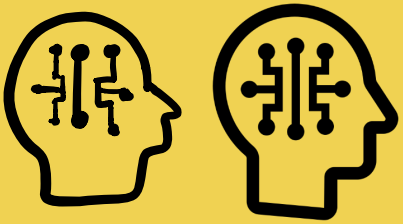


Lecture 4.
Neural Network Basics

Review

1. Intro to ML
2. Linear Classifier

1. Optimization

- SGD vs. ~~Random Search~~
- Analytic Gradient vs. ~~Numeric Gradient~~

2. Backpropagation

- Chain Rule
- Computational Graph
  - scalar
  - vector
  - matrix
- Implementation

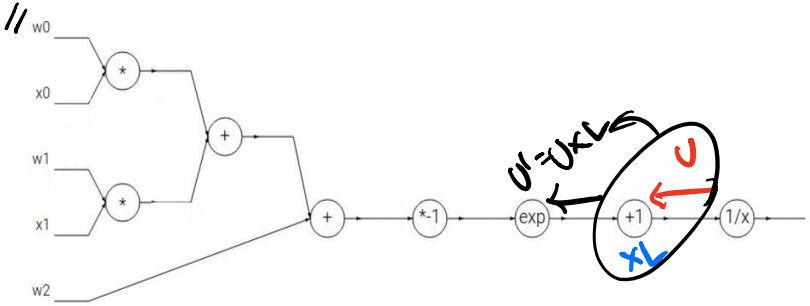
Review

$$w_{new} = w_{old} - lr \cdot \frac{\partial L}{\partial w}\Big|_{w = w_{old}}$$

$$b_{new} = b_{old} - lr \cdot \frac{\partial L}{\partial b}\Big|_{b = b_{old}}$$

$$f(w,x) = \frac{1}{1+e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

Backprop.



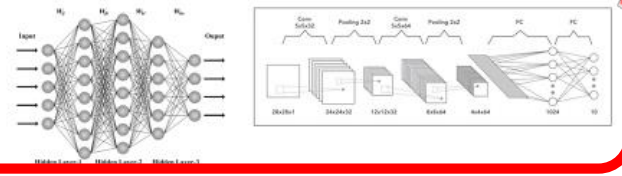
Review

Optimization : Calculating Gradient

IDEA 2. Analytic

지금은 simple linear classifier만 다루고 있지만,

앞으로 다룰 model은



then, how should we derive the gradient?

Multilayer Perceptron

Linear Classifier 단독으로는

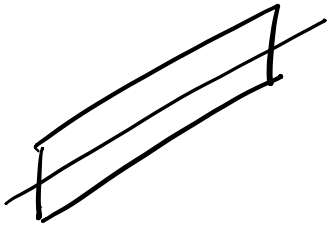
풀 수 없는 문제들 존재...

여러 개를 중첩시켜 보자!

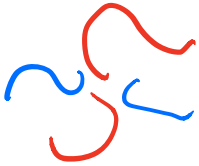
Today's Contents

1. **Limitation of Linear Classifier**
2. **Perceptron**
3. **MLP(Multilayer Perceptron)**
4. **Limitations of MLP**

Limitation of Linear Classifier



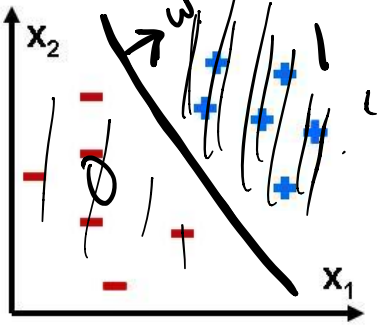
What's wrong with our "Linear" Classifier?



What about Nonlinear Classification Problems?

