5. Neural Network Training

문준원

mppn98@g.skku.edu

TNT ML Team

2024/09/26

TRAIN AND TEST

Contents

TRAIN AND TEST

Learning Objectives

Week 5 : Neural Network Training

이번 주에는 텐서플로에서 모델을 훈련하는 방법과 시그모이드 함수 외에 다른 중요한 활성화 함수와 신경망에서 각 유형을 어디에 사용하는지에 대해 알아봅니다. 또한 이진 분류를 넘어 다중 클래스 분류(3개 이상의 범주)로 넘어가는 방법도 배우게 됩니다. 다중 클래스 분류에서는 새로운 활성화 함수와 새로운 손실 함수를 소개합니다. 선택 사항으로 다중 클래스 분류와 다중 레이블 분류의 차이점에 대해서도 배울 수 있습니다. 아담 옵티마이저에 대해 알아보고, 이것이 신경망 훈련에 있어 일반 경사 하강을 개선한 이유에 대해서도 알아봅니다. 마지막으로 지금까지 살펴본 레이어 유형 외에 다른 레이어 유형에 대해 간략하게 소개합니다.

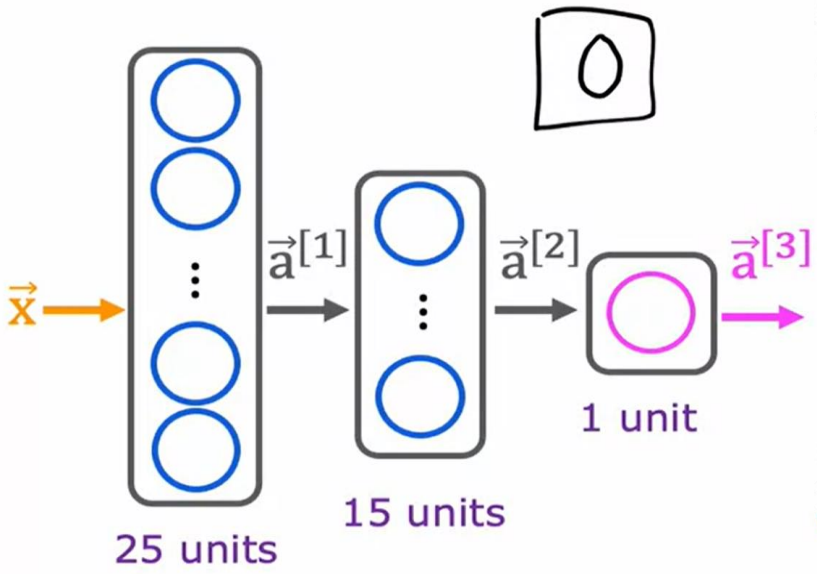
학습 목표

- TensorFlow를 사용하여 데이터에서 신경망 훈련하기
- 다양한 활성화 함수(시그모이드, ReLU, 선형)의 차이점을 이해합니다
- 어떤 유형의 레이어에 어떤 활성화 기능을 사용할지 이해합니다
- 비선형 활성화 함수가 필요한 이유 이해하기
- 멀티클래스 분류 이해
- 다중 클래스 분류를 구현하기 위한 소프트맥스 활성화 계산하기
- 다중 클래스 분류에 범주형 교차 엔트로피 손실 함수 사용
- 코드에서 다중 클래스 분류를 구현하는 데 권장되는 방법을 사용하세요
- (선택 사항): 다중 레이블과 다중 클래스 분류의 차이점 설명하기

TRAIN AND TEST

Neural Network Training

Train a Neural Network in TensorFlow




```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
   model = Sequential([
     Dense(units=25, activation='sigmoid'),
     Dense(units=15, activation='sigmoid'),
     Dense(units=1, activation='sigmoid'),
                          ])
from tensorflow.keras.losses import
BinaryCrossentropy
   model.compile(loss=BinaryCrossentropy())
   model.fit(X,Y,epochs=100)
```

① ② ③

epochs: number of steps in gradient descent

Given set of (x,y) examples

How to build and train this in code?

