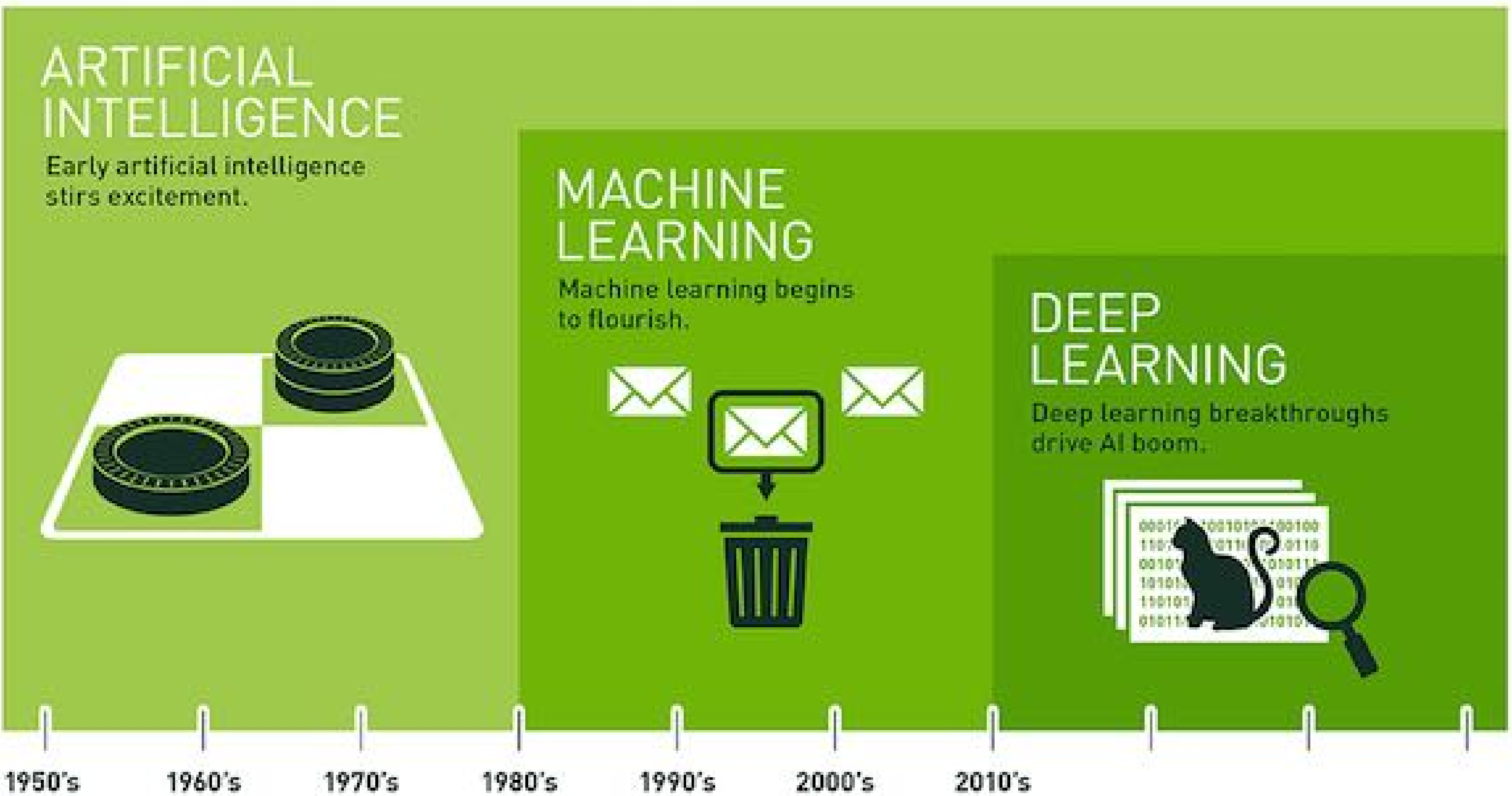
딥러닝

Deep learning refers to
artificial neural networks
that are composed of many layers.

딥러닝 정의



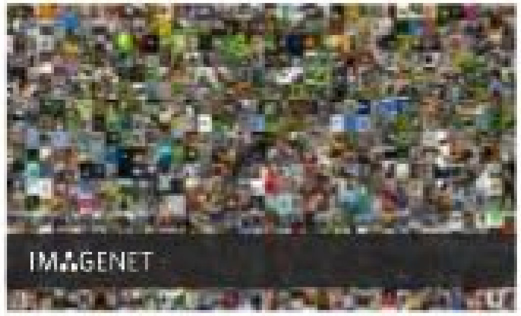
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Why is Deep Learning Hot Now?

Big Data Availability

facebook	350 millions images uploaded per day
Walmart	2.5 Petabytes of customer data hourly
YouTube	300 hours of video uploaded every minute