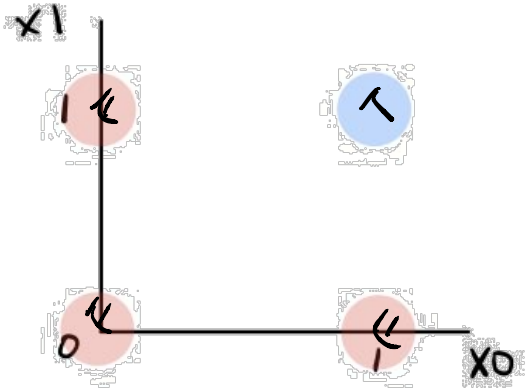
Limitation of Linear Classifier

Binary Classification : AND / OR / XOR Gate with Linear Classifier



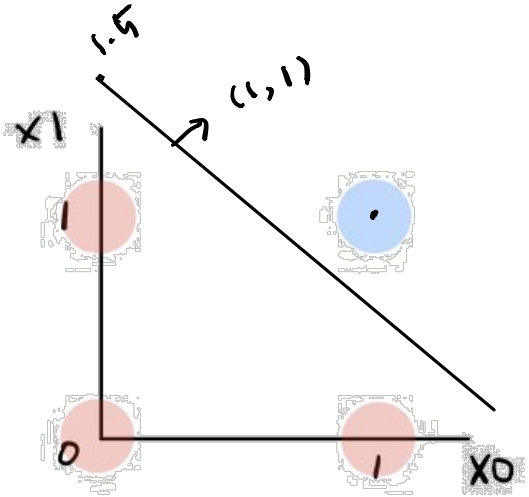
Define h(x) = 1 if x > 0, 0 otherwise

Linear Classification : AND Gate



x0	x1	ANS
1	1	T
1	0	F
0	1	F
0	0	F

Linear Classification : AND Gate



$x_0 + x_1 - 1.5 = 0$

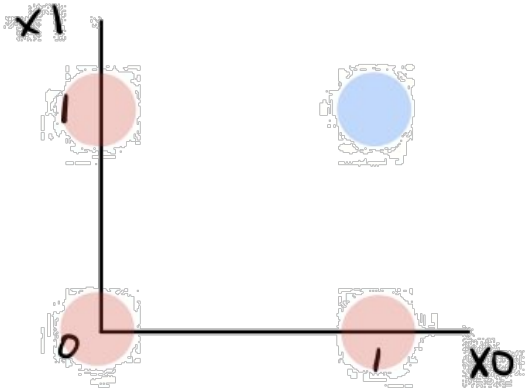
$(1,1) \rightarrow 1+1-1.5 = 0.5$

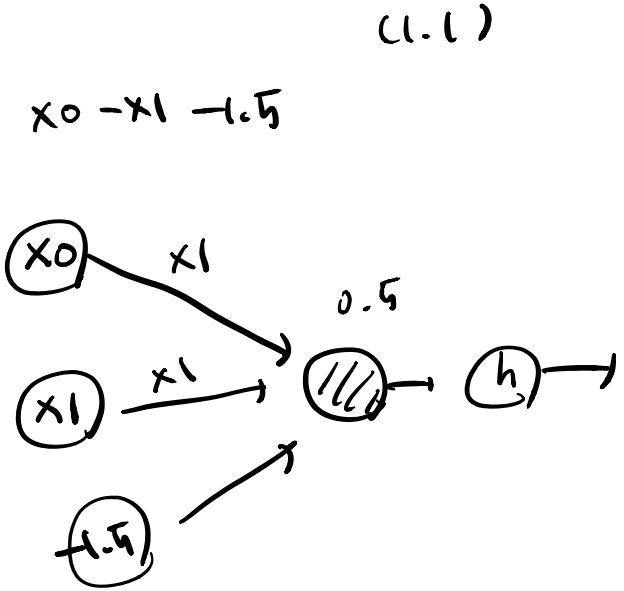
$h(0.5) \rightarrow$ "1." T

$(1,0) \rightarrow 1+0-1.5 = -0.5$

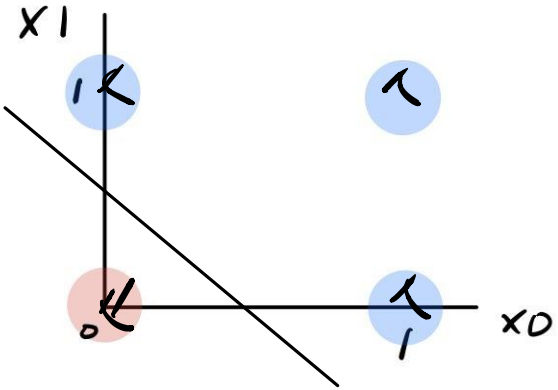
$h(-0.5) =$ "0". T

Linear Classification : AND Gate





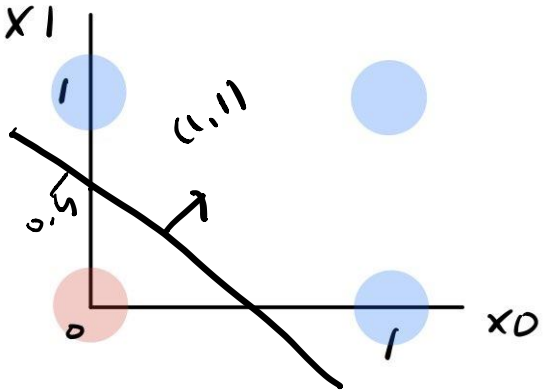
Linear Classification : OR Gate



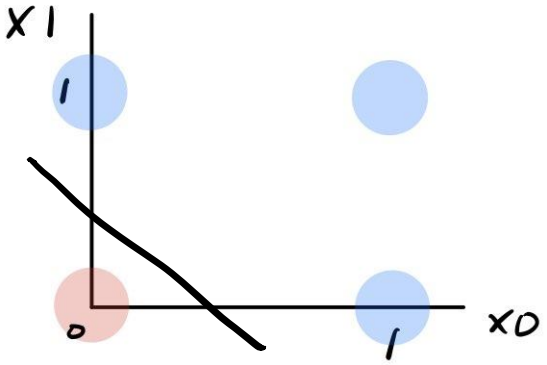
x0	x1	ANS
1	1	T
1	0	T
0	1	T
0	0	F

Linear Classification : OR Gate

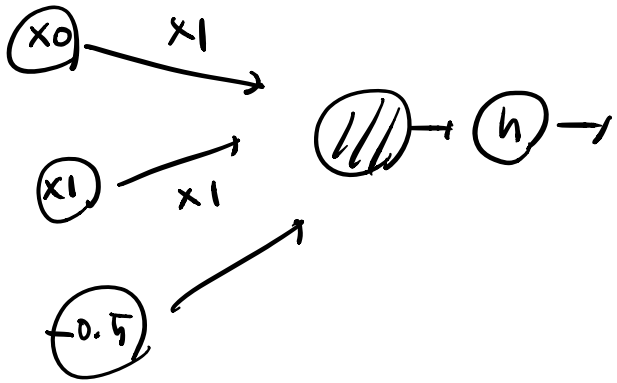
$x_0 + x_1 - 0.5 = 0$



Linear Classification : OR Gate

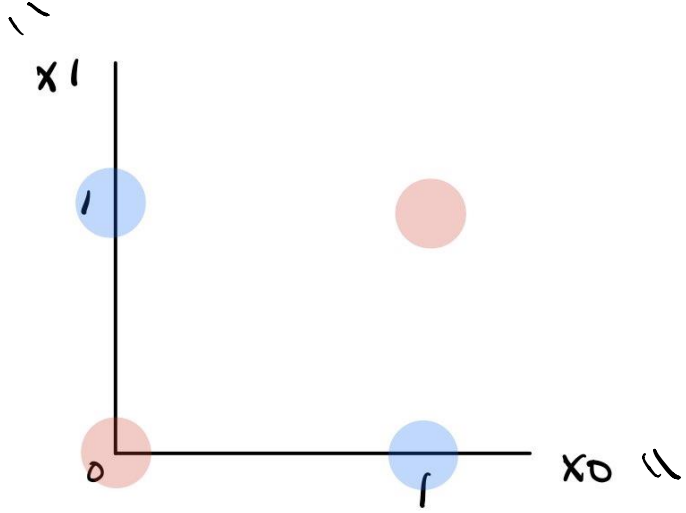


$x_0 + x_1 - 0.5 = 0$



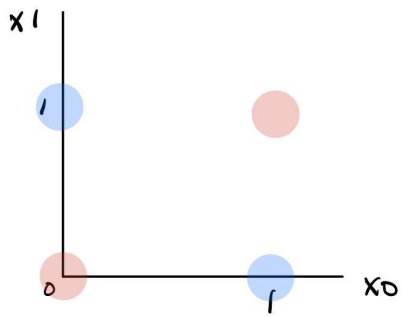
Linear Classification : XOR Gate

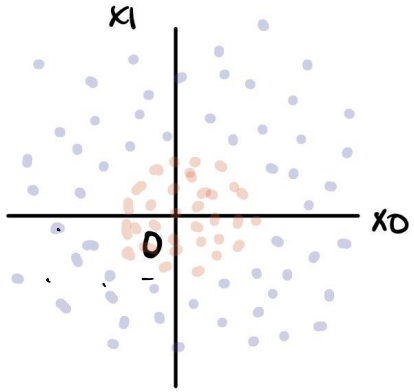
"Exclusive"



x0	x1	ANS
1	1	F
1	0	T
0	1	T
0	0	F

Limitation of Linear Classifier : Overcoming





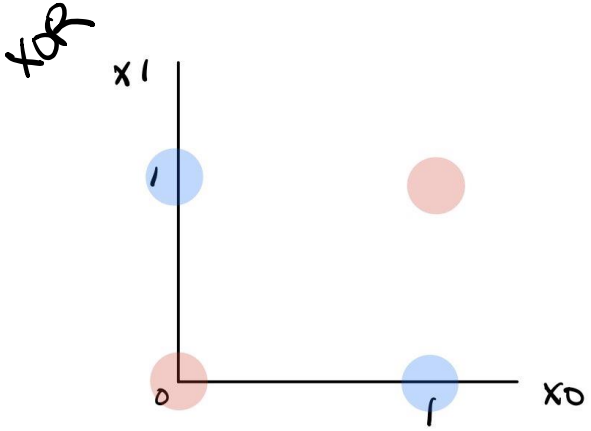
Linear Classifier 단독으로는 풀지 못하는 classification 문제 존재...!

then, how should we classify them?

