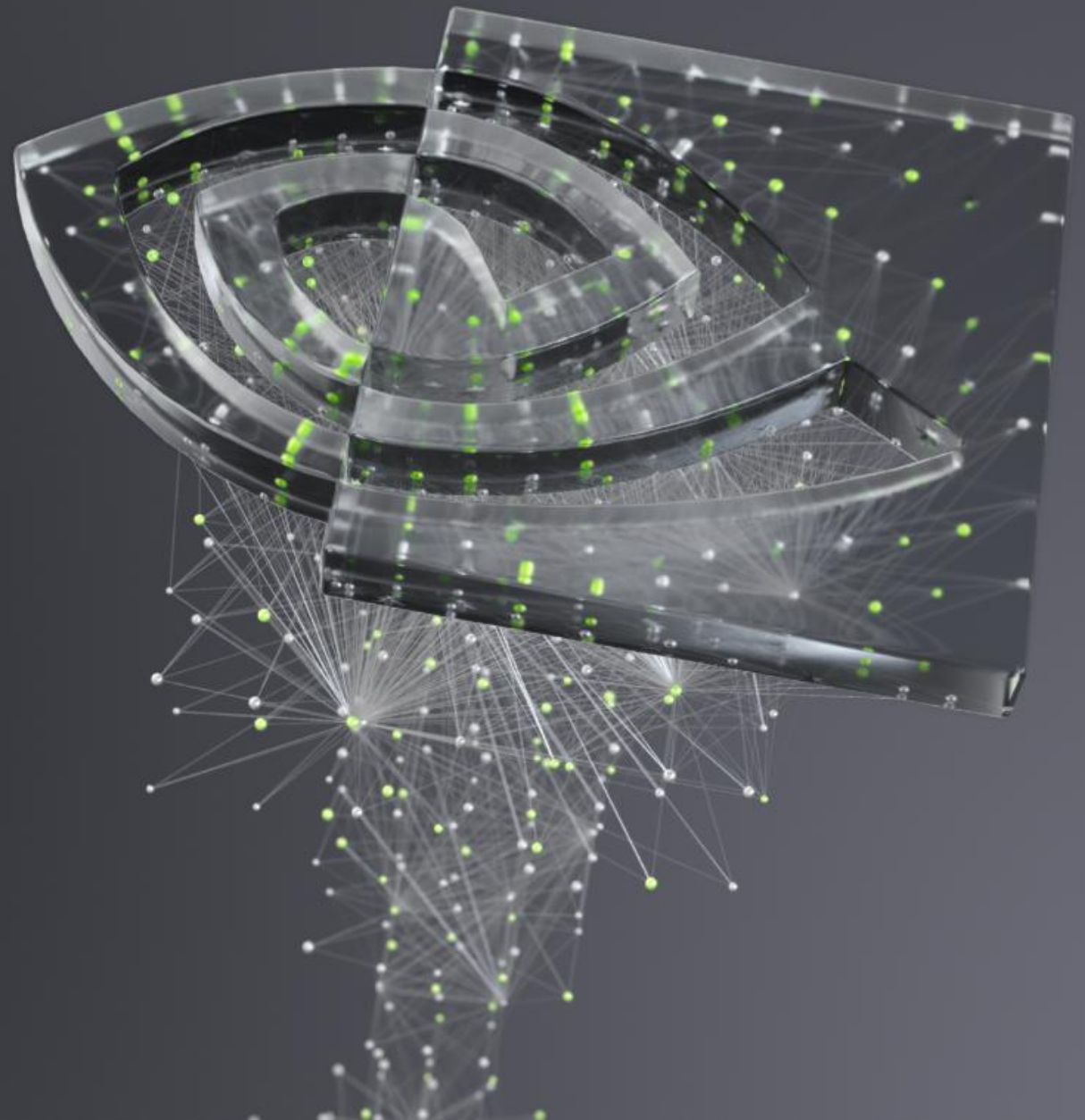
NVIDIA | DEEP LEARNING INSTITUTE

딥러닝의 기초

2부: 뉴럴 네트워크의 트레이닝 방식



목차

목차 – 2부

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

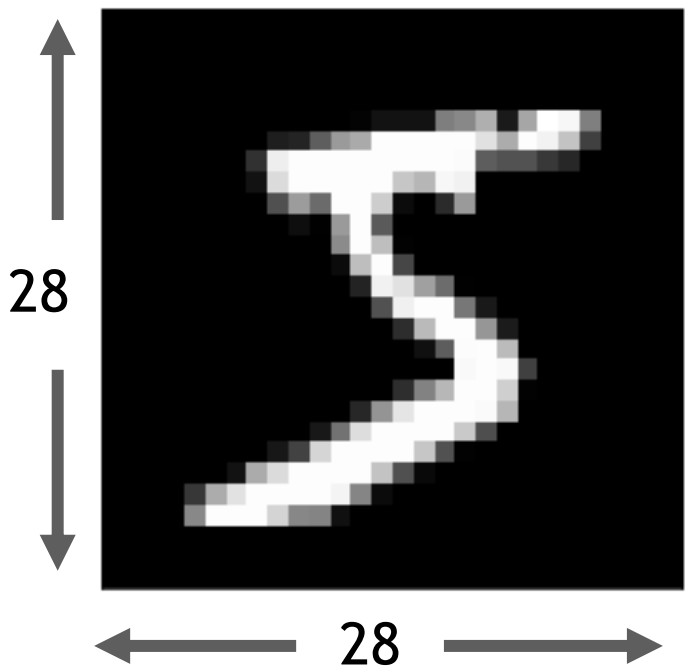
핸즈온 요약

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...]

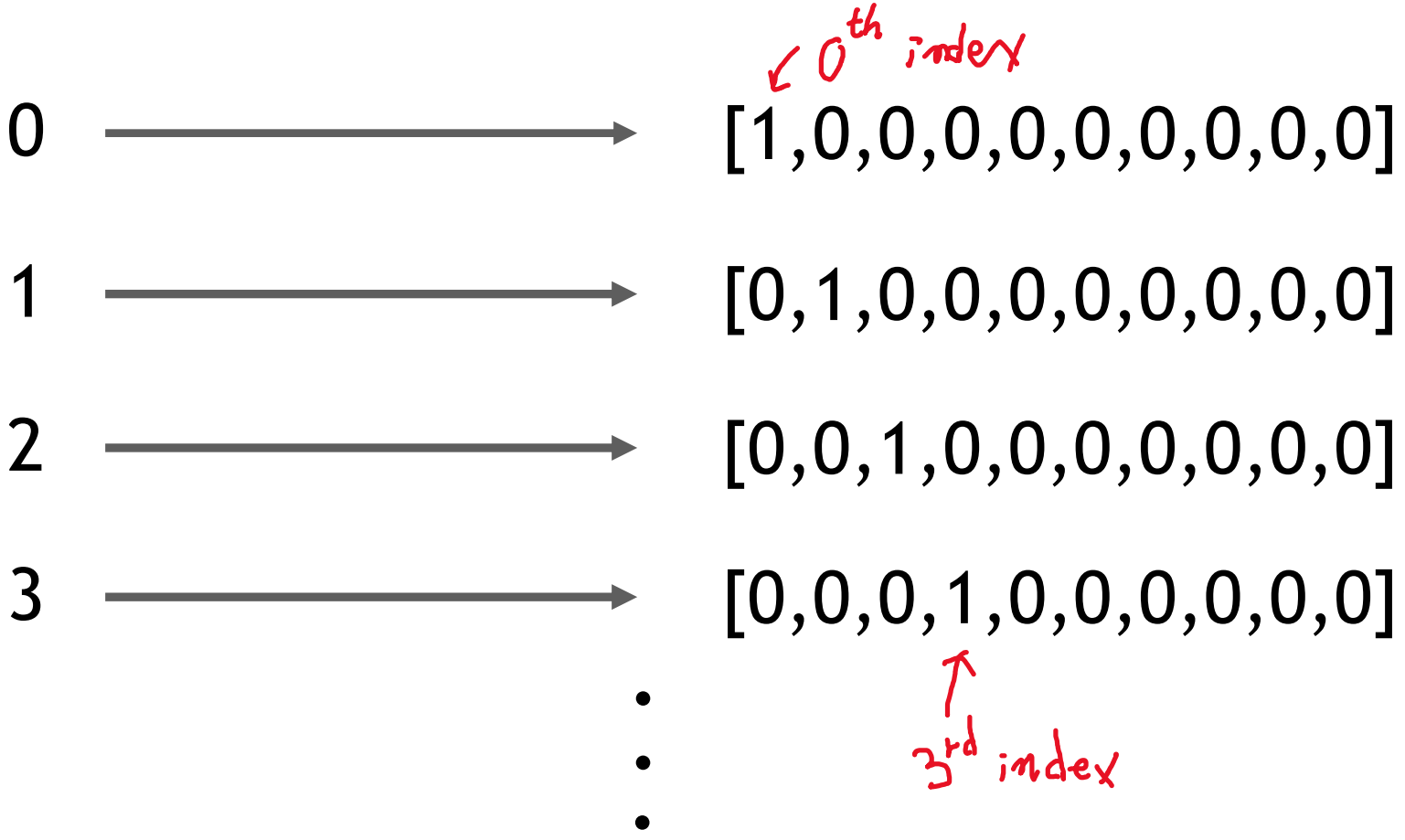
÷ 255 (정규화)

↓

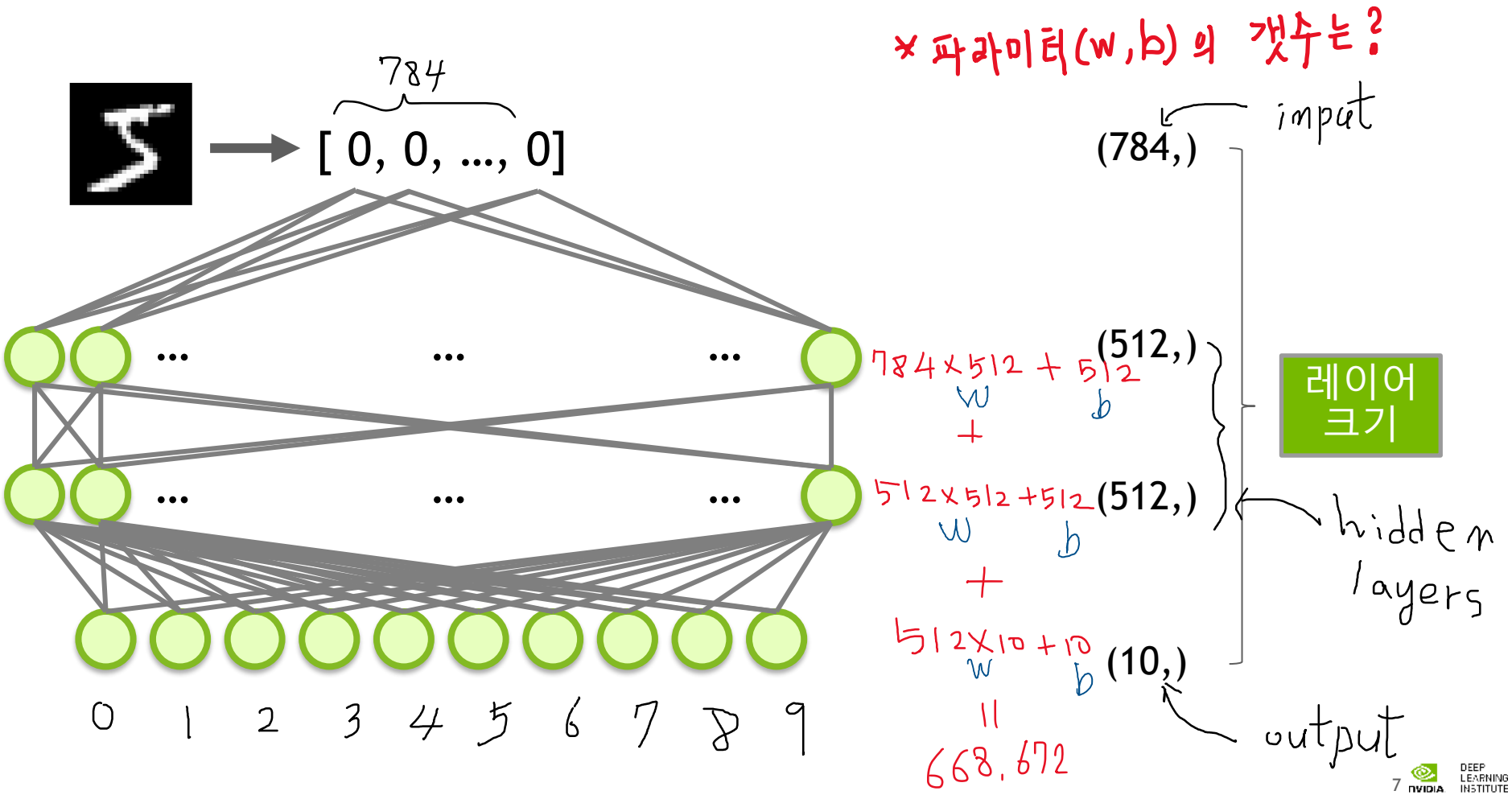
[0, 0, 0, 1, 0.3 ...]

데이터 준비 (Data preparation)

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

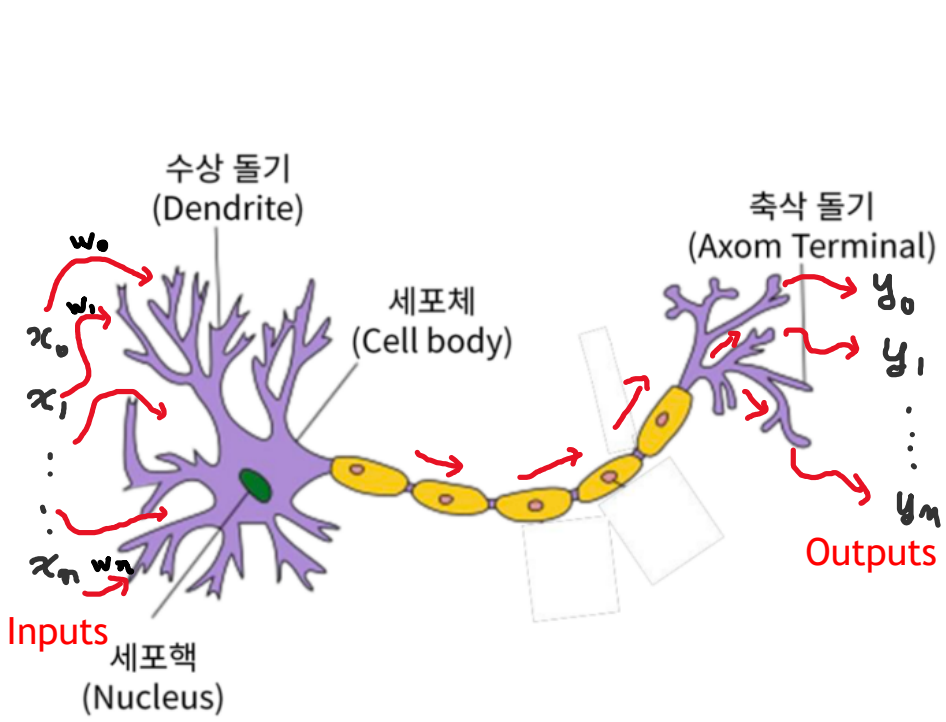


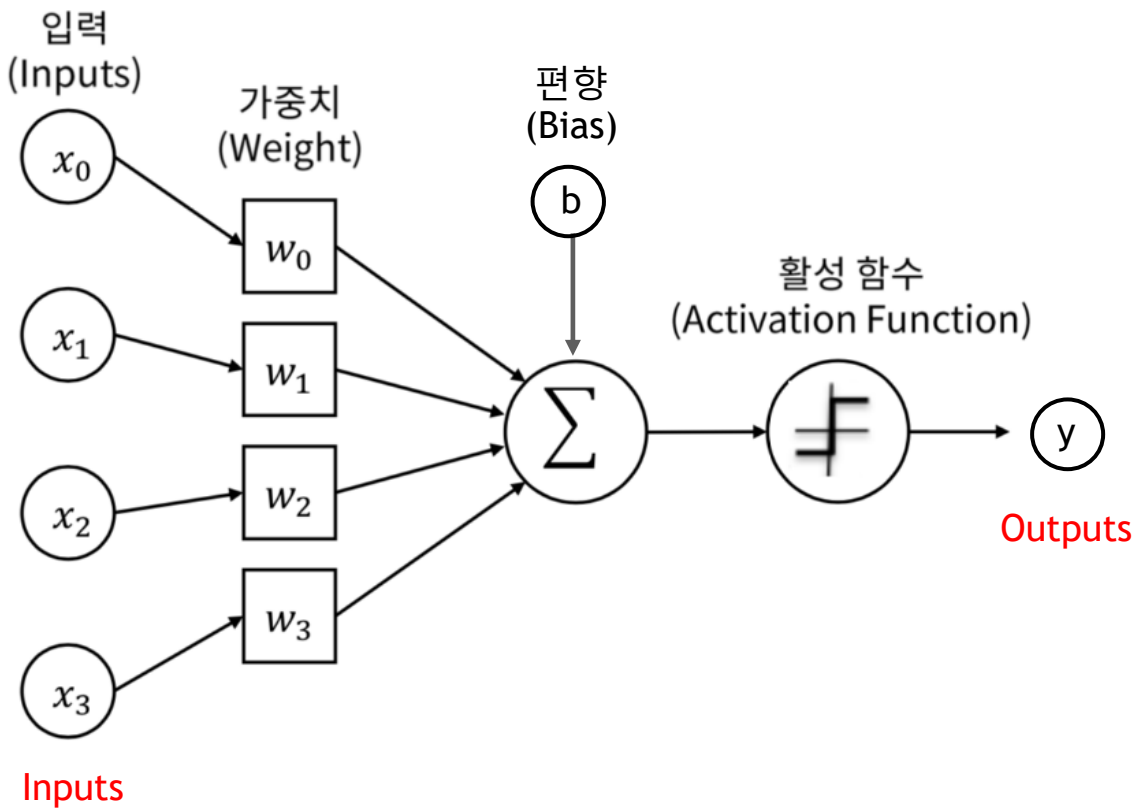
트레이닝되지 않은 모델


* 파라미터(w, b)의 갯수는?
784
[0, 0, ..., 0]
input
(784,)
(512,)
784x512 + 512
w
b
+
512x512+512
(512,)
w
b
+
512x10 + 10
w
b
(10,)
output
레이어 크기
hidden layers
11
668,672
0 1 2 3 4 5 6 7 8 9

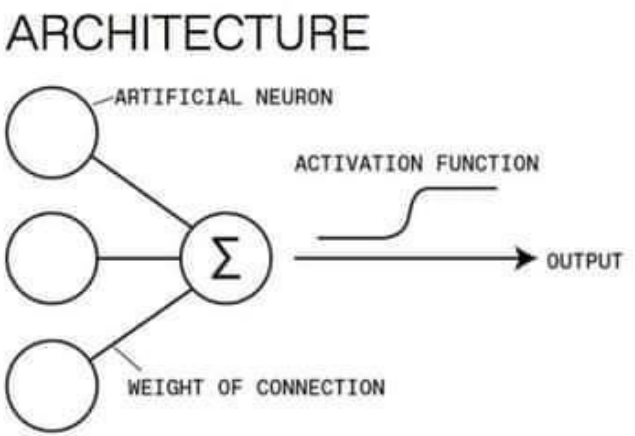

더욱 단순한 모델

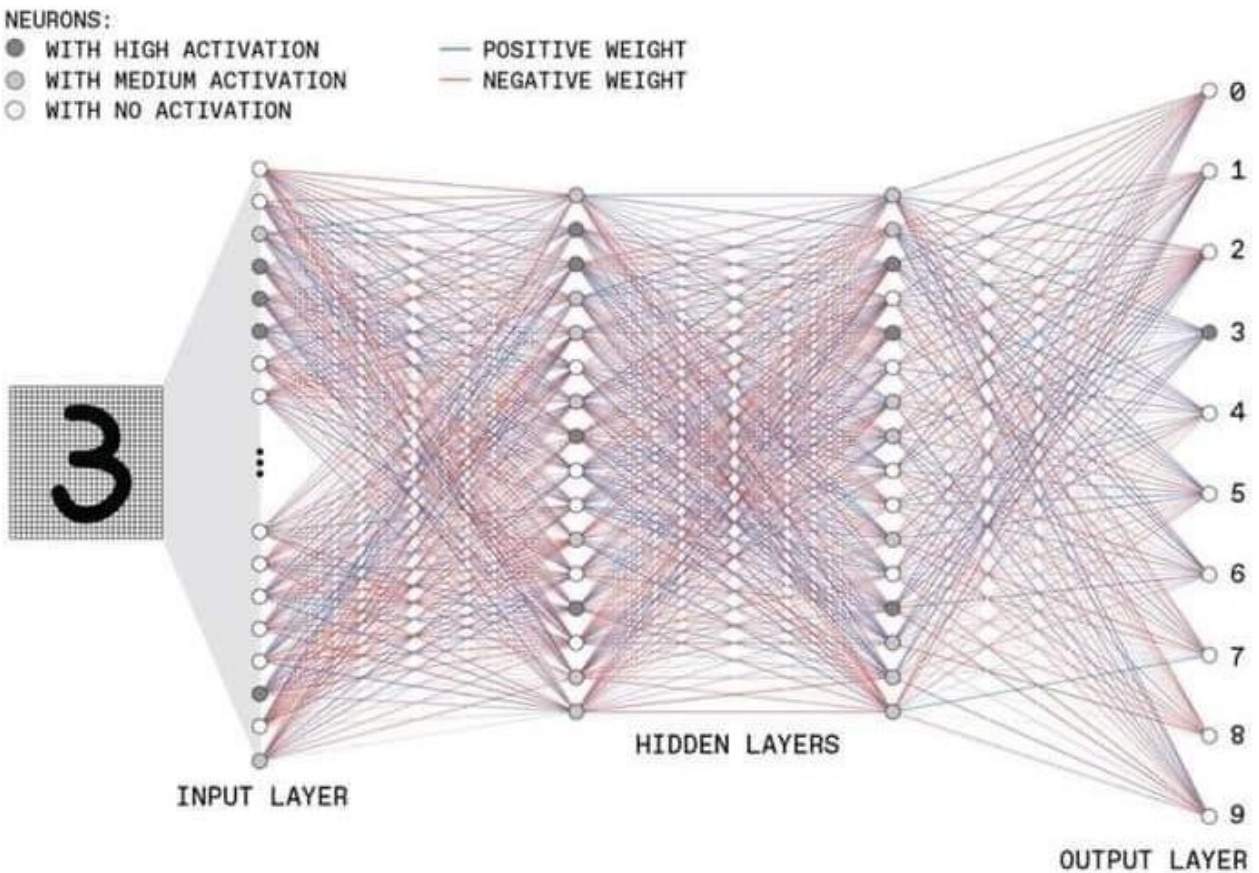
신경망(neuron) 구조





신경망(Neuron) 구조

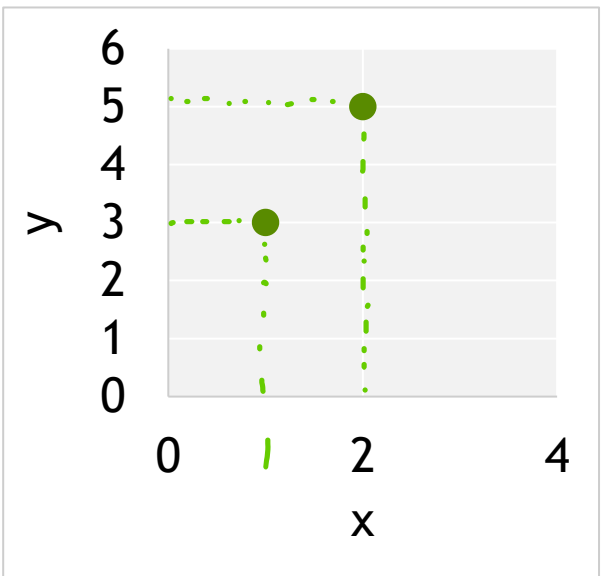


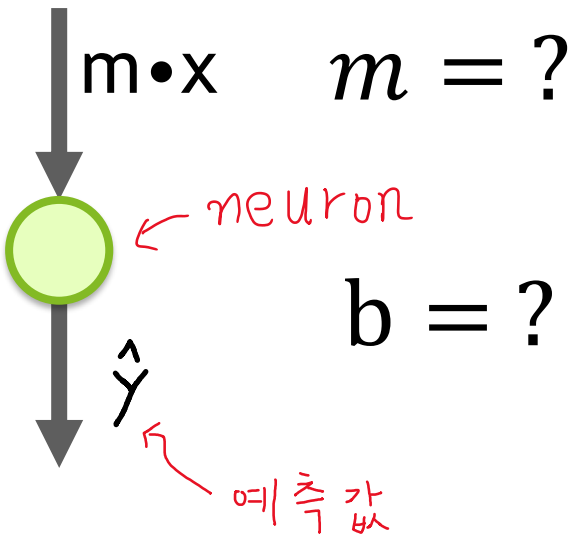


더욱 단순한 모델

$$y = mx + b$$

x	y
1	3
2	5

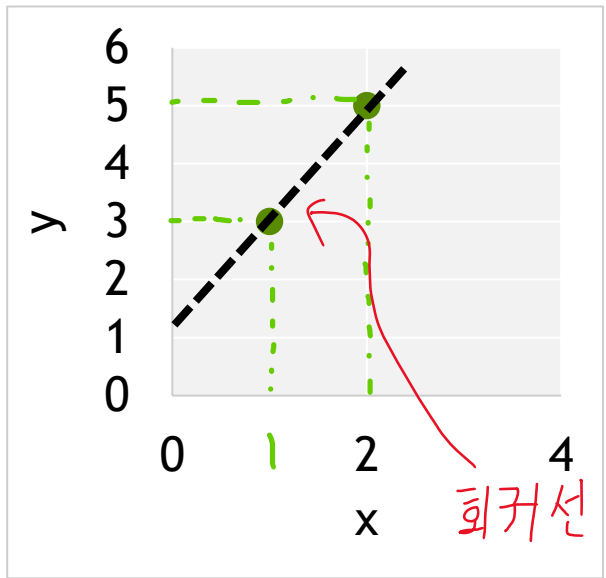


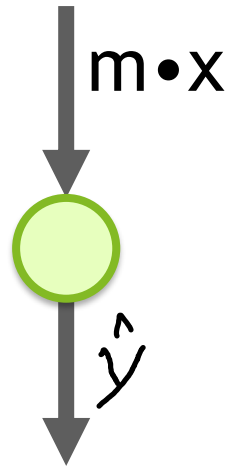


더욱 단순한 모델

$$y = mx + b$$

x	y
1	3
2	5





$m = ?$

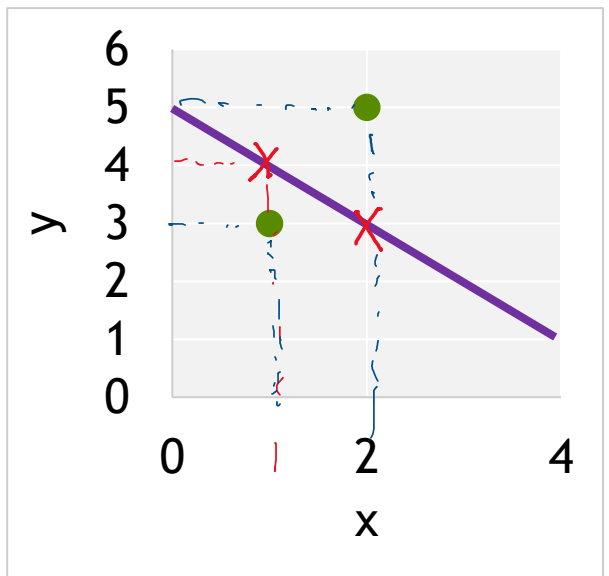
$b = ?$

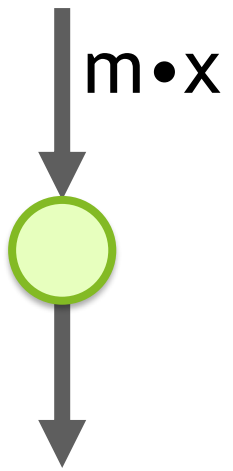
더욱 단순한 모델

$$y = mx + b$$

실제값(정답) 예측 값(가설)