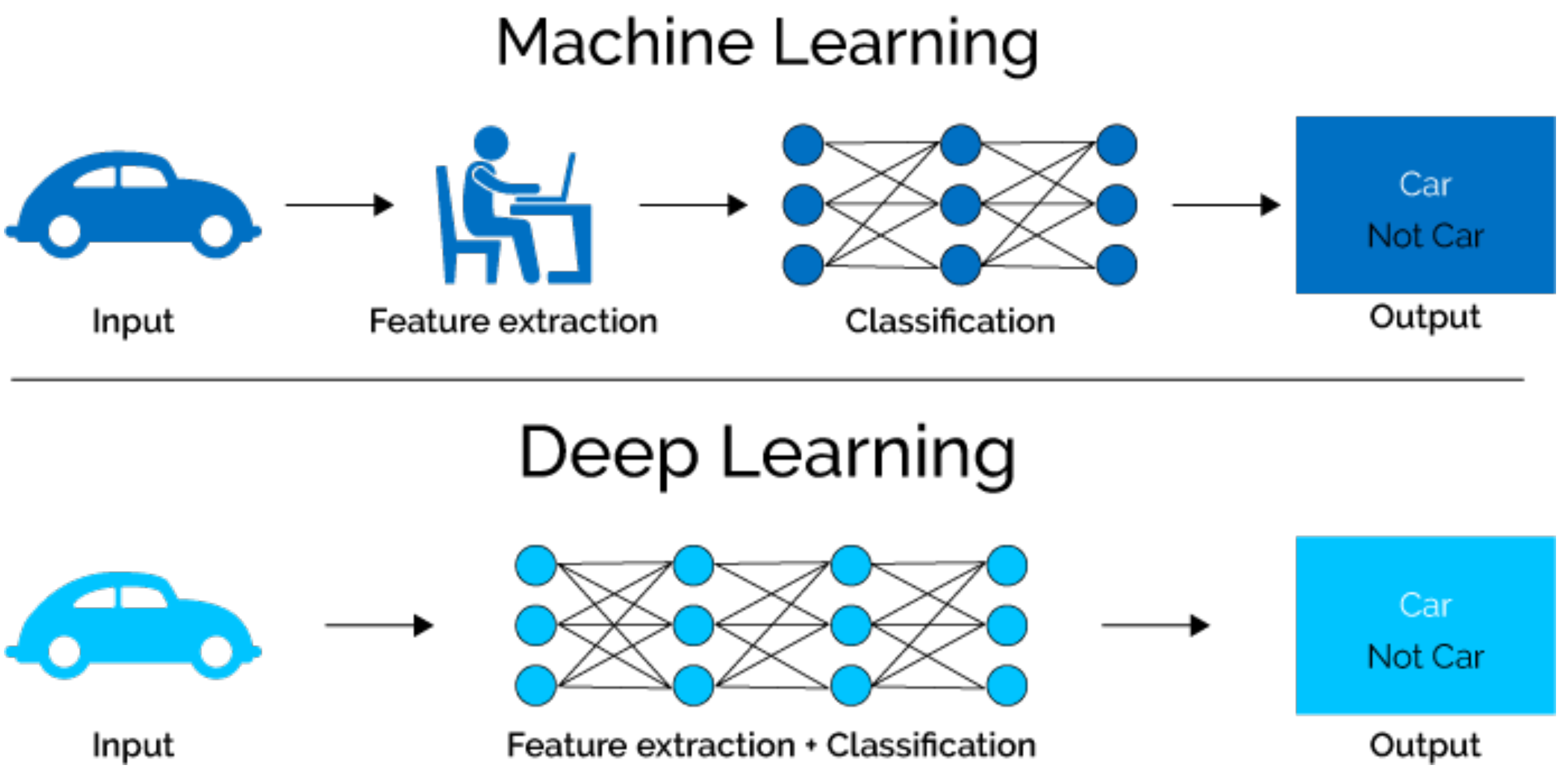
New ML Techniques

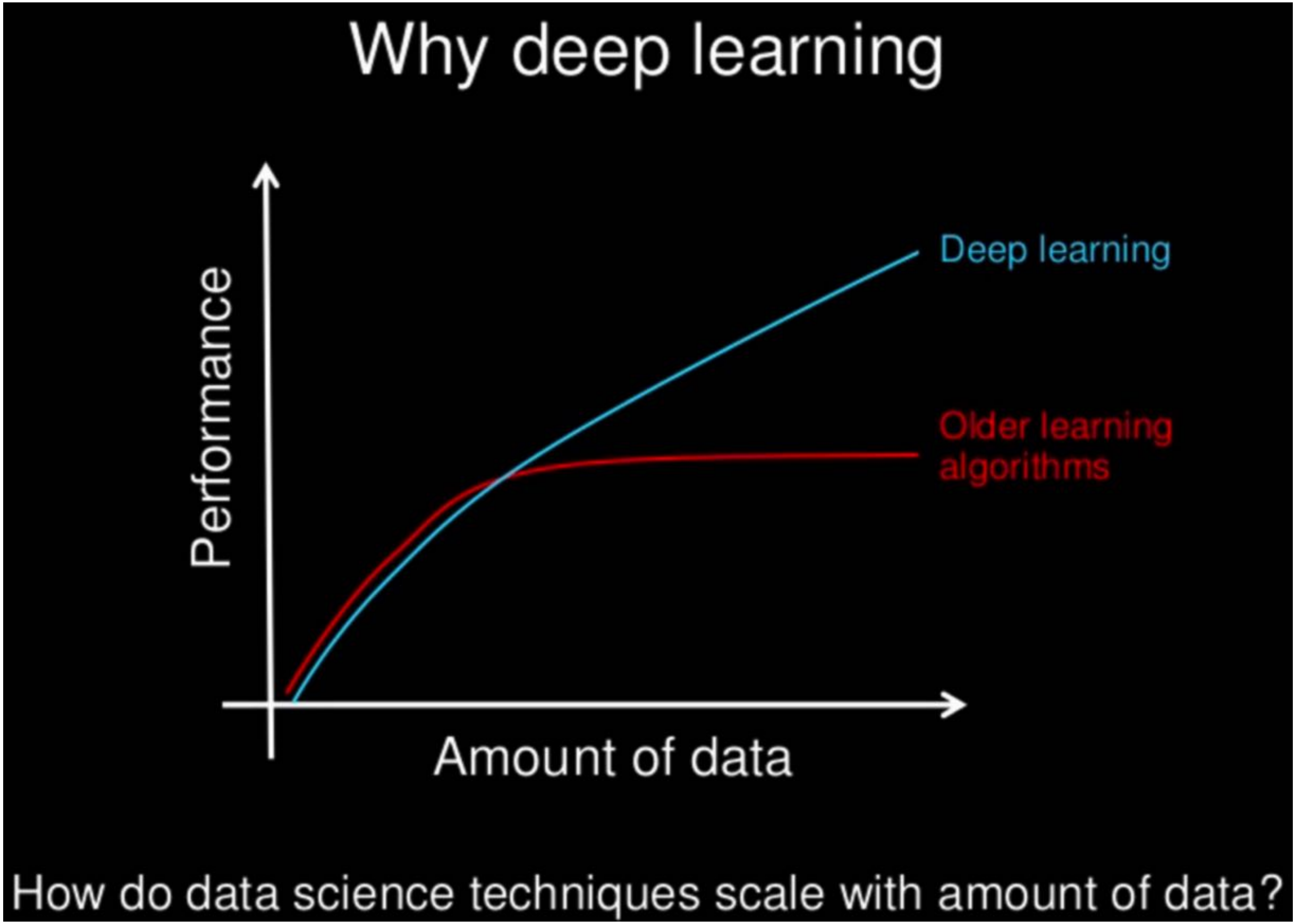


GPU Acceleration

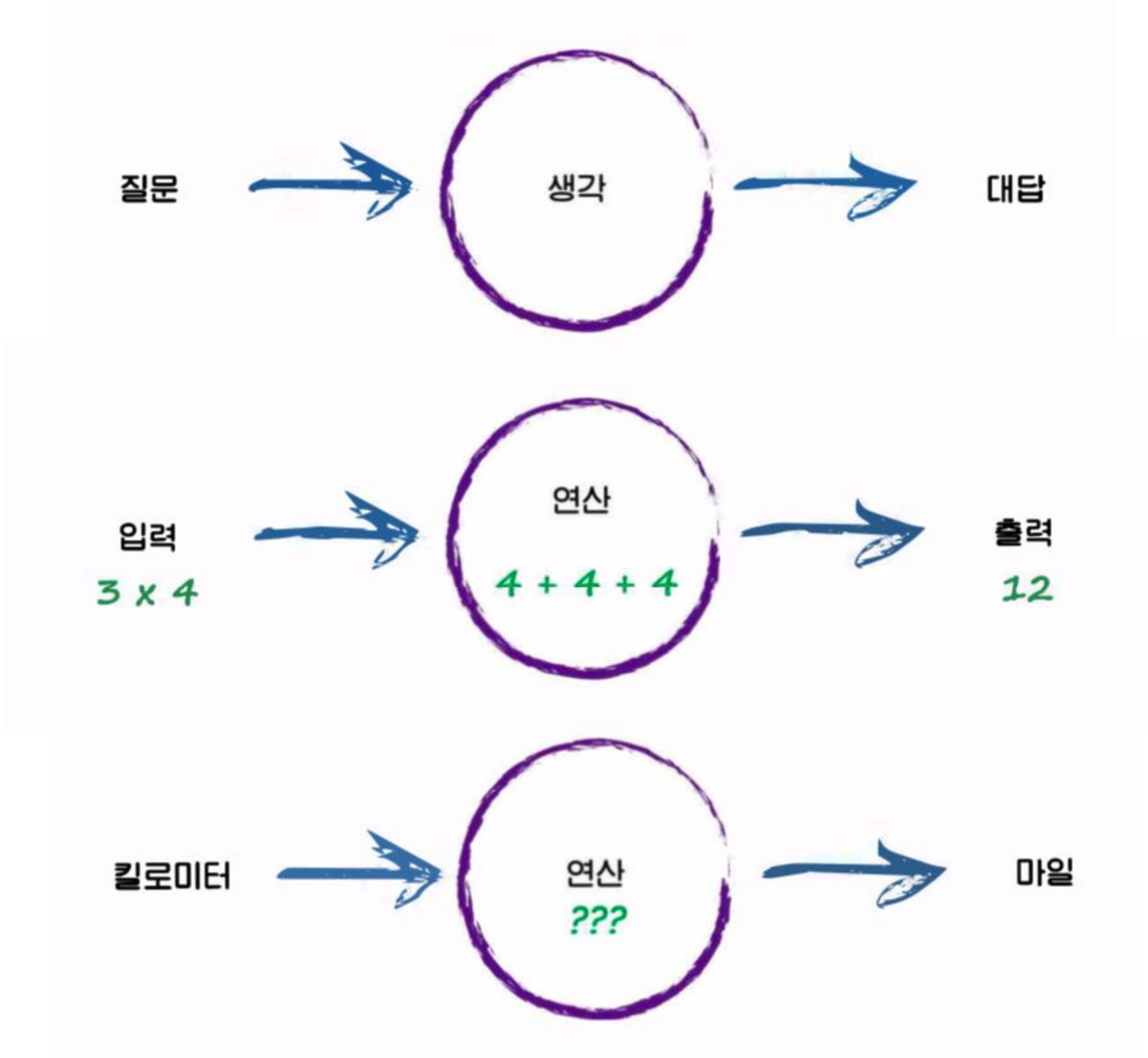
Why is deep learning a growing trend?