Limitation of Linear Classifier : Overcoming

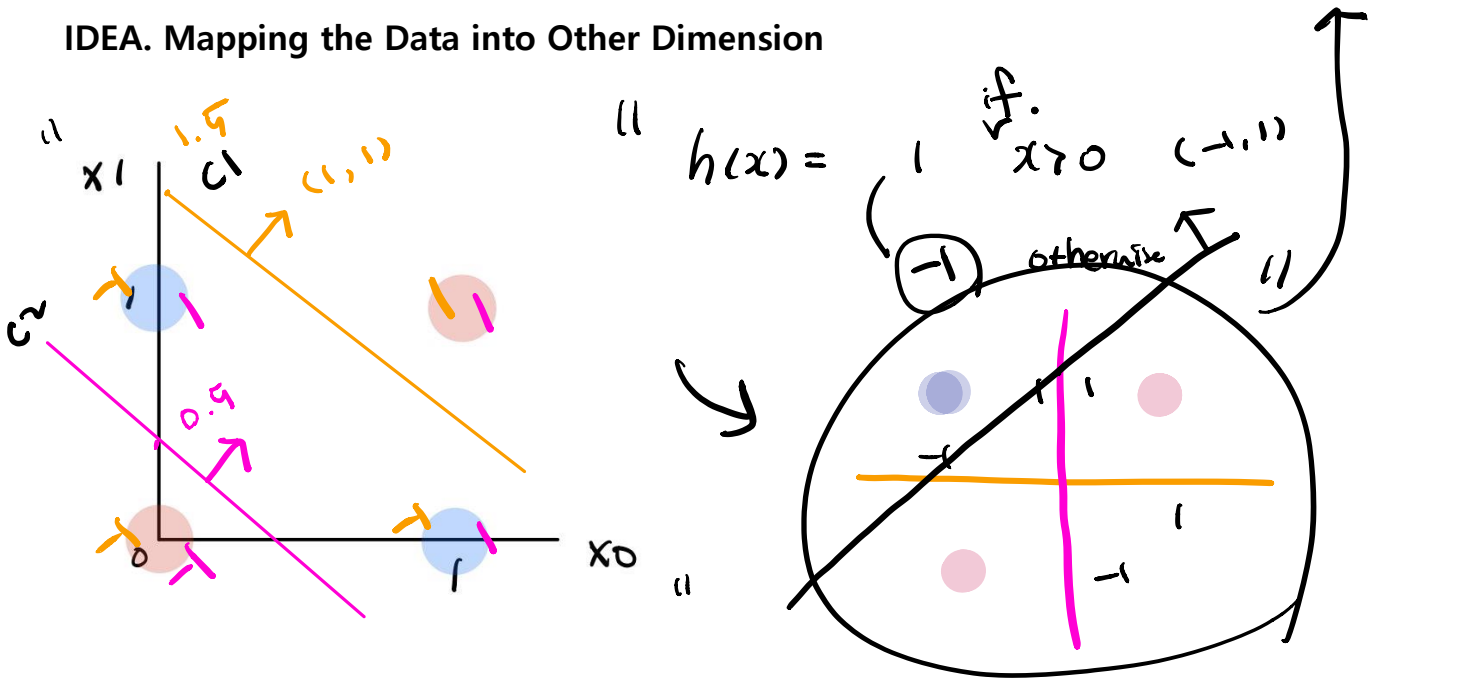
"IDEA. Mapping the Data into Other Dimension"

Map the Feature into other dimension, so that it becomes Linearly Separable



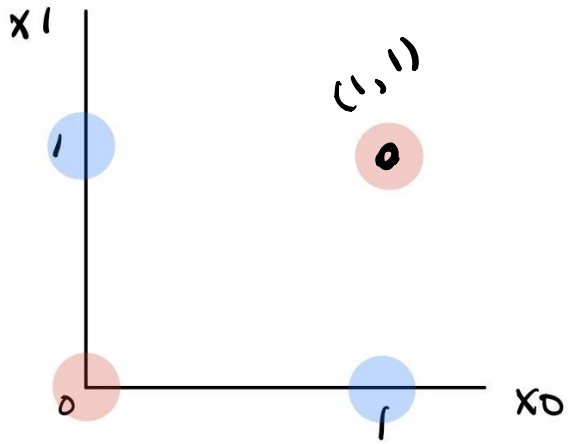
Limitation of Linear Classifier : Overcoming

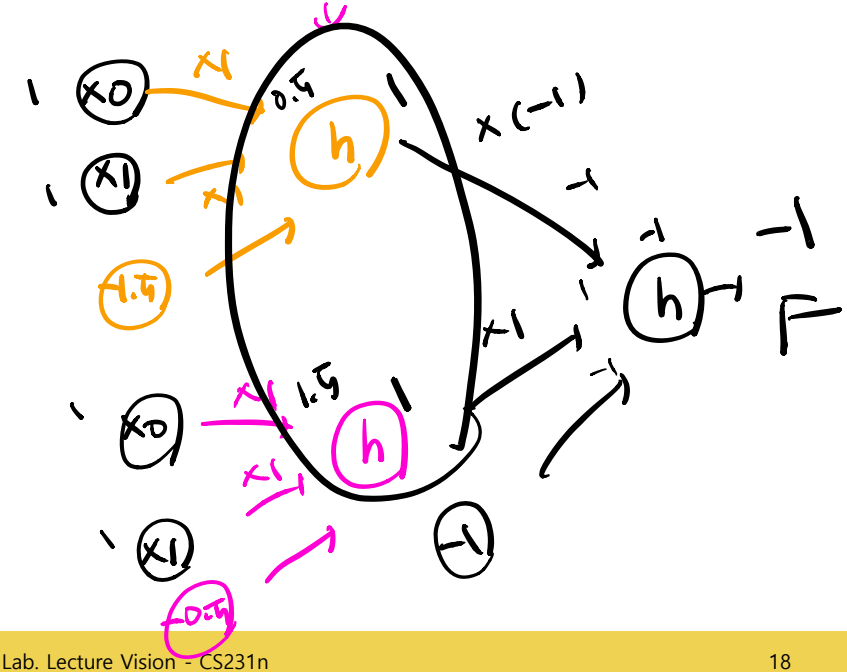
IDEA. Mapping the Data into Other Dimension



Limitation of Linear Classifier : Overcoming

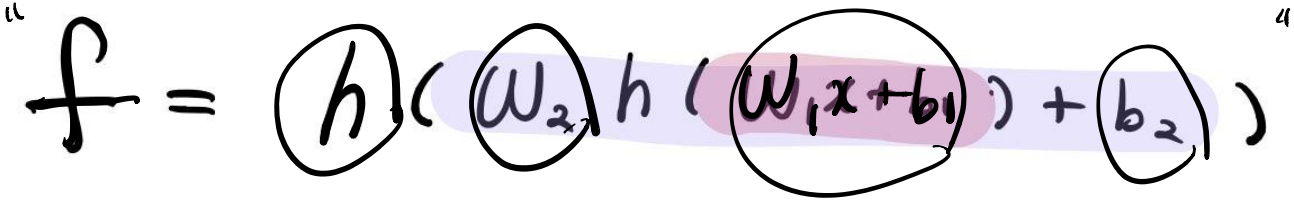
IDEA. Mapping the Data into Other Dimension





Limitation of Linear Classifier : Overcoming

IDEA. Mapping the Data into Other Dimension



Linear Classifier로 다른 차원에 Map된 Input을,

또다른 Linear Classifier로 Classify!

This model is called the **Multilayer Perceptron**, or **Artificial Neural Network**.

MLP ANN

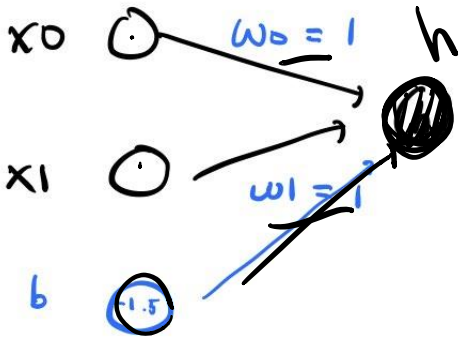
Limitation of Linear Classifier : Overcoming

What is a **"Perceptron"**...?

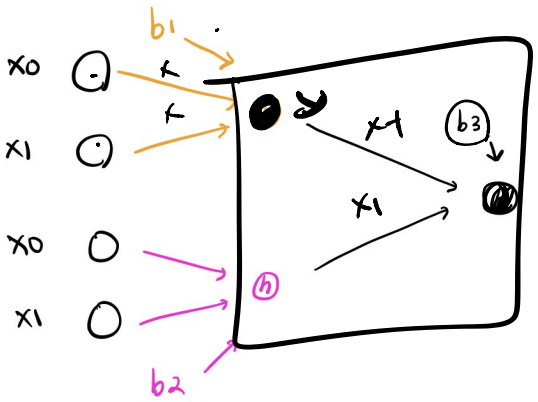
What does **"Neural"** mean...?

