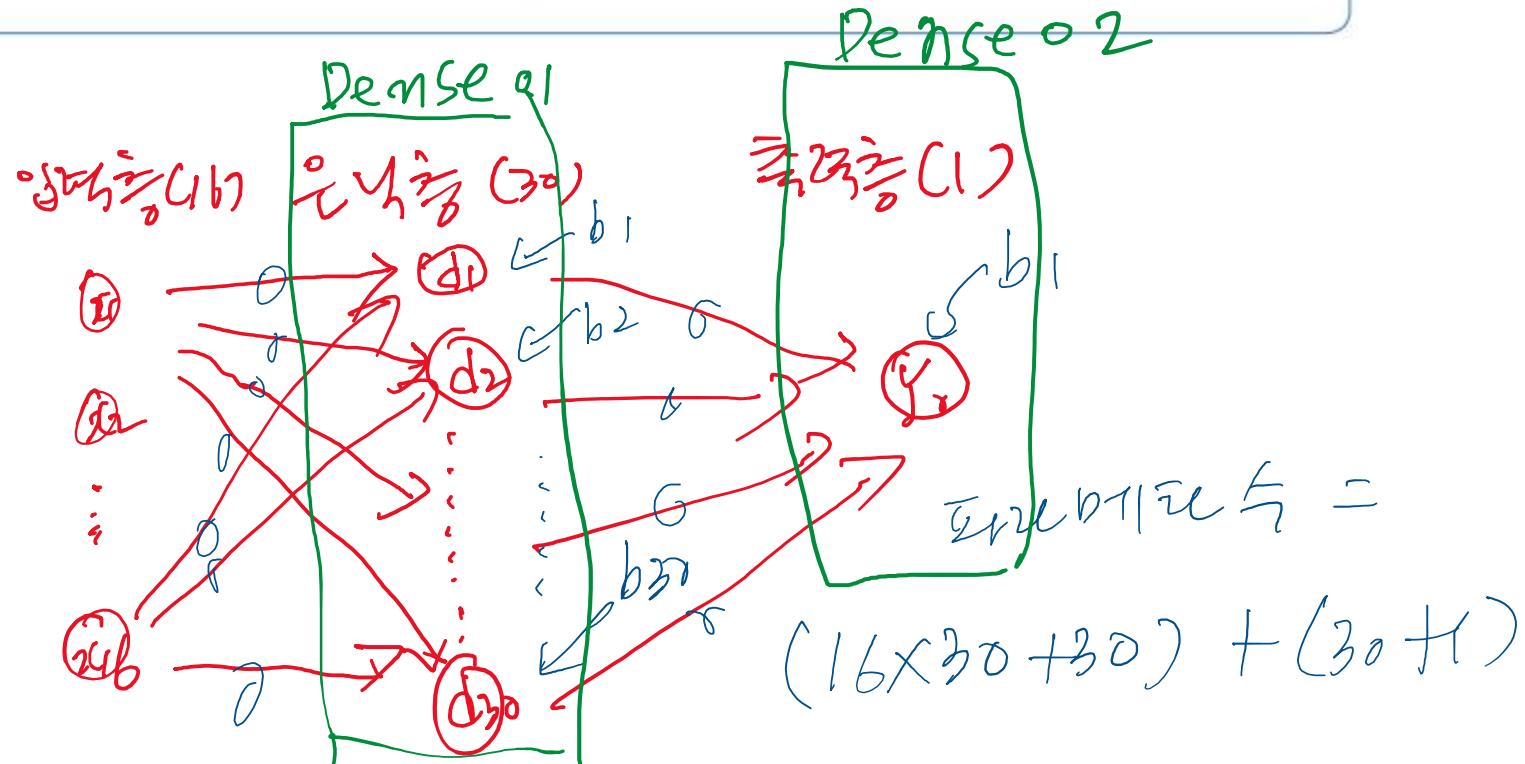
- 폐암 수술 환자 생존율 예측하기



실습 04 : 네트워크 구조

● 입력층, 은닉층, 출력층

```
model = Sequential()  
model.add(Dense(30, input_dim=16, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))
```



실습 05 : 다중 분류

- 아이리스(Iris) 품종 분류

iris setosa



petal sepal

iris versicolor



petal sepal

iris virginica



petal sepal

실습 05 : 데이터 셋 분할

- 데이터 셋

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|--------------|-------------|--------------|-------------|-------------|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