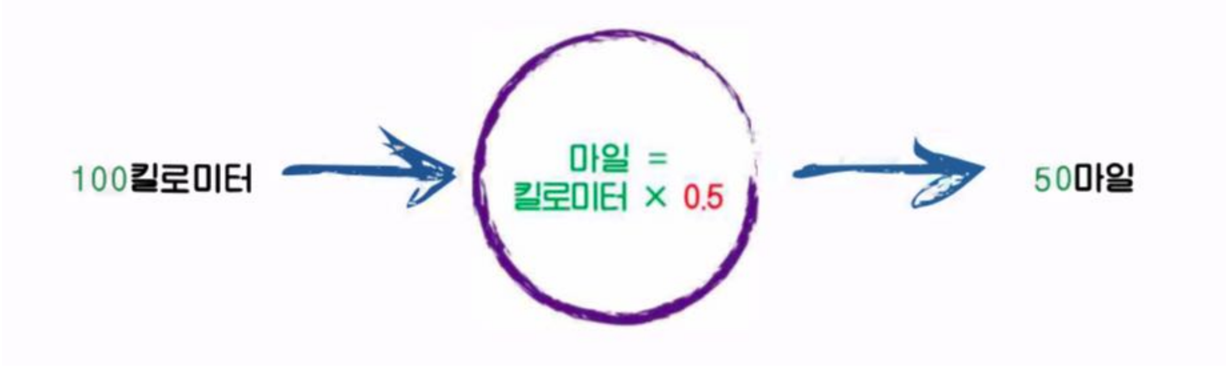
- **few feature engineering**
- state-of-the-art performance

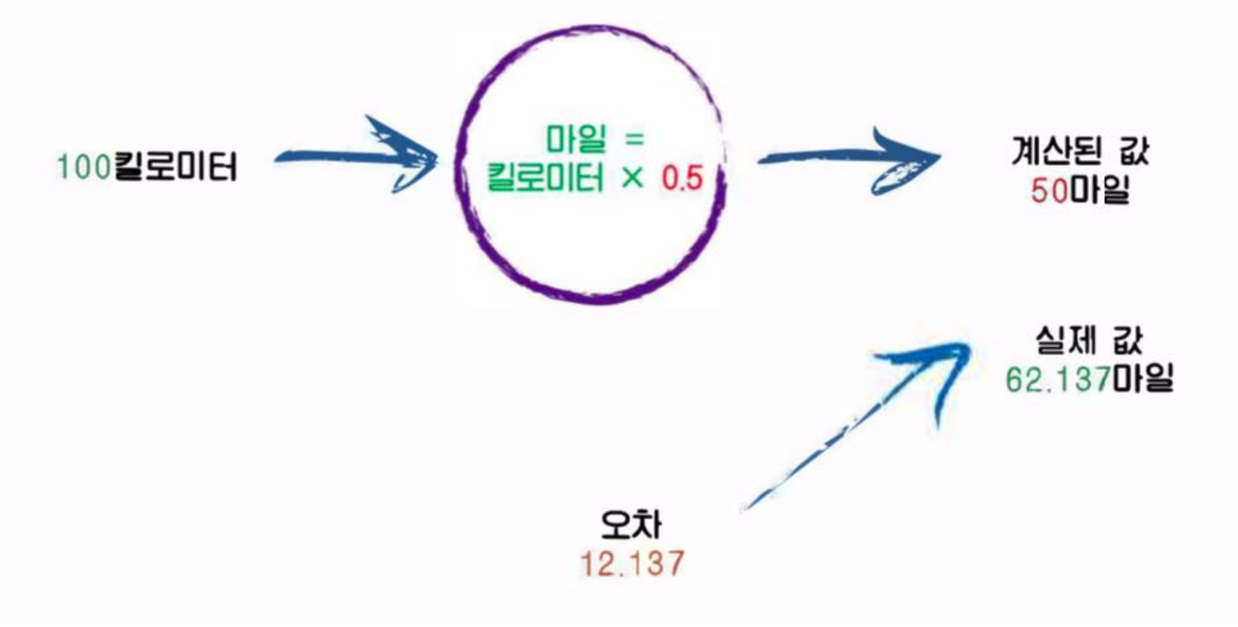

Machine Learning
Input
Feature extraction
Classification
Car
Not Car
Output
Deep Learning
Input
Feature extraction + Classification
Car
Not Car
Output

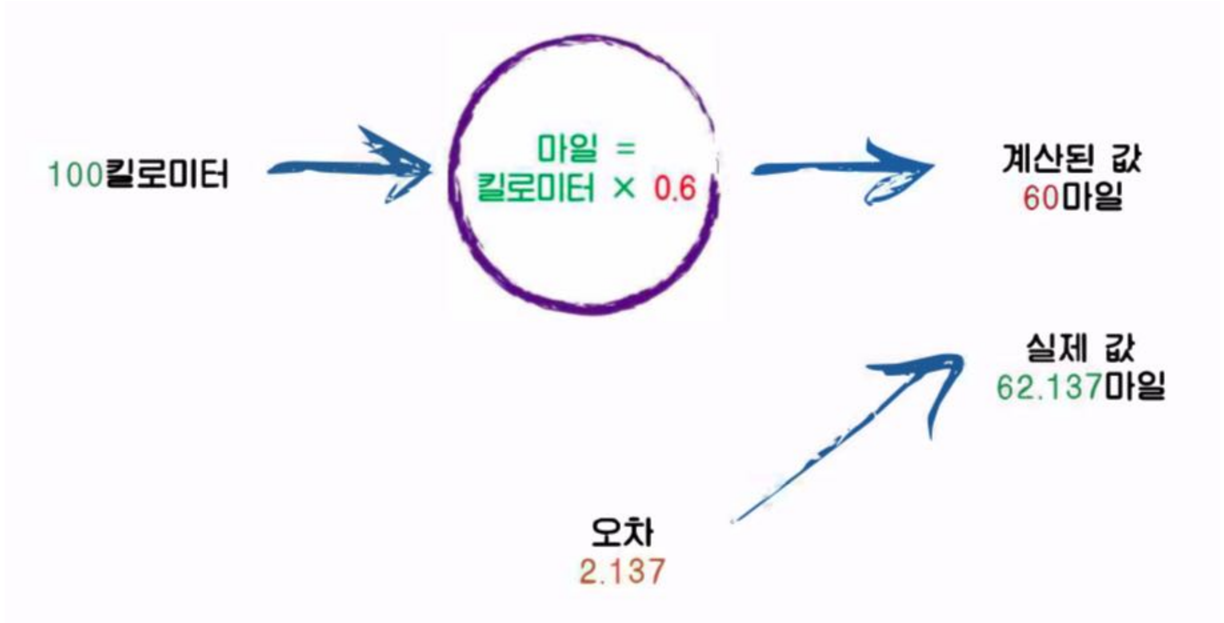

Why deep learning
Deep learning
Older learning algorithms
Performance
Amount of data
How do data science techniques scale with amount of data?