Neural Network Training

Model Training Steps

TensorFlow

1. specify how to compute output given input x and parameters w,b (define model)

$f_{\vec{w},b}(\vec{x})$ =?

2. specify loss and cost

$L(f_{\vec{w},b}(\vec{x}), y)$ 1 example

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)})$$

3. Train on data to minimize $J(\vec{w}, b)$

logistic regression

```
z = np.dot(w,x)+ b

f_x = 1/(1+np.exp(-z))
```

logistic loss

```
loss = -y * np.log(f_x)
   -(1-y) * np.log(1-f_x)
```

```
w = w - alpha * dj_dw
b = b - alpha * dj_db
```

neural network

```
model = Sequential([
        Dense(...)
        Dense(...)
        Dense(...)
                ])
```

binary cross entropy

```
model.compile(
loss=BinaryCrossentropy())
```

```
model.fit(X,y,epochs=100)
```

TRAIN AND TEST

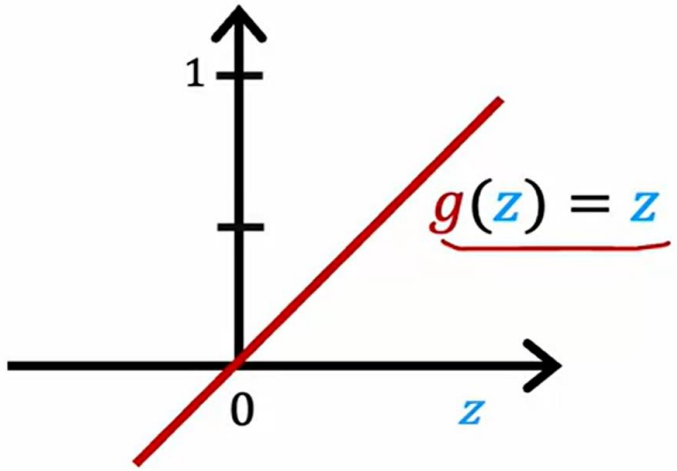
Activation Functions

Examples of Activation Functions

$$a_2^{[1]} = g(\underbrace{\vec{w}_2^{[1]} \cdot \vec{x} + b_2^{[1]}}_{z})$$

"No activation function"