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





무작위로 시작

$$m = -1$$

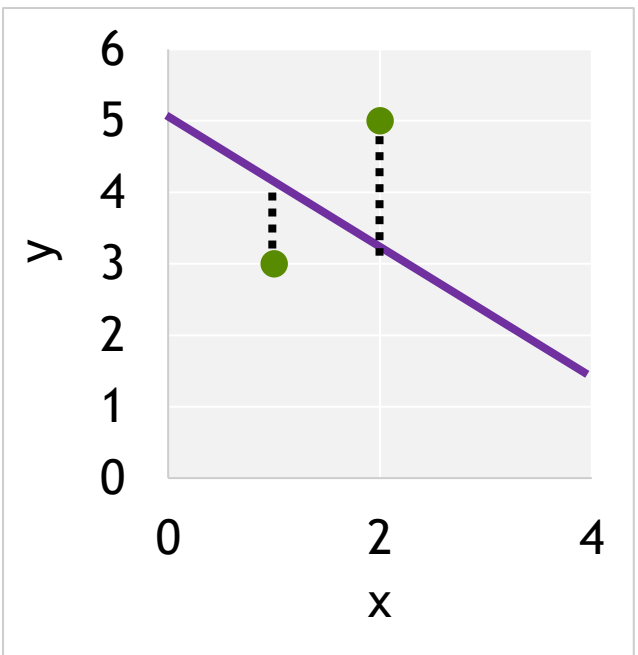
$$b = 5$$

더욱 단순한 모델

$$y = mx + b$$

오차(error)

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



* 평균 제곱 오차(Mean Square Error)

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

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

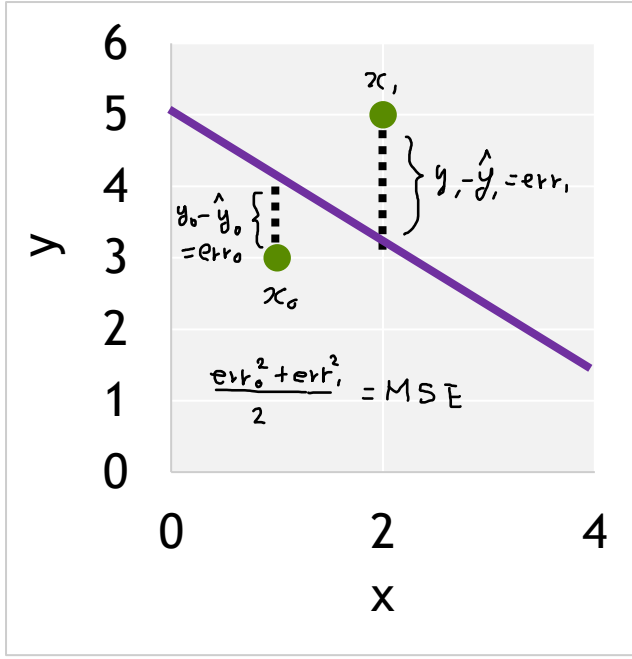
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2}$$