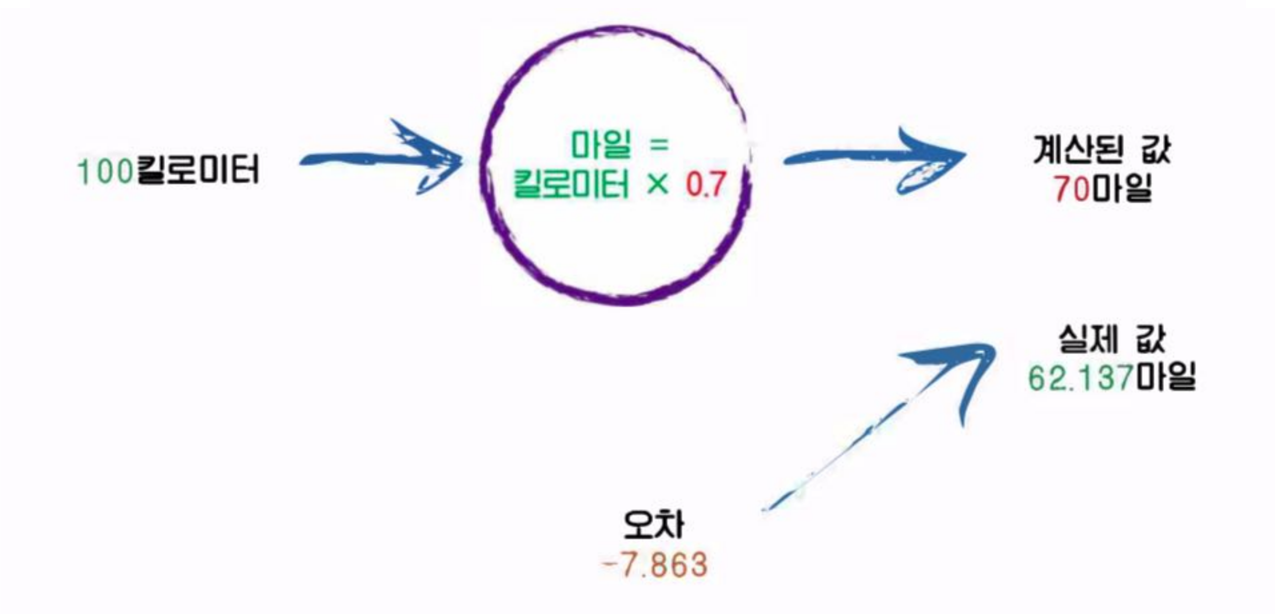
1. Machine Learning

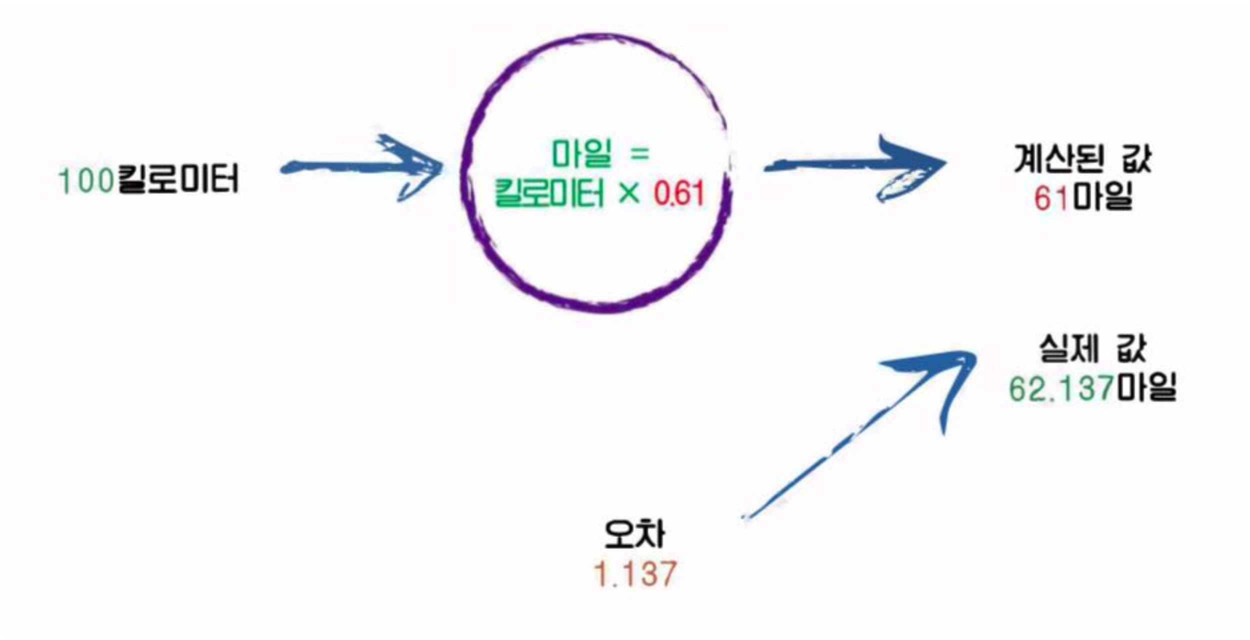

질문
생각
대답
입력
3 x 4
연산
4 + 4 + 4
출력
12
킬로미터
연산
???
마일


100킬로미터
마일 =
킬로미터 × 0.5
50마일


100킬로미터
마일 =
킬로미터 × 0.5
계산된 값
50마일
실제 값
62.137마일
오차
12.137


100킬로미터
마일 =
킬로미터 × 0.6
계산된 값
60마일
실제 값
62.137마일
오차
2.137


100킬로미터
마일 =
킬로미터 × 0.7
계산된 값
70마일
실제 값
62.137마일
오차
-7.863



어떤것의 동작원리를 정확히 파악할수 없을때 취할수있는한방

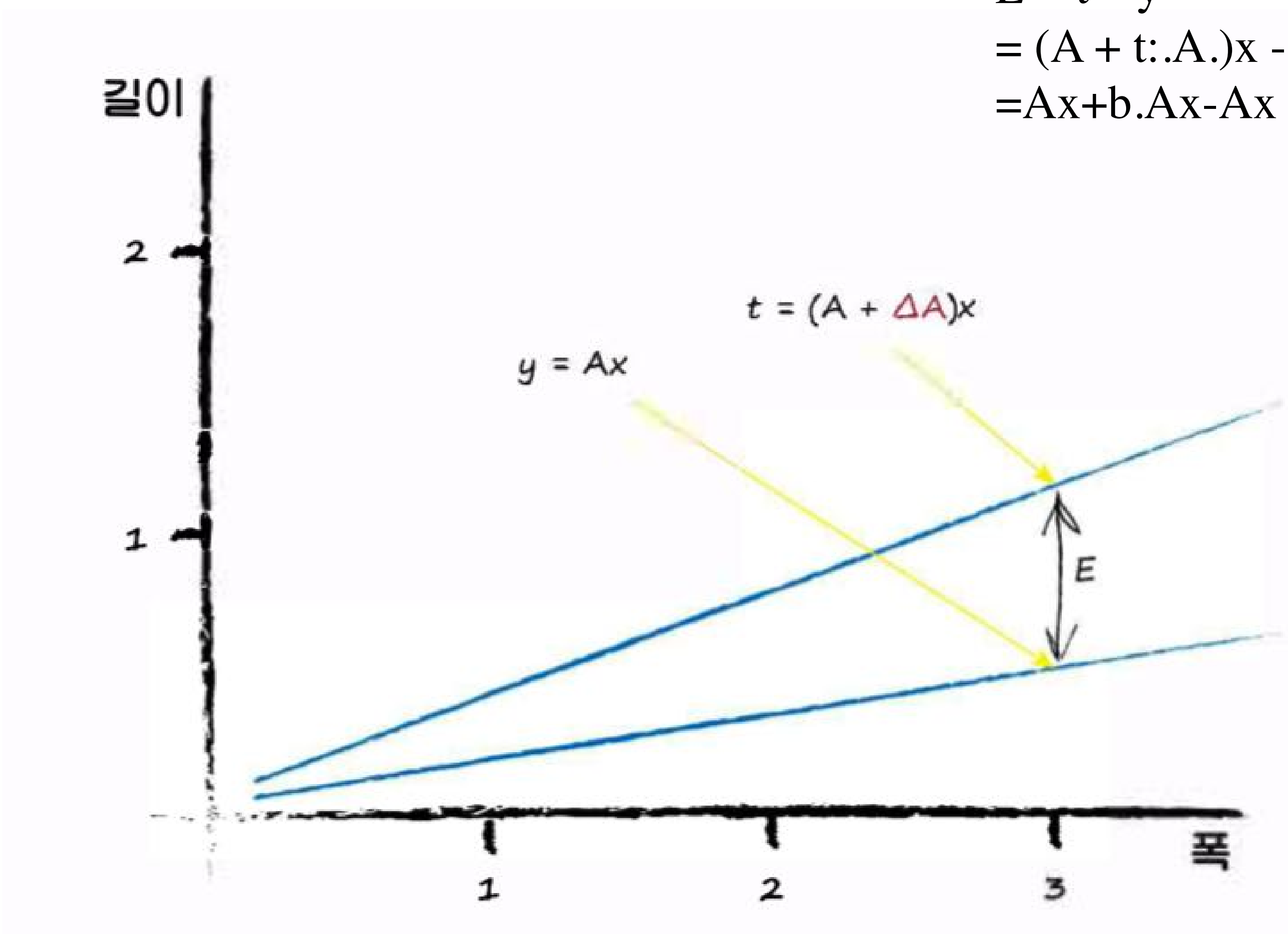
- 조정할 수 있는 매개변수 값율 포함하는 모델을 만들어보는 것
- 모델을 정교회해나가는좋은 방법은오치에 기초해 매개변수 값을 조정해나가는 것.