Perceptron : A Linear Classifier

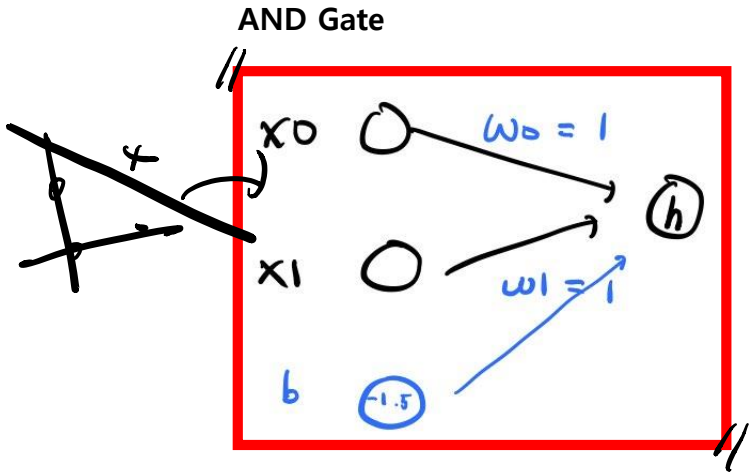
AND Gate

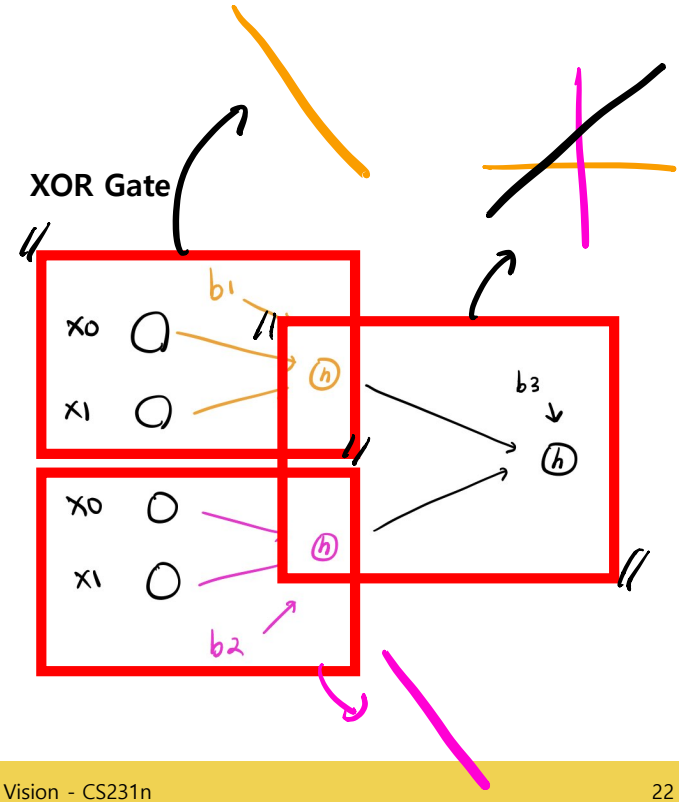


XOR Gate



Perceptron : A Linear Classifier

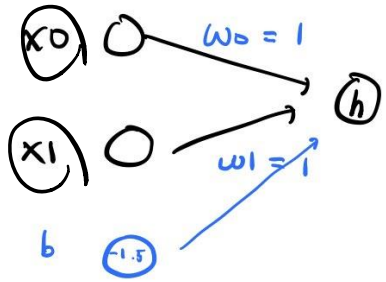




Perceptron : A Linear Classifier

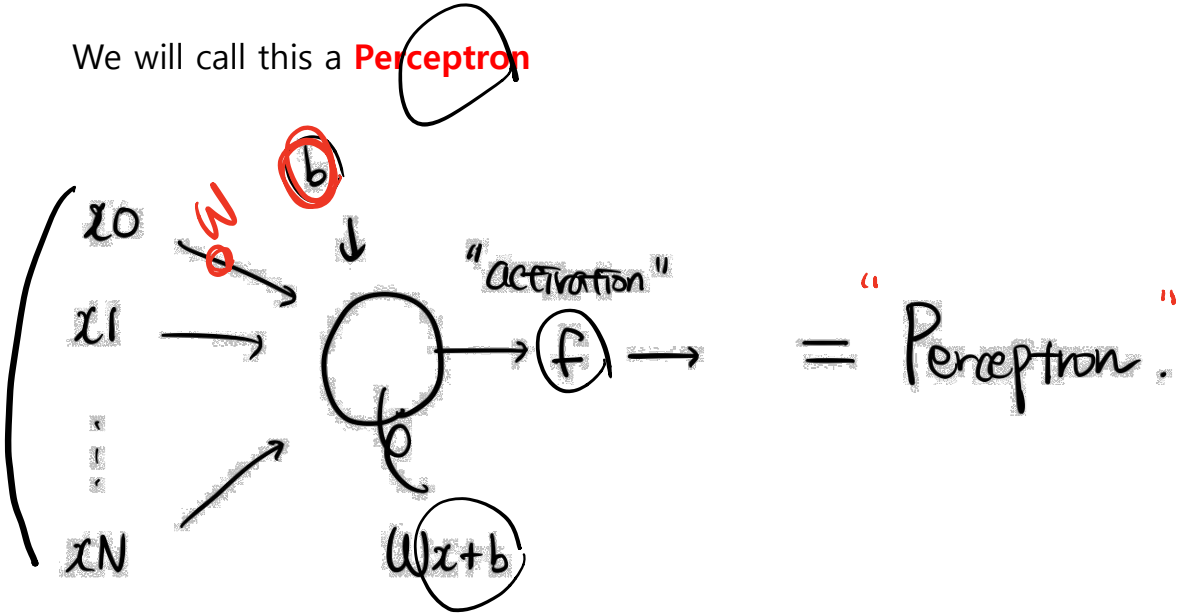
Each represents a (Decision Hyperplane) where