$\sqrt{2.5} = 1.6$

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



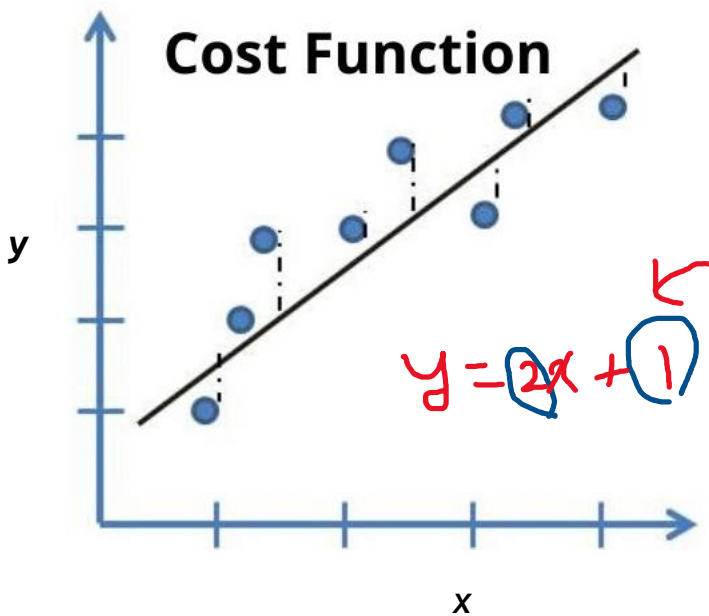

```python
data = [(1, 3), (2, 5)]
m = -1
b = 5

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)

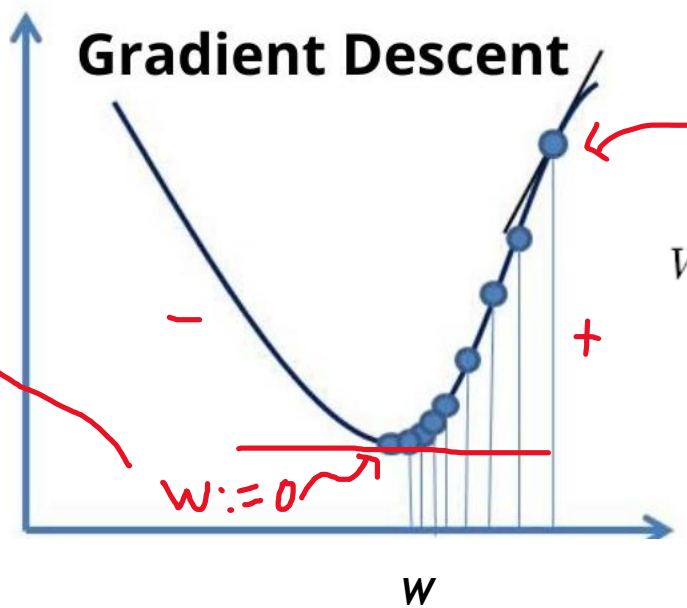
How to minimize cost(loss) $cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$

$\hat{y} = mx+b$

$Wx^{(i)}$: 예측값($\hat{y}$) $y^{(i)}$: 정답(실제값(y))



cost(W)



기울기 업데이트

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

step(learning rate)

- 경사하강법 쉽게 이해하기 : https://youtu.be/GEdLNvPIbiM

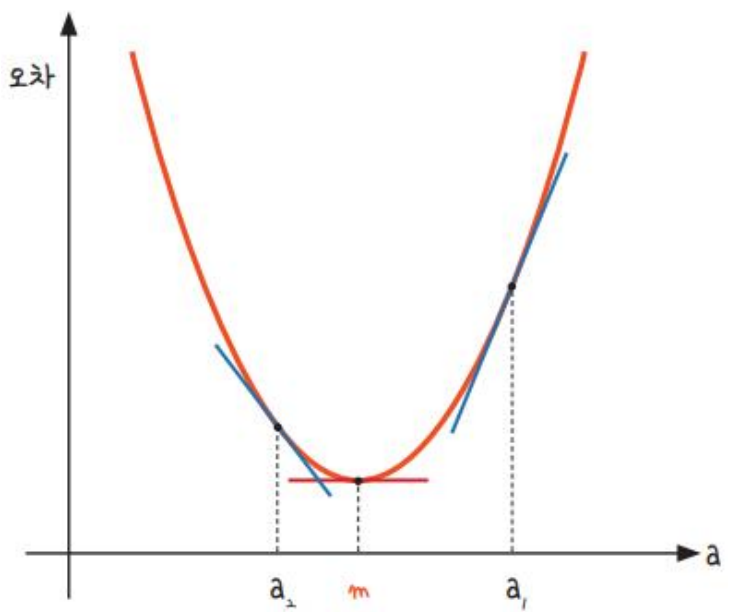
경사 하강법(Gradient Descent)

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

$\hat{y} = ax+b$

$Wx^{(i)}$: 예측값($\hat{y}$) $y^{(i)}$: 정답(실제값(y))

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



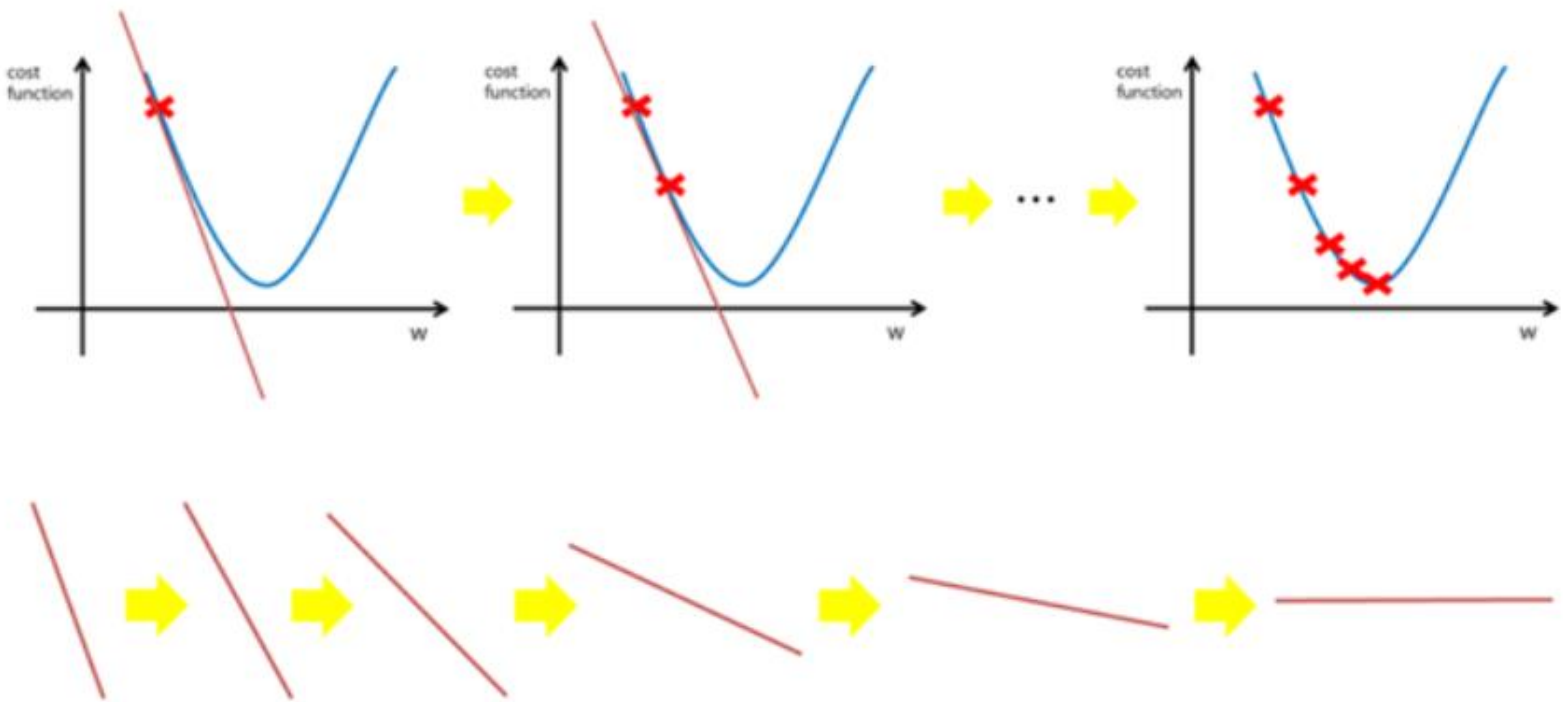
$\frac{\partial W}{\partial a}$: a로 편미분한 결과 $= \frac{2}{m}\sum -x_i(y_i - (ax_i + b))$

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

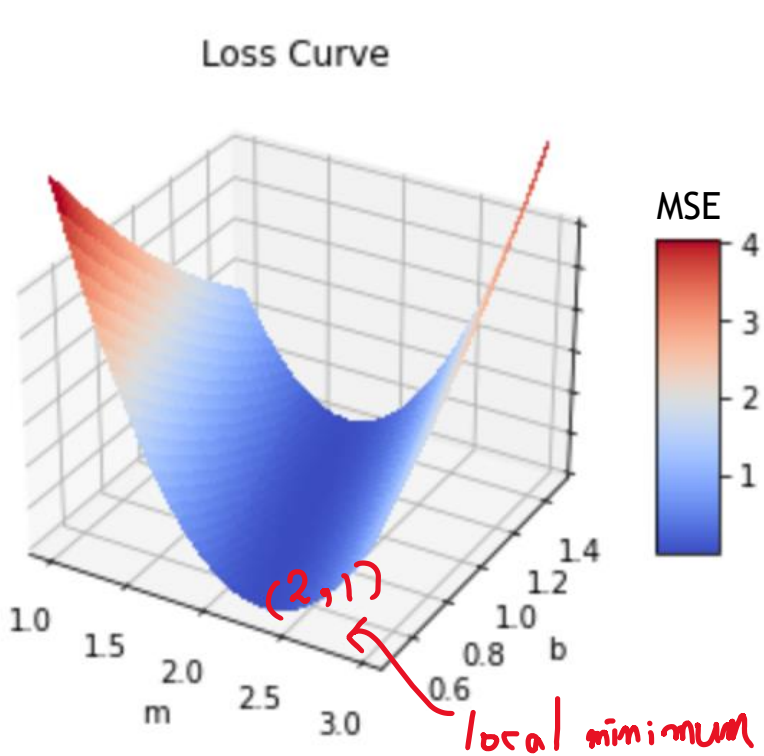
경사 하강법(Gradient Descent)

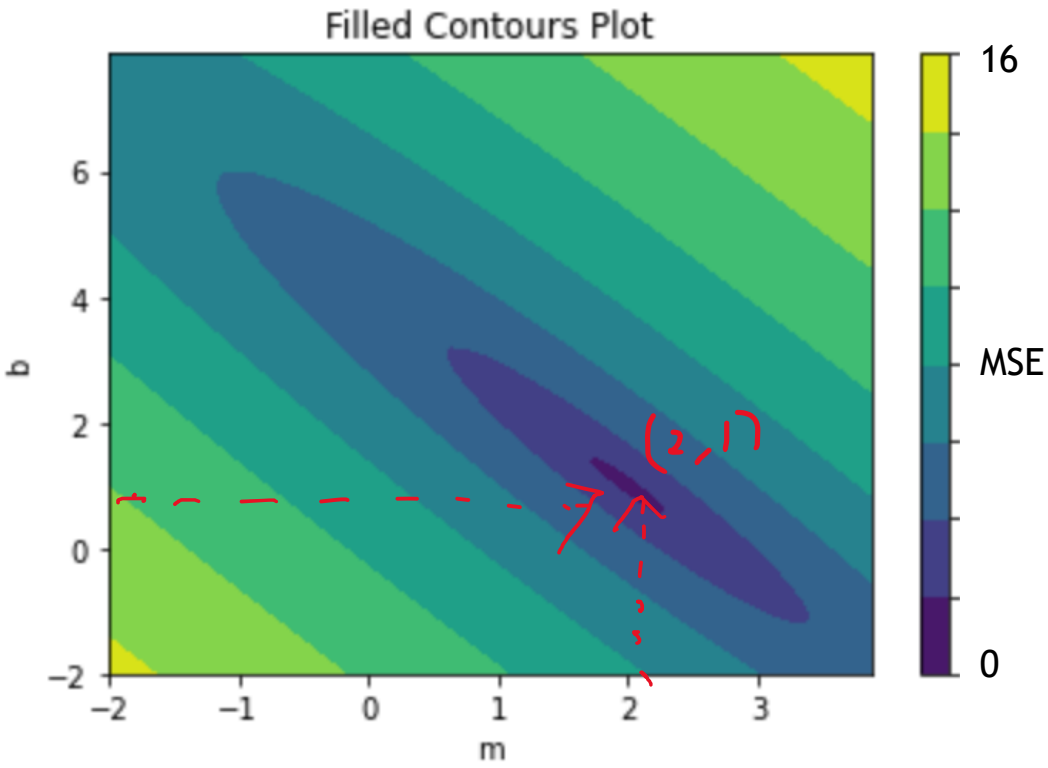
How to minimize cost(loss) $cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$

예측값($\hat{y}$) 정답(실제값(y))

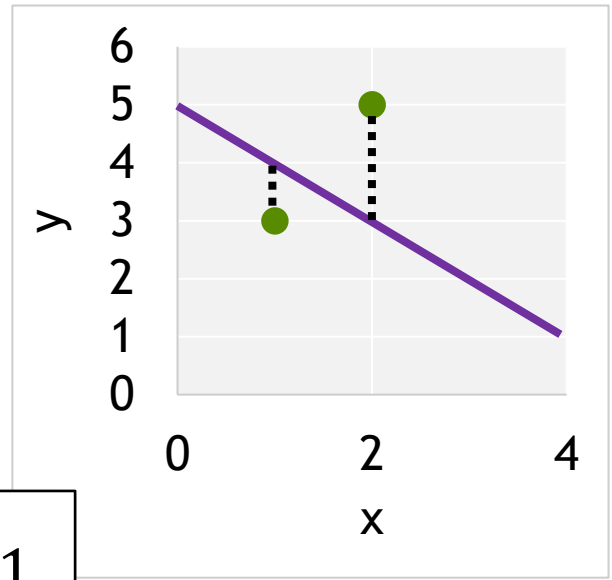


손실 곡선 (The Loss Curve)



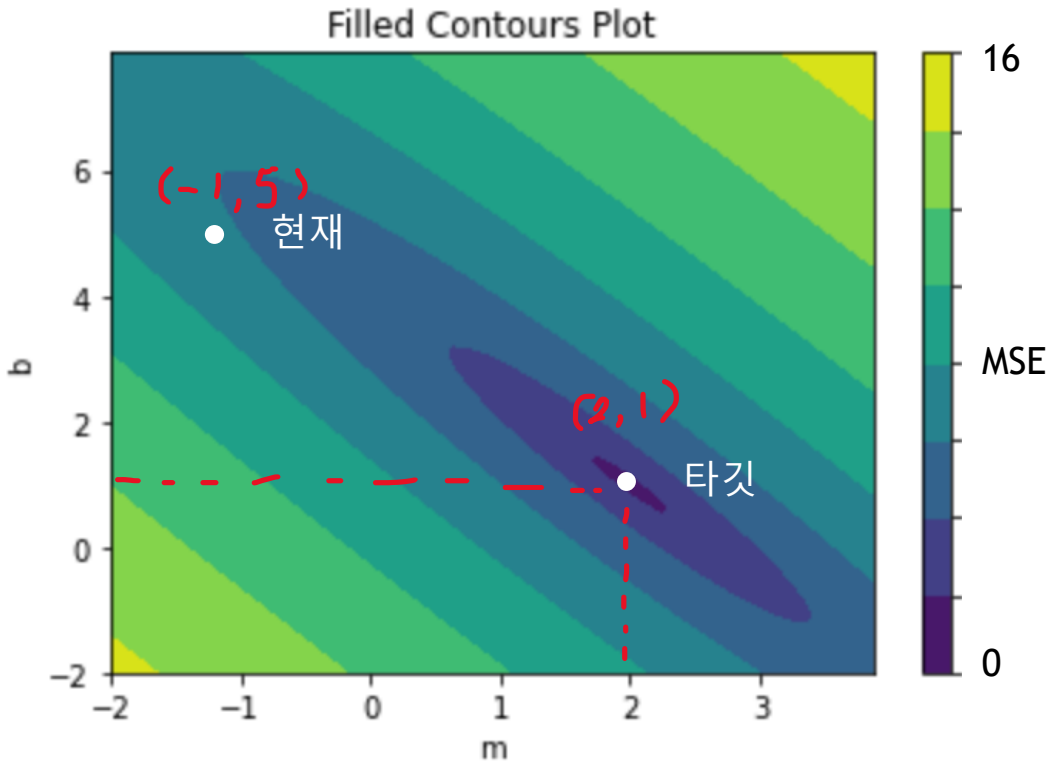


손실 곡선 (The Loss Curve)

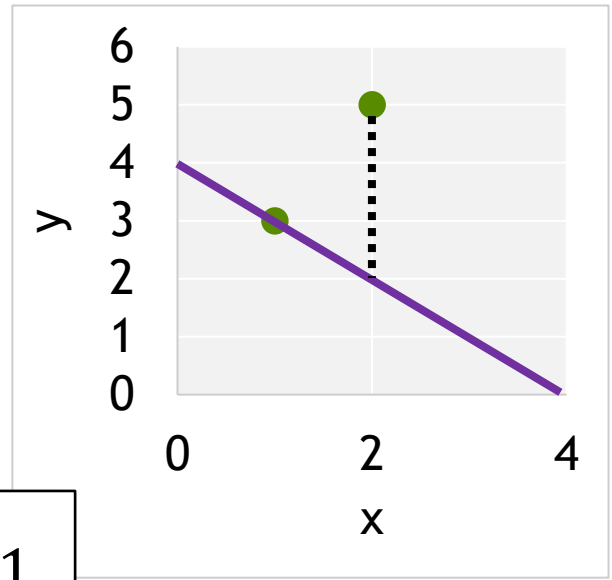


$m = -1$

$b = 5$

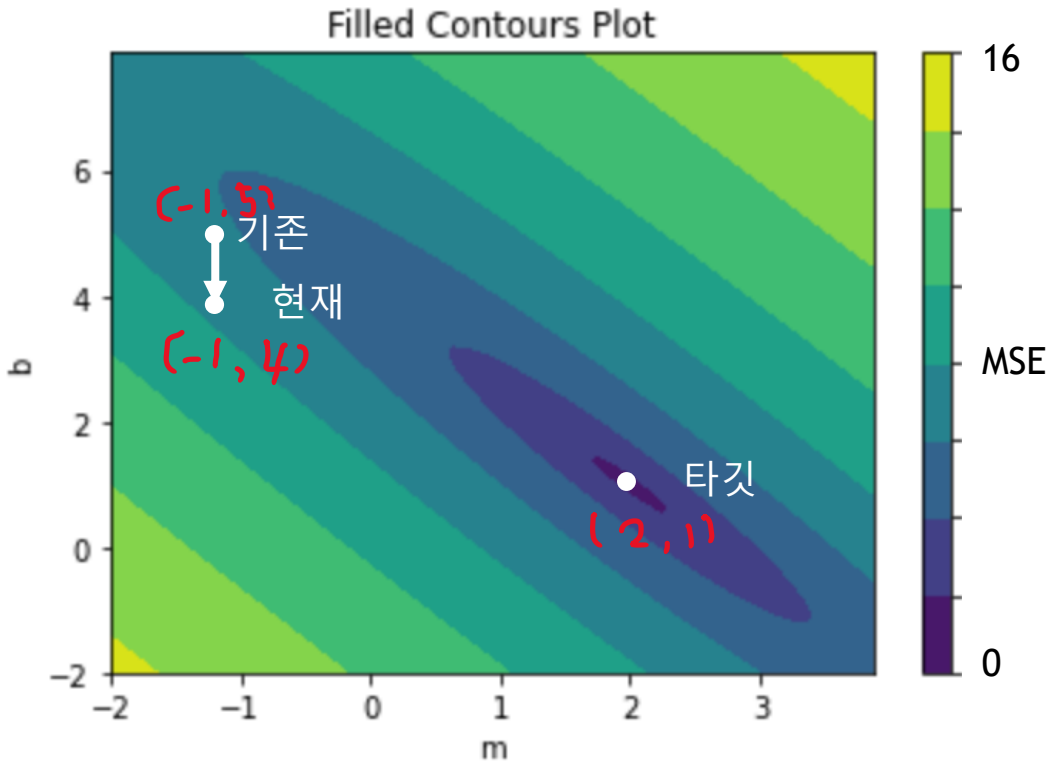


손실 곡선 (The Loss Curve)

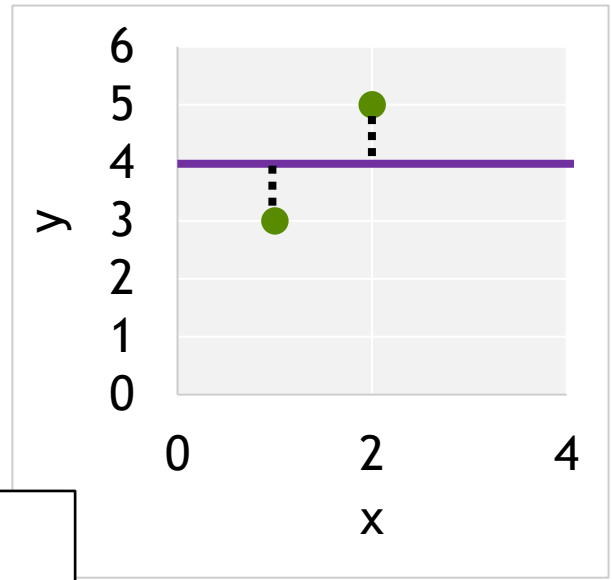


$m = -1$

$b = 4$

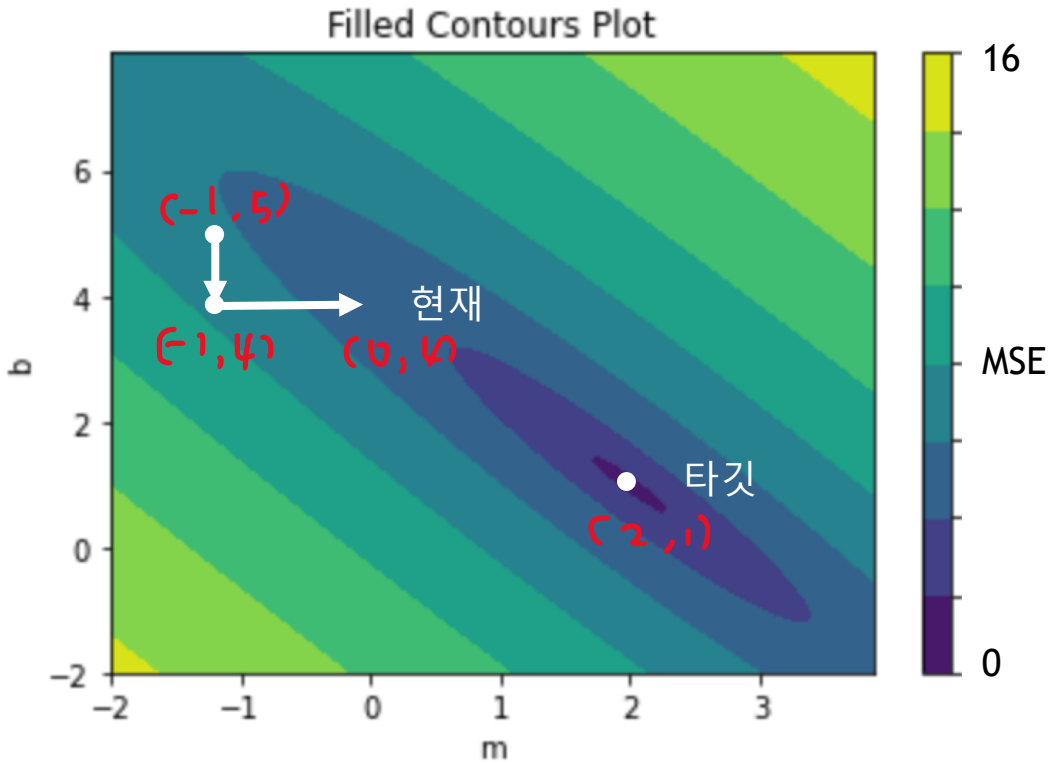


손실 곡선 (The Loss Curve)



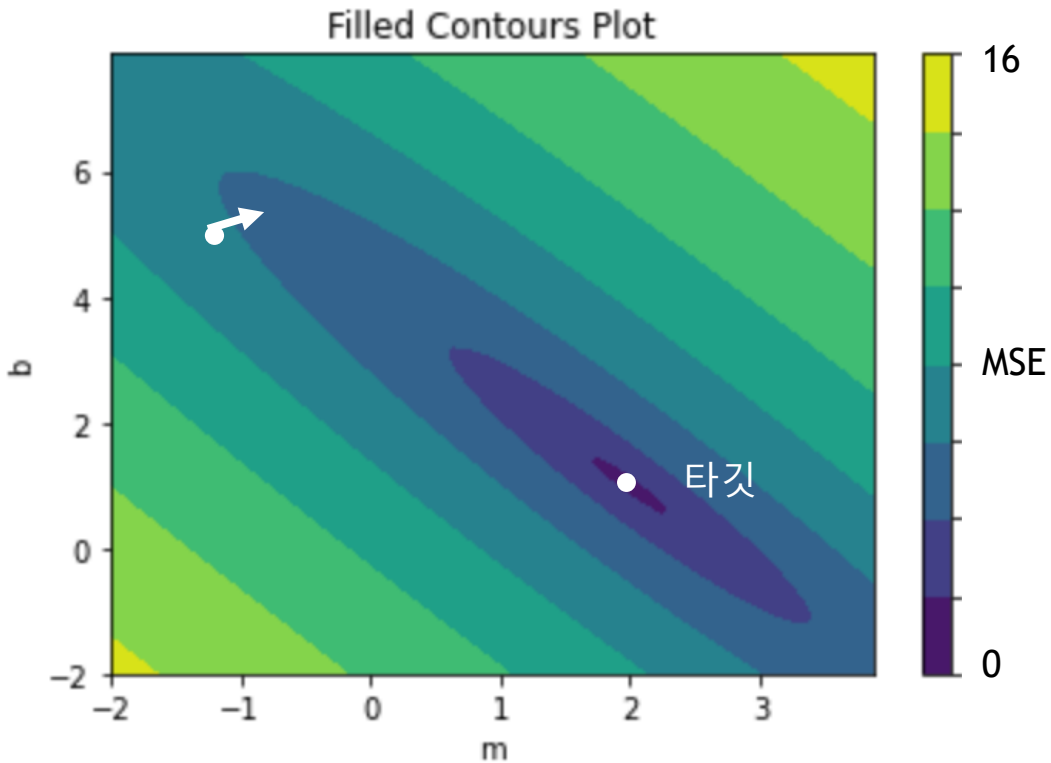
$m = 0$

$b = 4$



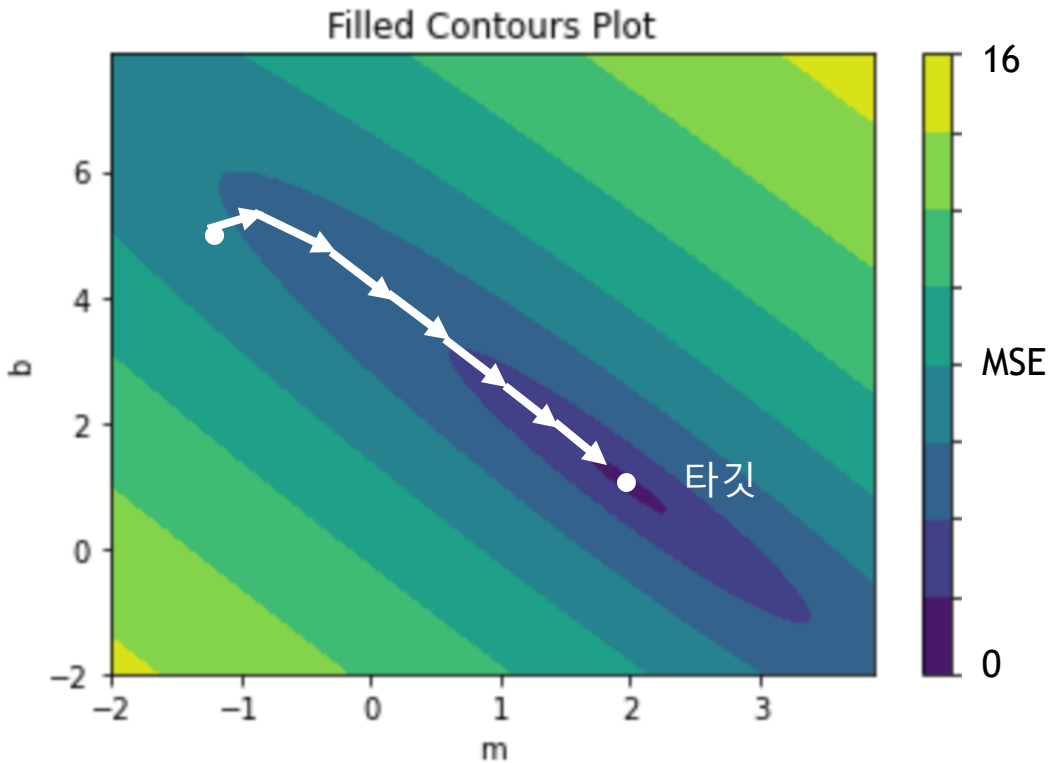
손실 곡선 (The Loss Curve)

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

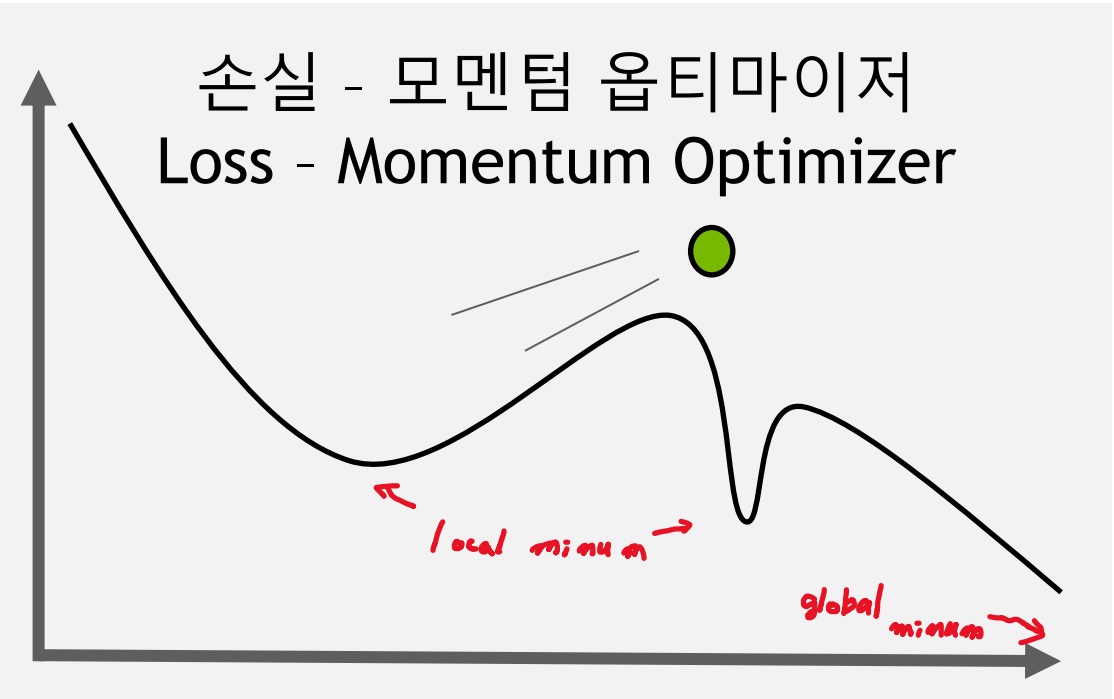


실습 01 : colab에서 경사 하강법 구현하기