E = t—y

= (A + t:.A.)x - Ax

=Ax+b.Ax-Ax = b.Ax

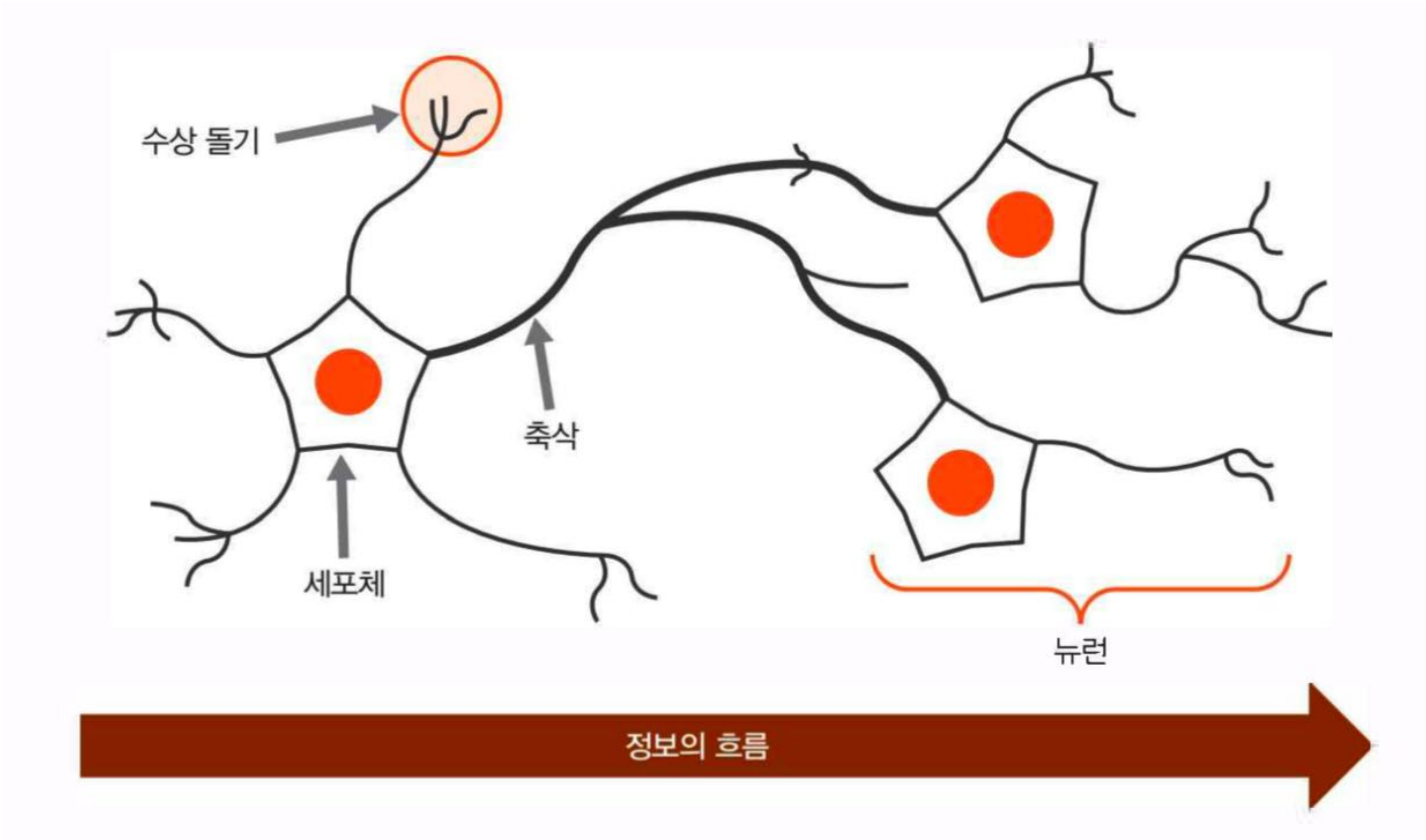
b.A=E/x

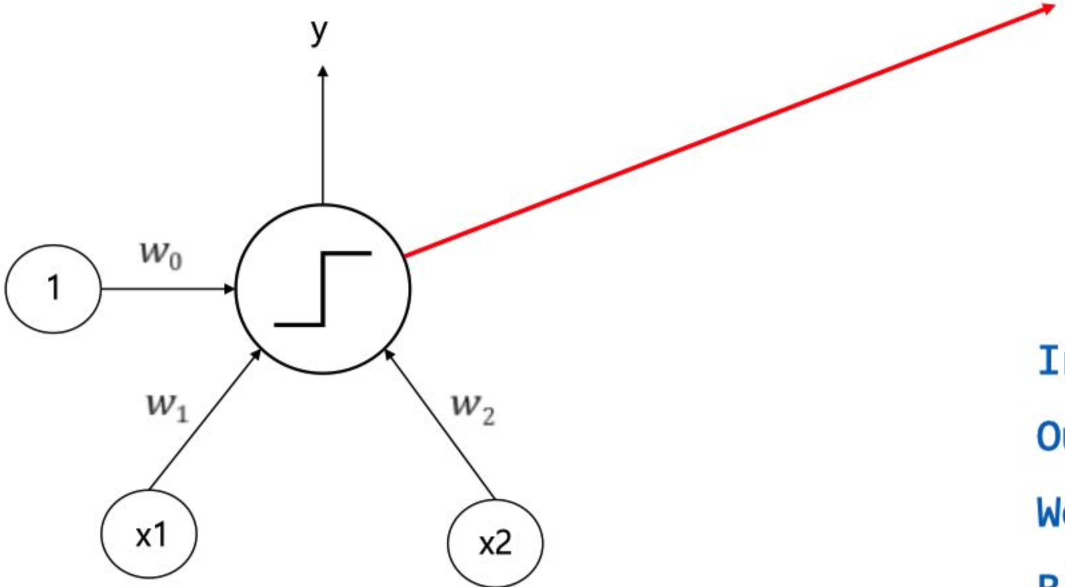


조정 과정의 문제점은 이전의 학습 데이터는 무시하고 최종 학습 데이터에만 맞춰 업데이트된다는 것
이를 해결하기 위해 학습률을 도입해 업데이트의 정도를 조정 해줍니다.
이를 통해 단일 학습 데이터가 힉습에 지배적인 영향을 주는 것을 방지할 수 있 습니다.
현실에서 학습 데이터는 잡음이 섞여 있거나 오차를 가집니다. 학습률을 이용한 업데이트는 이러한 데이터의 오류의 영향을 제한하는 효과

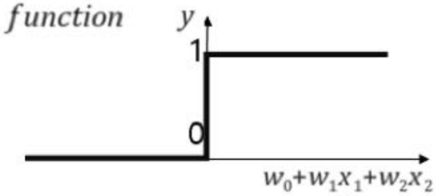
1. Perceptron

지능 = 뇌


수상 돌기
축삭
세포체
뉴런
정보의 흐름



hard thresholding function



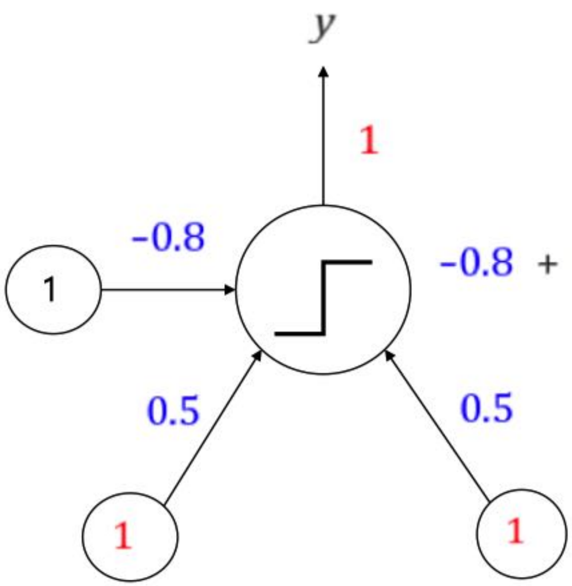
$$y = \begin{cases} 1 & w_0+w_1x_1+w_2x_2 \geq 0 \\ 0 & otherwise \end{cases}$$