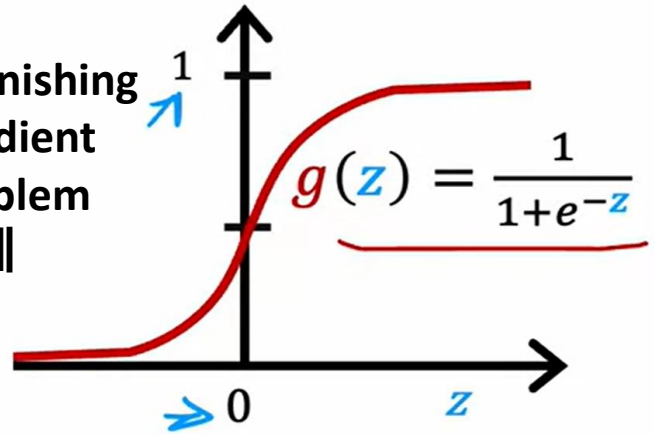
Linear activation function



$a = g(z) = \underbrace{\vec{w} \cdot \vec{x} + b}_{z}$

Sigmoid

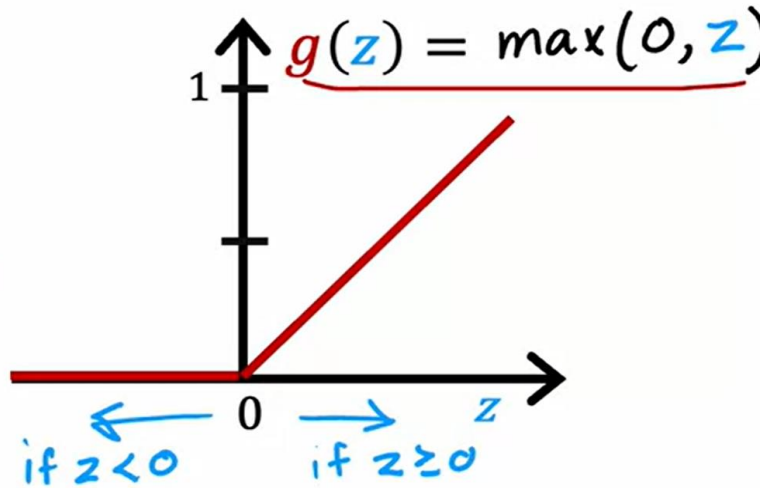
*Vanishing Gradient Problem 문제



$0 < g(z) < 1$

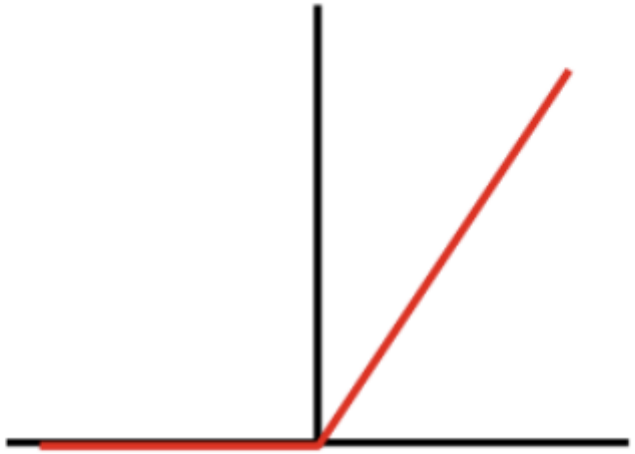
Later: softmax activation

ReLU Rectified Linear Unit



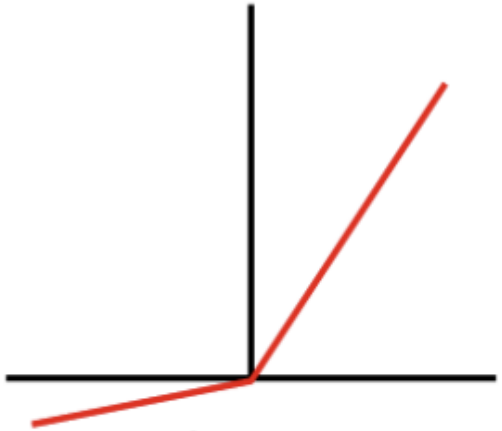
TRAIN AND TEST

Activation Functions



ReLU

$$\text{ReLU} = \begin{cases} x, if\ x \geq 0 \\ 0, if\ x < 0 \end{cases}$$



Leaky ReLU

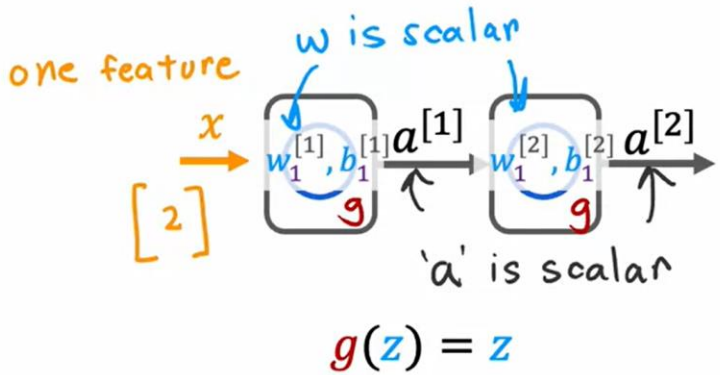
$$\text{Leaky ReLU} = \begin{cases} x, & if\ x \geq 0 \\ 0.01x, if\ x < 0 \end{cases}$$

TRAIN AND TEST

Activation Functions

Why do we need activation functions?