- colab 실습 코드
  - https://url.kr/dbajir

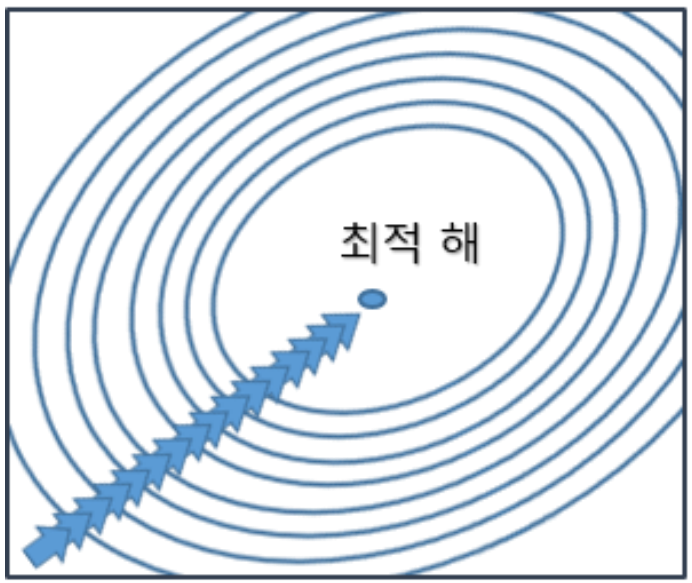
옵티마이저 (Optimizers)

- 손실(loss) 함수를 통해 얻은 손실 값으로 부터 모델을 업데이트(최적화) 하는 방법

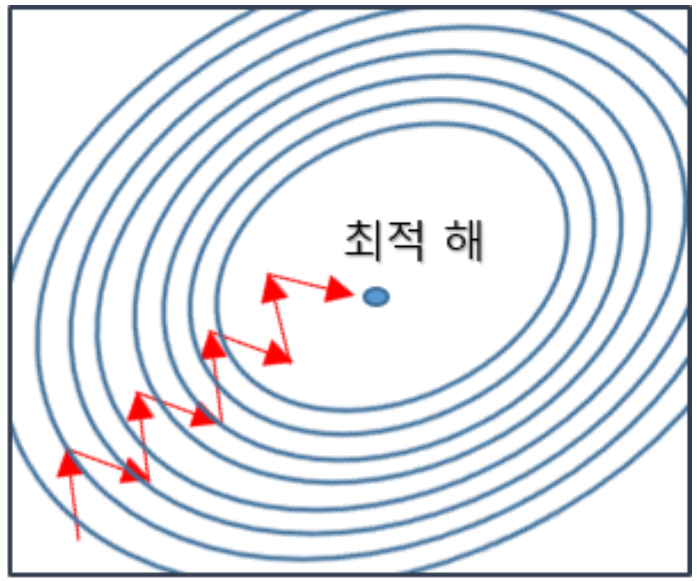


- Adam
- Adagrad
- RMSprop
- SGD

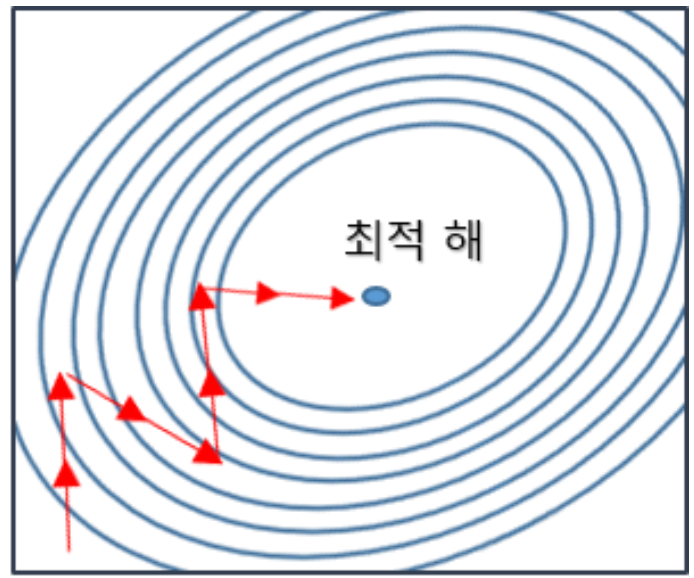
옵티마이저 (Optimizers)



경사 하강법

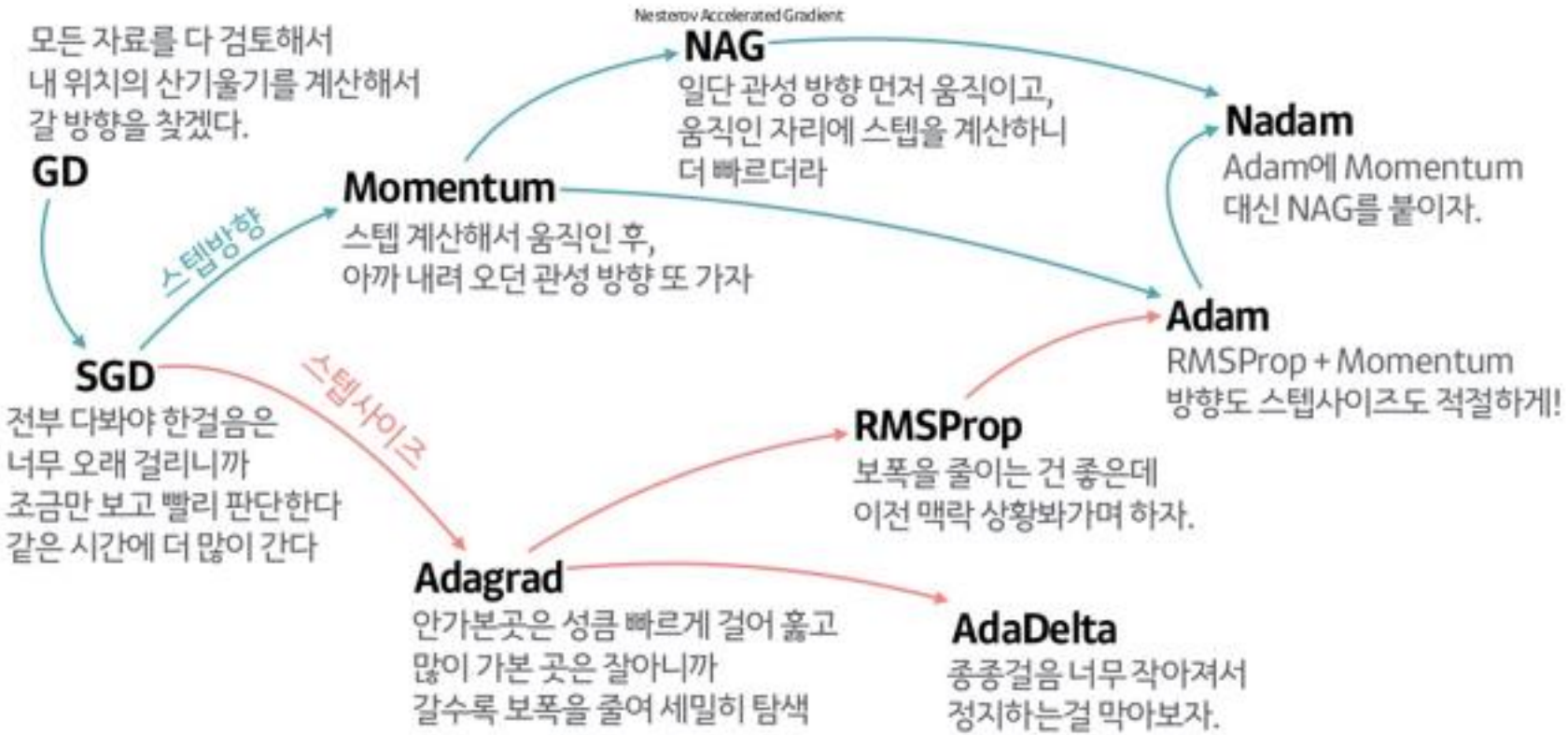


확률적 경사 하강법

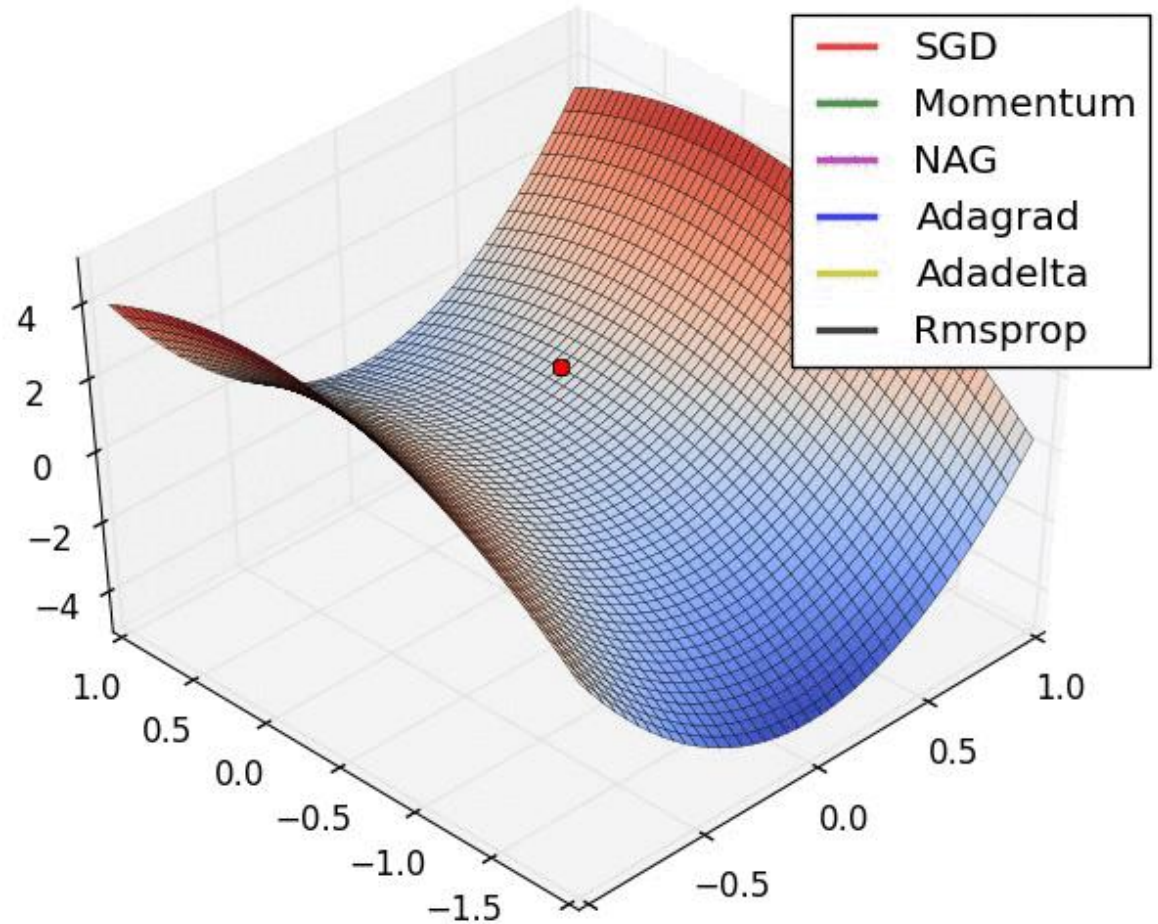


모멘텀

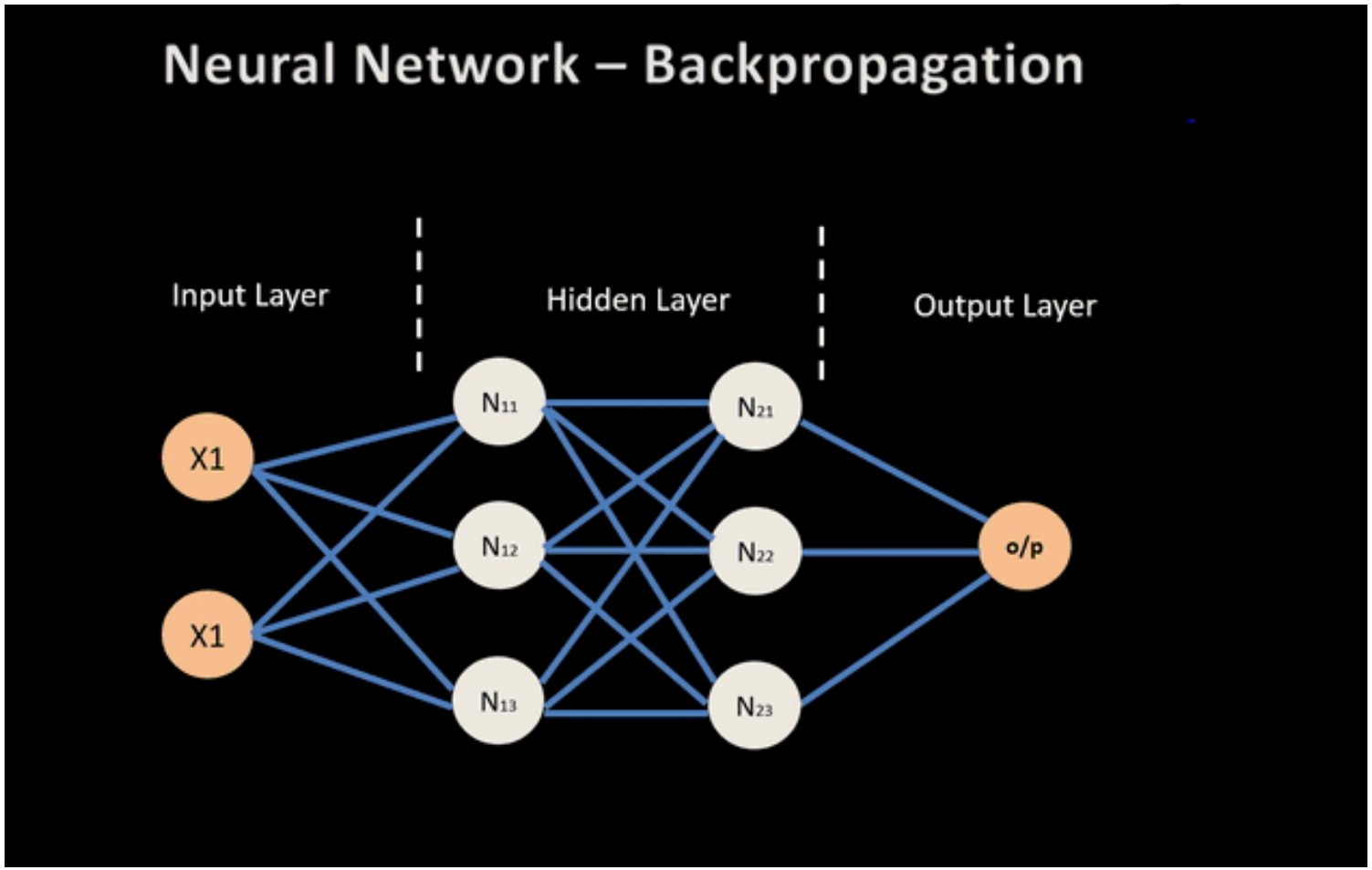
옵티마이저 (Optimizers)



옵티마이저 (Optimizers) 성능 비교



오차 역전파 (Error Backpropagation)



참조 : machinelearningknowledge.ai

1 오차 역전파, 체인 룰(CHAIN RULE)

$z = g(y), y = f(x)$

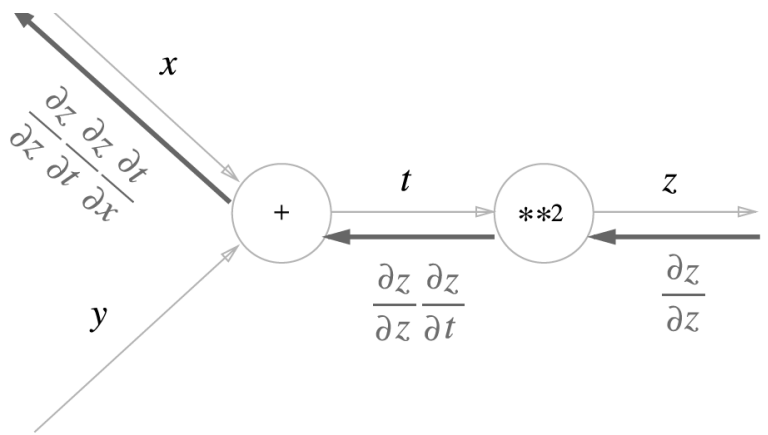
$z = (g \circ f)(x) = g(f(x))$

미분 →

$(g \circ f)'(x) = g'(y)f'(x)$

↓ Chain rule

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

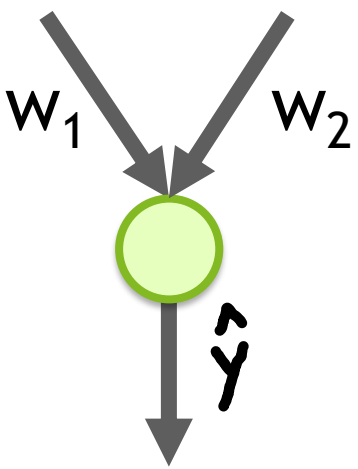


Backpropagation algorithm :

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

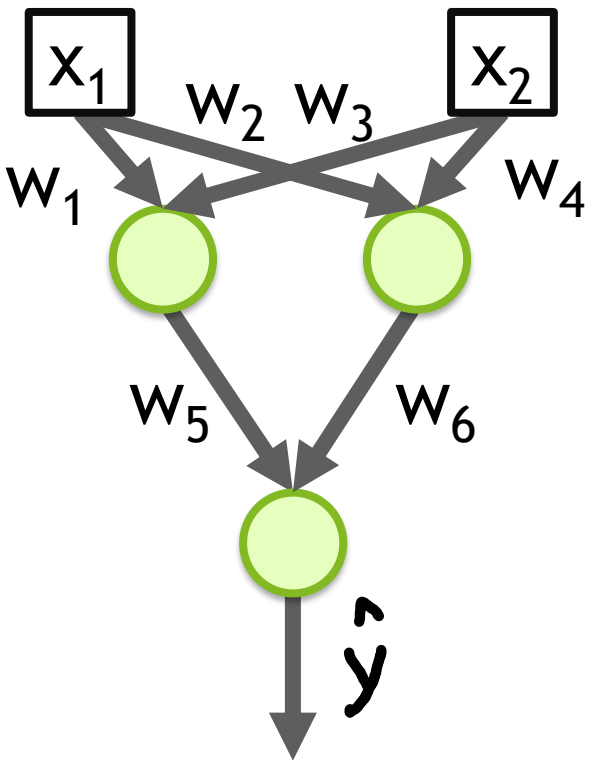

뉴런에서 네트워크로

네트워크 구축



- 더 많은 변수로의 확장

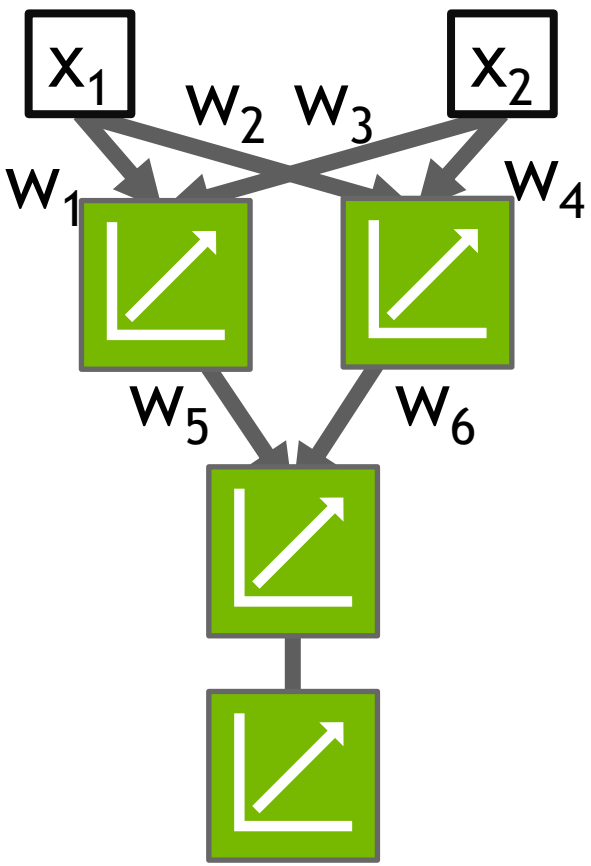
네트워크 구축



- 더 많은 변수로의 확장
- 뉴런들간의 연결

네트워크 구축

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

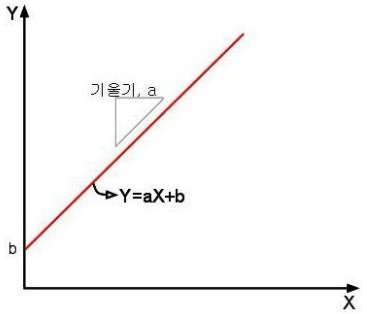


- 더 많은 변수로의 확장
- 뉴런들간의 연결
- 모든 회귀가 선형이면 출력 결과도 선형 회귀

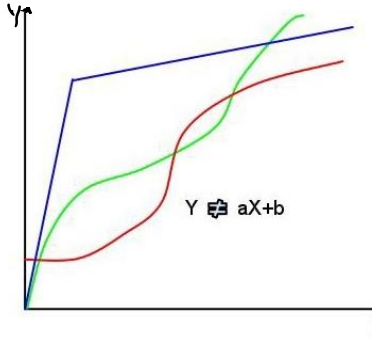
활성화 함수
ACTIVATION FUNCTIONS

선형(LINEAR), 비선형(NON-LINEAR)

- **선형(linear) 함수**
  - 어떤 값을 입력 했을 때 출력이 입력의 상수 배 만큼 변하는 함수(a, b의 선형 결합)
  - f(X) = aX + b : 선형 함수



- **비선형(non-linear) 함수**
  - 선형이 아닌 함수, 한 개의 직선으로 그릴 수 없는 함수

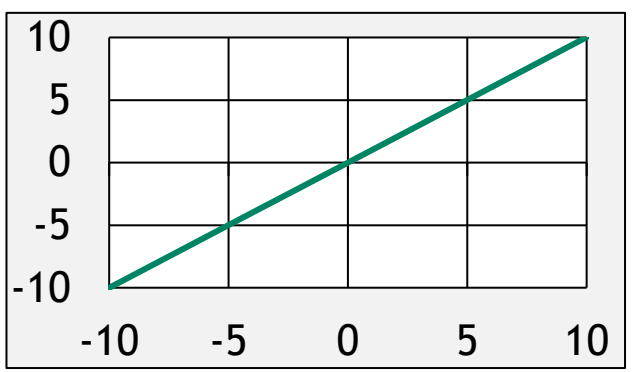


활성화 함수 (Activation Functions)

Linear(선형)

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

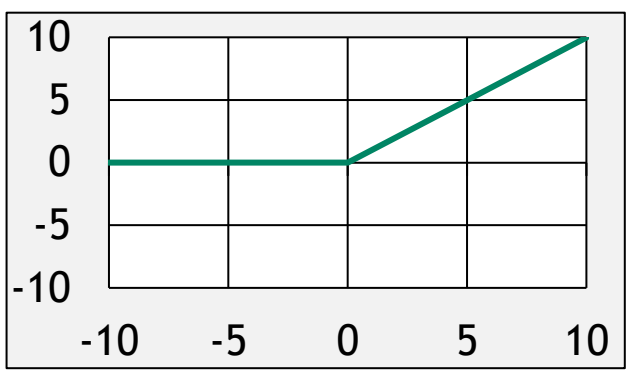
```
# Multiply each input
# with a weight (w) and
# add intercept (b)
y_hat = wx+b
```



ReLU

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

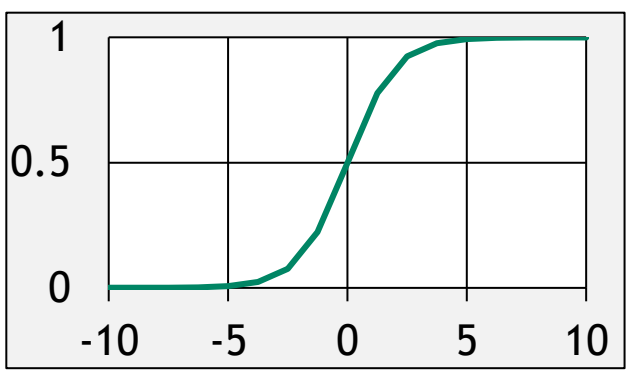
