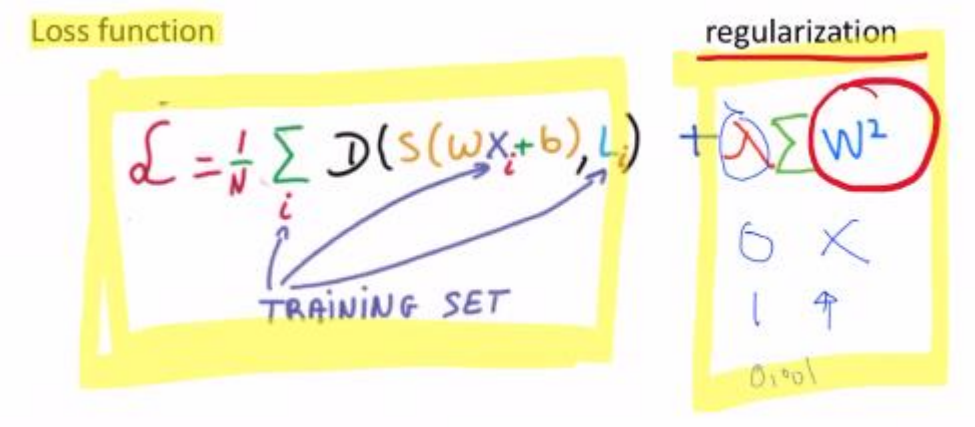
Overfitting을 막는 방법

데이터를 많이 모은다

피쳐를 줄인다

정규화

Regularization



W가 커지면 전체 값이 작아지도록 하는 Regularization Term을 붙인다.

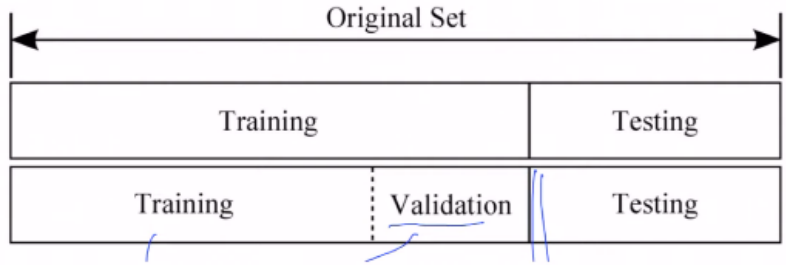
Performance evaluation

학습된 파라미터(W, b)를 사용한다.

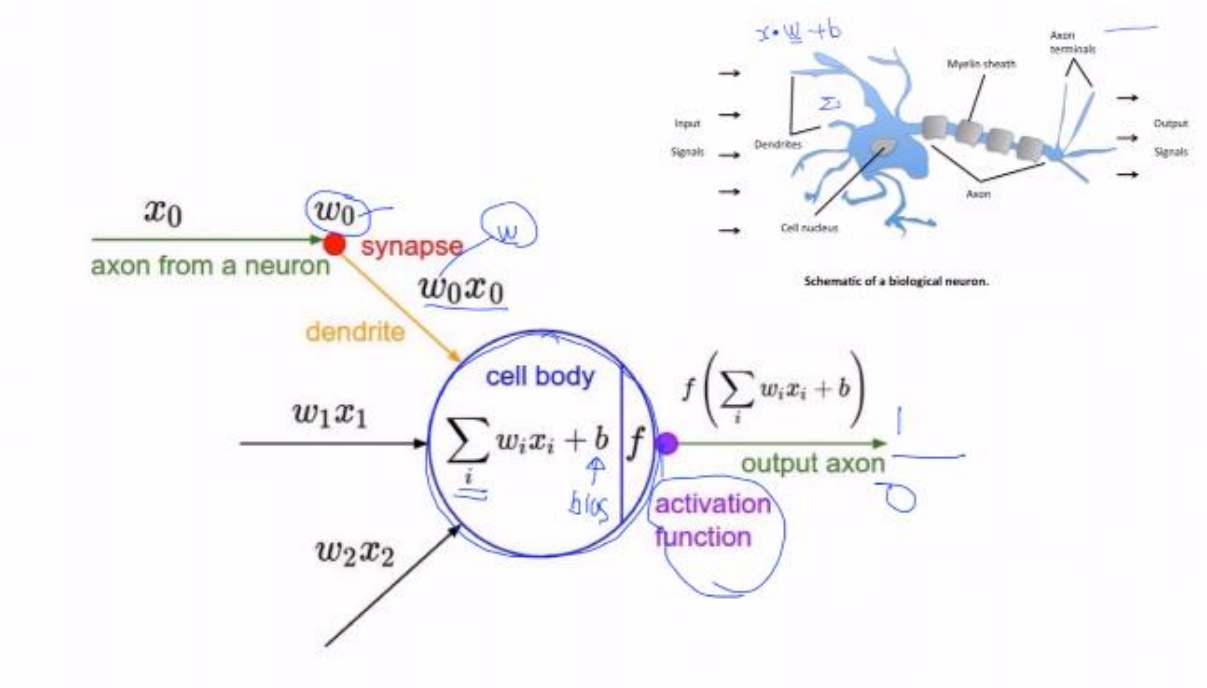
학습과정에 사용되지 않은 데이터를 사용한다.

> 왜?
>
> 모델이 학습할 때 사용한 training set에만 맞는 외운 답으로 100의 정확도를 내버린다. 일반화가 안됨.

Validation set : 학습 과정에서 성능 평가를 하기 위한 데이터.