Linear Example



$$a^{[1]} = w_1^{[1]} x + b_1^{[1]}$$

$$a^{[2]} = w_1^{[2]} a^{[1]} + b_1^{[2]}$$

$$= w_1^{[2]} (w_1^{[1]} x + b_1^{[1]}) + b_1^{[2]}$$

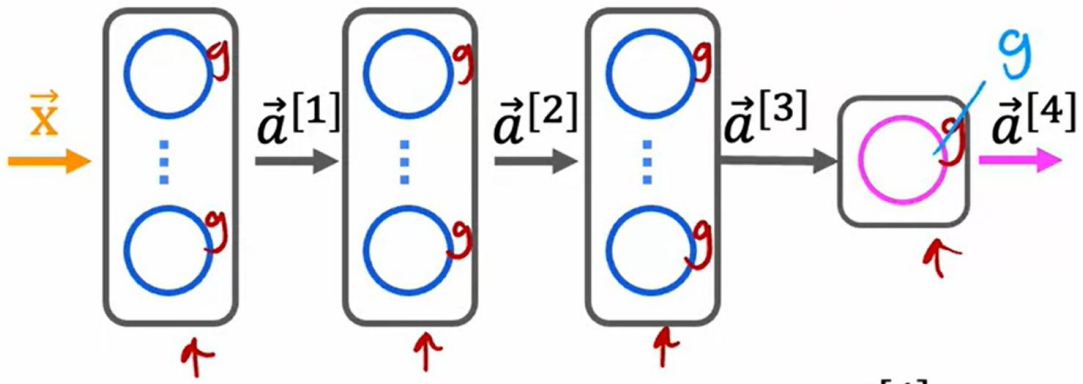
$$\vec{a}^{[2]} = (\underbrace{\vec{w}_1^{[2]} \vec{w}_1^{[1]}}_{w}) x + \underbrace{w_1^{[2]} b_1^{[1]} + b_1^{[2]}}_{b}$$

$$\vec{a}^{[2]} = w x + b$$

$f(x) = wx + b$ linear regression

Activation Functions

Example



$g(z) = z$

$$\vec{a}^{[4]} = \vec{w}_1^{[4]} \cdot \vec{a}^{[3]} + b_1^{[4]}$$

all linear (including output)
↳ equivalent to linear regression

$$\vec{a}^{[4]} = \frac{1}{1+e^{-(\vec{w}_1^{[4]} \cdot \vec{a}^{[3]} + b_1^{[4]})}}$$

output activation is sigmoid
(hidden layers still linear)
↳ equivalent to logistic regression

Don't use linear activations in hidden layers (use ReLU)

TRAIN AND TEST

Multiclass Classification

MNIST example

$y = 0$ 1 2 3 4 5 6 7 8 9

$y = 7$

multiclass classification problem:

target y can take on more than two possible values

TRAIN AND TEST

Multiclass Classification

- softmax

Logistic regression
(2 possible output values)

$z = \vec{w} \cdot \vec{x} + b$

$a_1 = g(z) = \frac{1}{1+e^{-z}} = P(y=1|\vec{x})$ 0.71

$a_2 = 1 - a_1 = P(y=0|\vec{x})$ 0.29

Softmax regression
(N possible outputs) $y = 1, 2, 3, \ldots, N$

$z_j = \vec{w}_j \cdot \vec{x} + b_j \quad j = 1, \ldots, N$

parameters $w_1, w_2, \ldots, w_N$ $b_1, b_2, \ldots, b_N$

$a_j = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}} = P(y=j|\vec{x})$

note: $a_1 + a_2 + \ldots + a_N = 1$

Softmax regression (4 possible outputs) $y = 1, 2, 3, 4$

$z_1 = \vec{w}_1 \cdot \vec{x} + b_1$

$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=1|\vec{x})$ 0.30

$z_2 = \vec{w}_2 \cdot \vec{x} + b_2$

$a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=2|\vec{x})$ 0.20

$z_3 = \vec{w}_3 \cdot \vec{x} + b_3$

$a_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=3|\vec{x})$ 0.15

$z_4 = \vec{w}_4 \cdot \vec{x} + b_4$

$a_4 = \frac{e^{z_4}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}} = P(y=4|\vec{x})$ 0.35