```
# Only return result
# if total is positive
linear = wx+b
y_hat = linear * (linear > 0)
```



Sigmoid(시그모이드)

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

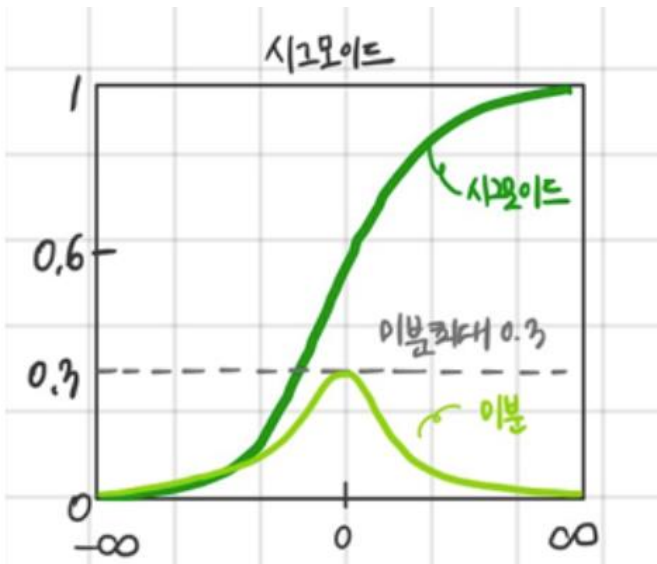
```
# Start with line
linear = wx + b
# Warp to - inf to 0
inf_to_zero = np.exp(-1 * linear)
# Squish to -1 to 1
y_hat = 1 / (1 + inf_to_zero)
```



가중치 소실 문제 (Vanishing gradient)

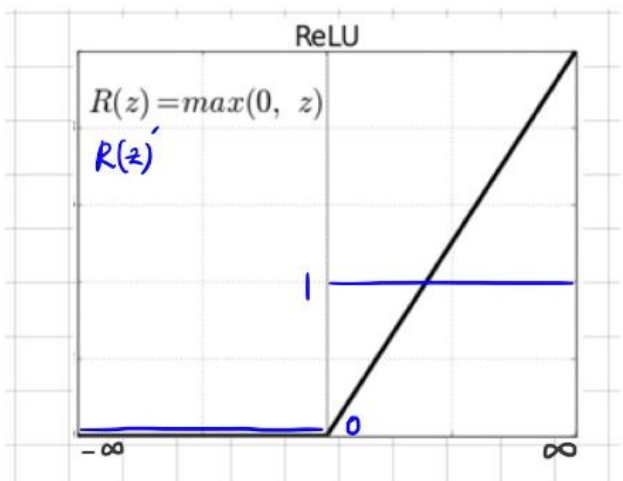
Sigmoid(시그모이드)

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



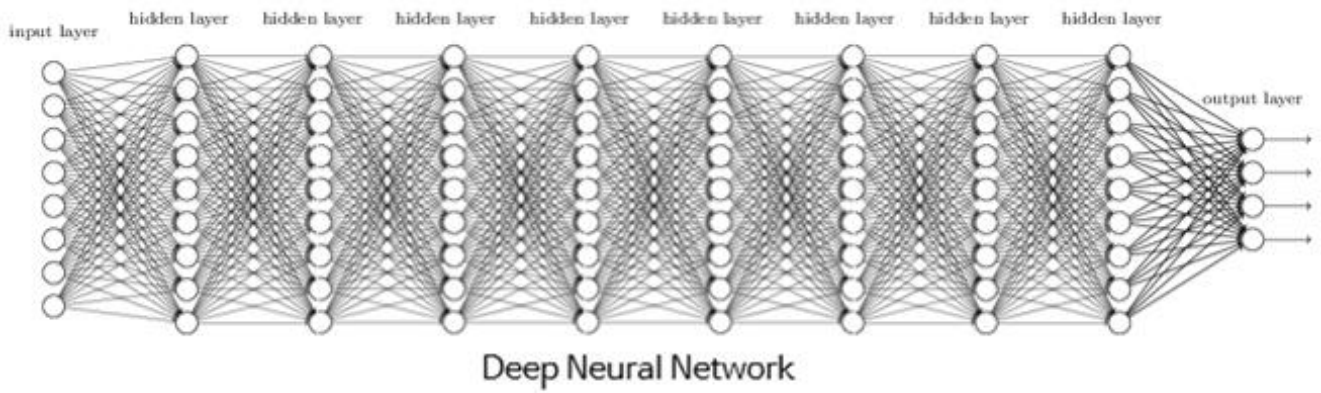
ReLU

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



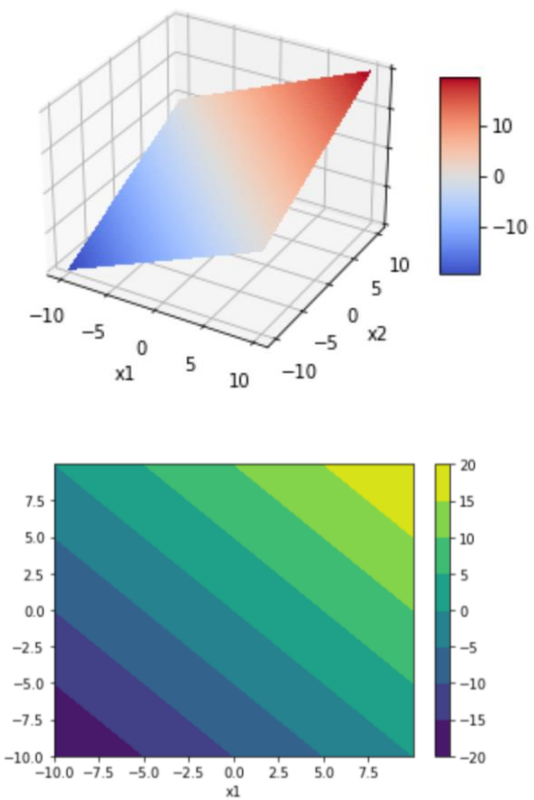
$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \qquad f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

가중치 소실 문제 (Vanishing gradient)

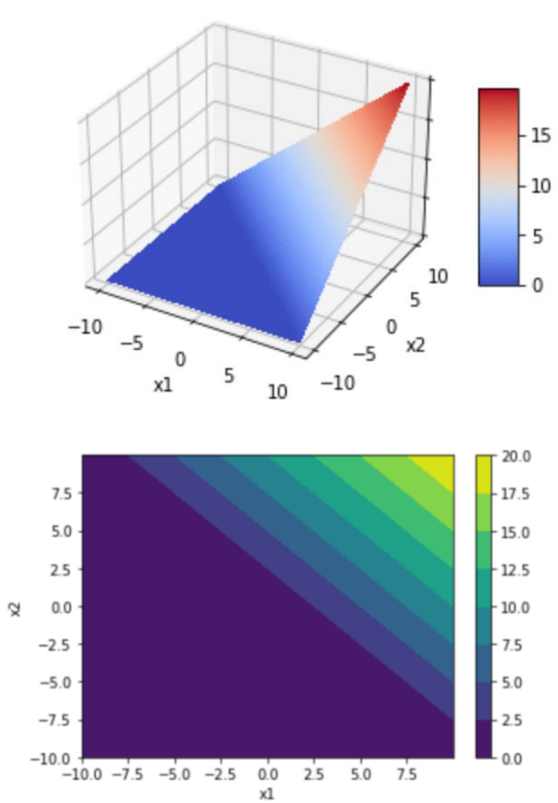


활성화 함수 (Activation Functions)

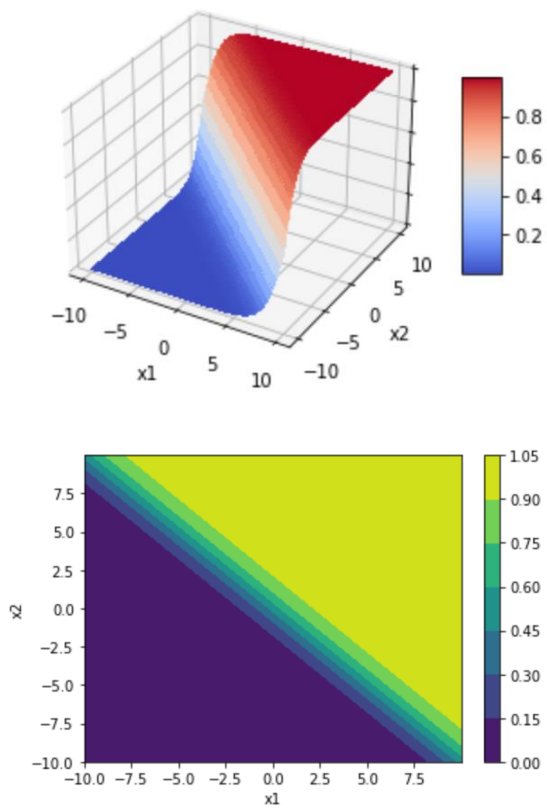
Linear(선형)



ReLU

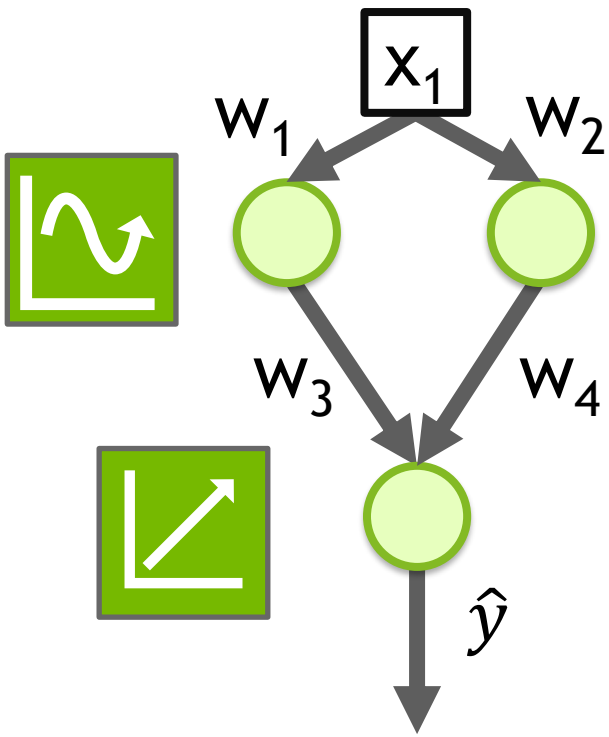


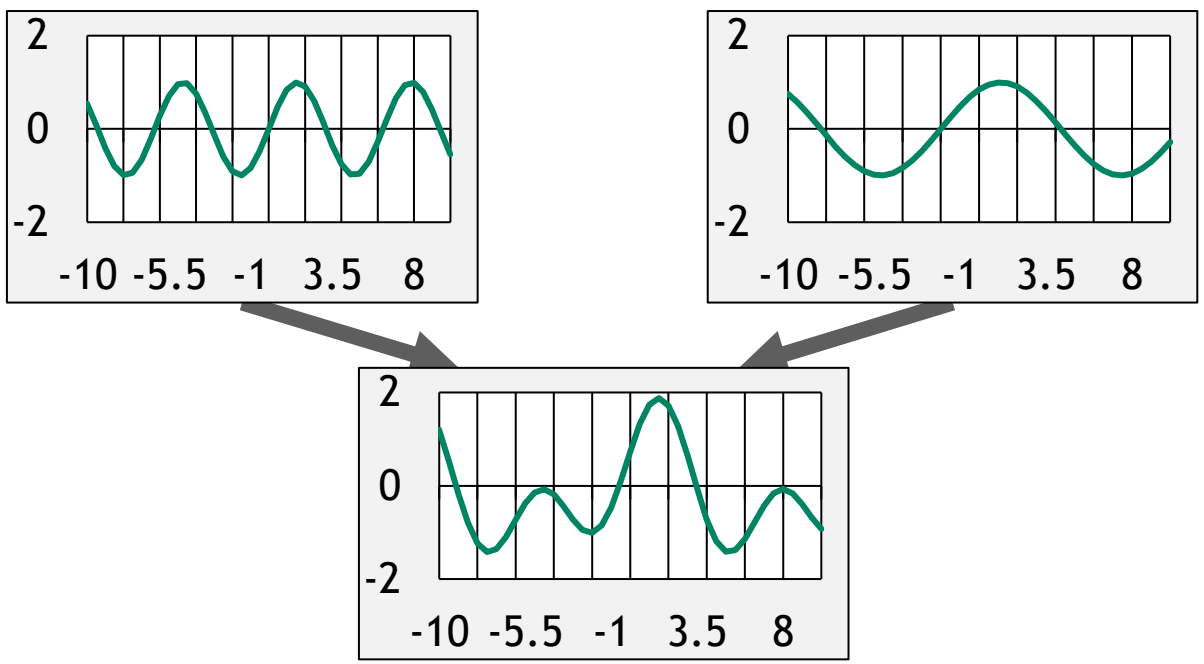
Sigmoid(시그모이드)



활성화 함수 (Activation Functions)

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

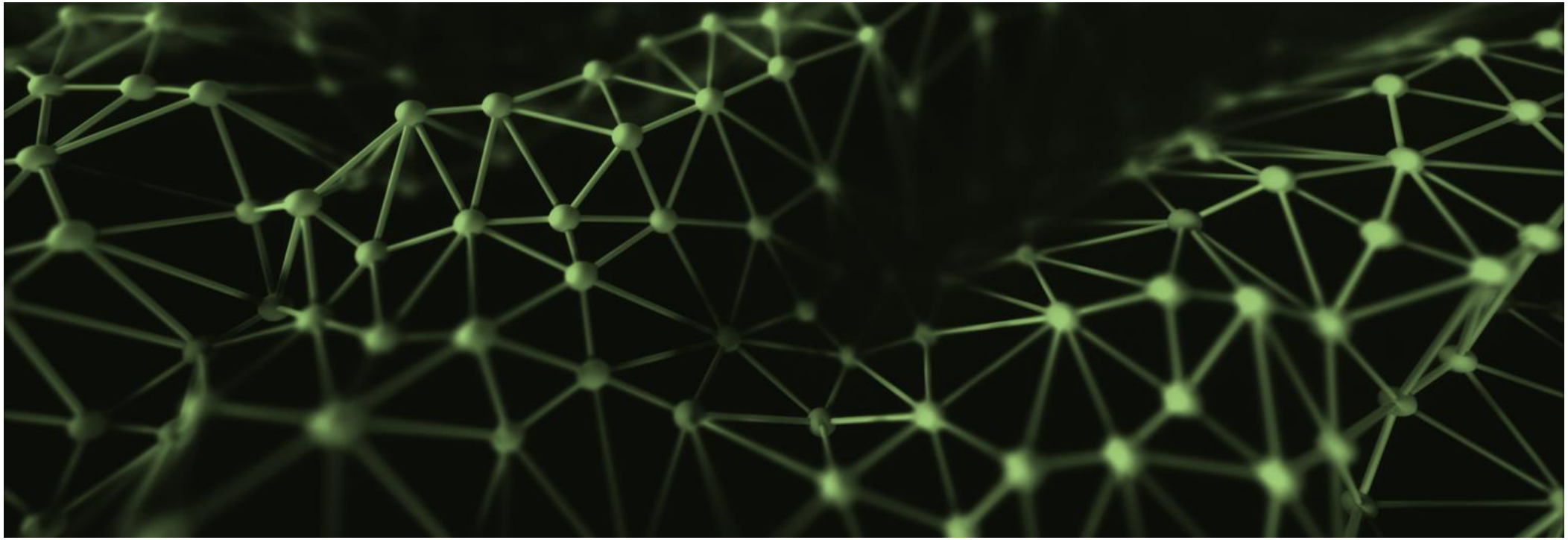




과적합
OVERFITTING

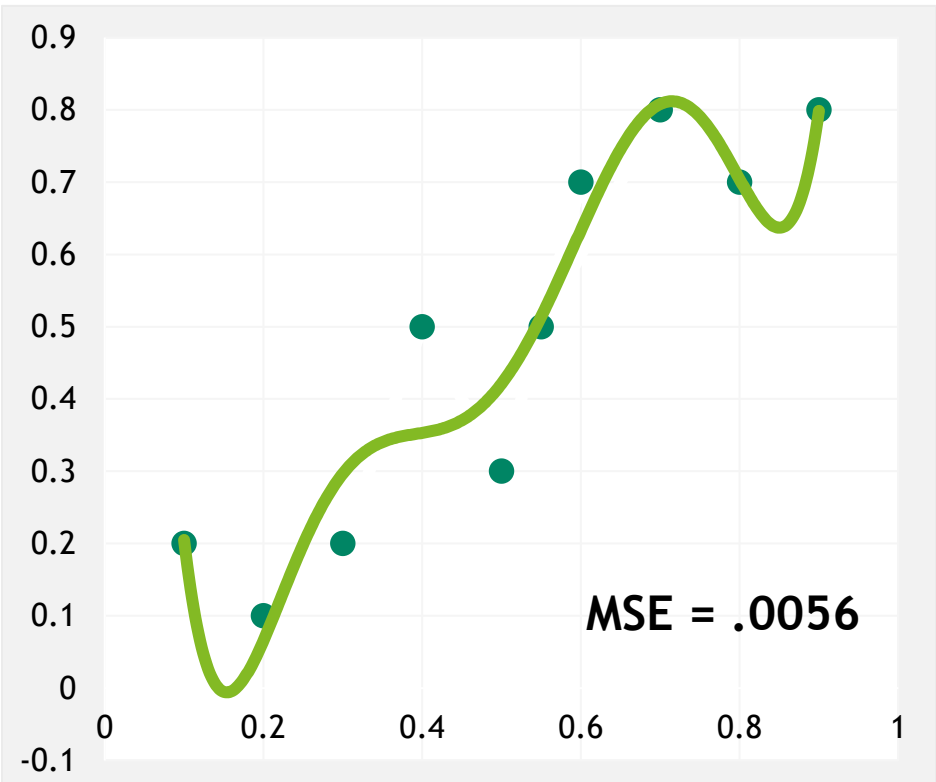
과적합 (Overfitting)

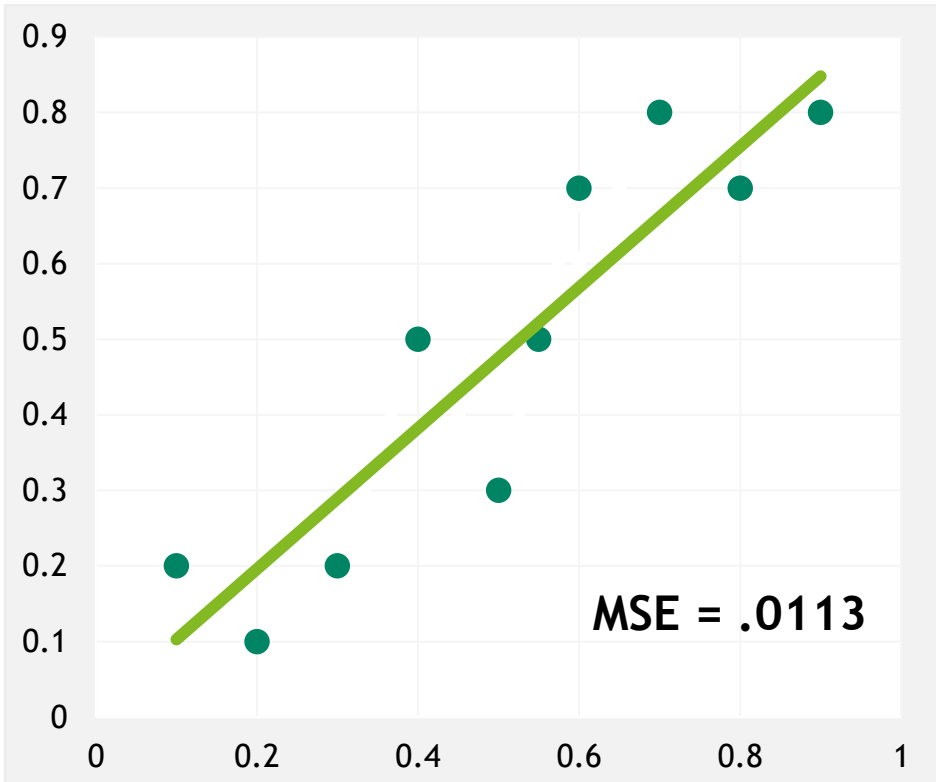
거대한 뉴럴 네트워크를 구축하면 되지 않을까?



과적합 (Overfitting)

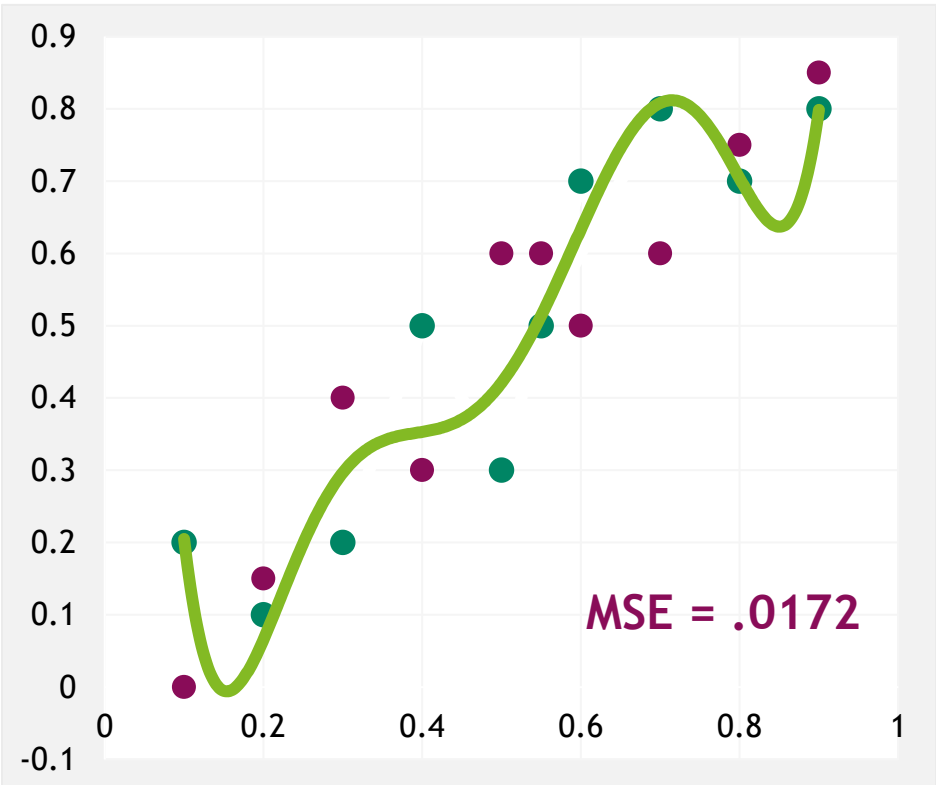
어떤 추세선이 더 나은가?

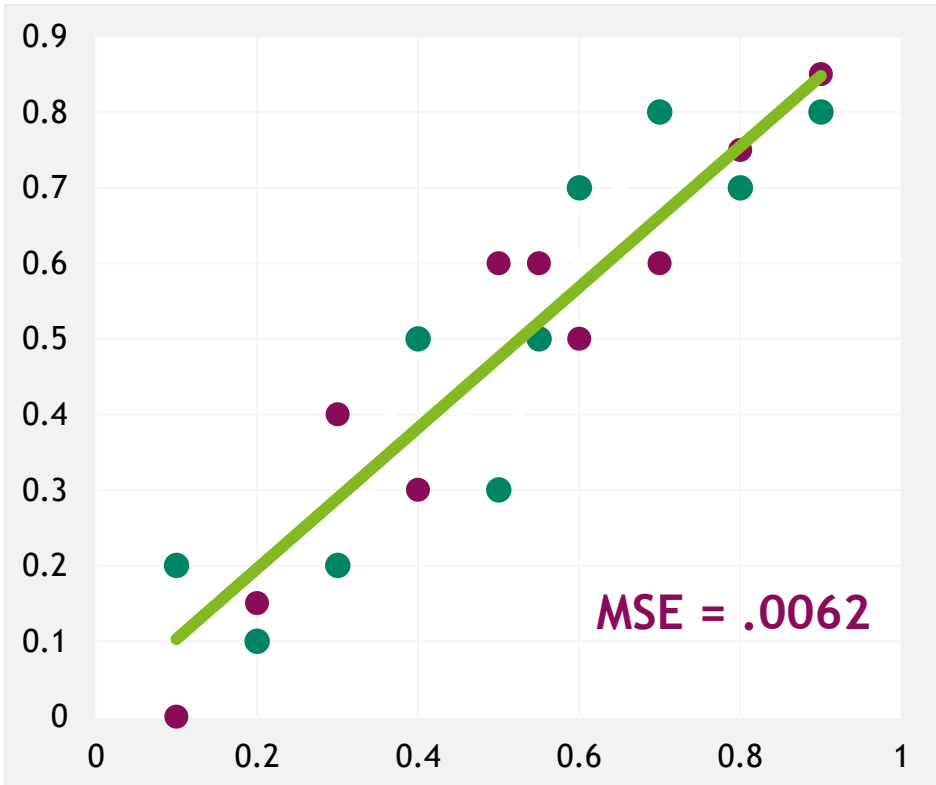




과적합 (Overfitting)

어떤 추세선이 더 나은가?





트레이닝 데이터와 검증 데이터 비교

암기 회피(Avoid memorization)

트레이닝 데이터 (Training data)

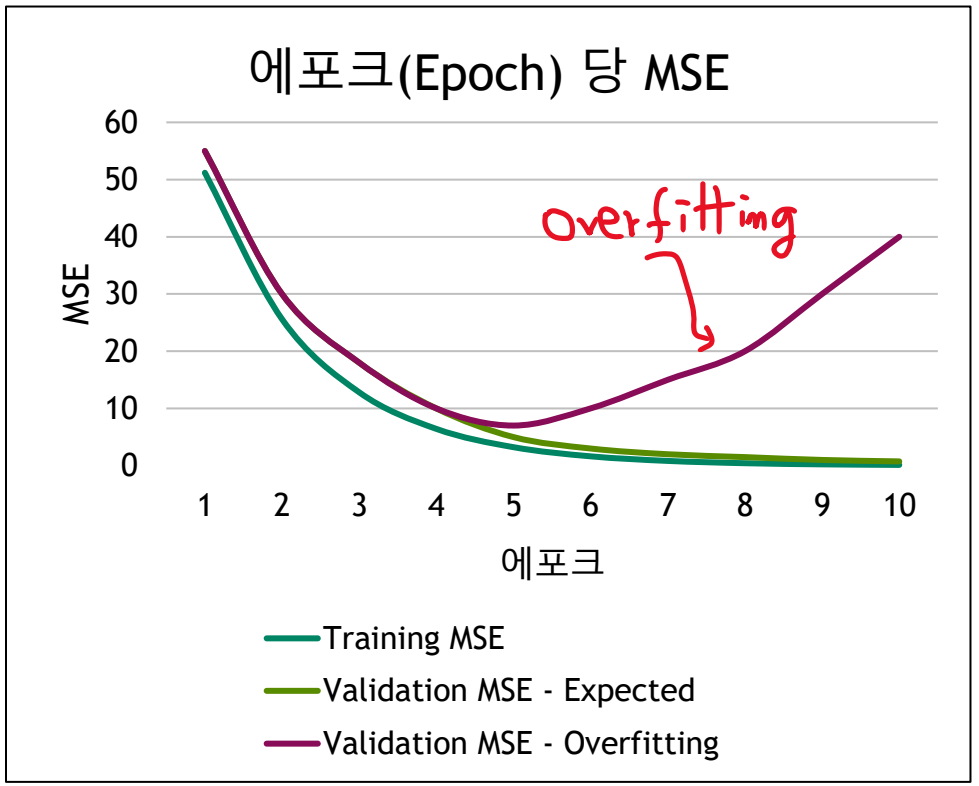
- 학습해야 하는 모델의 핵심 데이터세트

검증 데이터 (Validation data)

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

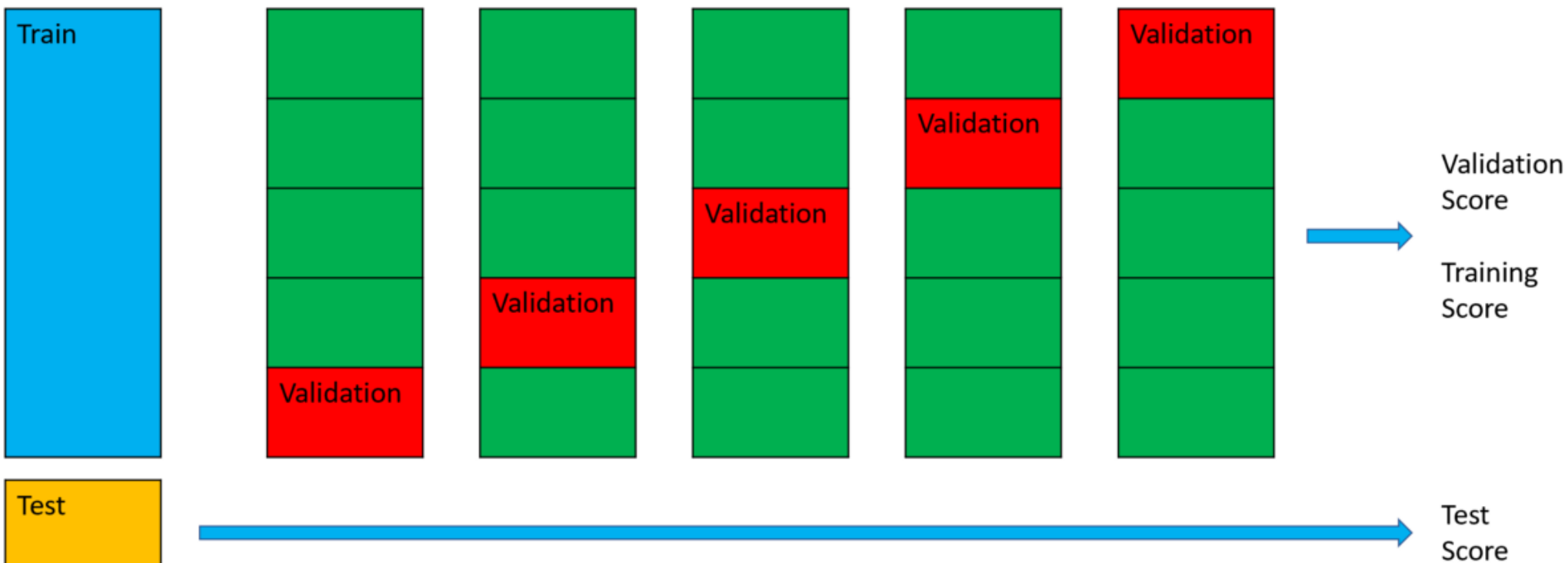
과적합 (Overfitting)

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

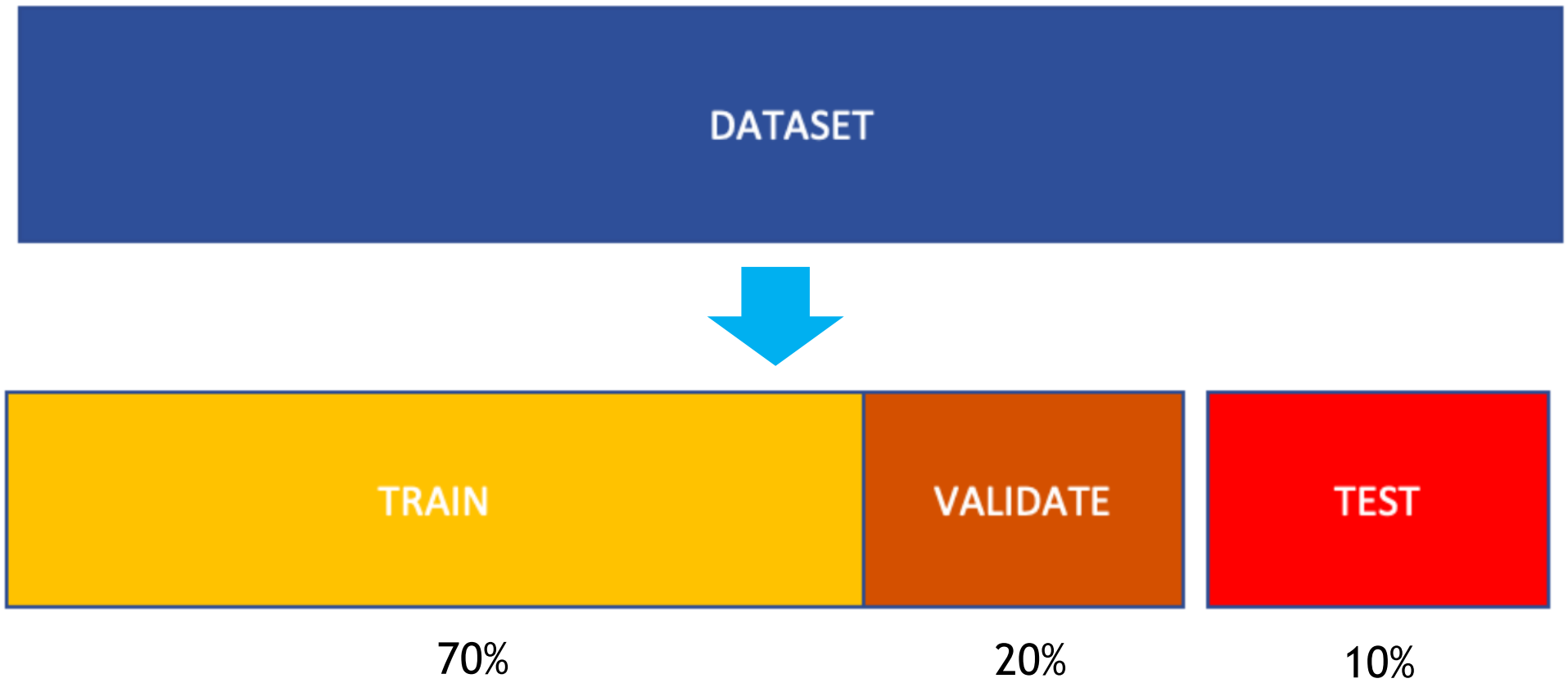