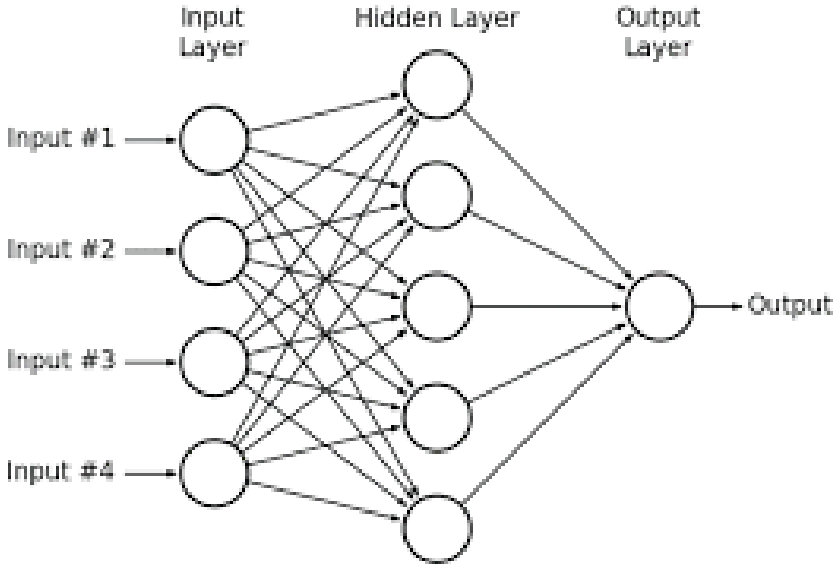


Perceptron : 인간의 신경 세포의 형태를 모사



Activation Function 활성함수 : Output을 결정하는데 사용되는 함수

MLP : 여러 개의 Perceptron을 쌓아서 만든다.

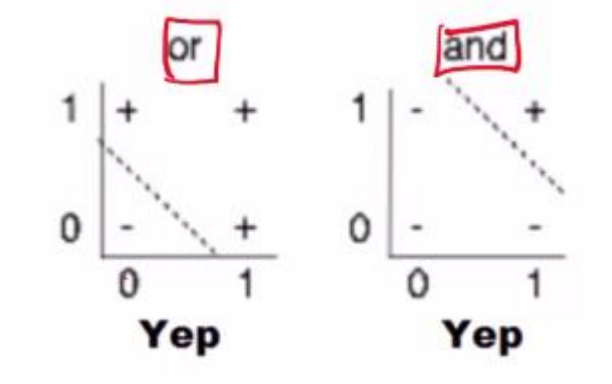


MLP 머신.

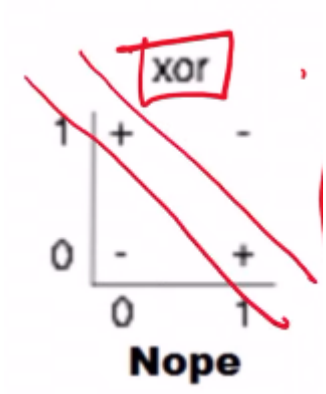
근데 이제 사람이 training을 직접 한...

컴퓨터에 기본적으로 필요한 논리 연산은 AND, OR, XOR이다.

Simple AND/OR Problem : linear separable 얘네는 선형으로 쉽게 할 수 있다.

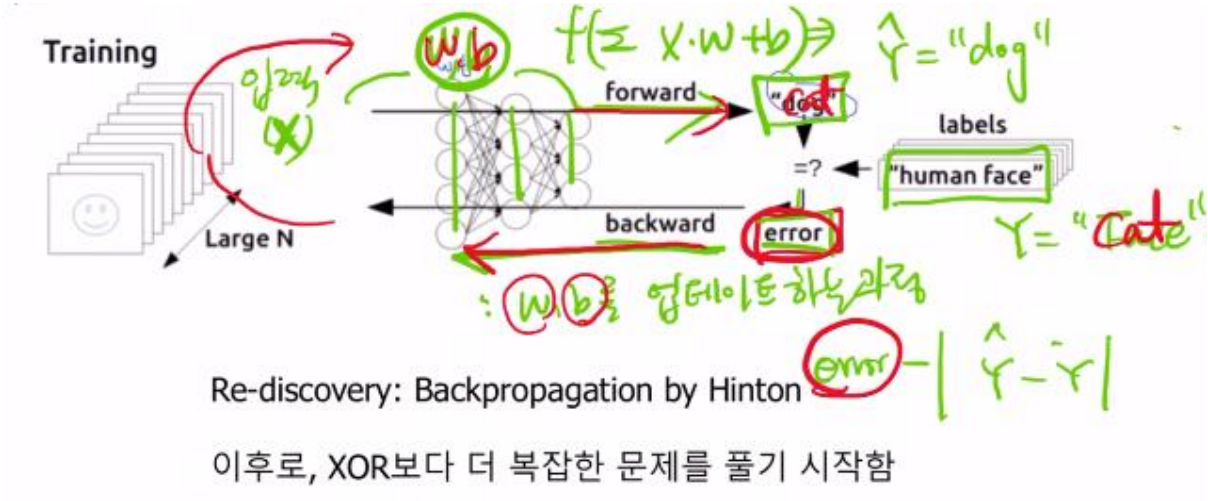


근데! XOR은 안돼~ Perceptron 모델로 못풀어요.. (1969년)