TRAIN AND TEST

Multiclass Classification

- softmax

$$\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Cost

Logistic regression

$$z = \vec{w} \cdot \vec{x} + b$$

$$a_1 = g(z) = \frac{1}{1+e^{-z}} = P(y=1|\vec{x})$$

$$a_2 = 1 - a_1 = P(y=0|\vec{x})$$

$$loss = -y \log a_1 - (1-y) \log(1-a_1)$$

if $y=1$ if $y=0$

$$J(\vec{w}, b) = \text{average loss}$$

Softmax regression

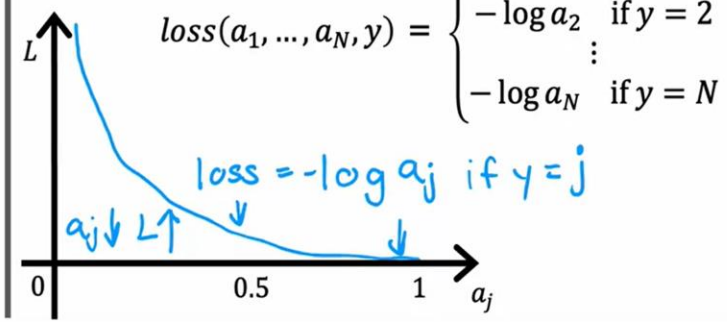
$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \cdots + e^{z_N}} = P(y=1|\vec{x})$$