Inputs: x1, x2

Output: y

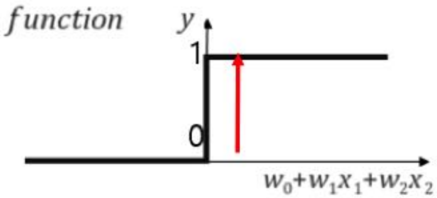
Weights: w0, w1, w2

Bias: 1



$$-0.8 + 0.5\times 1 + 0.5\times 1 = 0.2$$

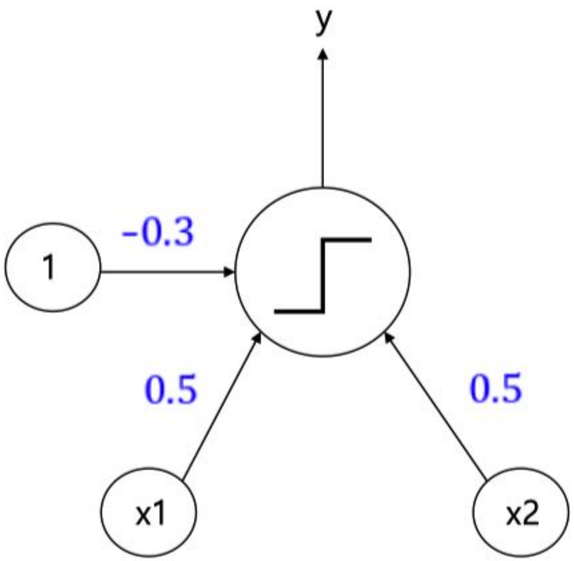
hard thresholding function



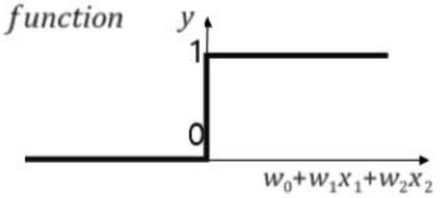
$$y = \begin{cases} 1 & w_0+w_1x_1+w_2x_2 \geq 0 \\ 0 & otherwise \end{cases}$$

AND Gate

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



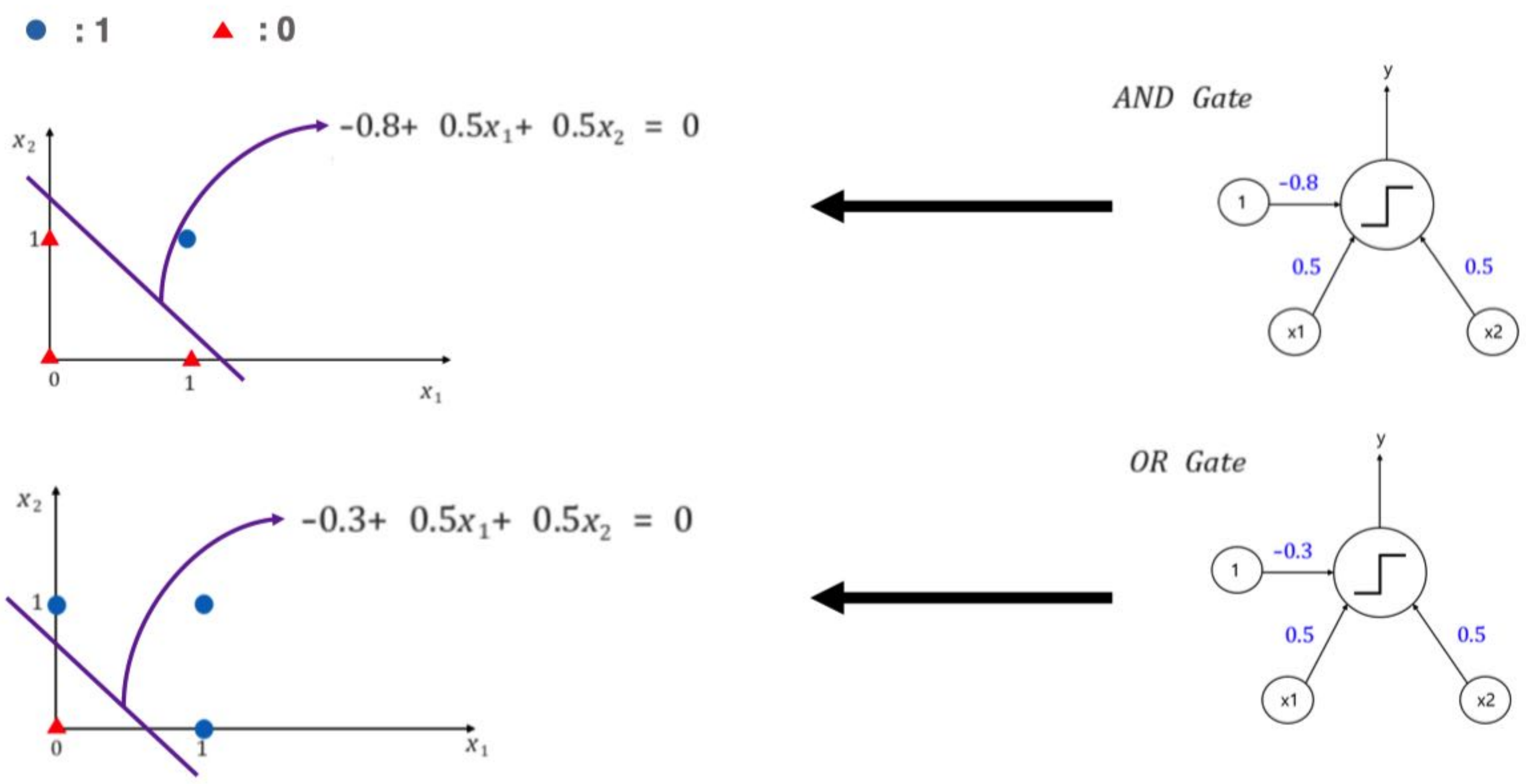
hard thresholding function



$$y = \begin{cases} 1 & w_0+w_1x_1+w_2x_2 \geq 0 \\ 0 & otherwise \end{cases}$$

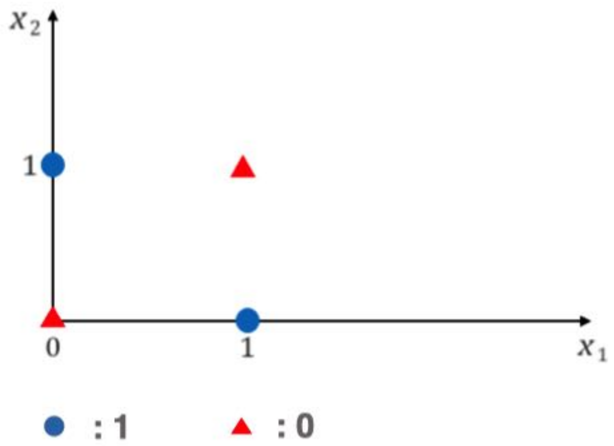
OR Gate

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1


: 1
: 0
$-0.8+\ 0.5x_1+\ 0.5x_2\ =\ 0$
x_2
x_1
0
1
1
AND Gate
y
1
-0.8
0.5
0.5
x1
x2
$-0.3+\ 0.5x_1+\ 0.5x_2\ =\ 0$
x_2
x_1
0
1
1
OR Gate
y
1
-0.3
0.5
0.5
x1
x2

Is it possible to solve XOR problem using a single layer perceptron?

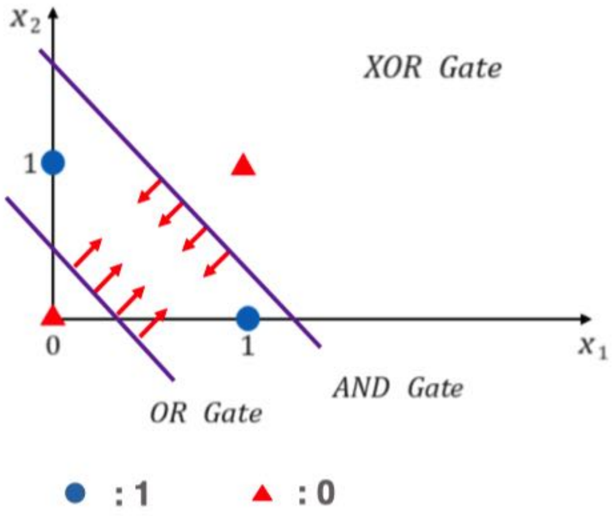
No. Single layer perceptron can only solve linear problem. XOR problem is non-linear.