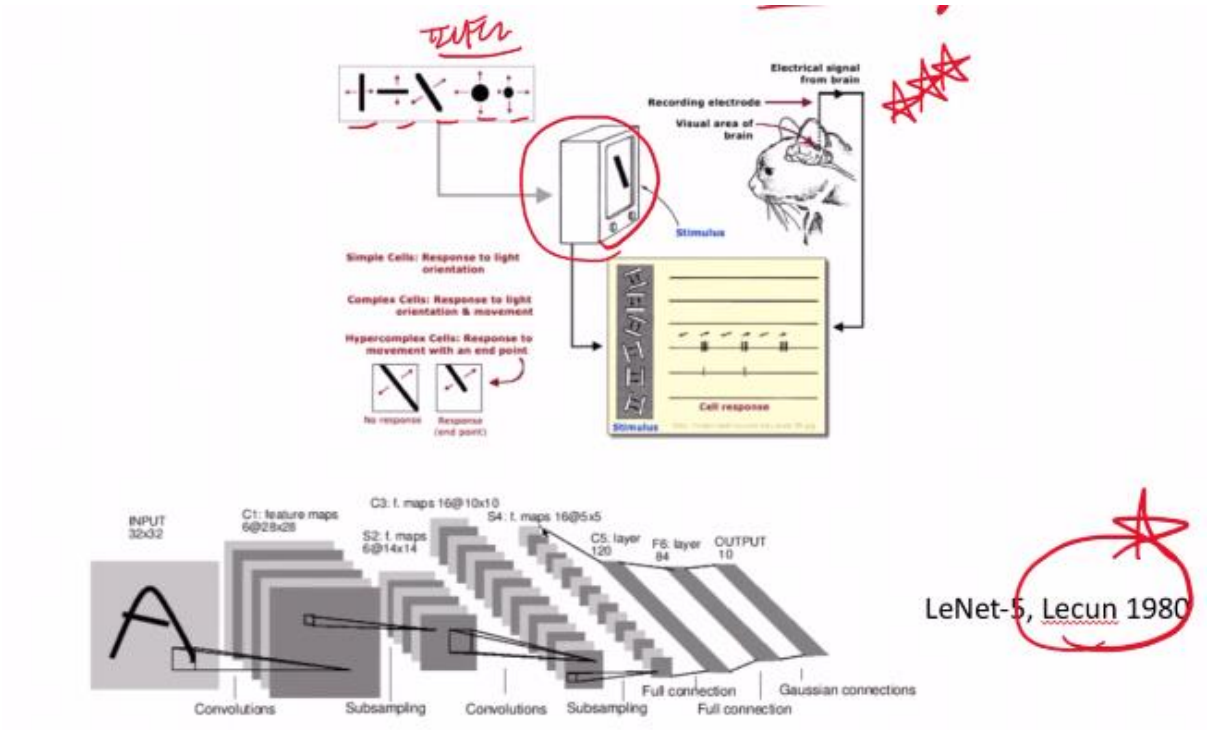
근데 이런 난제가 풀렸다. (1974, 1982 by Paul Werbos, 1986 by Hinton)

Backpropagation : 역방향으로 오차를 전파시키면서 각 layer의 가중치를 업데이트 하고 최적의 학습결과를 찾아가는 방법

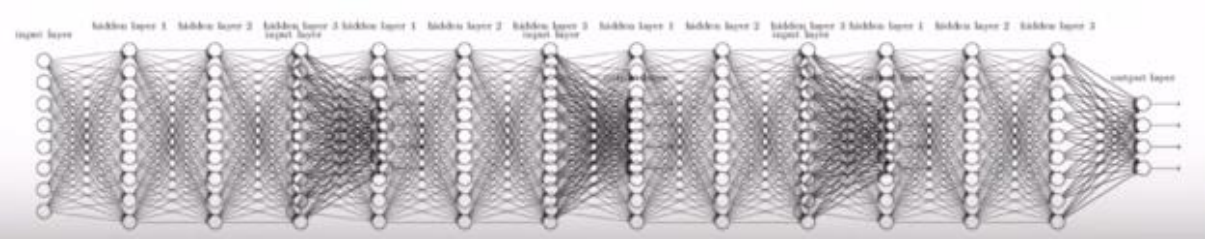


Convolutional Neural Networks :

고양이의 이미지 인식 뇌 전극 실험으로 이미지를 인식할 때 뇌는 이미지 조각의 집합으로 Feature을 기술함을 알아 냄.



근데 다시 난관에 봉착..



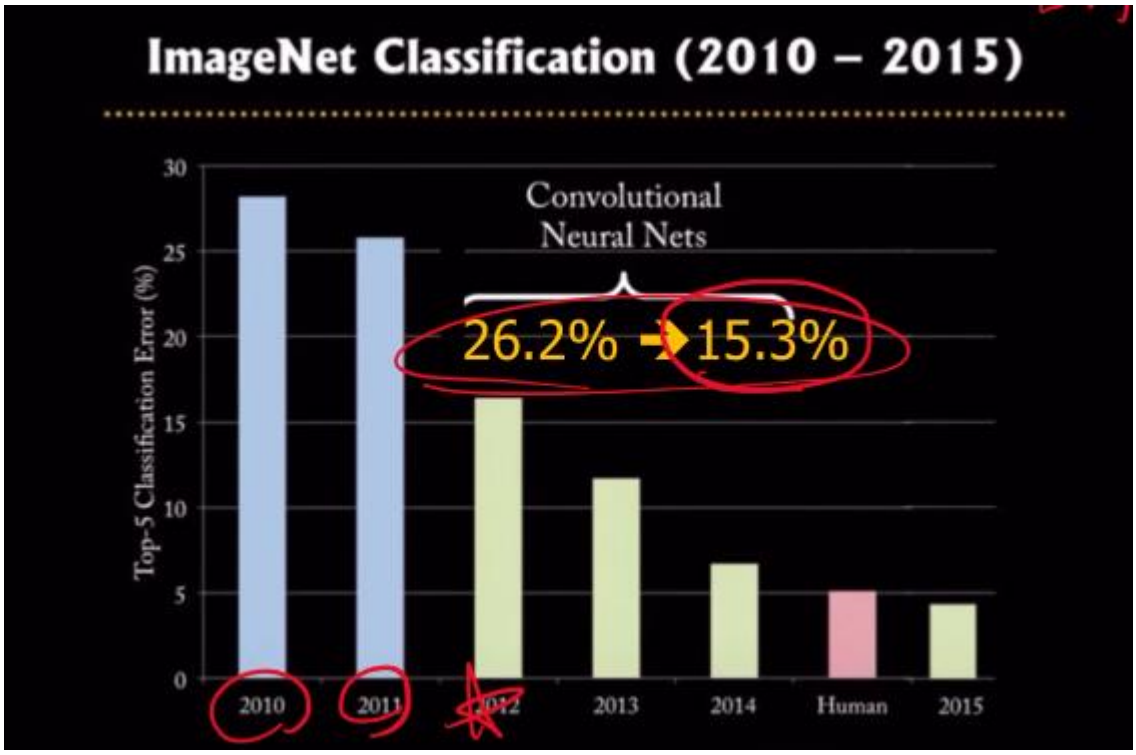
은닉층을 많이 쌓다 보니 학습이 잘 안되더라 (Gradient vanishing). 그러다가 SVM과 Random Forest 같은 방법론들이 나오면서 NN을 뛰어넘었다.