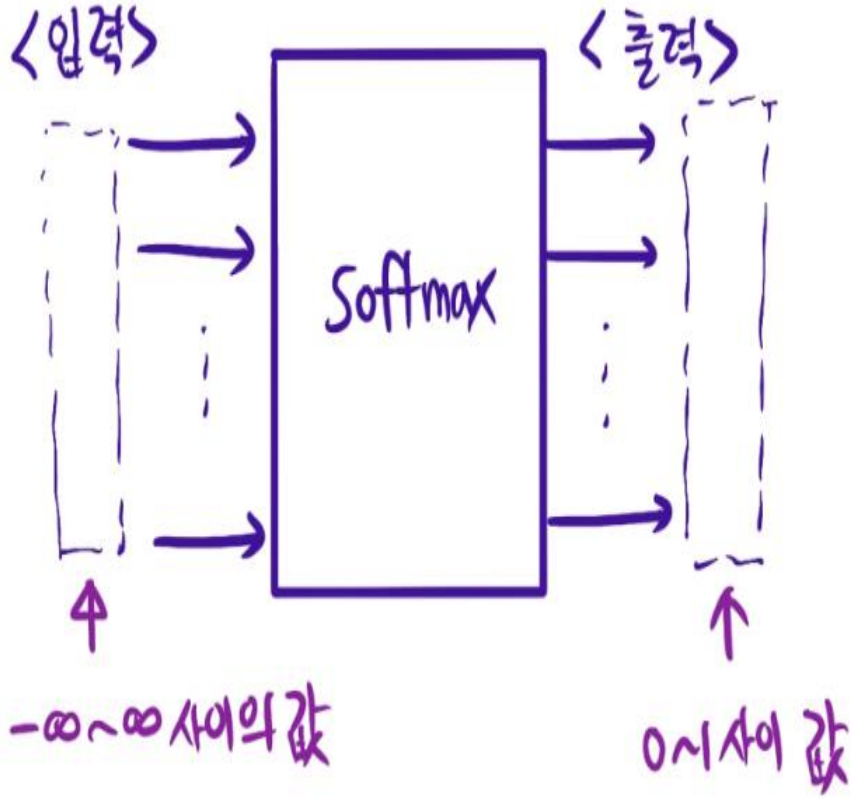
$$\vdots$$

$$a_N = \frac{e^{z_N}}{e^{z_1} + e^{z_2} + \cdots + e^{z_N}} = P(y=N|\vec{x})$$

Crossentropy loss

$$loss(a_1, \ldots, a_N, y) = \begin{cases} -\log a_1 & \text{if } y=1 \\ -\log a_2 & \text{if } y=2 \\ \vdots & \\ -\log a_N & \text{if } y=N \end{cases}$$



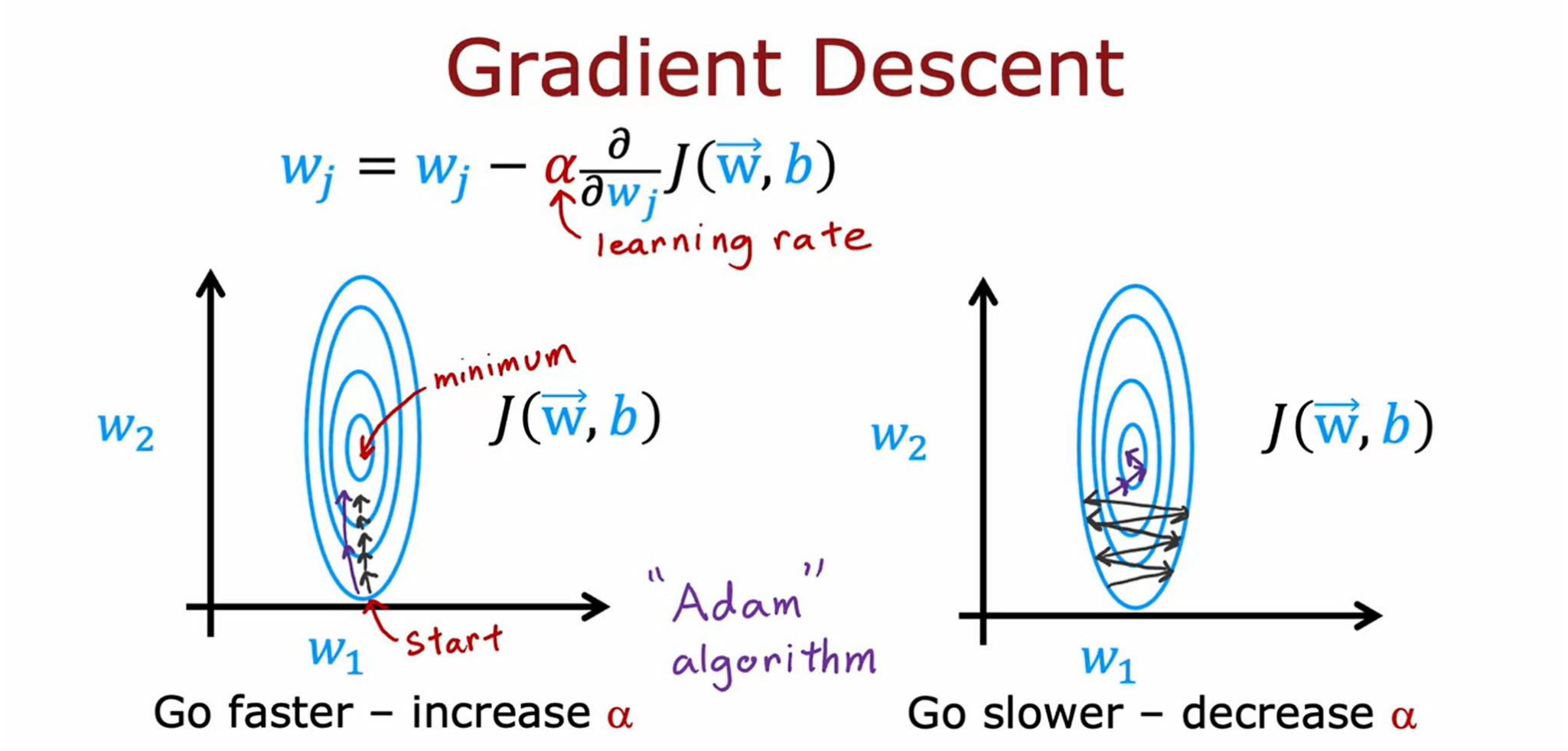


Multiclass Classification

- Where should we use Softmax?