$X = df.iloc[:, 0:4]$

입력 데이터

$y = df.iloc[:, 4]$

결과

실습 05 : 특성(FEATURE)

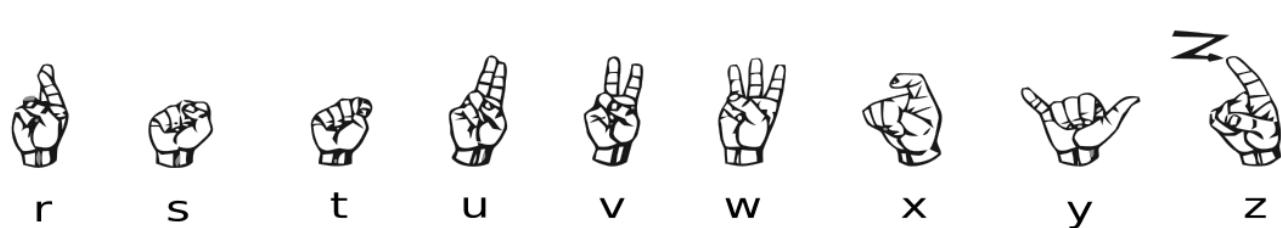
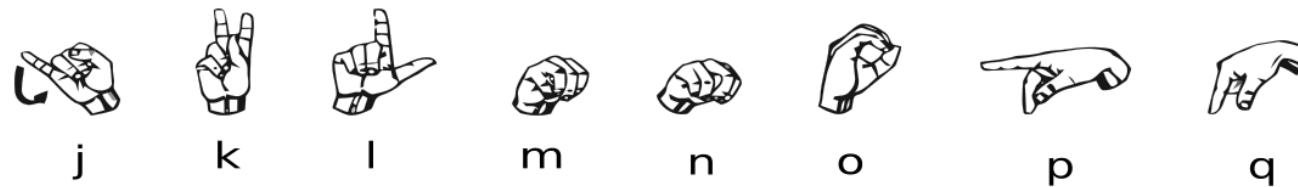
- 샘플 수: 150
- 속성 수: 4
 - 정보 1: 꽃받침 길이(sepal length, 단위: cm)
 - 정보 2: 꽃받침 너비(sepal width, 단위: cm)
 - 정보 3: 꽃잎 길이(petal length, 단위: cm)
 - 정보 4: 꽃잎 너비(petal width, 단위: cm)
- 클래스: Iris-setosa, Iris-versicolor, Iris-virginica

A complex network graph is displayed against a dark gray background. The graph consists of numerous small, semi-transparent white and light green circular nodes, connected by a dense web of thin, light gray lines representing edges. The nodes are scattered across the frame, with a higher density in the upper half and some isolated nodes in the lower half.