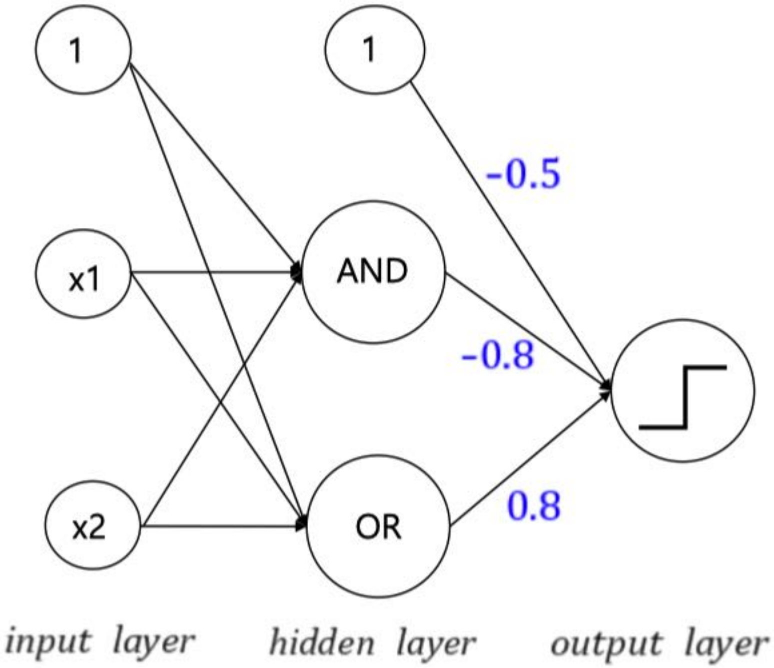


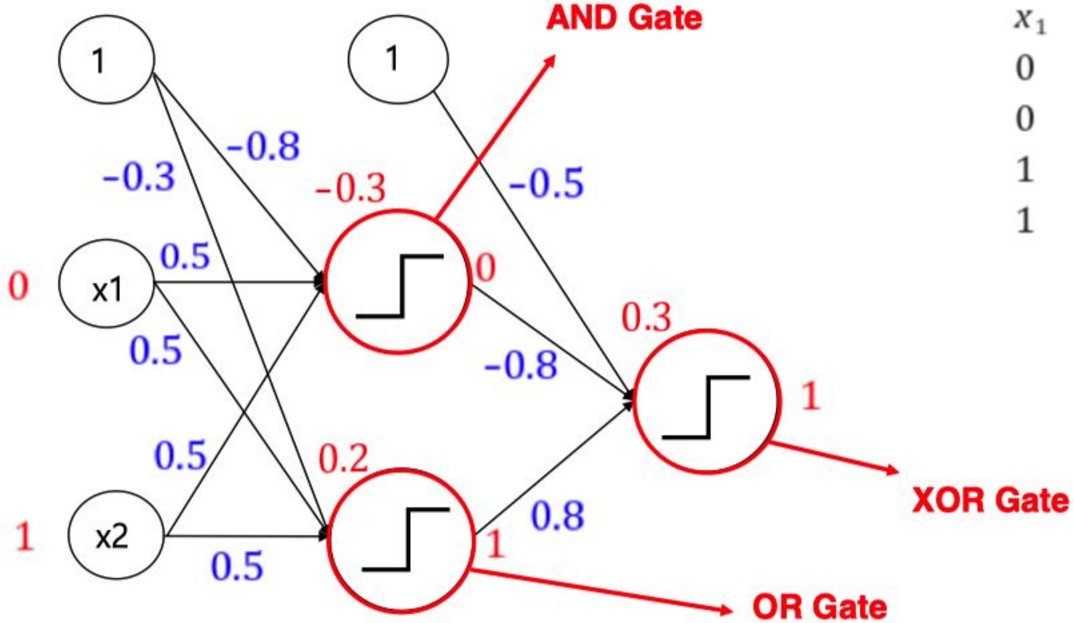
XOR Gate

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

But if we use 2 single layer perceptron, we can solve XOR problem. This model is called multi layer perceptron.







OR Gate

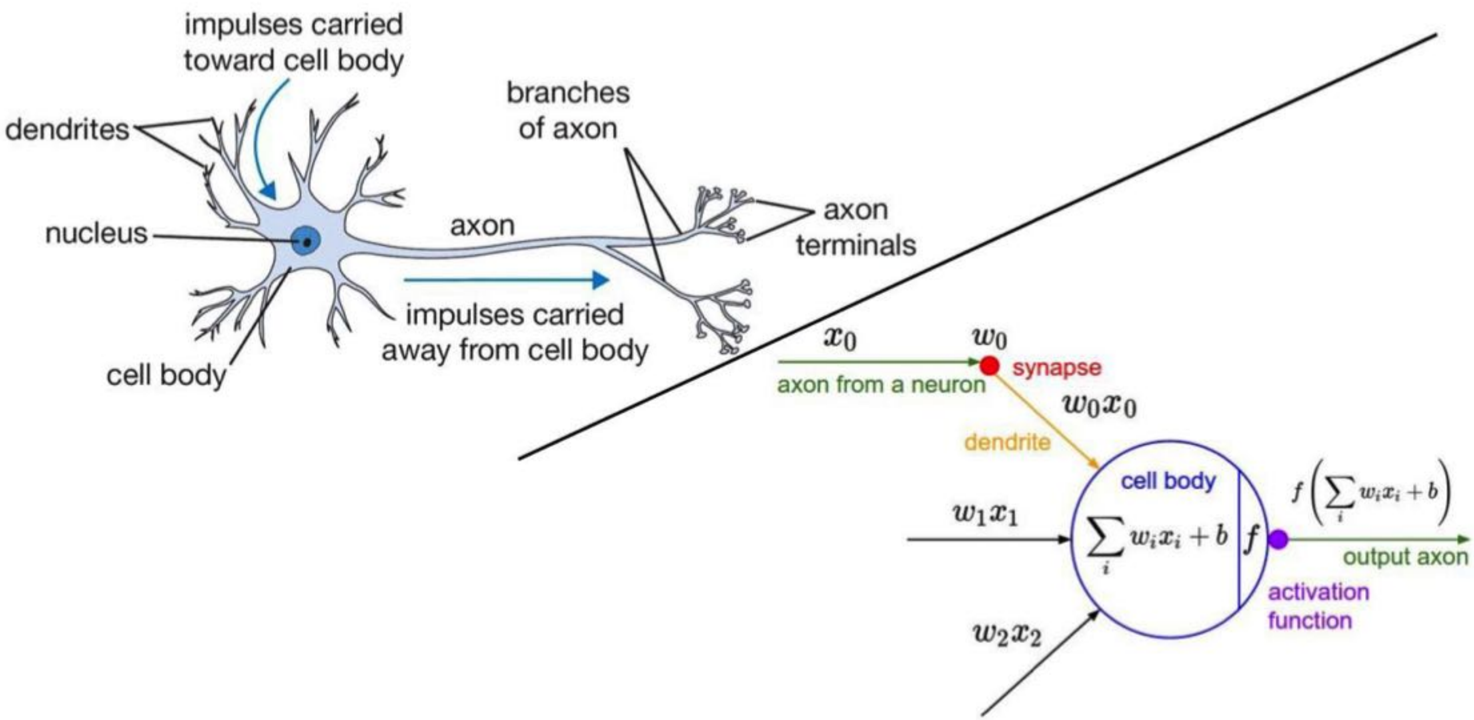
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

AND Gate

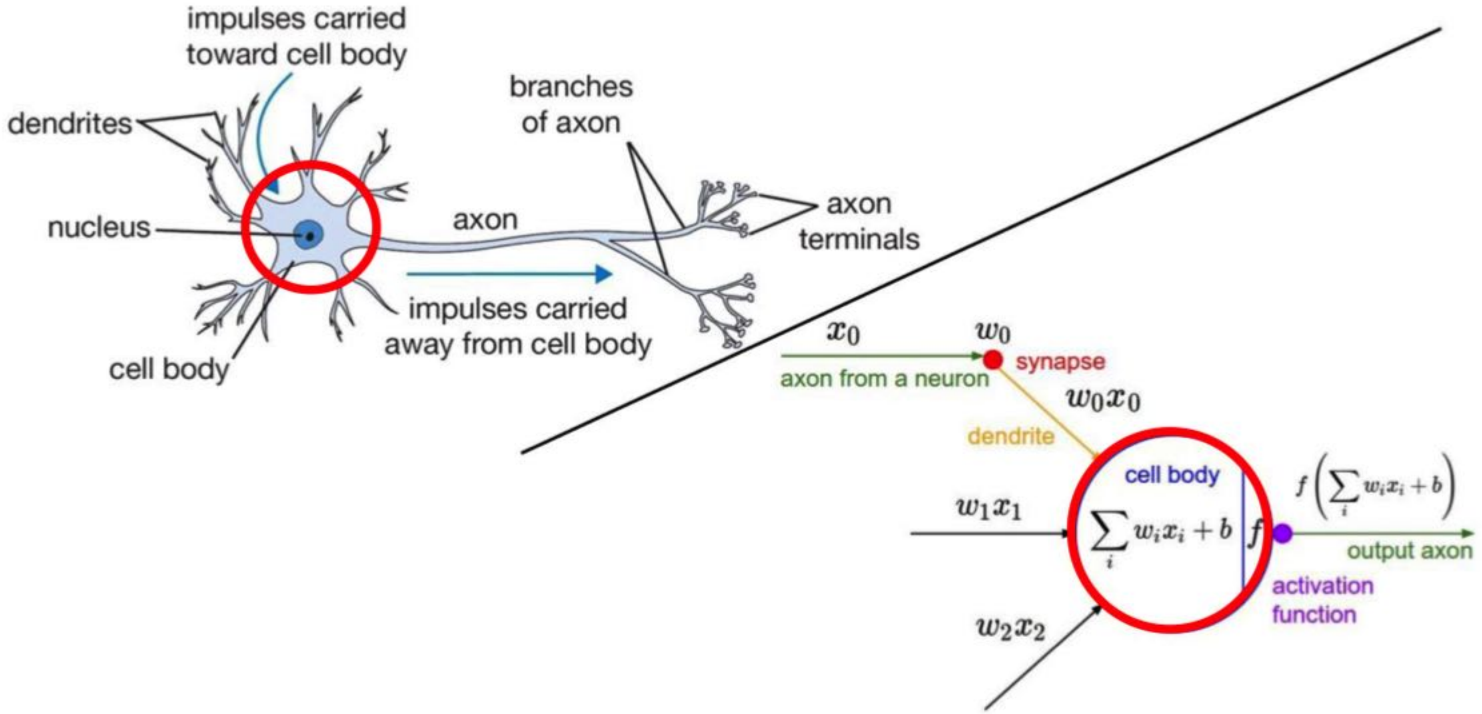
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