-> By using the Softmax activation function, you can obtain the probabilities that the model calculates for each class. These probability values represent the likelihood that the model believes the user belongs to a particular class, allowing you to provide information such as, “You are 51% likely to be in Class 1.”

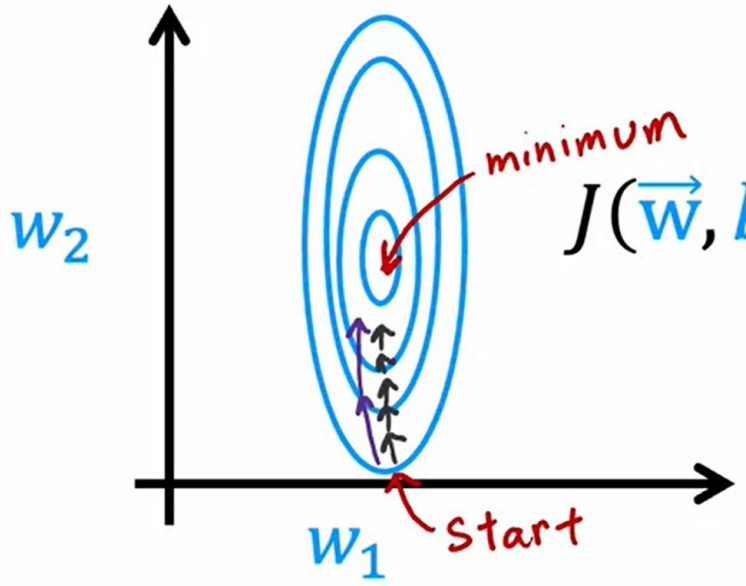
Additional Neural Network Concepts



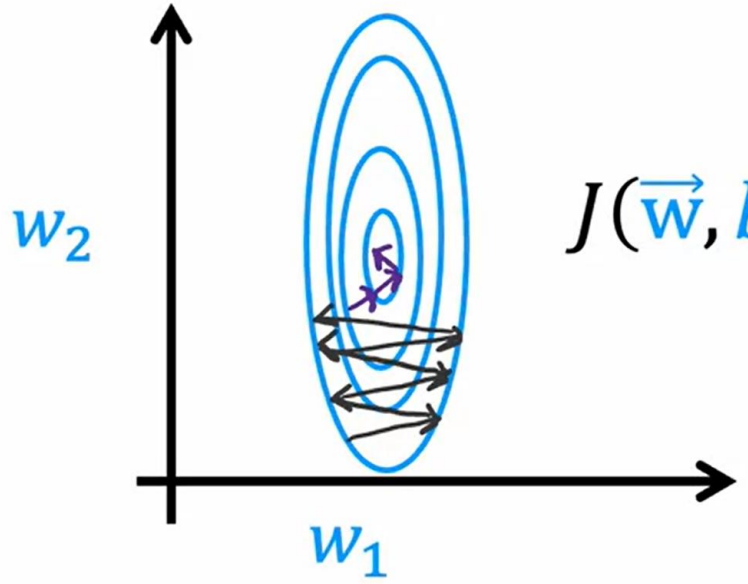
TRAIN AND TEST

Additional Neural Network Concepts

Adam Algorithm Intuition



If w_j (or b) keeps moving in same direction, increase α_j.



If w_j (or b) keeps oscillating, reduce α_j.

TRAIN AND TEST

TRAIN AND TEST