1. Input들에 가중치를 곱한 값들을 받아서,
2. 그들의 Linear Combination을 취한 뒤,
3. 특정 함수를 통과시켜 다음 Node의 Input으로 보냄



Perceptron : A Linear Classifier

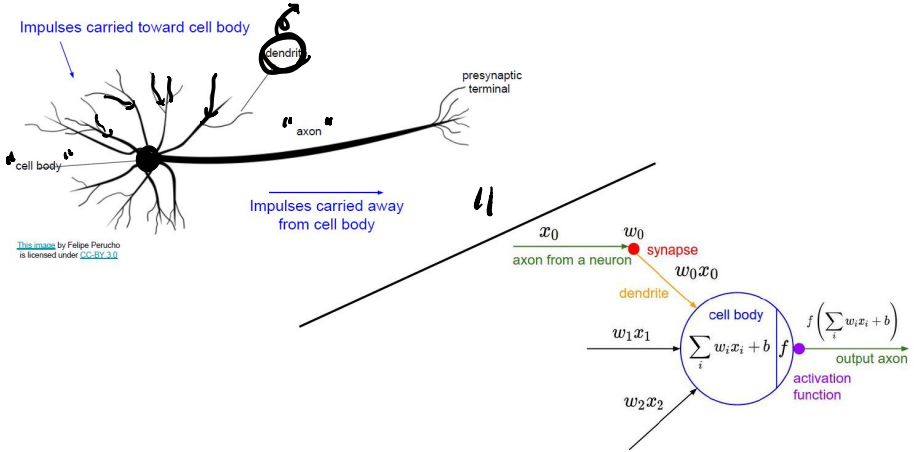
We will call this a **Perceptron**



Perceptron : Analogy to Neurons

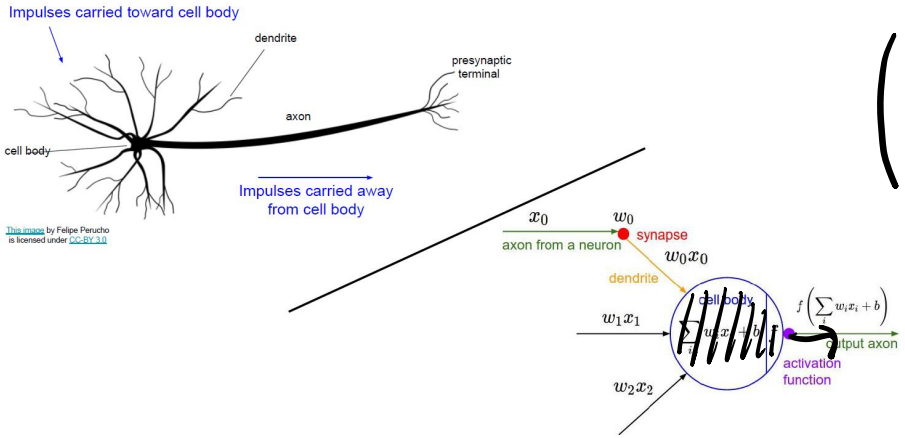
Perceptron Model은 신경계의 **Neuron**을 모방한 것

1. Dendrite로 신호 수신
2. Cell Body에서 신호 합침
3. 신호의 강도 조절하여, Axon을 통해 신호 출력



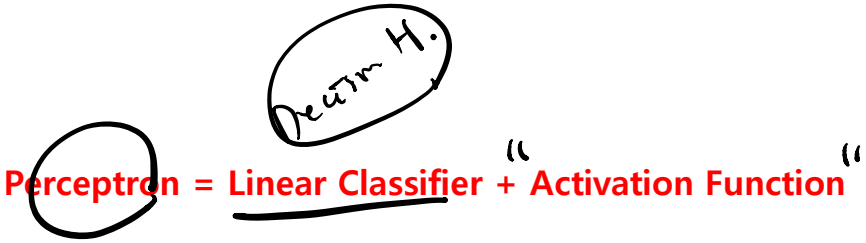
Perceptron : Analogy to Neurons

Perceptron Model은 신경계의 **Neuron**을 모방한 것



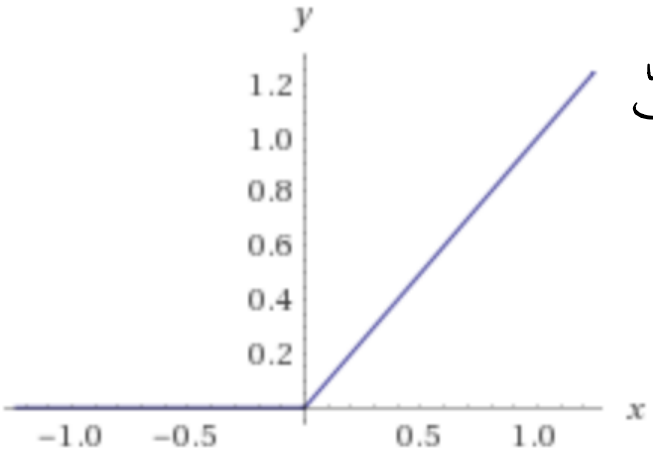
1. Input들에 가중치 곱한 값들 받아
2. 그들의 Linear Combination 취한 뒤,
3. 특정 함수를 통과시켜 출력

Perceptron : Activation Function



Perceptron : Activation Function

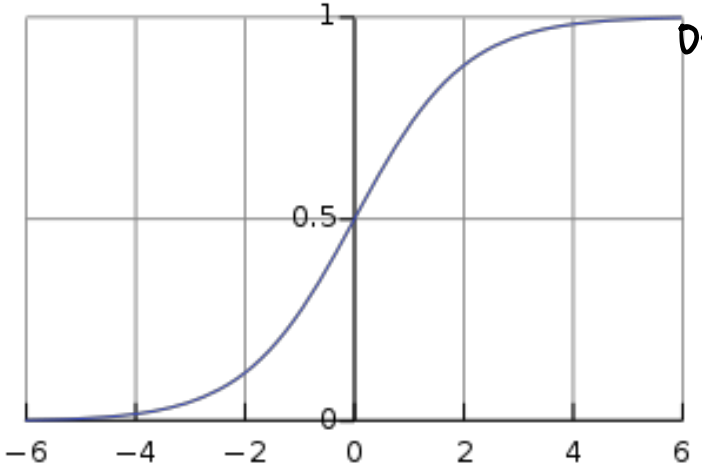
1. RELU



$y = \max(0, x)$

Perceptron : Activation Function

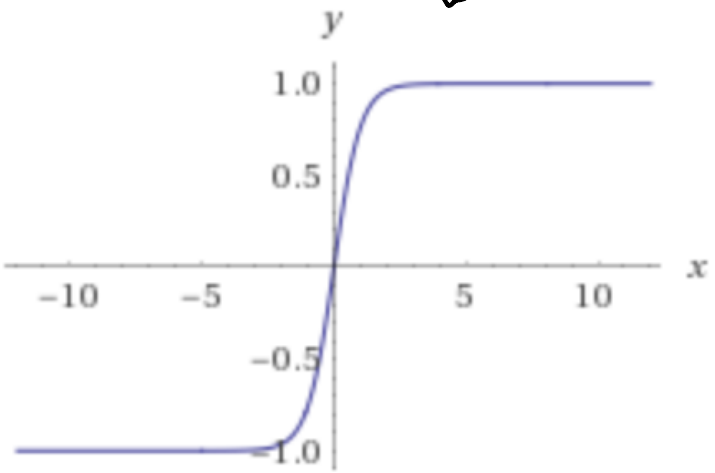
2. Sigmoid



$0 < y = \frac{1}{1+e^{-x}} < 1$

Perceptron : Activation Function

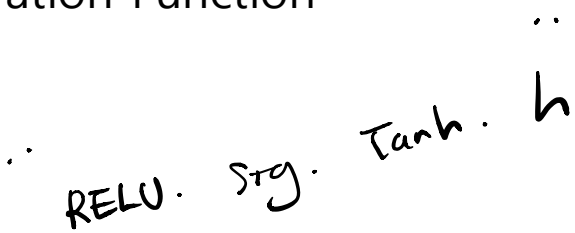
3. Tanh



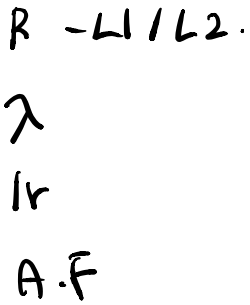
[-1, 1]

$$y = \tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}.$$

Perceptron : Activation Function



They are also … **Hyperparameters**!!!

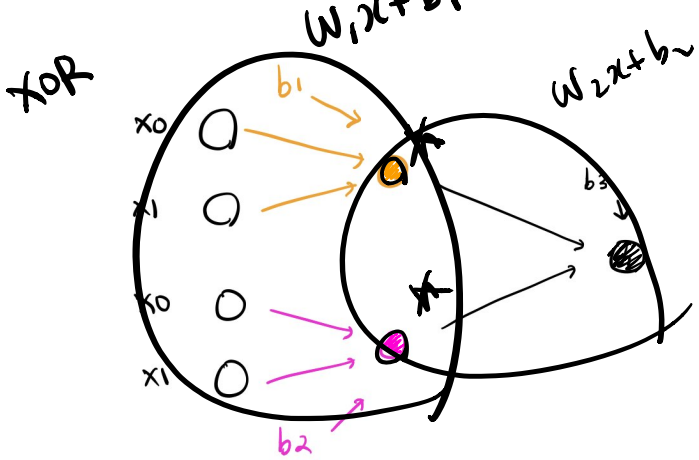


Perceptron : Activation Function

$W_2(W_1x+b_1)+b_2$

$= W'x+b'$

But, what if there is no Activation Function…?