종합 정리

연습 문제

미국 수화 알파벳



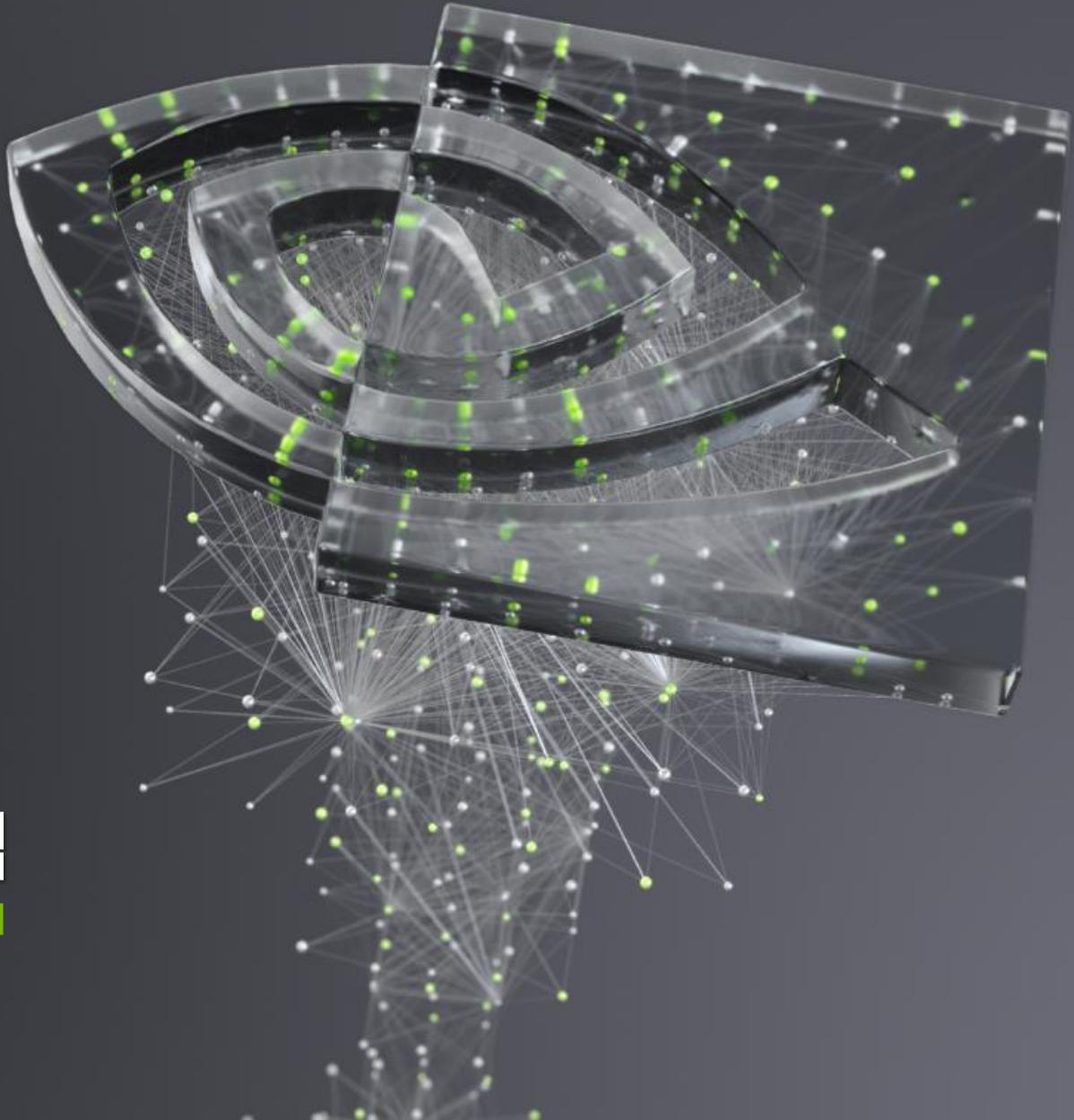


시작하겠습니다!