Breakthrough Hinton, Bengio, Lecun : 딥러닝 3대 거장

2006년 : W, b의 초기화를 잘 해보자

2007년 : Dropout

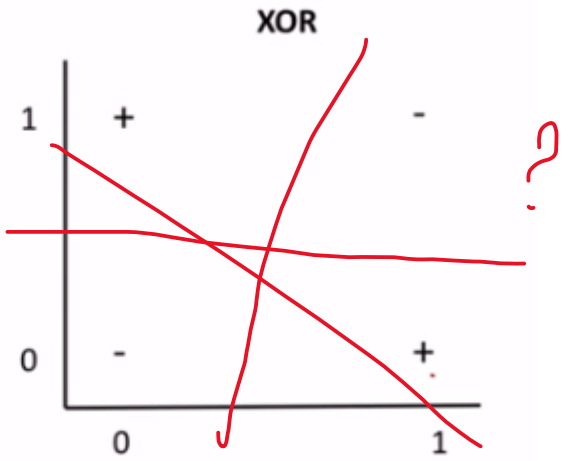
ImageNet 분류 문제에서 CNN이 엄청난 성능을 보이면서 각광받기 시작



XOR 문제를 어떻게 해결했을까??

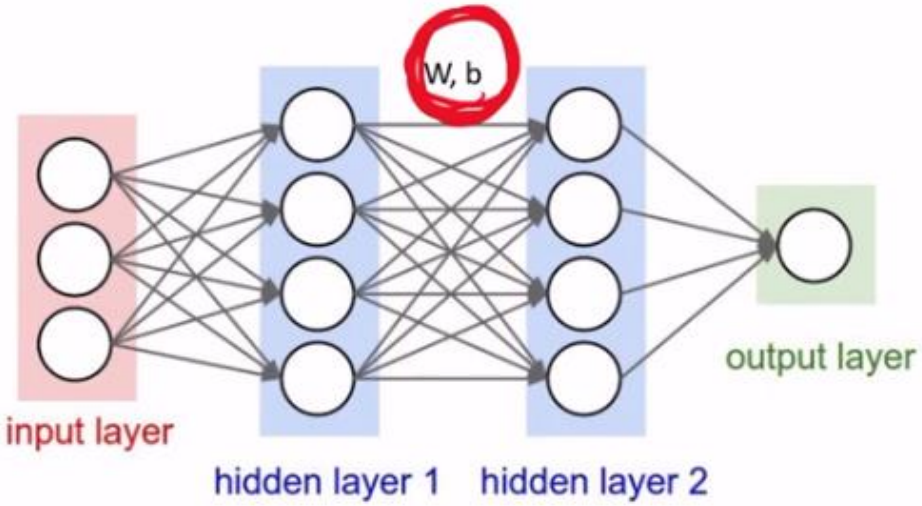
Perceptron은 왜 XOR을 못 풀었을까요? -> 선형 분류가 안돼서.

X1	X2	XOR	
0	0	0	-
0	1	1	+
1	0	1	+
1	1	0	-



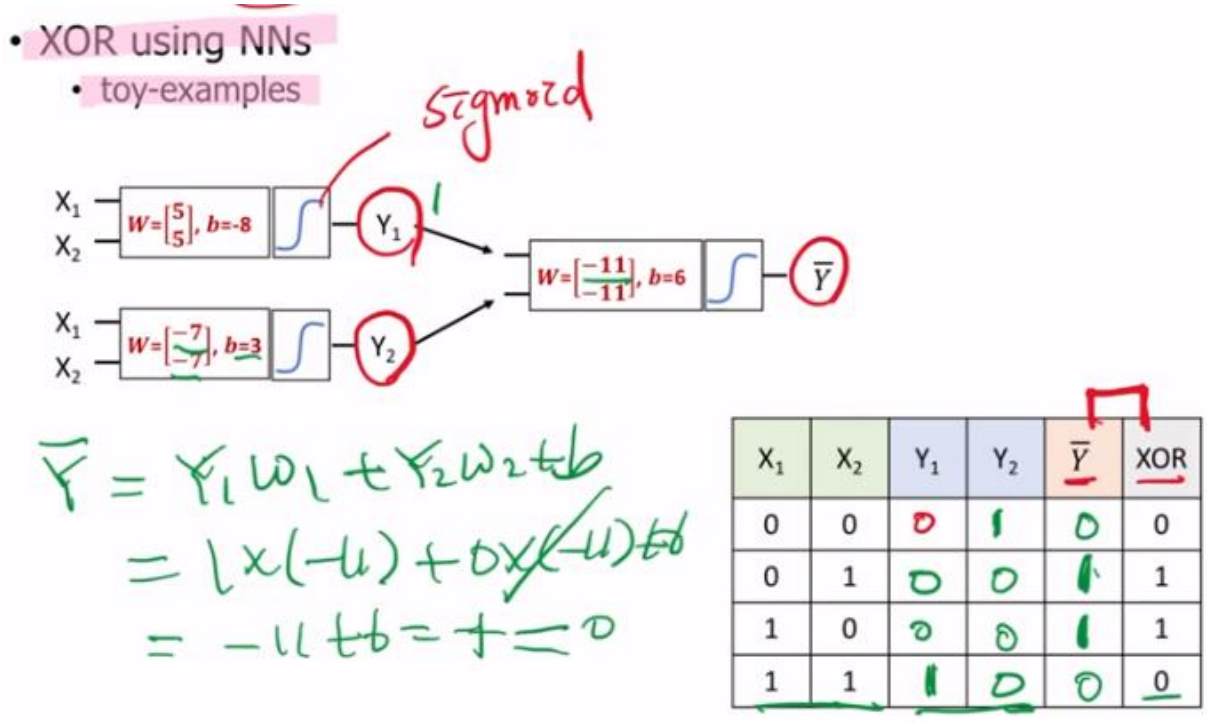
MLP로는 가능!