Perceptron : Activation Function

Activation Function gives the capacity for Linear Classifiers to,

handle Nonlinear Classification Problems

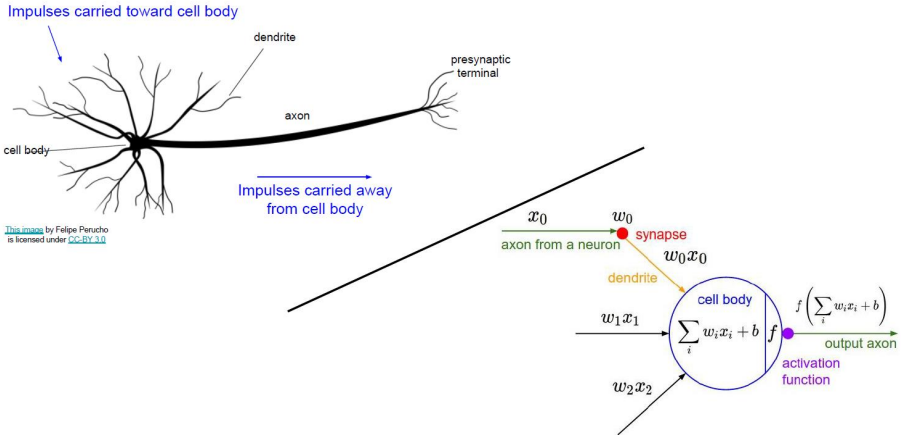
MLP : Idea

XOR

" Human Neural Network "

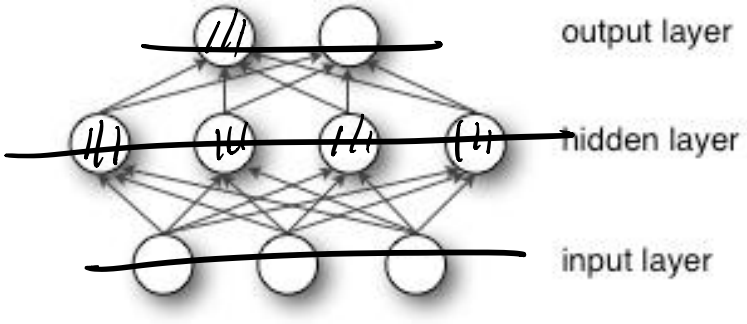
MLP : Idea

"Neuron and Perceptron"

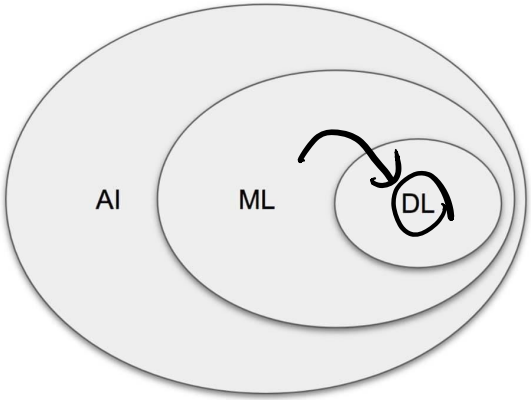


MLP : Idea

Multilayer Perceptron / Artificial Neural Network



MLP : Intro to DL



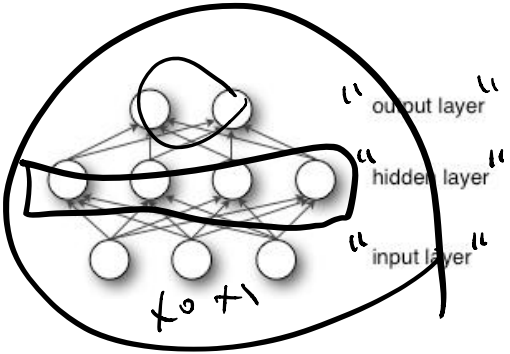
"Deep" learning.

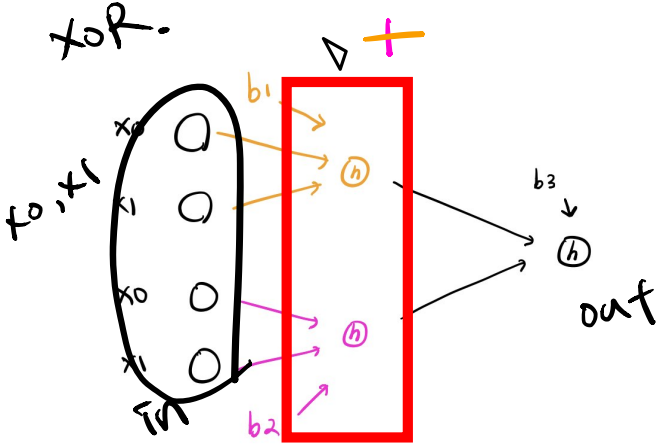
복잡한 Dataset 처리

via ANNs

MLP : Hidden Layer

Multilayer Perceptron / Artificial Neural Network

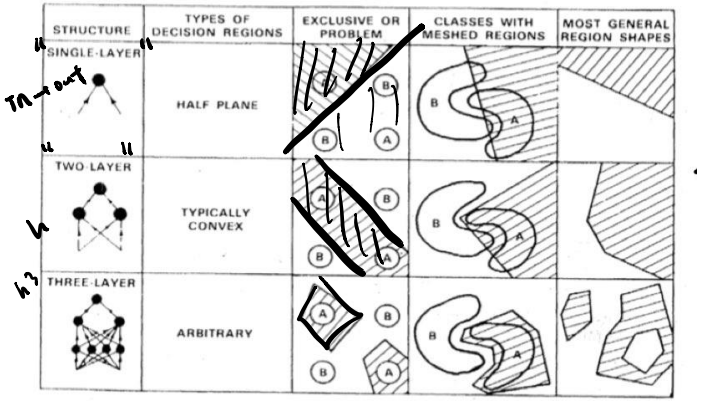




MLP : Hidden Layer

What does a Hidden Layer do?

Map the Input Feature into other dimension, so that it becomes Linearly Separable



MLP : Idea

However, can we guarantee that

the MLP will eventually give the correct classification?

MLP : The Universal Approximation Theorem

" **1개의 hidden layer를 가진 MLP를 이용해 어떠한 함수도 근사할 수 있다.** "

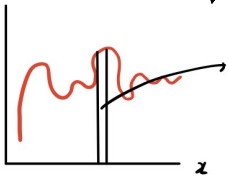
** when Activation Function is nonlinear

** when Hyperparameters are appropriately chosen

MLP : The Universal Approximation Theorem

Idea of Proof (Optional)

NN은 아주 (복잡한 함수)를 근사하는 것.



이 작은 조각들은 과.