부록: 그래디언트 하강법

컴퓨터가 미적분에 대한 편법을 사용할 수 있도록 돋기



에러를 통한 학습

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

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

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

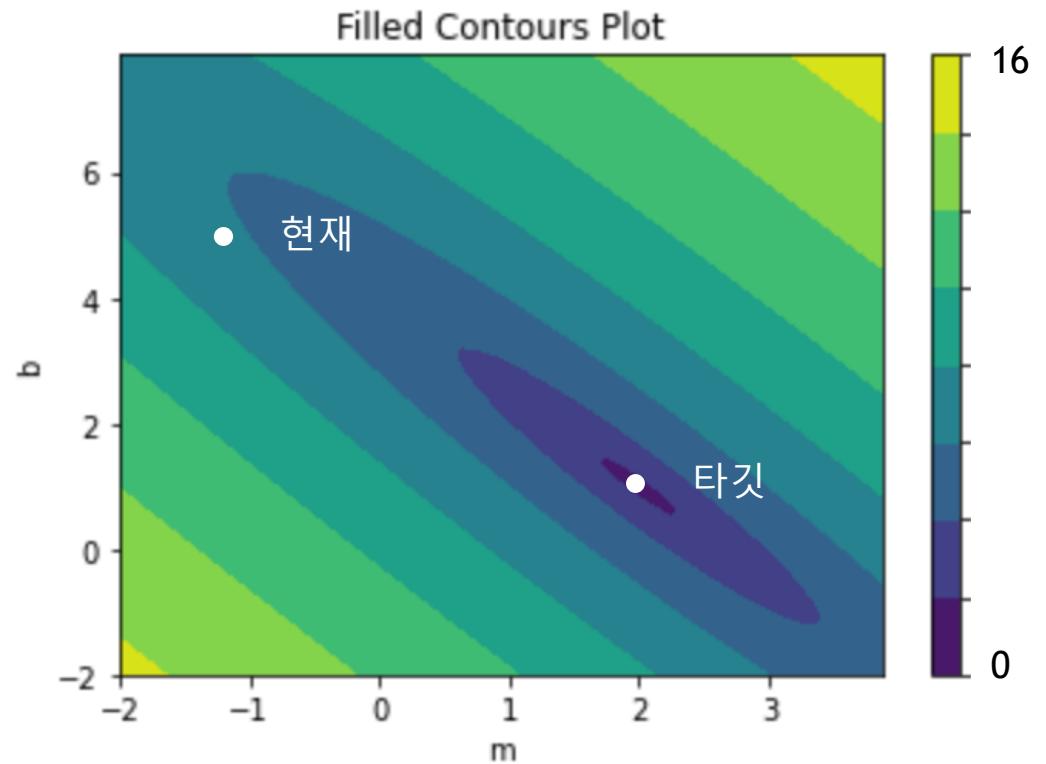
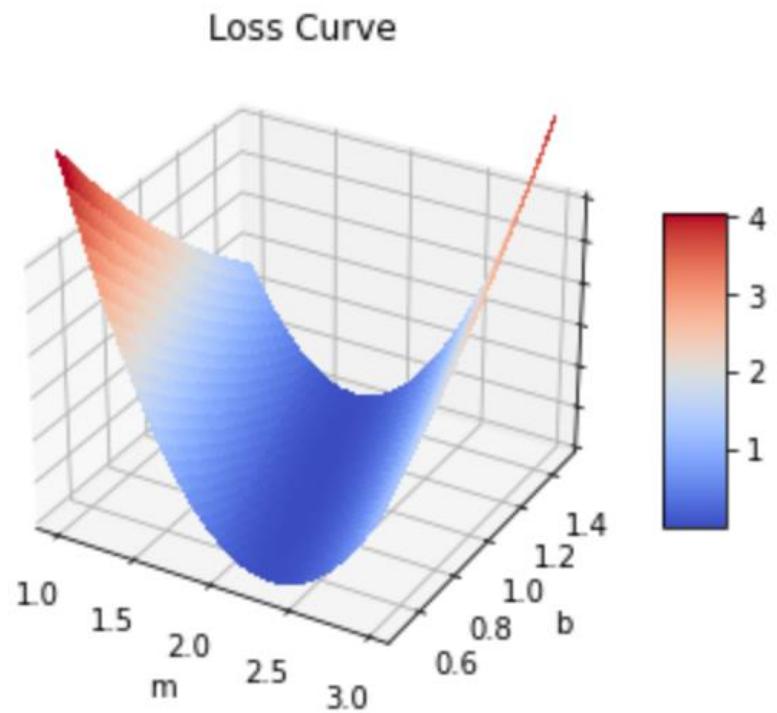
$$\frac{\partial MSE}{\partial m} = -7$$