```
model.fit(x_train, y_train, epochs=20, verbose=1, validation_data=(x_valid, y_valid))
```

Train Data, validation data, test data



Train Data, validation data, test data



실습 02 : 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

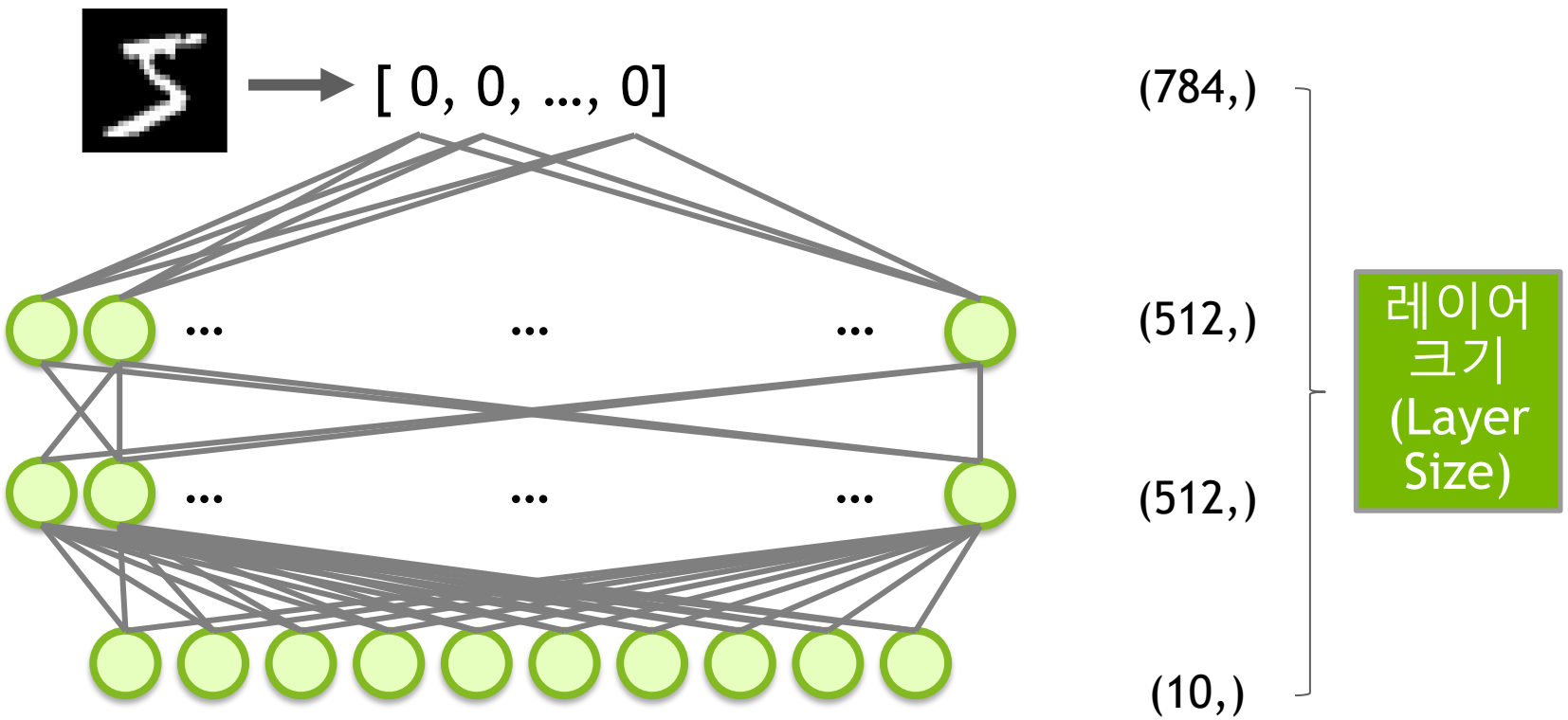
- 다중 선형회귀 : 공부한 시간, 과외수업 횟수에 따른 성적 예측

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

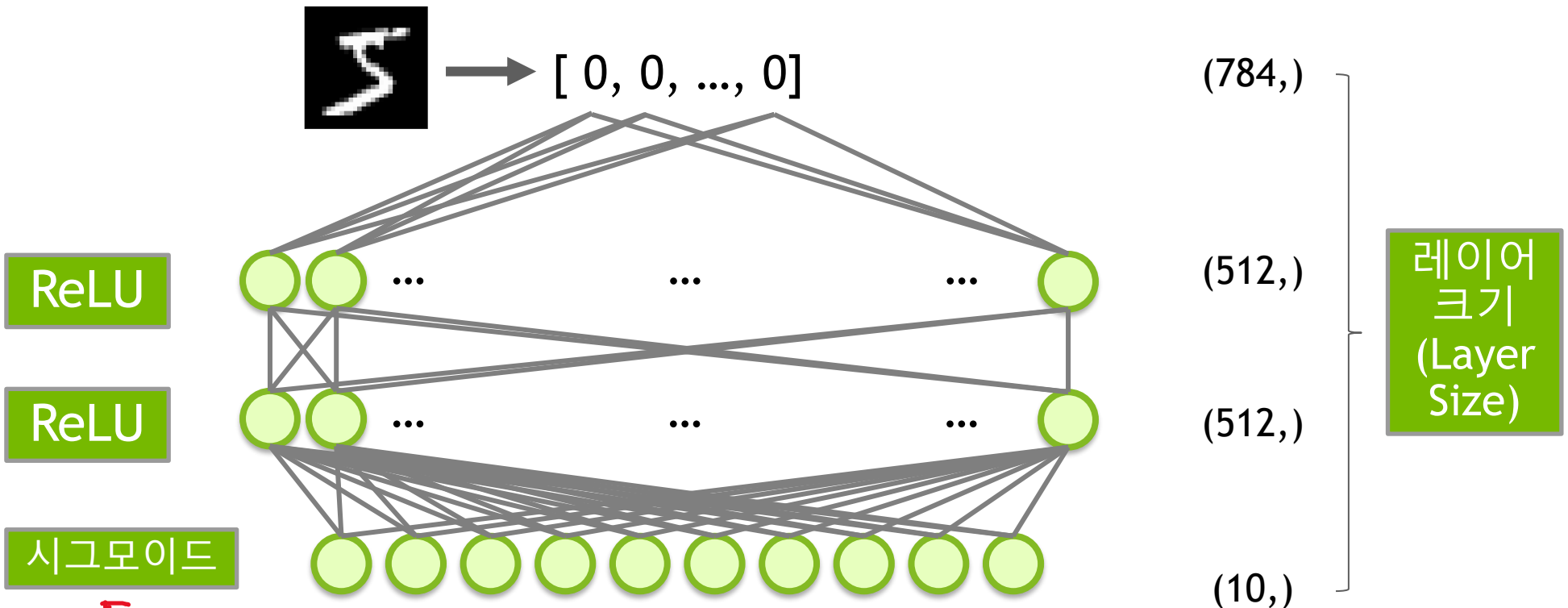
- colab 실습 코드 :
  - https://url.kr/b5zkqu

회귀에서 분류로
FROM REGRESSION TO CLASSIFICATION

MNIST 모델



MNIST 모델

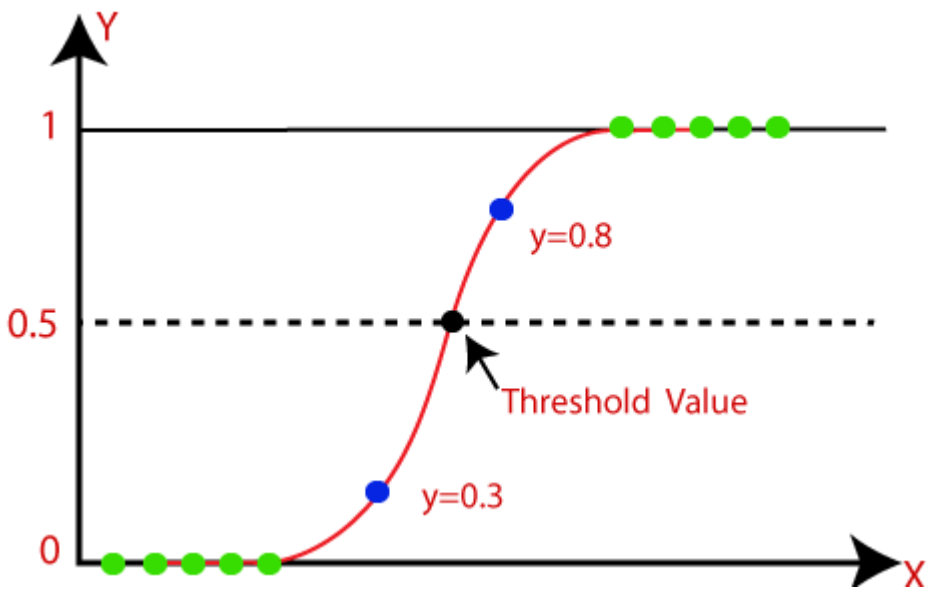


확률값(0~1) : 10개의 이진분류기 필요 → Softmax분류기

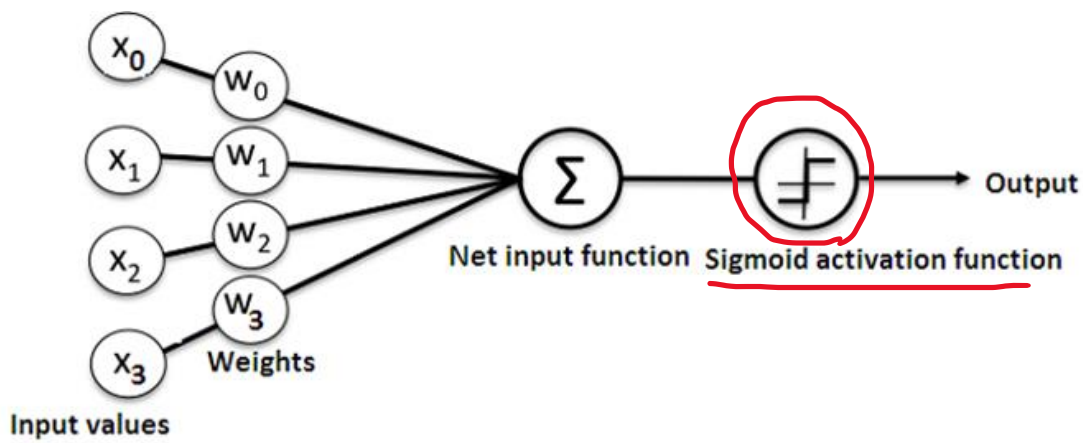
Sigmoid(Logistic) 활성화 함수

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

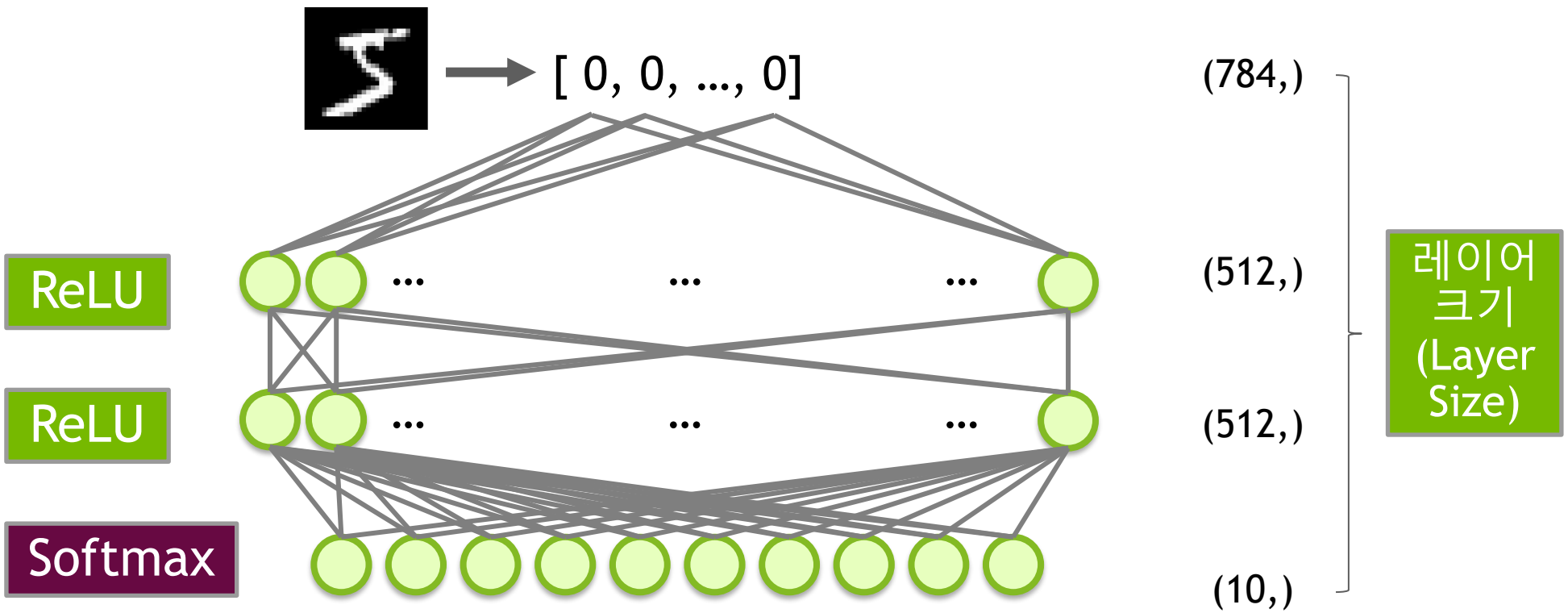
$wx+b$



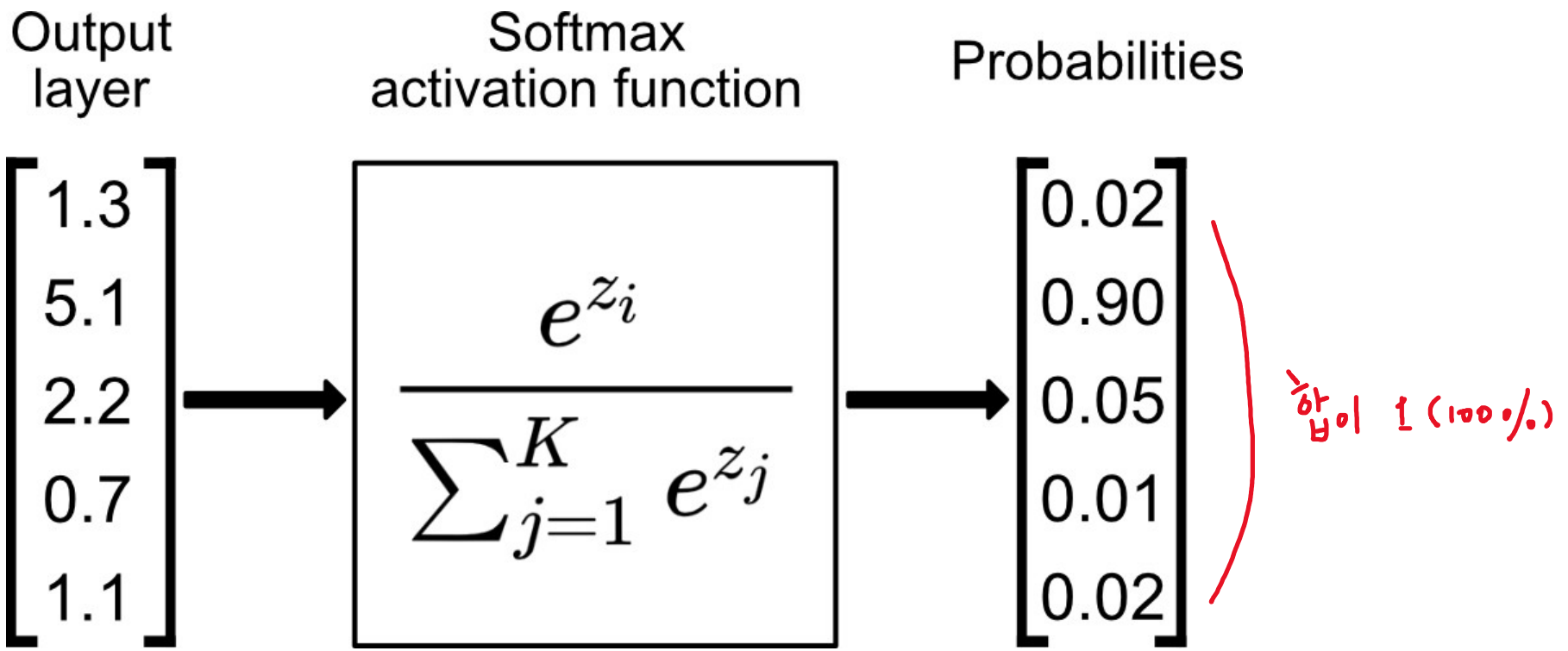
- 이진 분류
  - 시험 합격 여부 : Pass or Fail
  - 스팸 필터 : Spam or Ham
  - SNS 피드 노출 : Show or Hide



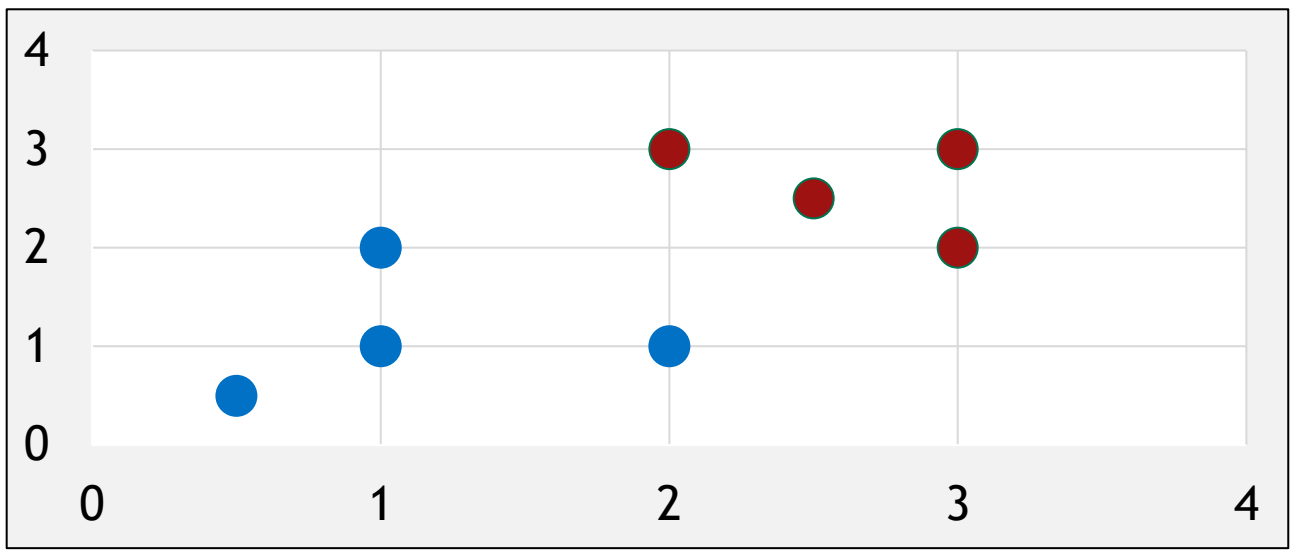
MNIST 모델 : 다중 분류



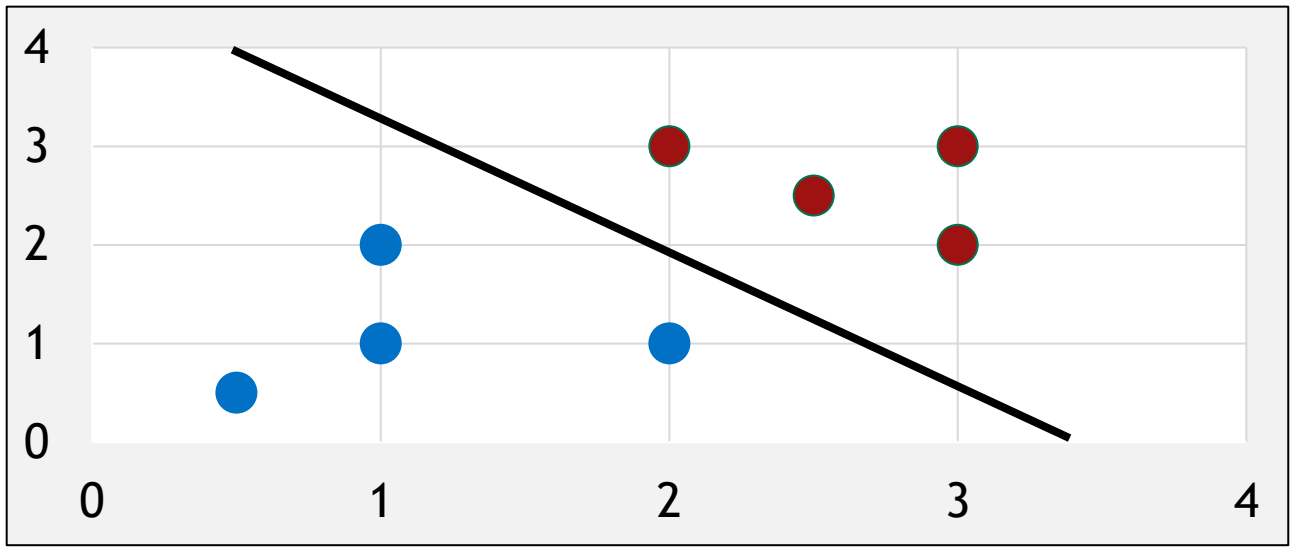
SoftMax 활성화 함수 : 다중 분류



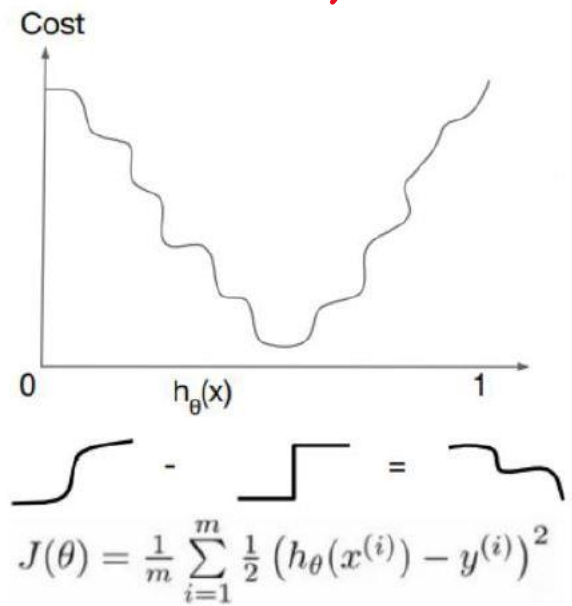
RMSE로 확률 예측?



RMSE로 확률 예측?

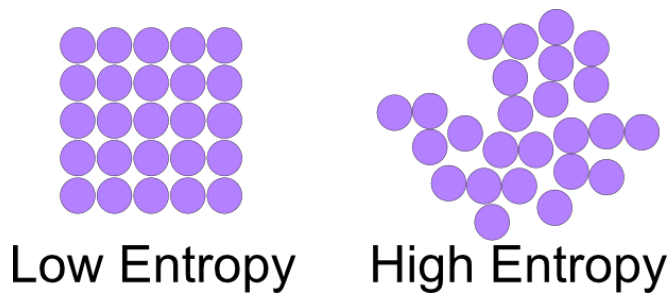


non convex function



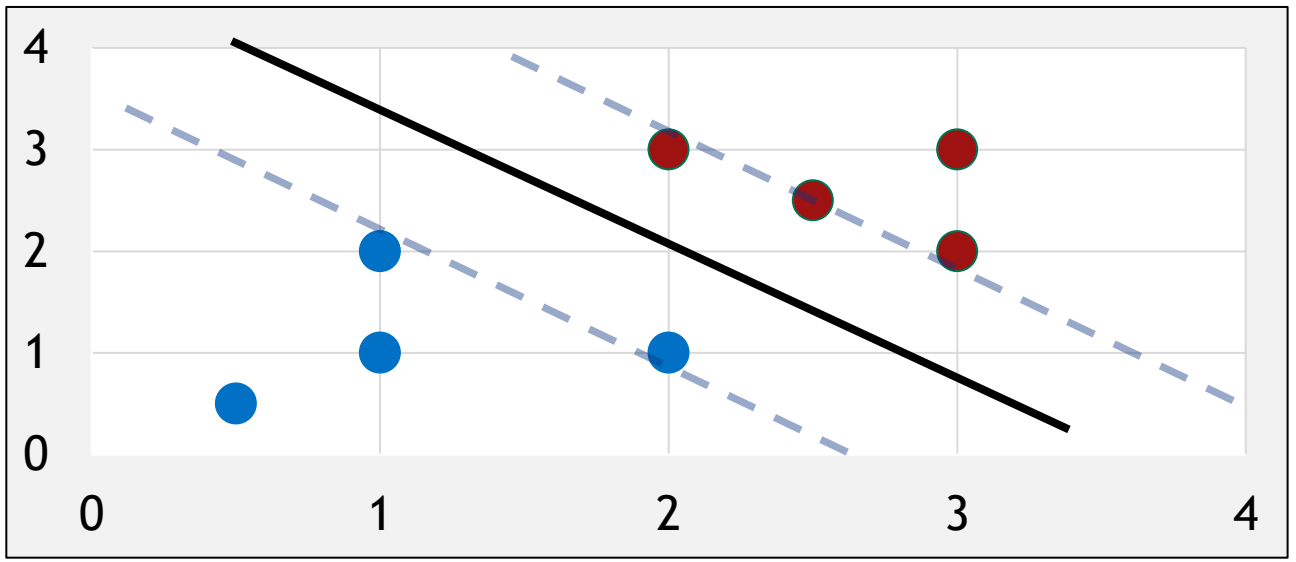
크로스 엔트로피 (Cross Entropy)

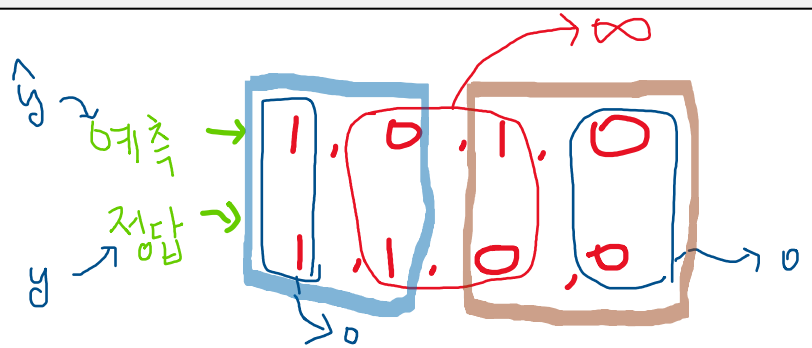
- **엔트로피(ENTROPY)**
  - 무질서 측도
  - 값이 클수록 데이터가 혼재, 분류가 잘 안된 상태
  - 값이 작을 수록 데이터가 잘 분류된 상태

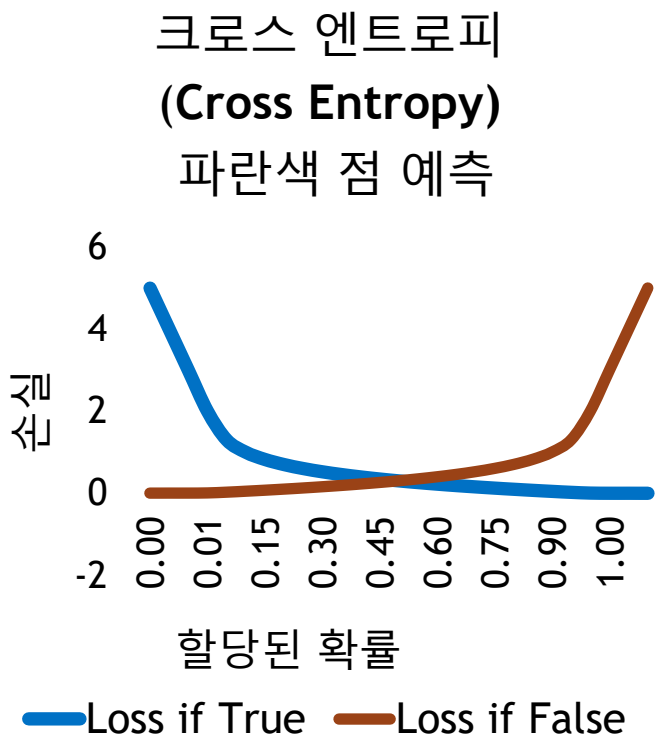


- **크로스 엔트로피(CROSS ENTROPY)**
  - 모델에서 예측한 확률 값이 실제 값과 비교했을 때 틀릴 수 있는 정보량
    - 그 값이 적을 수록 모델이 데이터를 더 잘 예측하는 모델임을 의미
  - 딥러닝에서 예측 값과 정답 값의 크로스 엔트로피 값을 줄이기 위해 가중치(w)와 편향(b)을 업데이트하며 학습(오차 역전파)

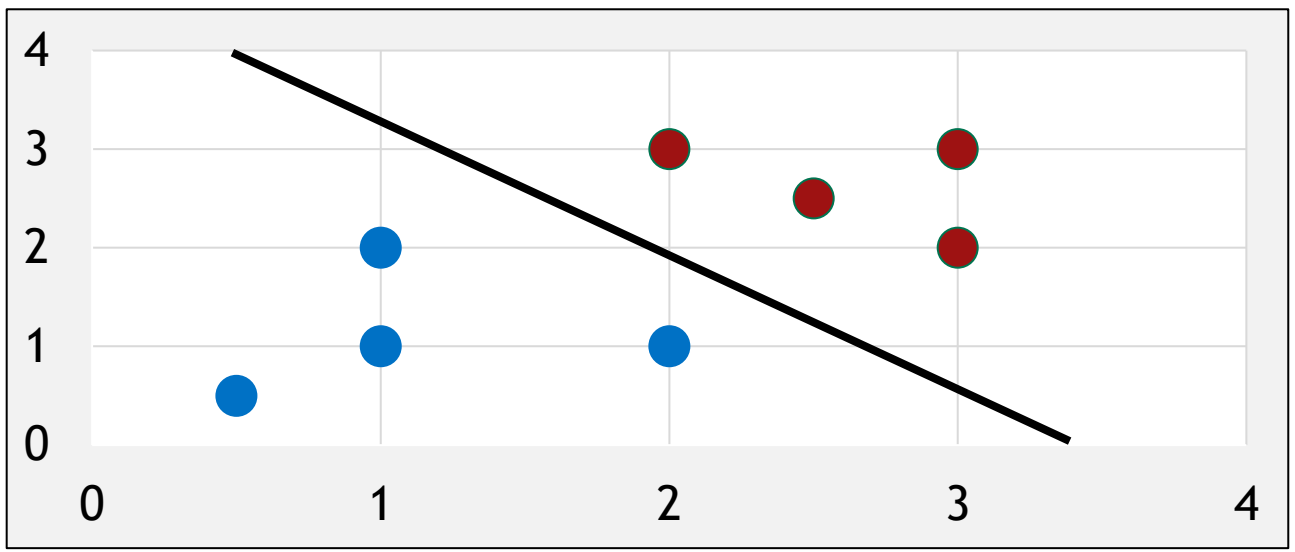
크로스 엔트로피 (Cross Entropy)

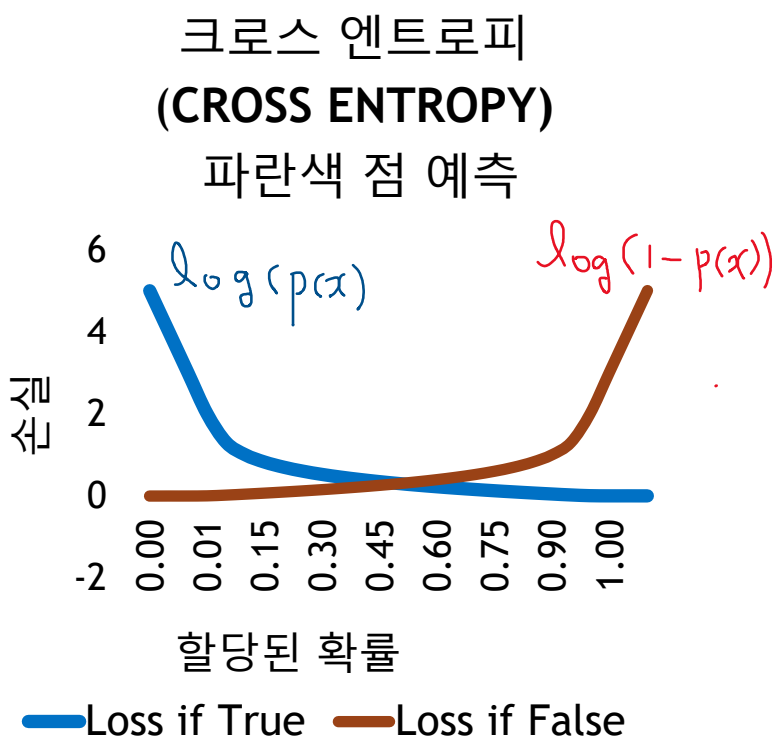






크로스 엔트로피 (Cross Entropy)





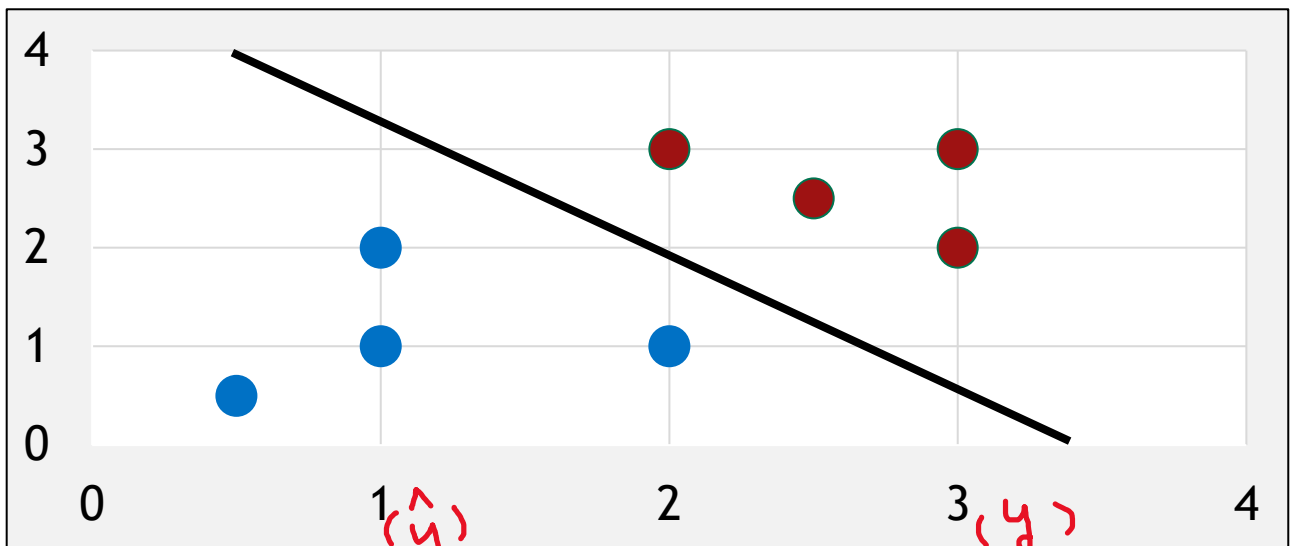
$$Loss = -(t(x) \cdot \log(p(x)) + (1 - t(x)) \cdot \log(1 - p(x)))$$

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

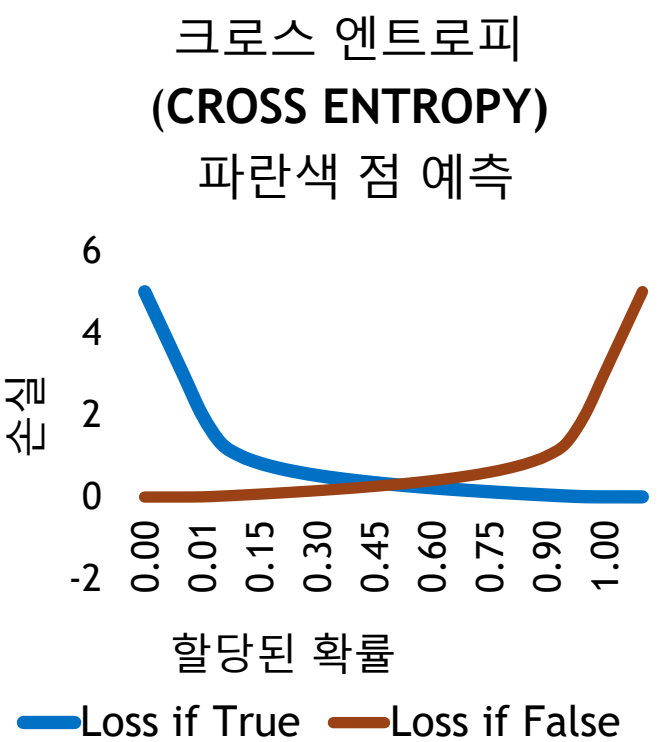
$p(x) = probability\ prediction\ of\ point\ x$ ← $\hat{y}$

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

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
```