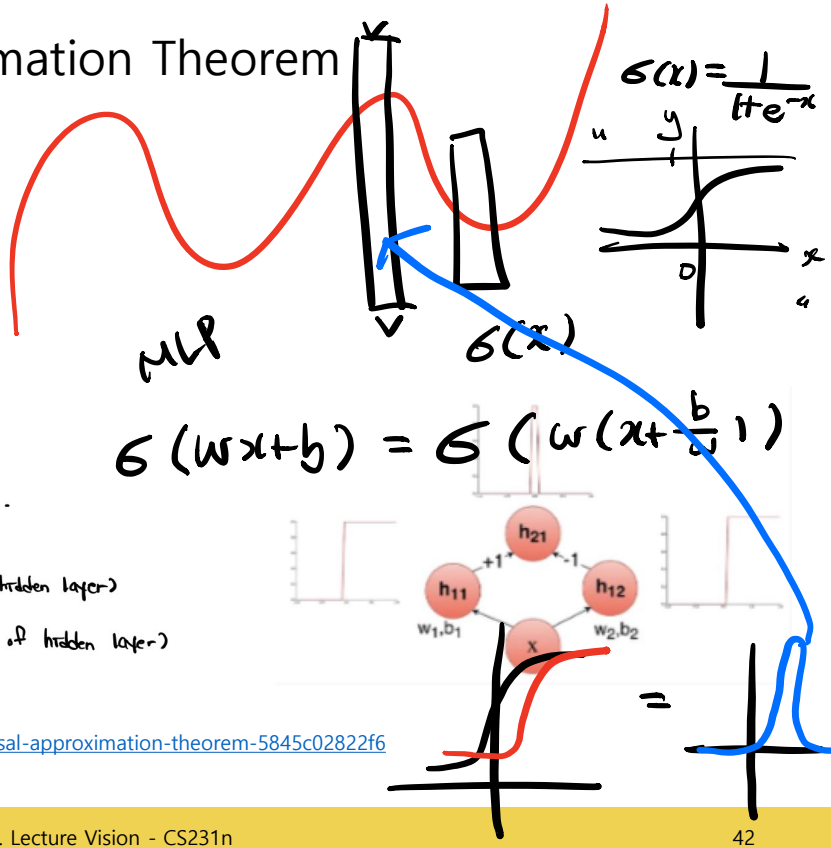
How--?

Sigmoid 2개로, tower를 build 가능.

↳

weight로 sigmoid 위치 결정, (into hidden layer)

두 sigmoid 빼서 tower 형성, (out of hidden layer)



For more detail, https://hackernoon.com/illustrative-proof-of-universal-approximation-theorem-5845c02822f6

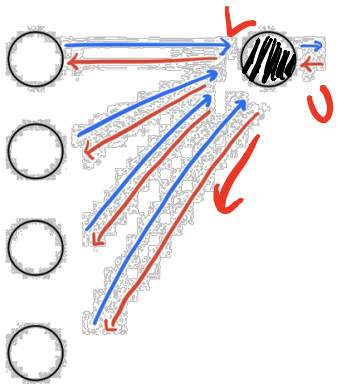
MLP1: Looking Back at "Backpropagation"

MLP를 통해, 복잡한 Classification Function을 근사하고 있다.

Thus, we should utilize **Backpropagation** to compute the gradient

MLP : Looking Back at Backpropagation

with Backpropagation,



→ : feed forword

perception에서 계산 결과가

다음 perception의 input으로 들어감.

← : backpropagation

앞 perception에서의 gradient가

선 perception의 upstream gradient를

계산함.

즉

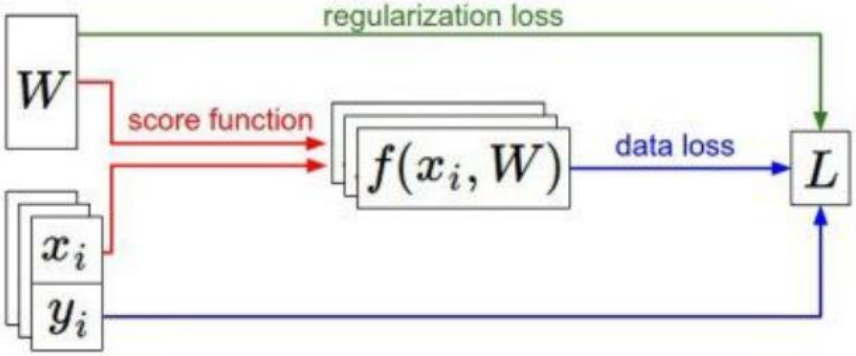
Chain rule에 의해,

(upstream gradient) x (local gradient)