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

$$\frac{\partial MSE}{\partial b} = -3$$

$m = -1$
 $b = 5$

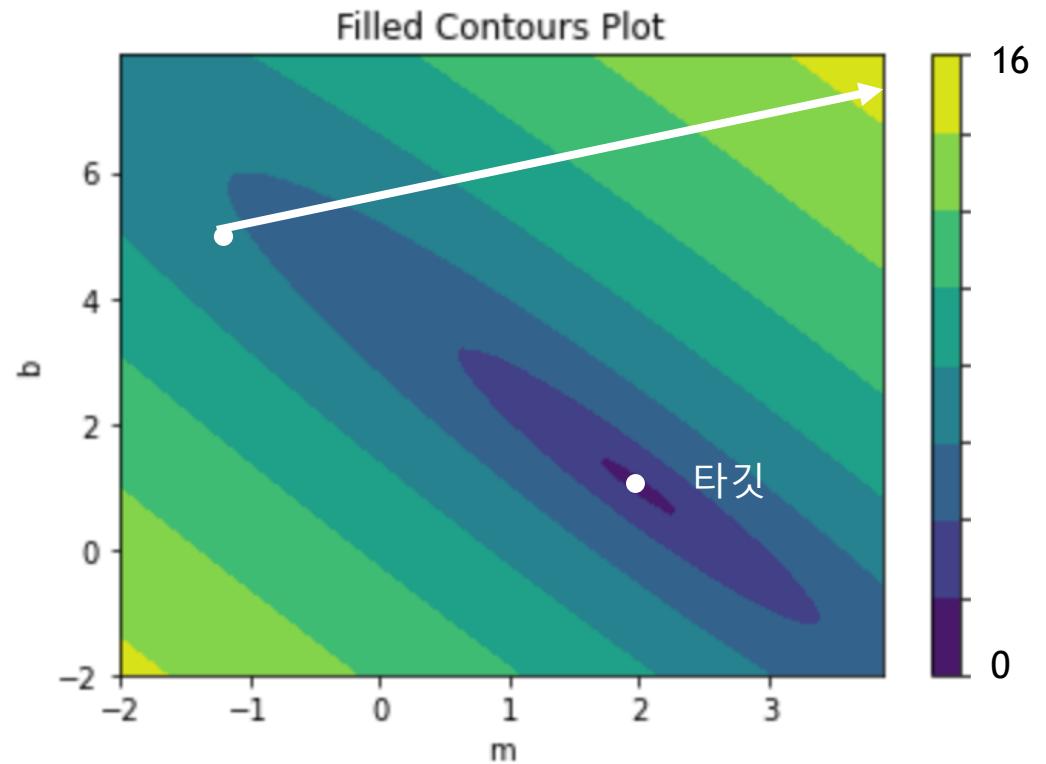
손실 곡선 (THE LOSS CURVE)



손실 곡선 (THE LOSS CURVE)

$$\frac{\partial MSE}{\partial m} = -7$$

$$\frac{\partial MSE}{\partial b} = -3$$

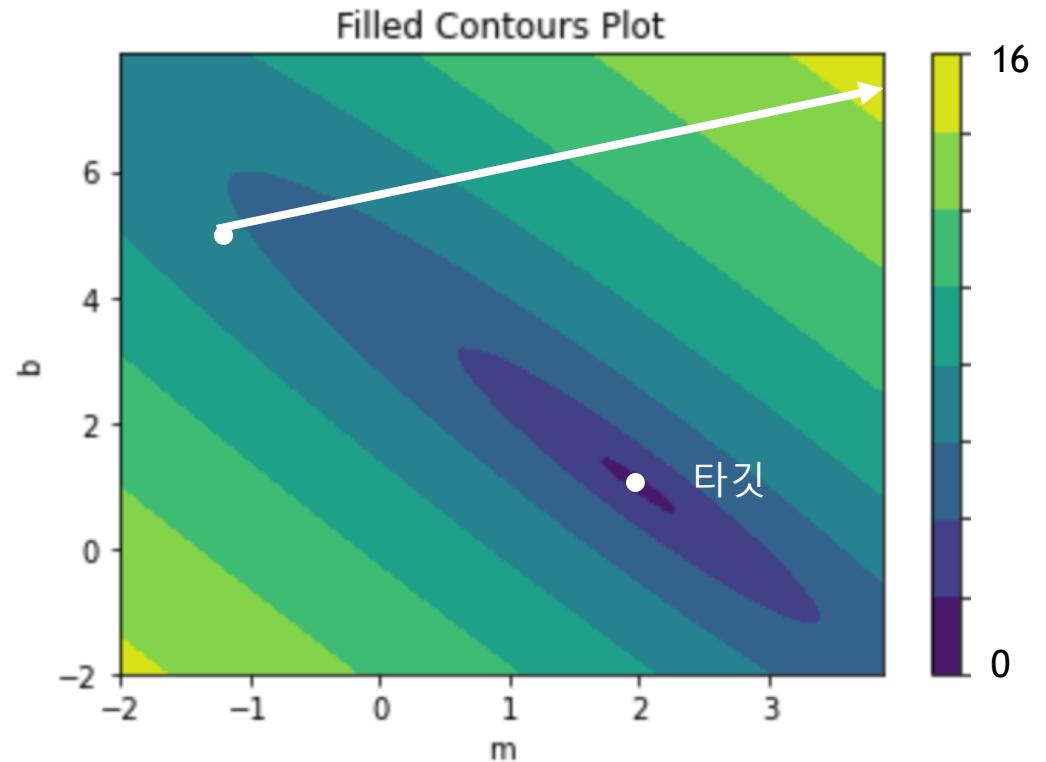


손실 곡선 (THE LOSS CURVE)