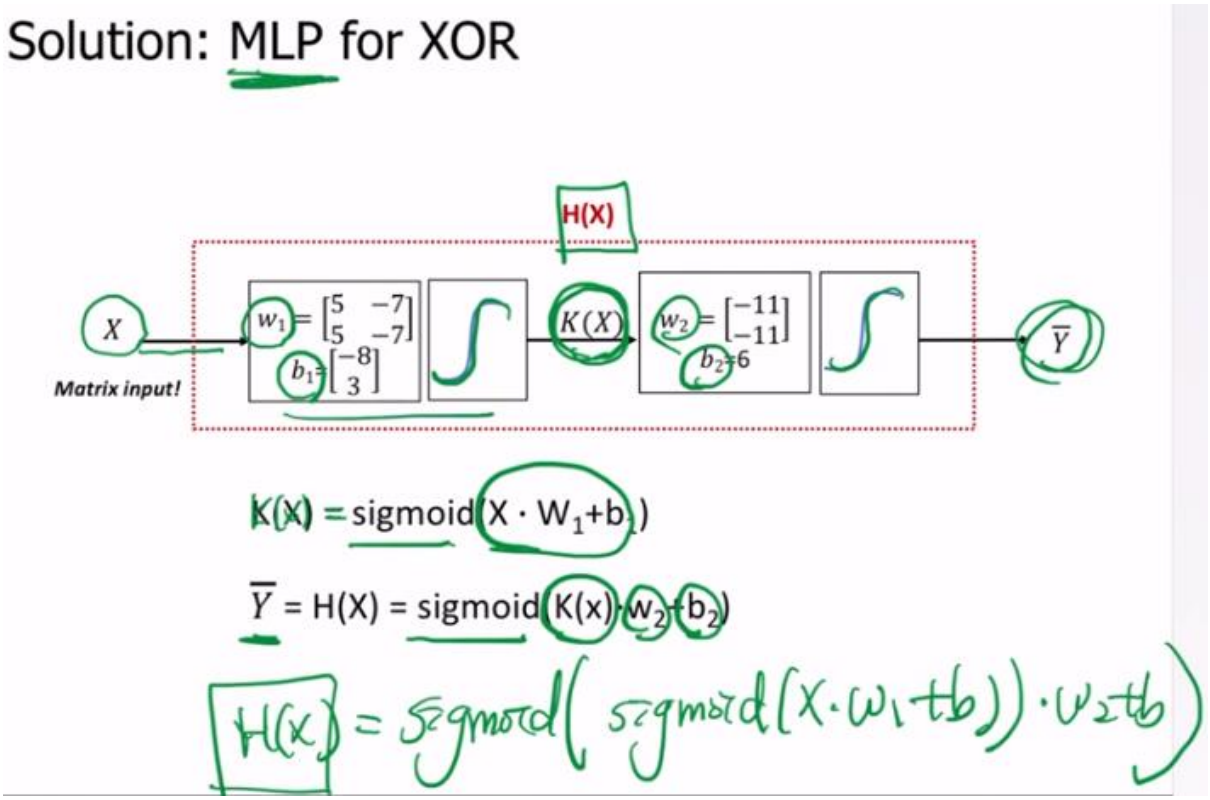
- Multiple logistic regression units can separate XOR
- But! "No one on earth had found a viable way to train"



*Marvin Minsky

MLP가 XOR문제를 푸는 파라미터 값





2 layer perceptron으로 해결.

자 더 깊게 쌓은 Deep Neural Network (DNN)을 통한 어려운 문제에 도전.

그런데 Gradient Vanishing 문제가 발생했다. Backpropagation이 앞으로 갈수록 희미 해져 학습이 안된다.

Geoffrey Hinton 선생님의 이에 대한 원인 분석

- 잘못된 비선형 함수를(Activation Function) 사용했다.
- Weight 초기화를 멍청하게 해왔다.
- Labeled된 데이터가 너무 적었다.
- 컴퓨터가 너무 느렸다.