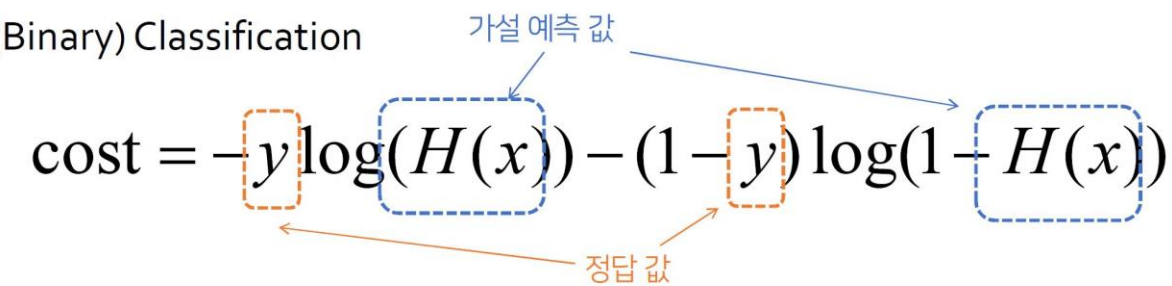
크로스 엔트로피 (binary) Cross Entropy

Linear Regression

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

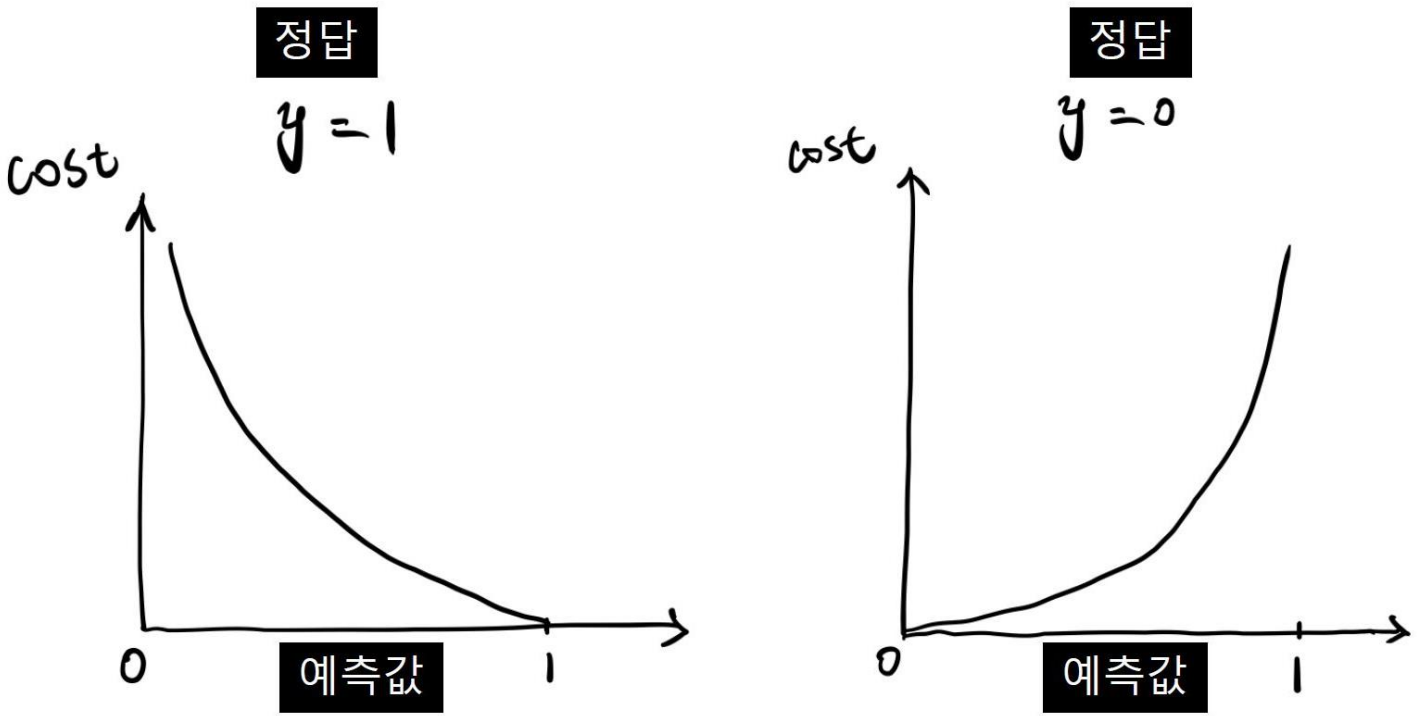
가설 예측 값 $\hat{y}(y_{pred})$ 정답 값

(Binary) Classification



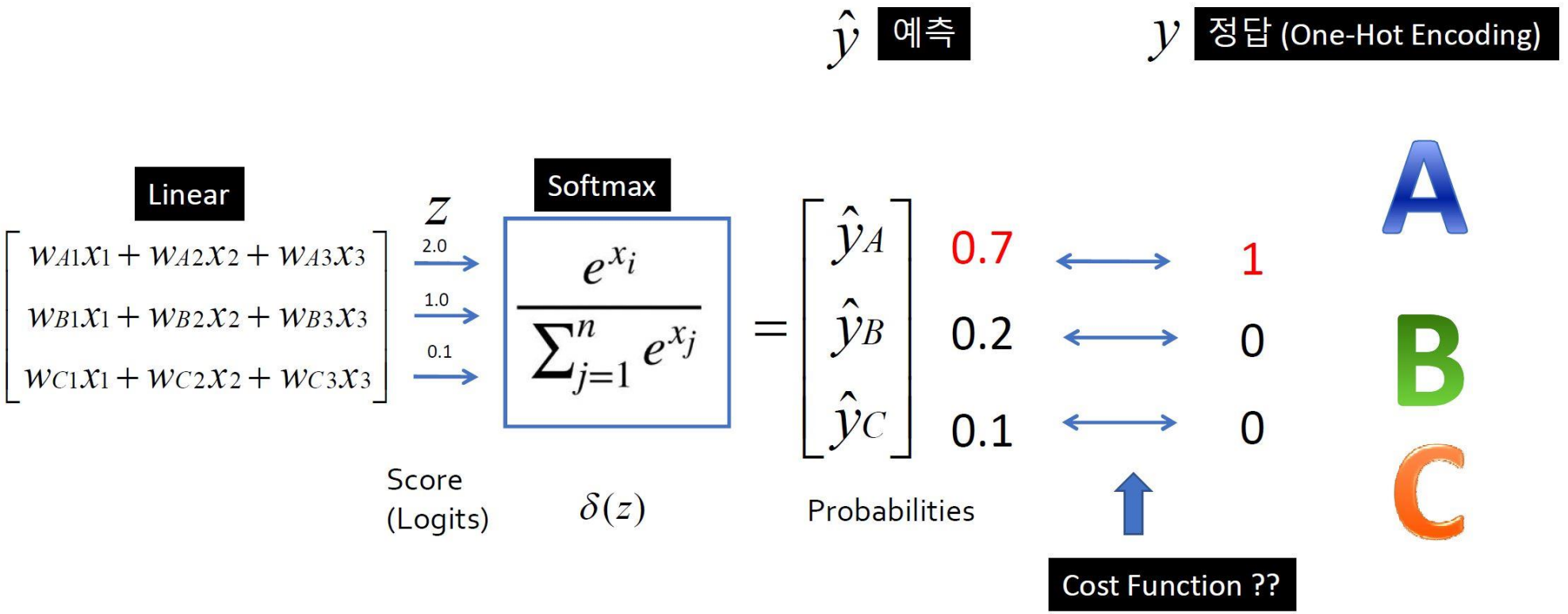
$Y_{actual/\ target/\ label}$	y_{pred}	Loss(cost)	Loss(cost)
1	1	log(1)	0
0	0	log(1)	0
1	0	log(0)	∞
0	1	log(0)	∞

크로스 엔트로피 (binary) Cross Entropy



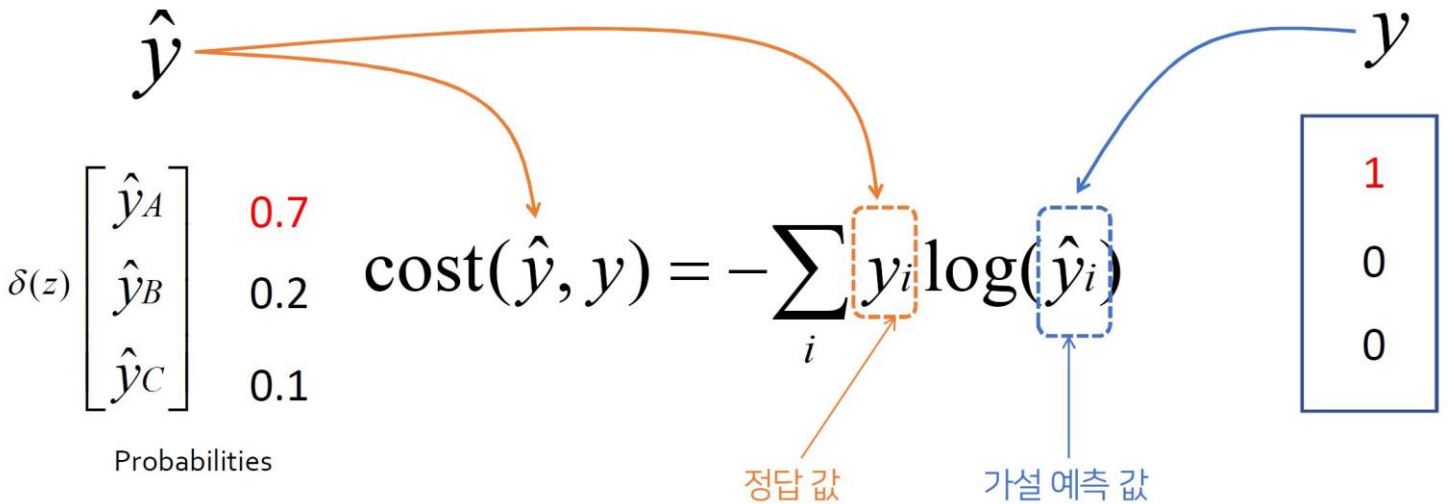
$$\text{cost}(H(x), y) = \begin{cases} -\log(H(x)) & \text{if } y = 1 \\ -\log(1 - H(x)) & \text{if } y = 0 \end{cases}$$

크로스 엔트로피 (Cross Entropy)



크로스 엔트로피 (Cross Entropy)

Cost Function



* 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(mean_square_error)	Gradient Descent (SDG, RMSProp, Adam…)	-
이진 분류	sigmoid	binary_crossentropy	Gradient Descent (SDG, RMSProp, Adam…)	-
다중 분류	softmax	categorical_crossentorpy	Gradient Descent (SDG, RMSProp, Adam…)	범주형 one-hot-encoding
다중 분류	softmax	sparse_categorical_crossentropy	Gradient Descent (SDG, RMSProp, Adam…)	정수형

실습 03 : 이진 분류(binary classification)