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

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

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



손실 곡선 (THE LOSS CURVE)

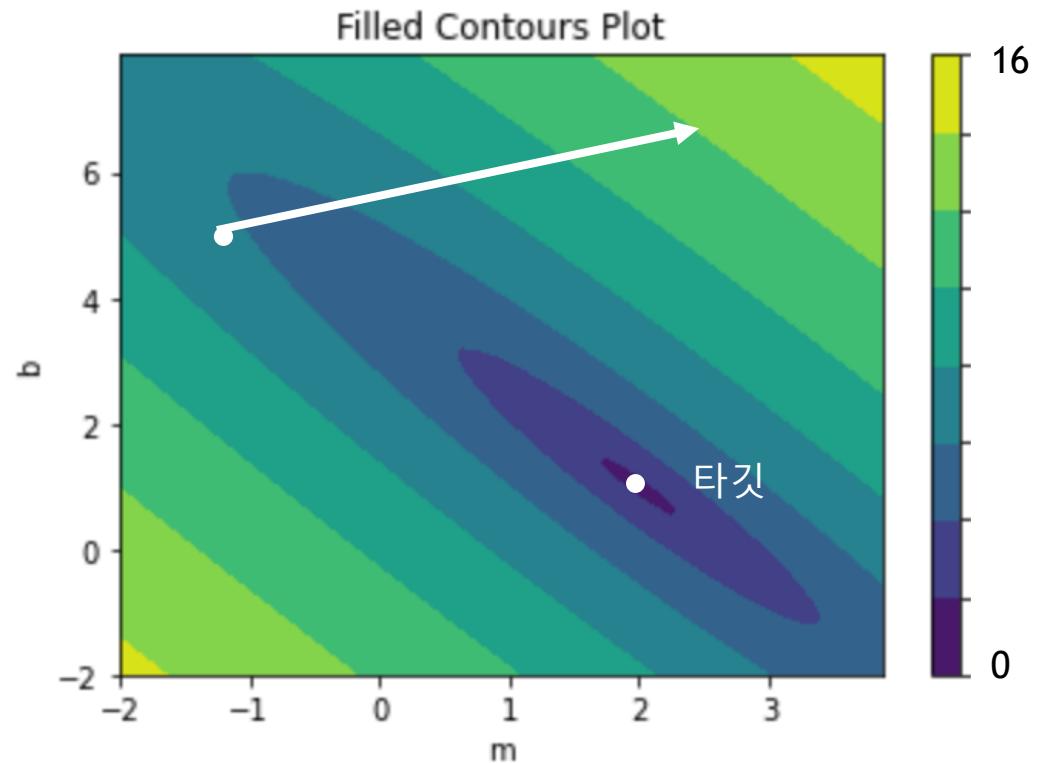
$$\frac{\partial MSE}{\partial m} = -7$$

$$\frac{\partial MSE}{\partial b} = -3$$

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

$$\lambda = .5$$

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



손실 곡선 (THE LOSS CURVE)