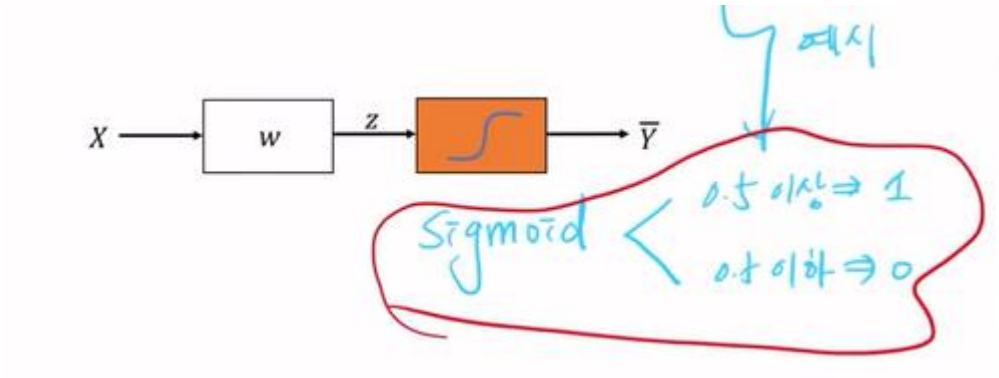
Activation Function

활성함수는 신경학적으로 볼 때 뉴런 발사(Firing of a Neuron)의 과정에 해당함

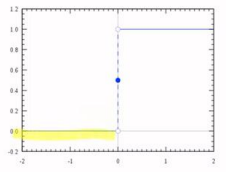
최종 출력 신호를 다음 뉴런으로 보내줄지 말지 결정함

뉴런이 다음 뉴런으로 신호를 보낼 때 입력신호가 일정 기준 이상이면 보내고 기준에 달하지 못하면 보내지 않을 수 있다.

많은 종류의 활성화 함수가 있고, Activation의 결정이 결과에 크게 영향을 미침



1. **Step func**
    A. 입력이 양수 일 때는 1, 음수 일 때는 0 의 신호를 보내는 이진 함수
    B. 미분이 불가능한 함수로 모델 Optim 과정에 사용이 어려워 신경망에서는 안 쓰임



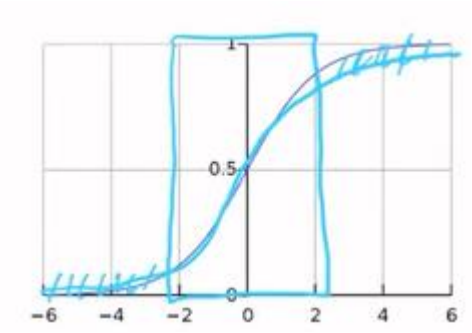
2. Sigmoid func

A. 단일 퍼셉트론에서 사용했던 활성함수

B. 입력을 (0,1)사이로 정규화함

C. 역전파 단계에서 NN layer 을 거칠 때 마다 작은 미분 값이 곱해져, Gradient Vanishing 을 야기함.

D. 3 개 이상의 layer 에서 사용을 권장하지 않음



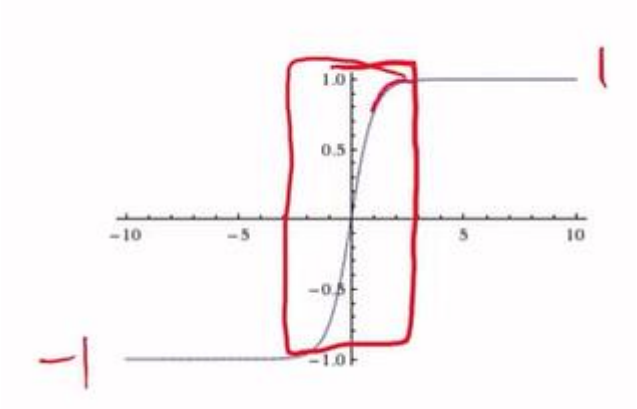
미분 값이 작아요.

3. Tanh func

A. Sigmoid 를 보완하고자 제안됐음

B. 입력을 (-1,1)사이의 값으로 정규화

C. 기울기 값이 급격하긴 한데 그래도 Gradient Vanishing 발생



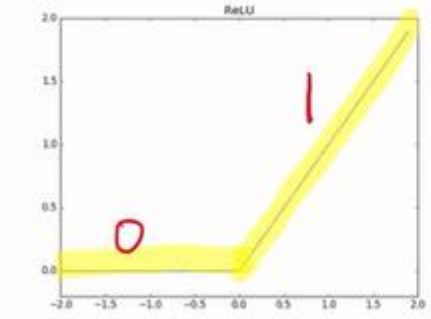
4. ReLU (Rectified Linear Unit)