MLP : Looking Back at Backpropagation

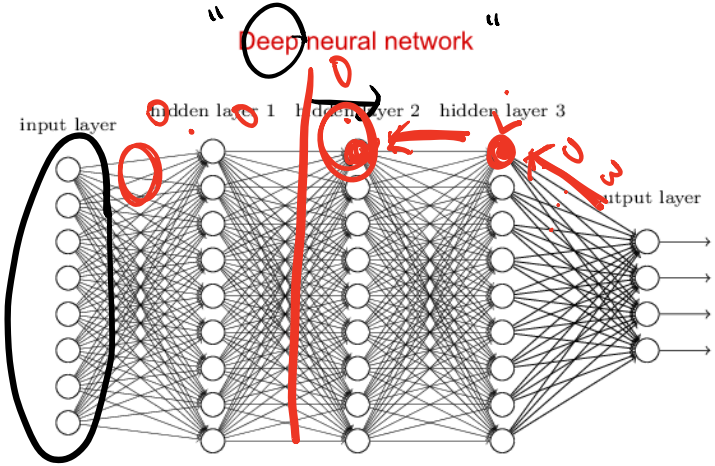
지난 Lecture까지는,

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \lambda R(W)$$

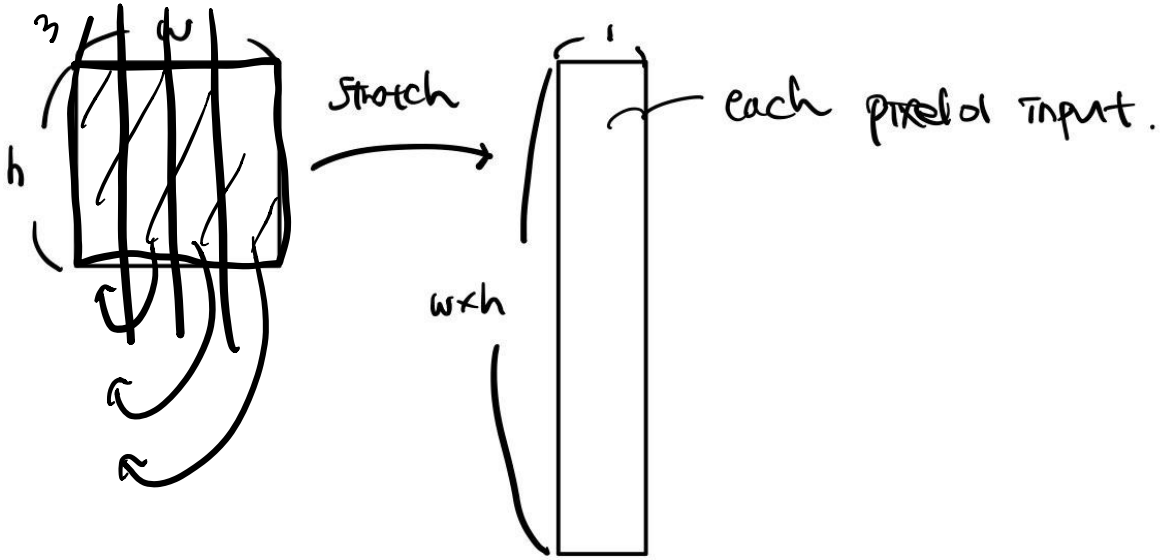


MLP : Looking Back at Backpropagation

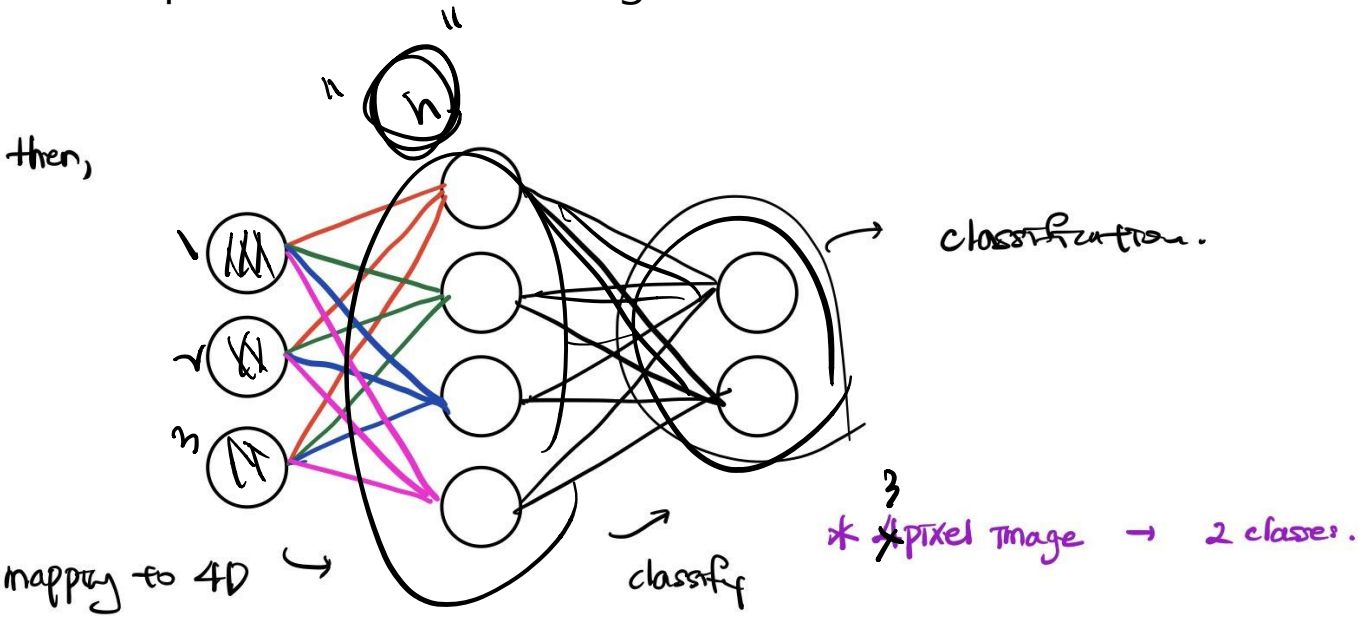
but now,



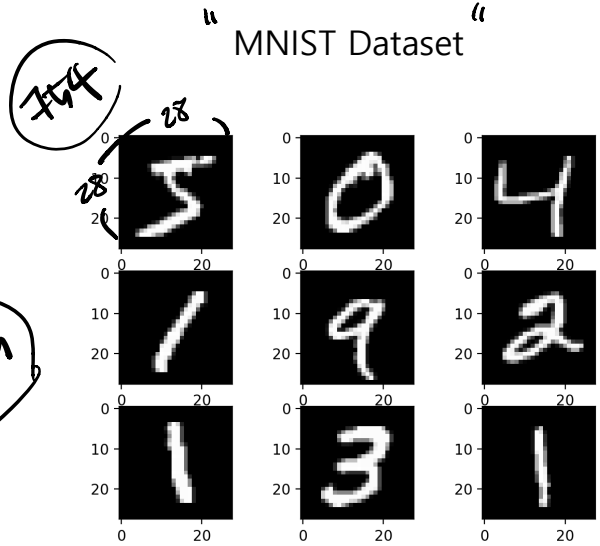
MLP : Implementation on Image Classification



MLP : Implementation on Image Classification



MLP : Implementation on Image Classification



MLP : Implementation on Image Classification

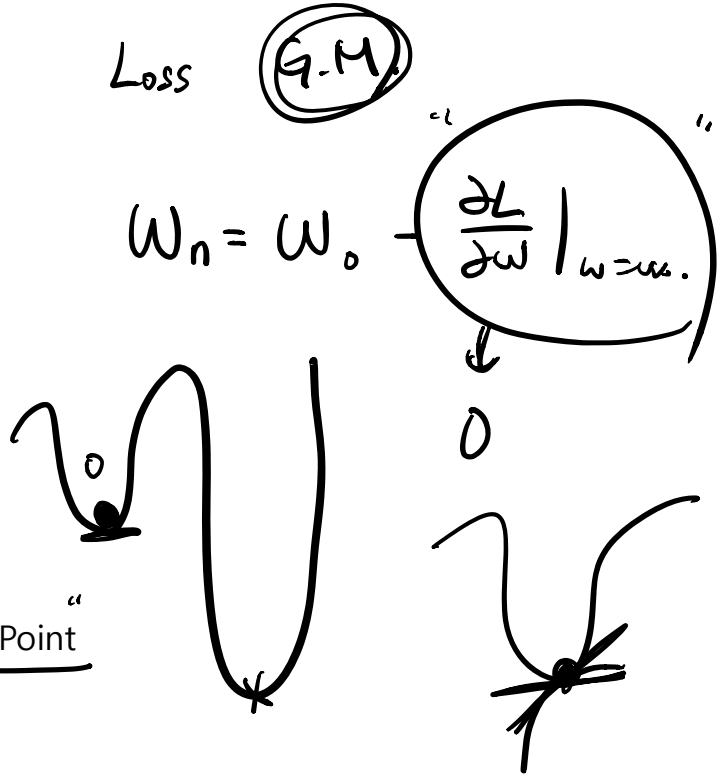
4. Limitations of MLP

MLP + Backpropagation 이론이 나온 것은, 1980s

However, 2010s부터 본격적으로 쓰이기 시작

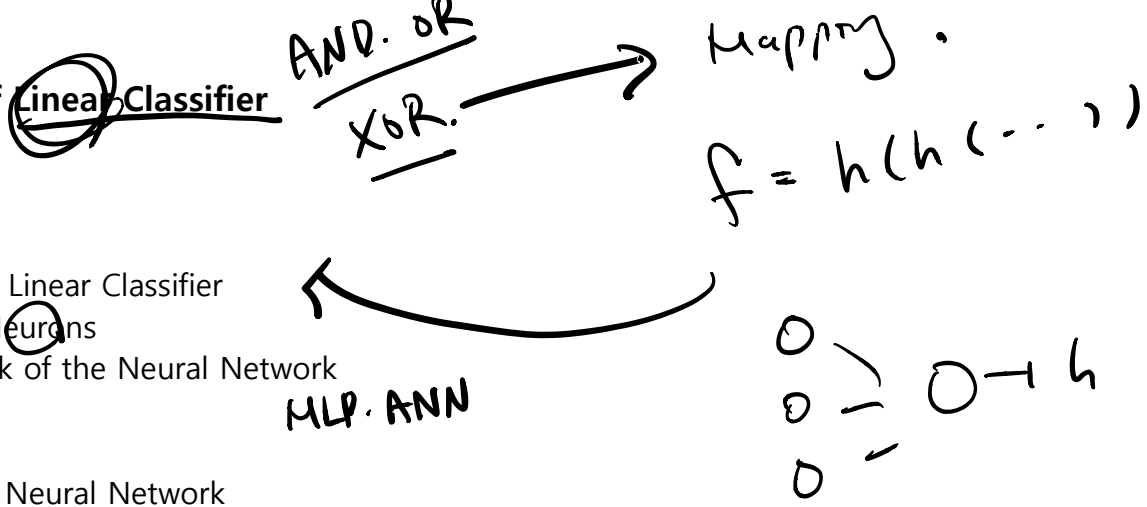
Limitations of MLP

1. Needs a lot of Labeled Data
2. Vanishing Gradient
3. Overfitting (Reg.)
4. Gets stuck in the Local Minima / Saddle Point



"Review"

1. **Limitation of Linear Classifier**
   - XOR Gate
2. **Perceptron**
   - Perceptron = Linear Classifier
   - Analogy to Neurons
   - Building Block of the Neural Network
3. **MLP**
   - MLP and the Neural Network
   - The Universal Approximation Theorem
   - Where Backpropagation becomes Important
4. **"Limitations of MLP"**



Preview on Next Lecture(s)