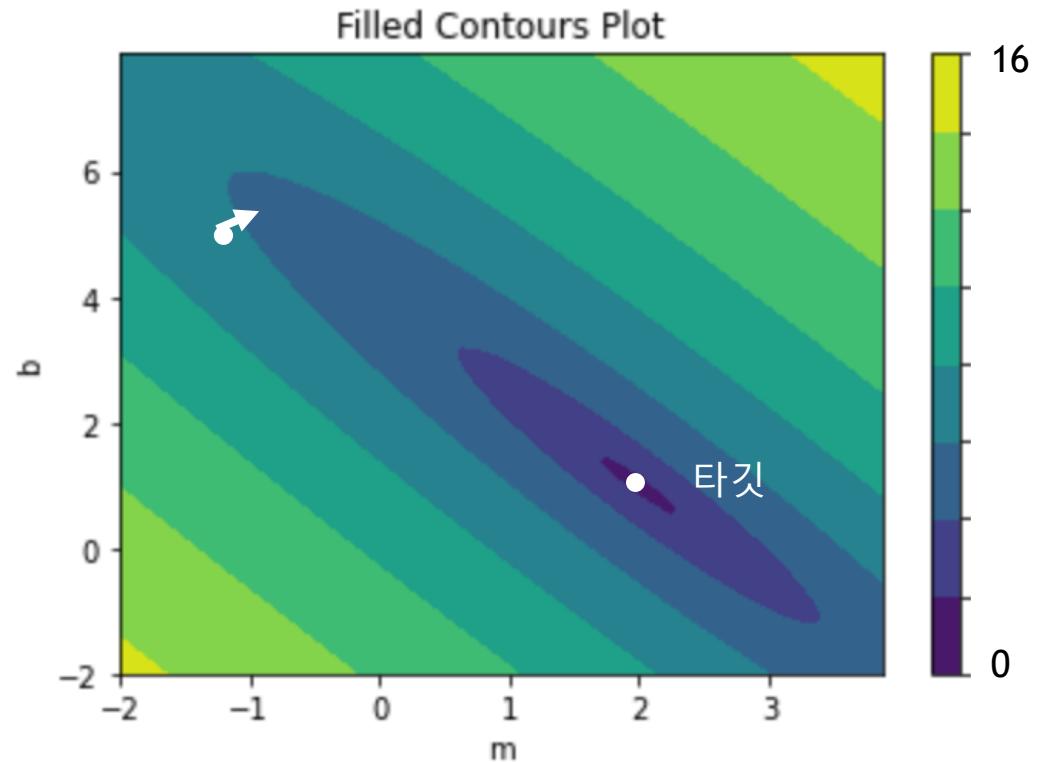
$$\frac{\partial MSE}{\partial m} = -7$$