A. 현재 가장 인기있음

B. 양수에서 Linear function 과 같으며 음수는 0 을 출력하는 함수

C. 미분값을 0 또는 1 로 가지기 때문에 Gradient Vanishing 이 발생하지 않는다.

D. 함수도 단순해서 sigmid 나 tanhq 다 6 배 정도 학습이 빠르다



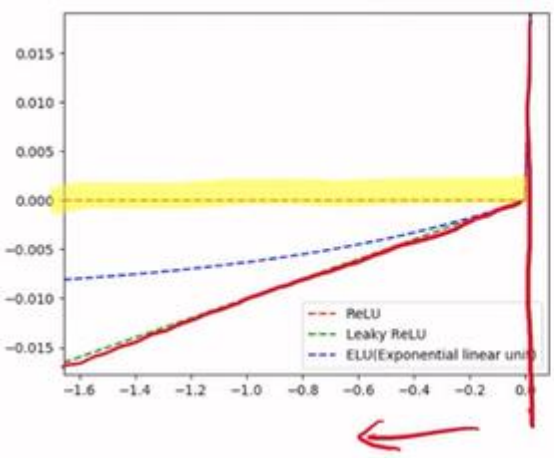
$Relu(x)=max(0,x)$

5. Leaky ReLU

A. 0 미만으로는 싹 0 인 dying ReLU 현상을 해결하기 위해 제시된 함수

B. 0 미만에서 작은 기울기를 부여함. 보통 0.01

C. Leaky ReLU 로 성능이 향상 되었다는 보고가 있으나 항상 그렇진 않다.



$Leaky\ Relu(x)=max(0.01*x,x)$

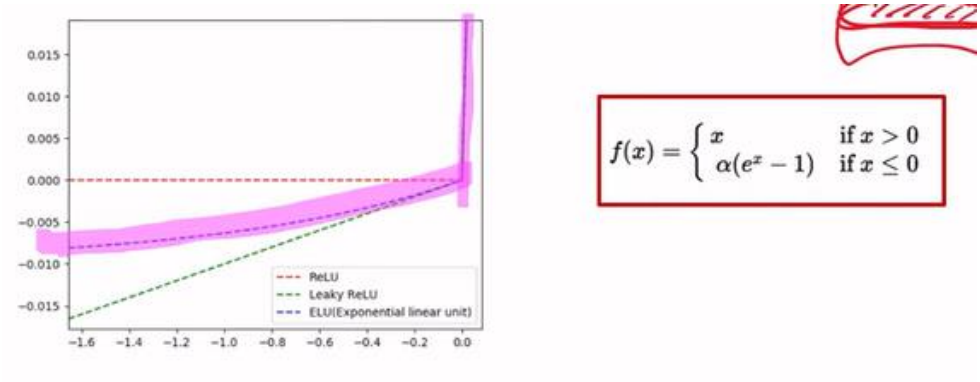
6. ELU (Exponential Linear Unit)

A. ReLU 의 threshold 를 -1 로 낮춘 함수를 exp 를 통해 근사한 것

B. Dying ReLU 문제 해결

C. 출력 값이 거의 zero_centerd 에 가까움

D. 하지만 exp()를 계산해야 하는 비용이 듦



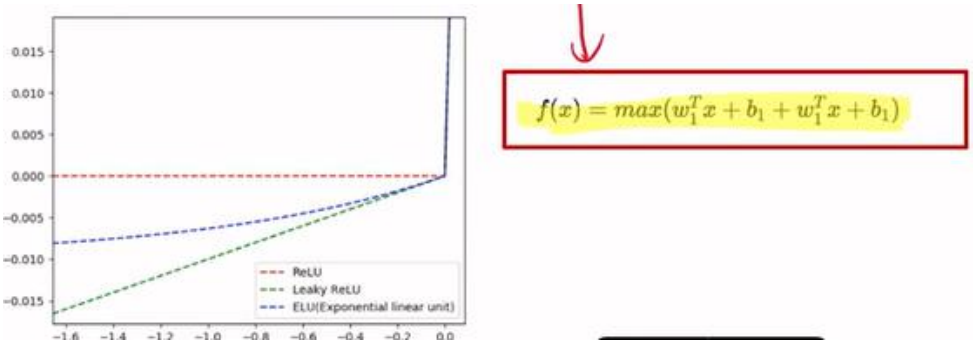
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$$

7. **Maxout**

A. ReLU 와 LeakyReLU 를 일반화 한 것.

B. ReLU 의 장점을 모두 가지며, dying ReLU 문제도 해결

C. 한 뉴런에 대해 파라미터가 두 배이기 때문에 전체 파라미터가 증가하는 단점



$$f(x) = max(w_1^T x + b_1 + w_1^T x + b_1)$$

그럼 어떤 Activation Function 을 사용해야 하는가?

1. 가장 먼저 ReLU 를 사용해본다
2. 다음으로 Leaky ReLU, Maxout, ELU 를 시도해본다

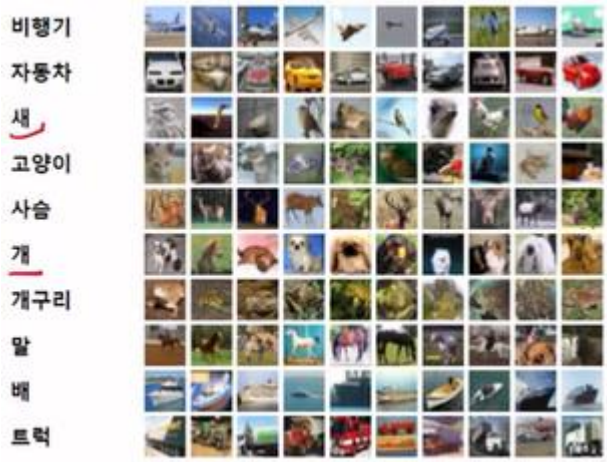
The compatibility of activation functions and initialization.
Dataset: CIFAR-10

Init method	maxout	ReLU	tanh	Sigmoid
LSUV	**93.94**	**92.11**	89.28	n/c
OrthoNorm	93.78	91.74	89.48	n/c
OrthoNorm-MSRA scaled	–	91.93	–	n/c
Xavier	91.75	90.63	**89.82**	n/c
MSRA	n/c†	90.91	89.54	n/c

n/c symbol stands for "failed to converge, Architecture FitNets-17

Maxout > ELU, Leaky ReLU >= ReLU > tanh > sigmoid

CIFAR-10



- 32x32픽셀의 60000개 이미지
- 각 이미지는 10개의 클래스로 라벨링

Weight Initialization

같은 ReLU 인데 Random Seed 에 따라 이렇게 차이가 난다!

지금까지 선형 회귀나 Softmax 같은 알고리즘 에서는 -1~1 의 난수를 Weight 로 사용

Neural Network 에서는 W 가 0 이 되면 절대 안된다. 왜? Gradient Vanishing 되니까.

그럼 어떻게 초기화 할까

초기의 RBM 은 복잡도가 너무 높다.