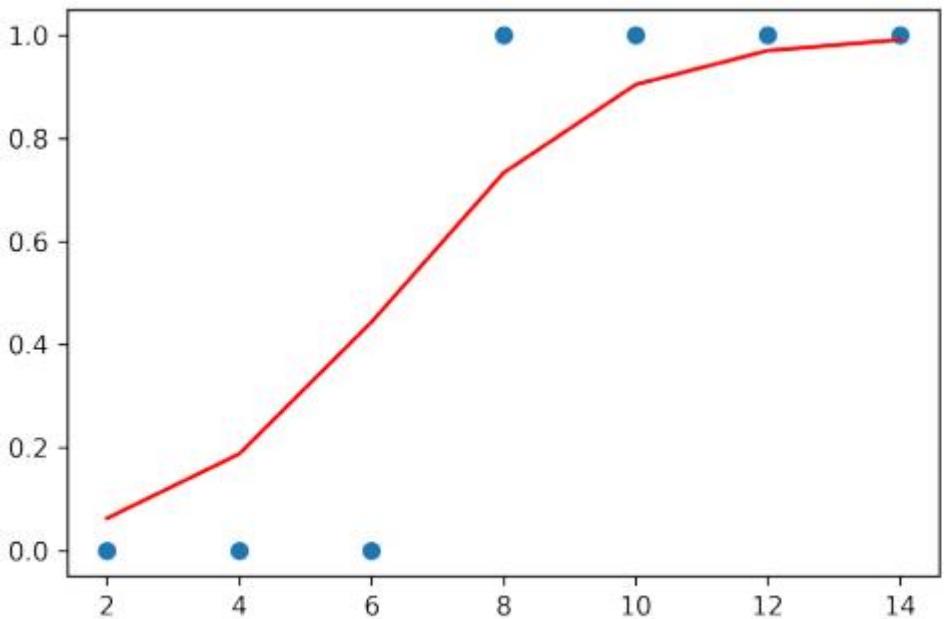
- 공부한 시간에 따른 합격(x), 불합격(y) 예측

```
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 03 : 이진 분류

- 공부한 시간에 따른 합격(x), 불합격(y) 예측



7시간을 공부할 경우, 합격 예상 확률은 59.7%입니다.

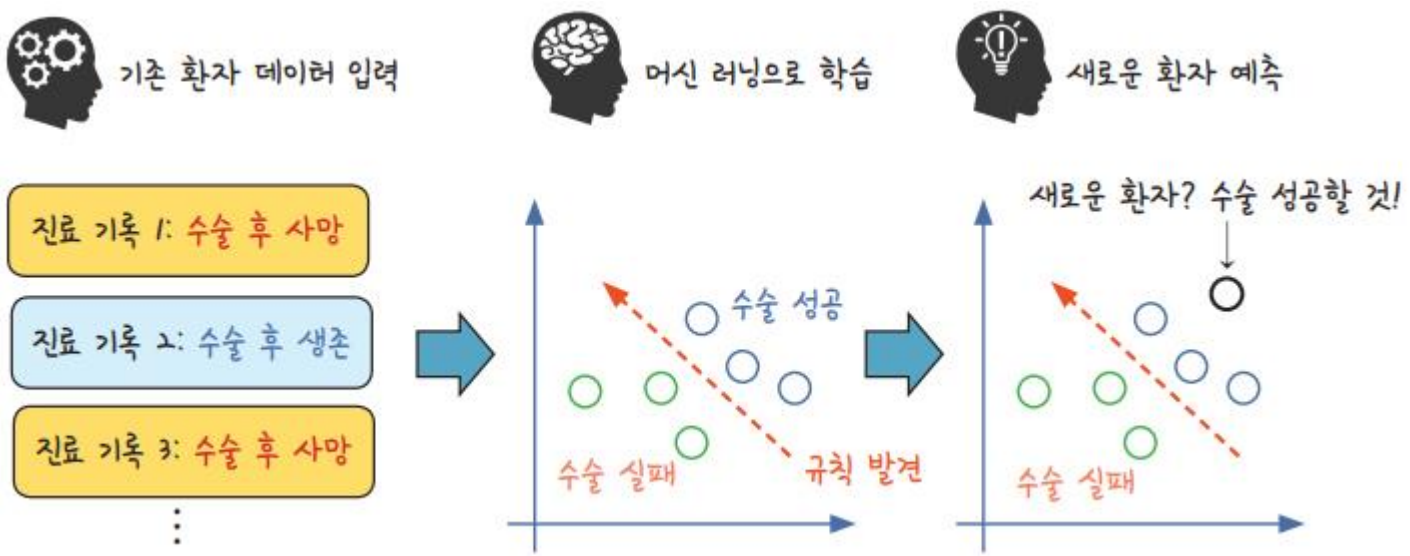
실습 04 : 이진 분류

- 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 05 : 이진 분류

- 폐암 수술 환자 생존율 예측하기

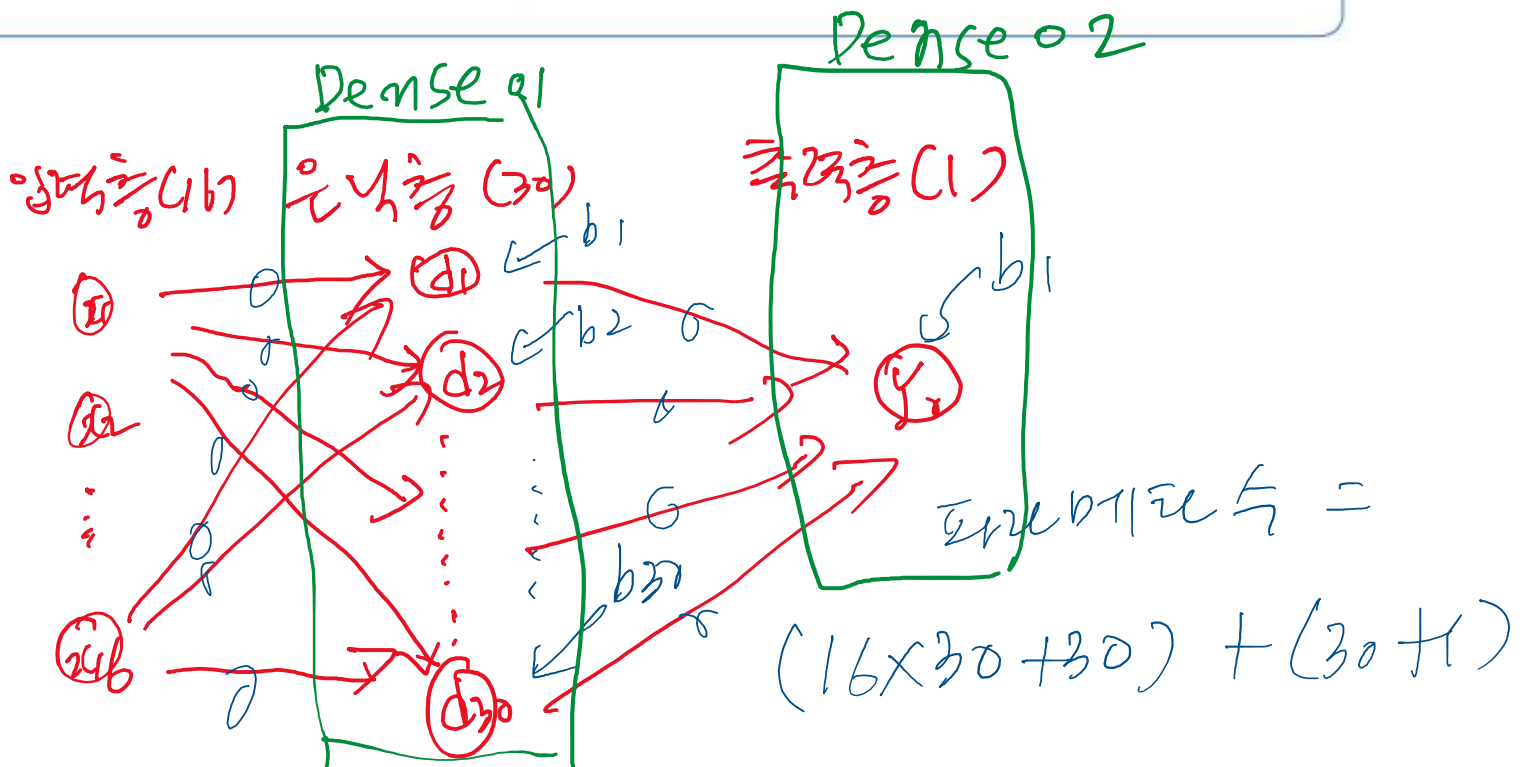


- colab 실습 코드 :
  - https://url.kr/8lg1m2

실습 05 : 네트워크 구조

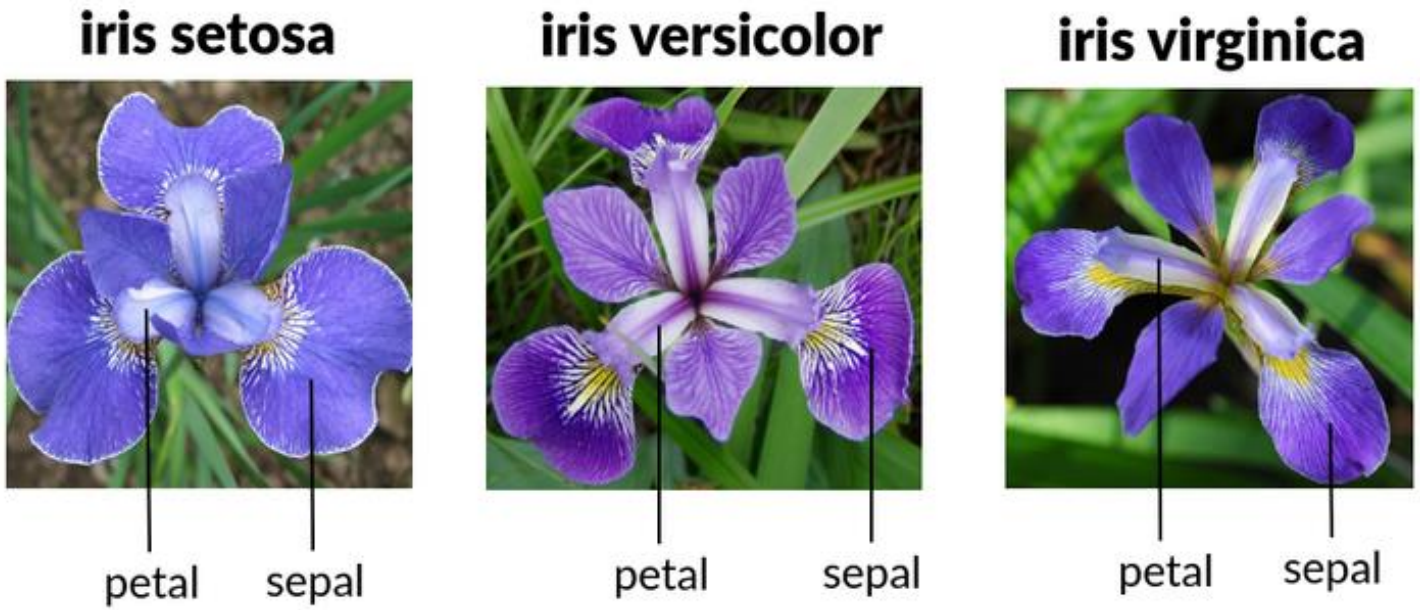
- 입력층, 은닉층, 출력층

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



실습 06 : 다중 분류

- 아이리스(Iris) 품종 분류



- colab 실습 코드 :
  - https://url.kr/a4zdin

실습 06 : 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]

입력데이터

y = df.iloc[:, 4]

정답

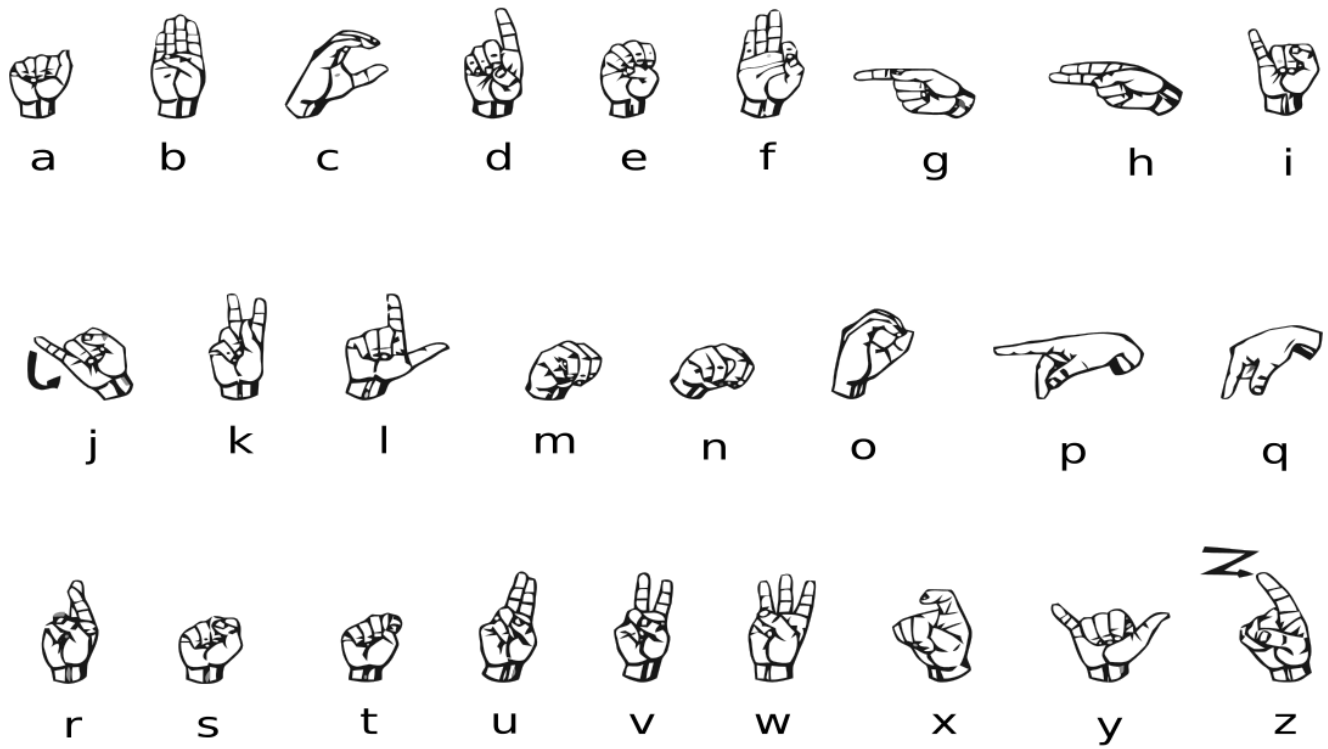
실습 06 : 특성(feature)

- 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

종합 정리

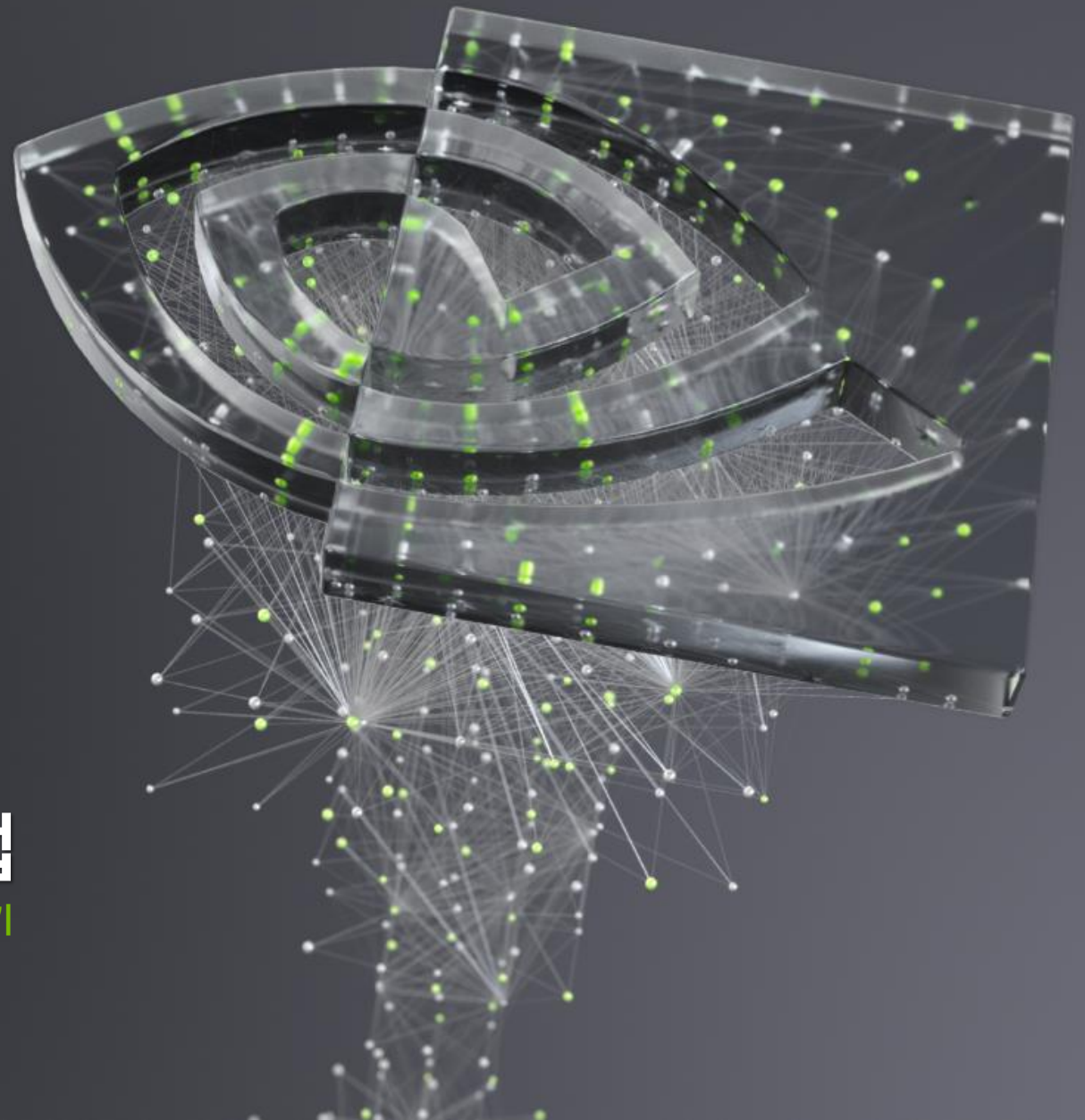
연습 문제

미국 수화 알파벳




시작하겠습니다!

NVIDIA | DEEP LEARNING INSTITUTE



부록: 그래디언트 하강법

컴퓨터가 미적분에 대한 편법을 사용할 수 있도록 돕기

에러를 통한 학습

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

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

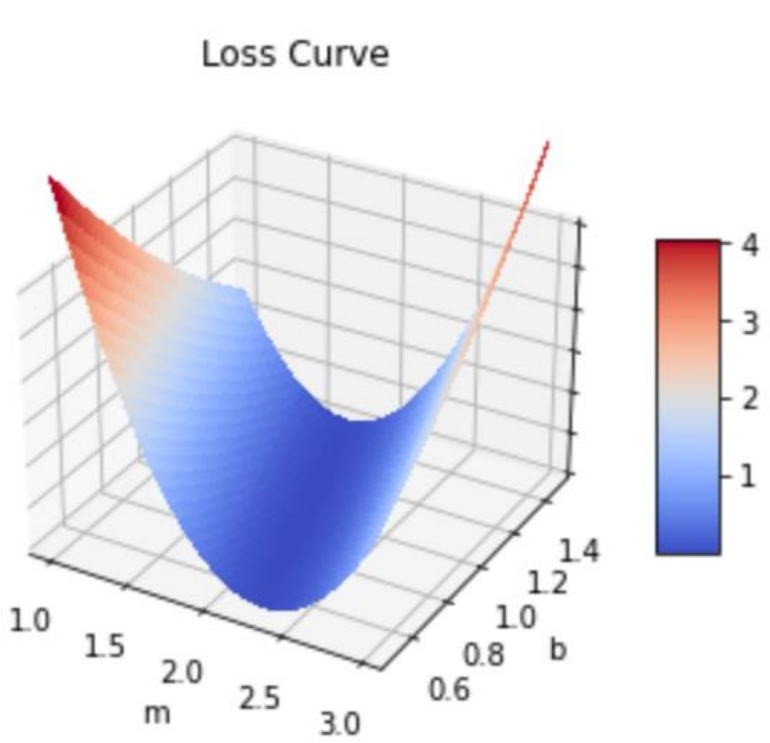
$$\frac{\partial MSE}{\partial m} = 9m + 5b - 23 \qquad \frac{\partial MSE}{\partial b} = 5m + 3b - 13$$

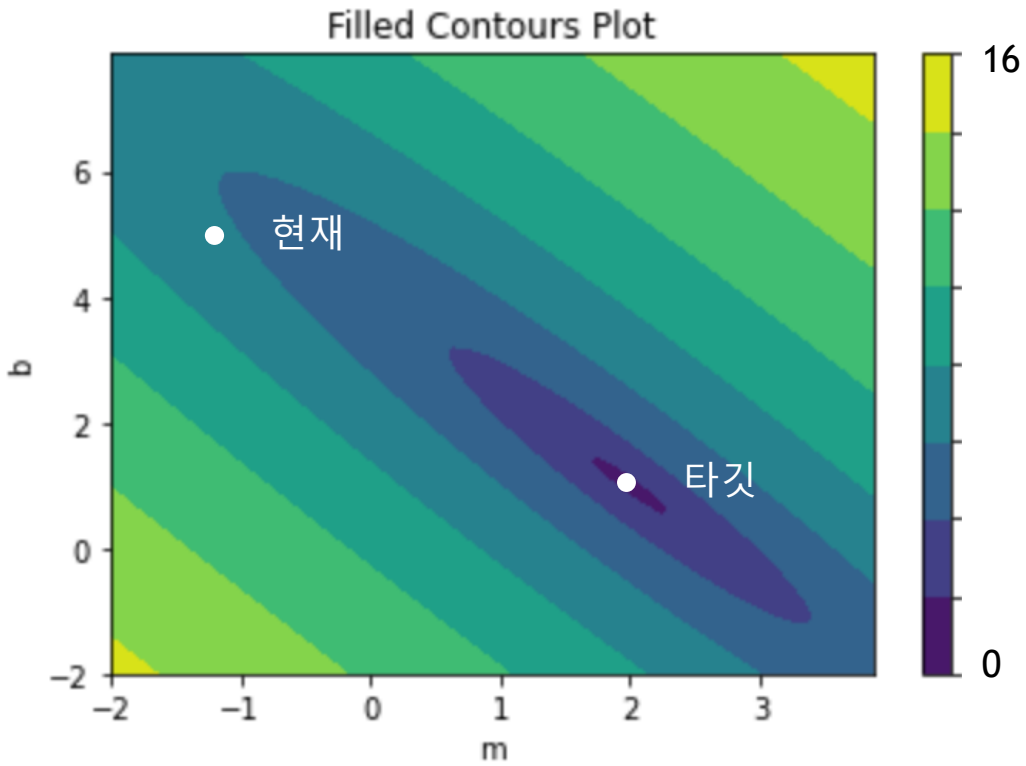
$$\frac{\partial MSE}{\partial m} = -7 \qquad \frac{\partial MSE}{\partial b} = -3$$

$m = -1$

$b = 5$

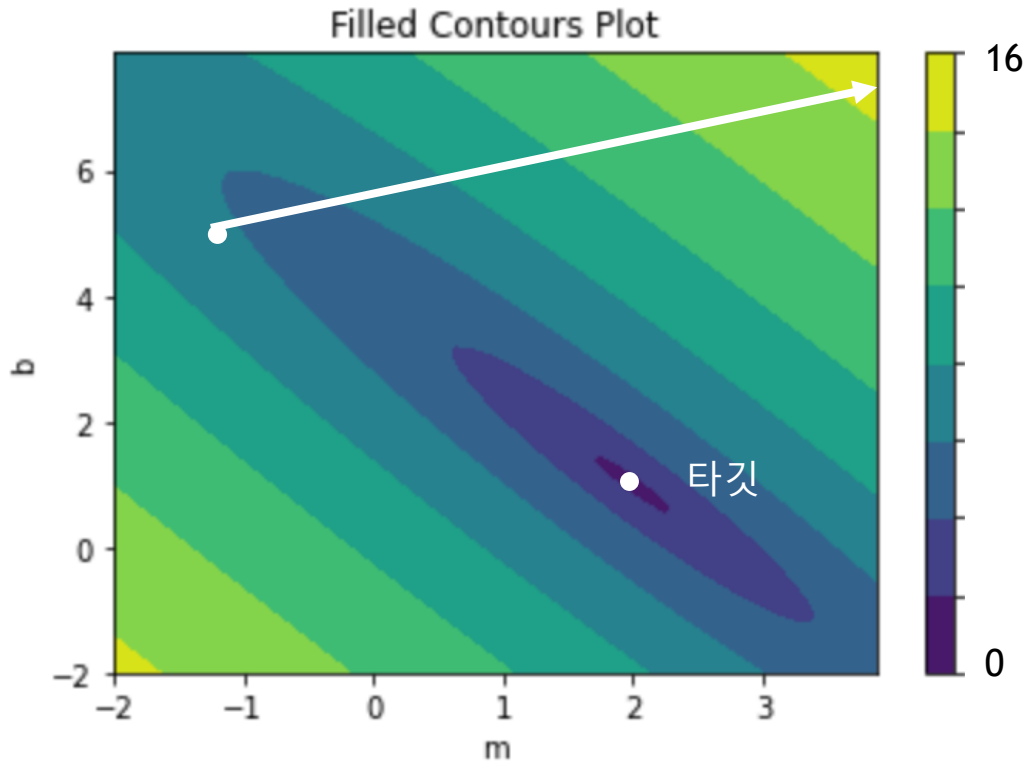
손실 곡선 (The Loss Curve)





손실 곡선 (The Loss Curve)

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

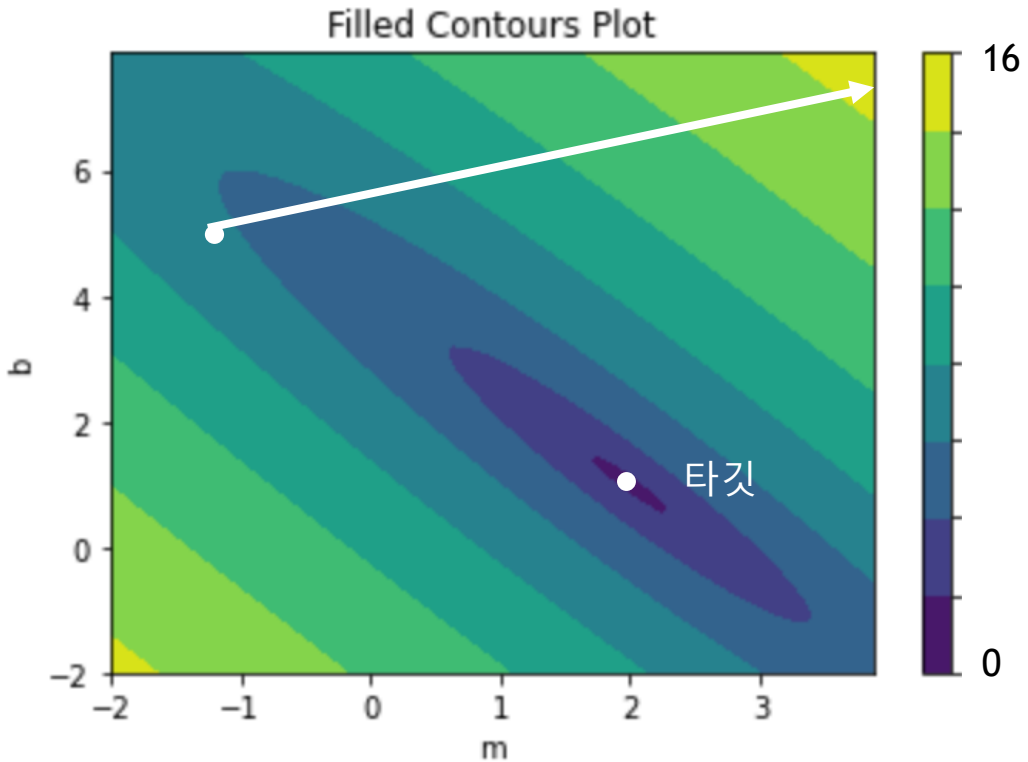


손실 곡선 (The Loss Curve)

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

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

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



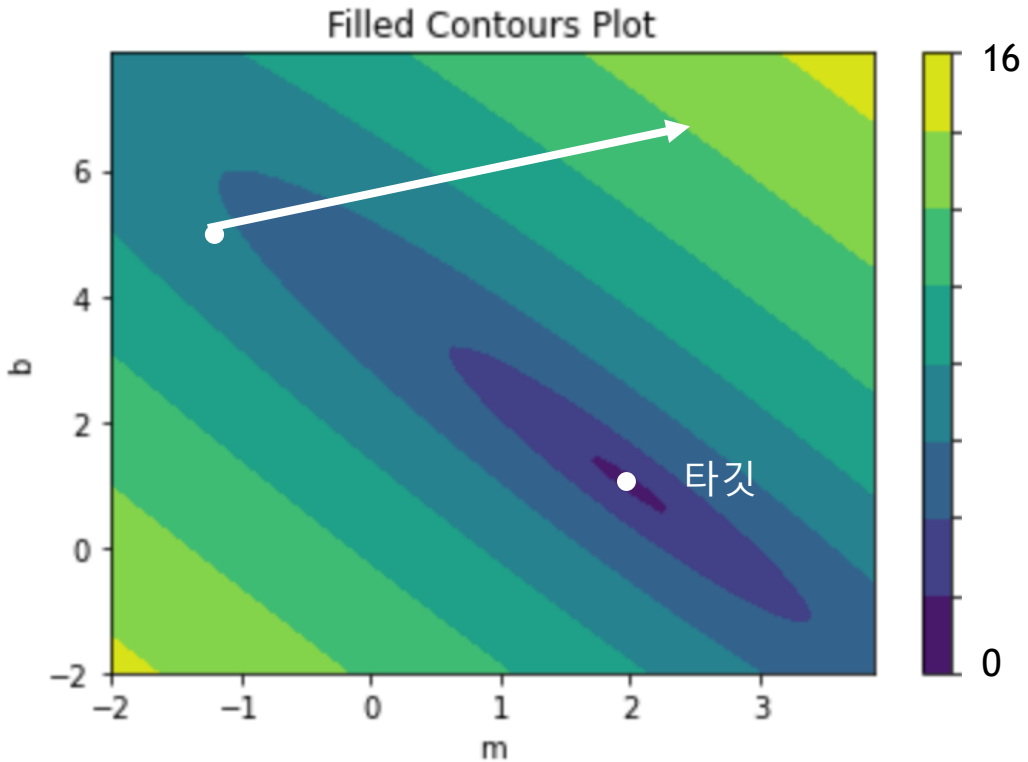
손실 곡선 (The Loss Curve)

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

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

$$\lambda = .5$$

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



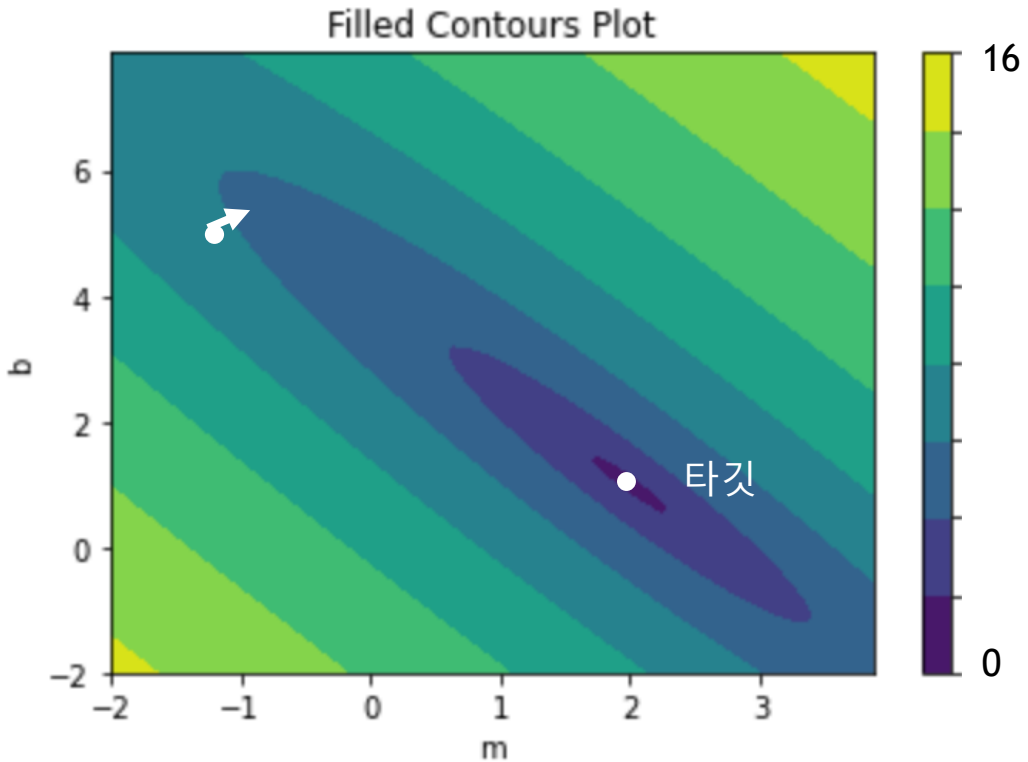
손실 곡선 (The Loss Curve)

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

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

$$\lambda = .005$$

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

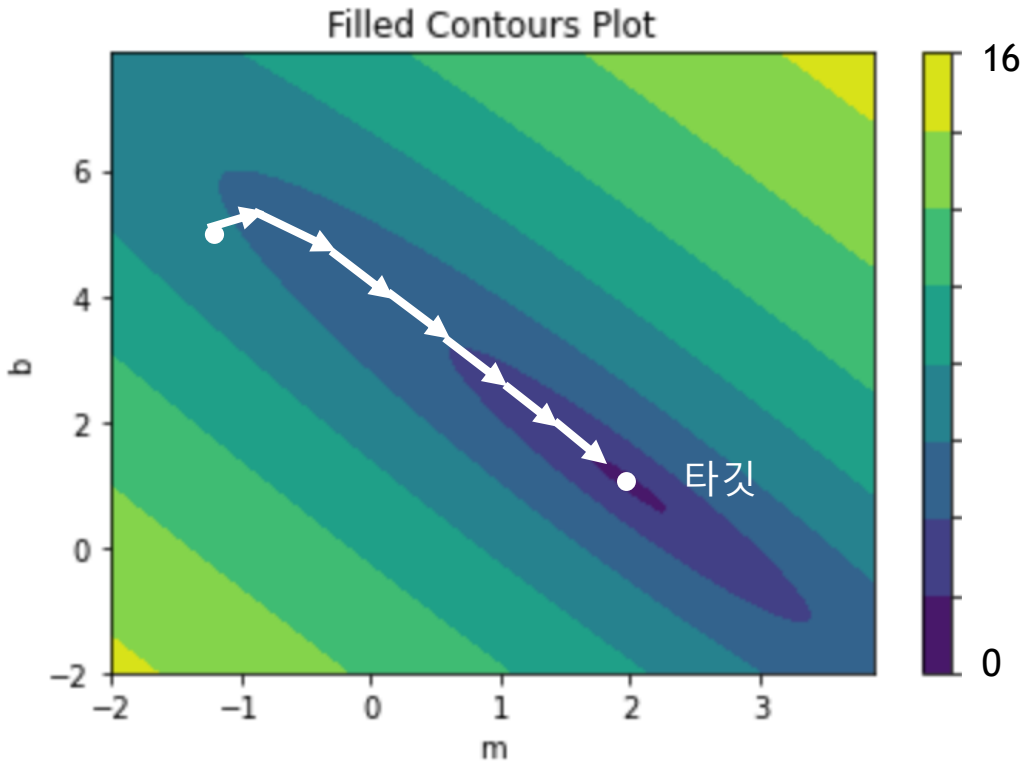


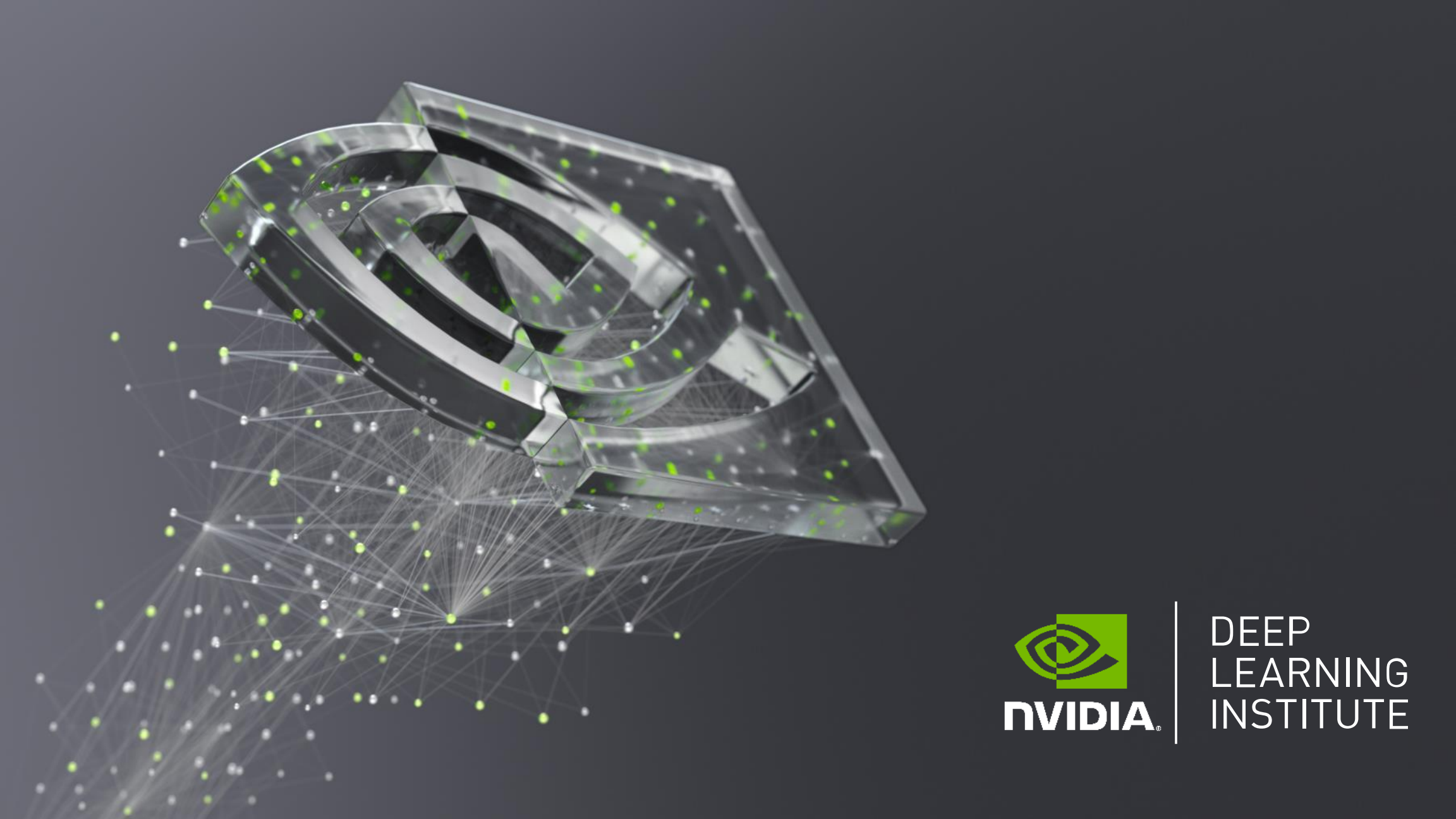
손실 곡선 (The Loss Curve)

$\lambda = .1$

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

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




NVIDIA
DEEP
LEARNING
INSTITUTE