$$\frac{\partial MSE}{\partial b} = -3$$

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

$$\lambda = .005$$

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

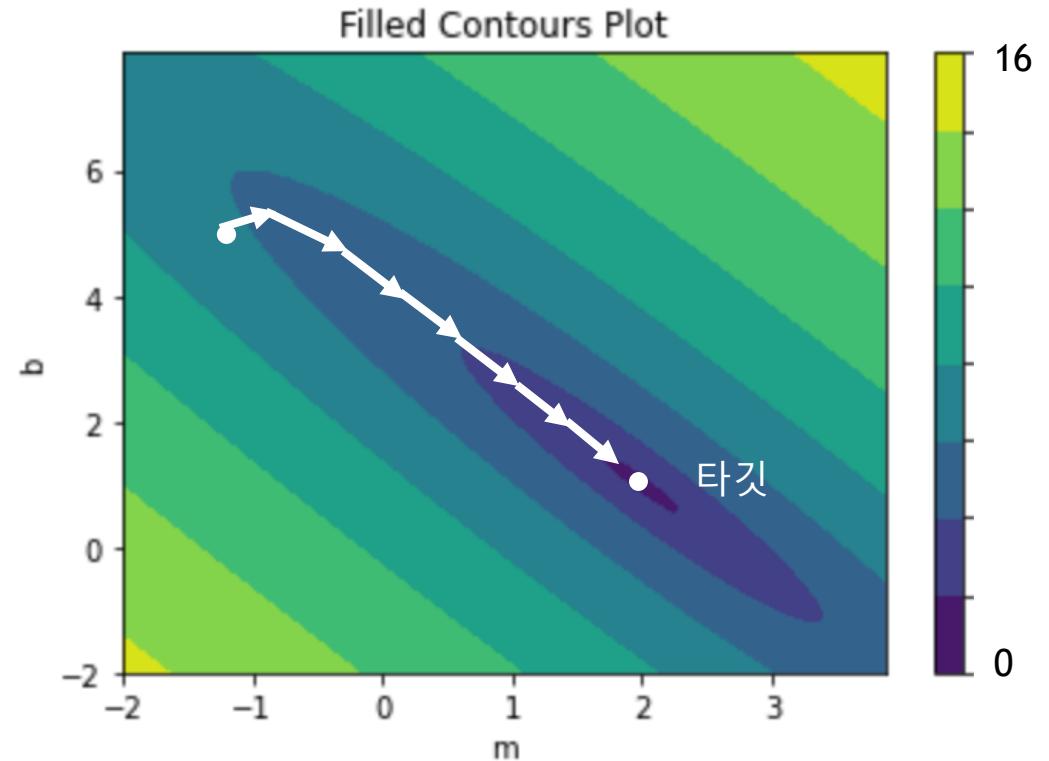


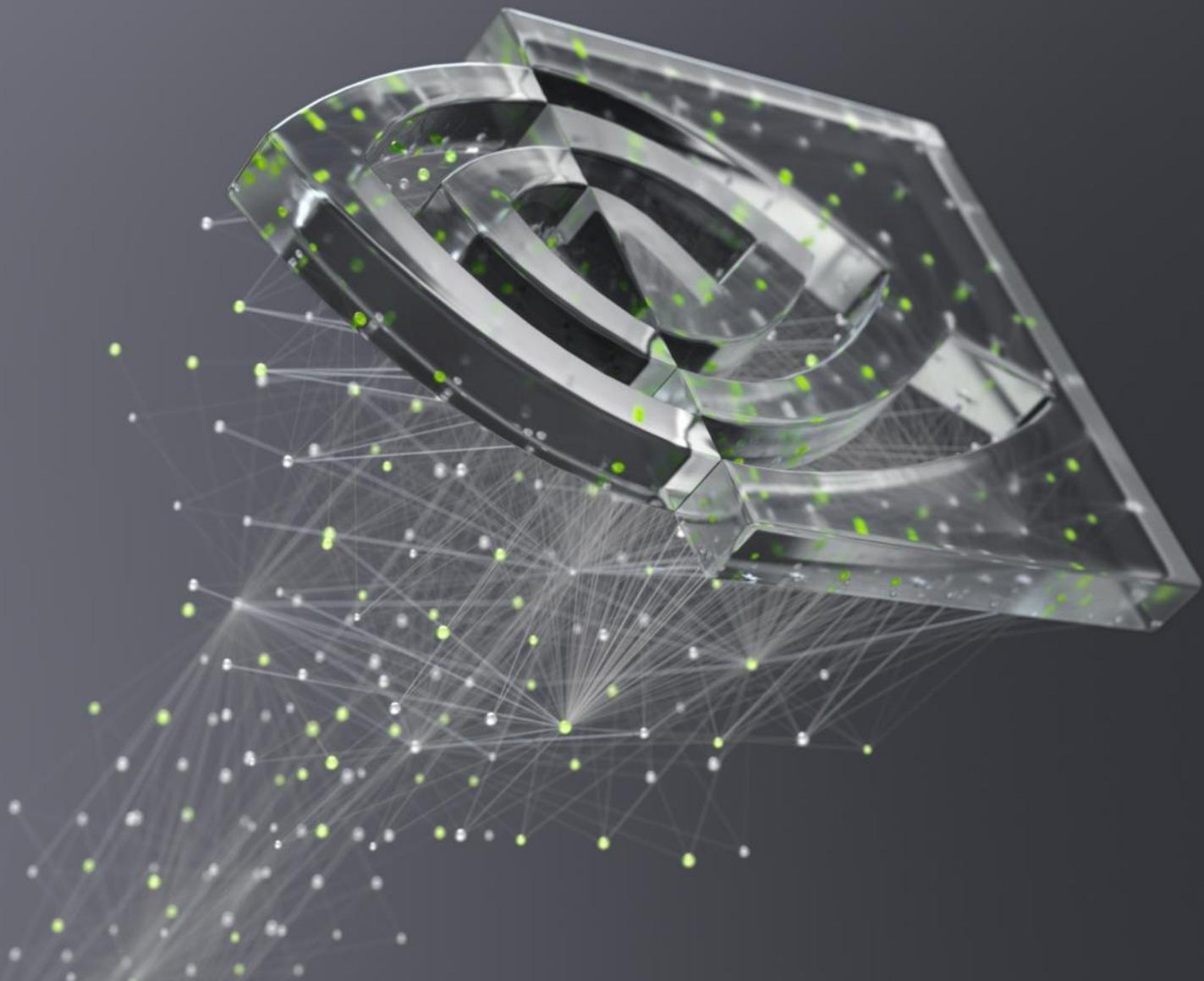
손실 곡선 (THE LOSS CURVE)

$$\lambda = .1$$

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

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





DEEP
LEARNING
INSTITUTE