2010 년 : Xavier initialization

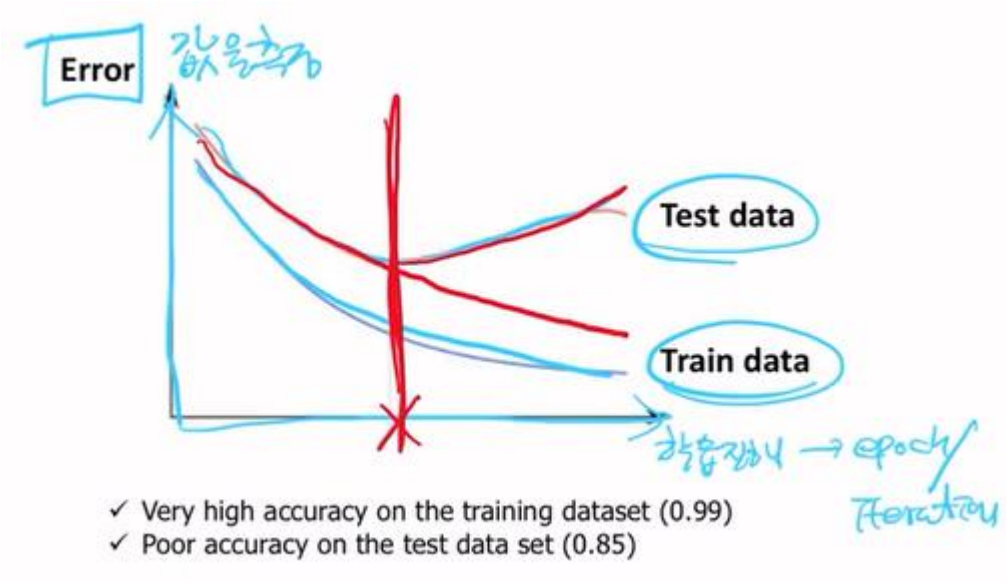
2015 년 : He's initialization

= 노드의 입출력 수에 비례에서 초기화를 결정짓는 방법

지금도 다양한 방법론이 제시되고 있다.

Dropout and Ensemble

overfitting 을 피하자.



이를 피하는 방법들:

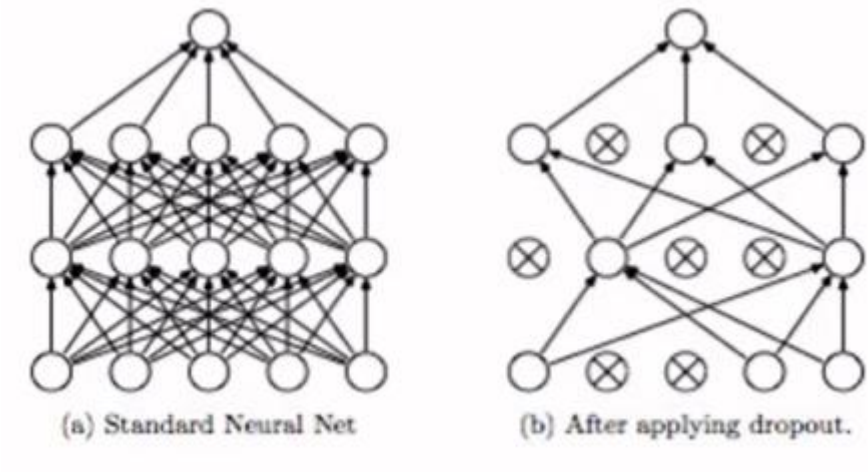
More Training data

Reduce the number of features

Regularization (DROPOUT)

학습시에 신경망에서 일부 유닛을 임시로 제외

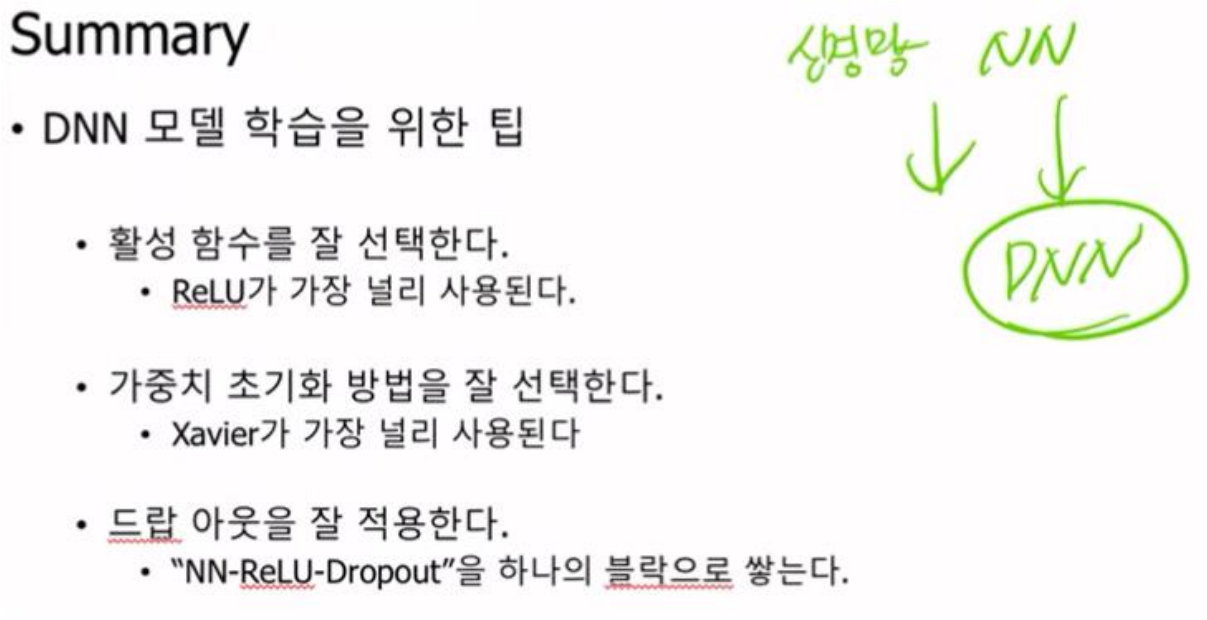
테스트 시에는 모든 유닛을 사용한다.



torch.nn.Dropout(p=0.5(default), inplace)

왜 성능이 향상될까?

Dropout 을 통해 Ensemble model 학습과 같은 효과가 있기 때문이다.



Optimizer : 발전 방향