XOR Gate

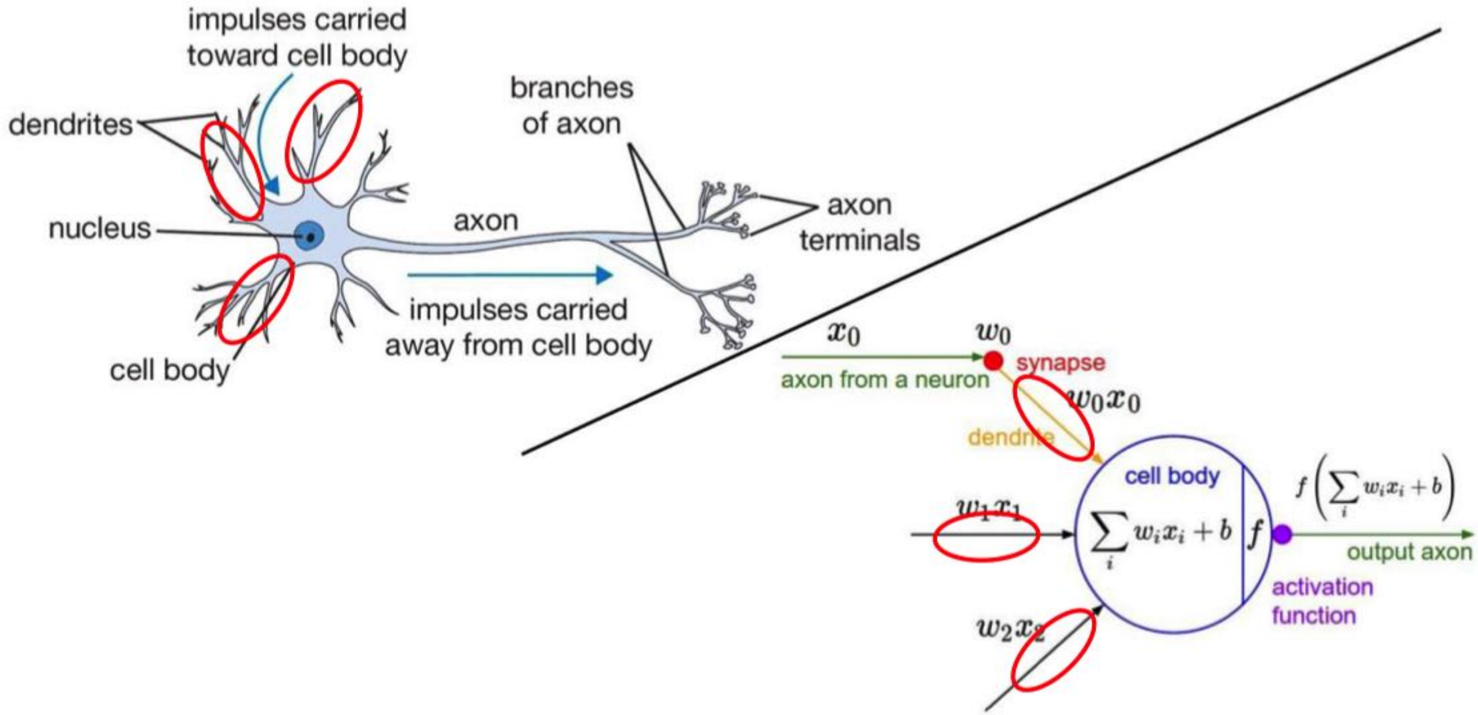
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



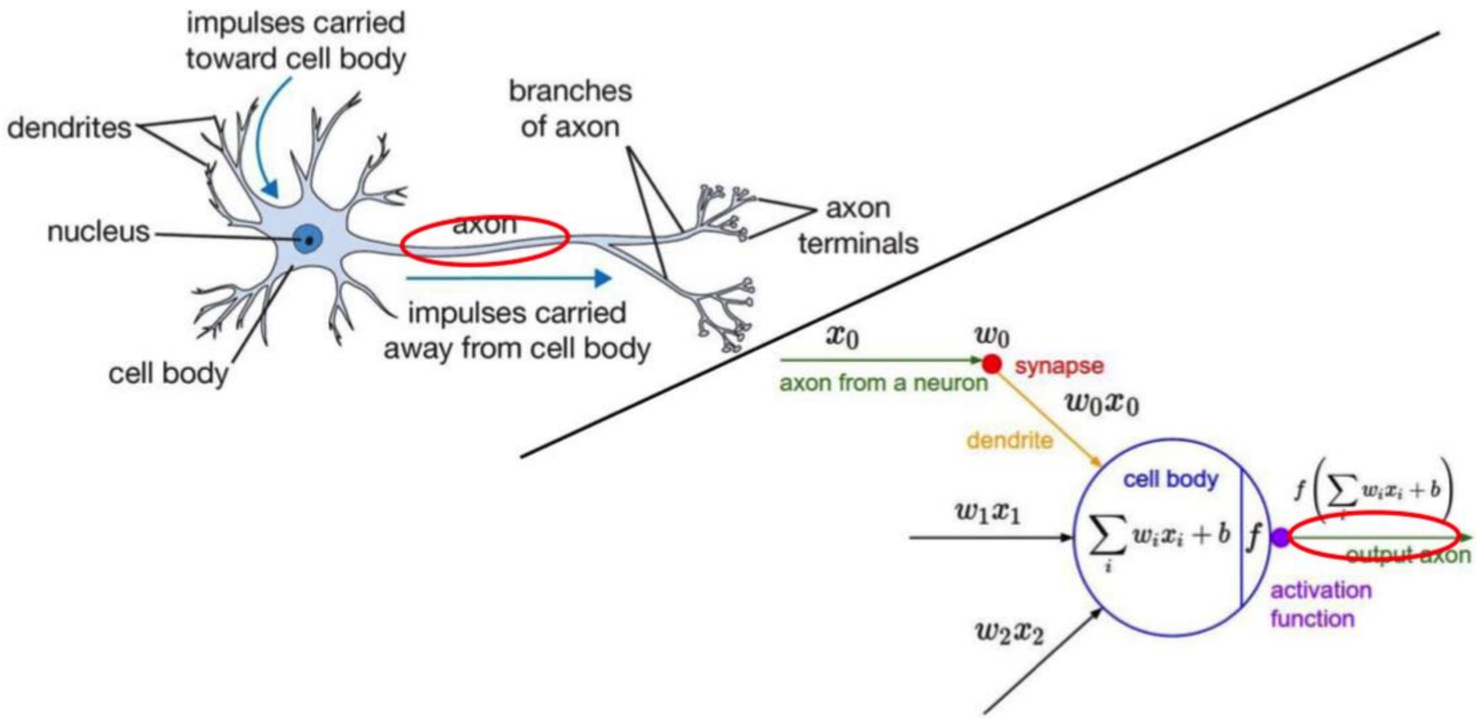
impulses carried toward cell body
dendrites
branches of axon
axon
axon terminals
nucleus
impulses carried away from cell body
cell body
x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation function
$w_2 x_2$

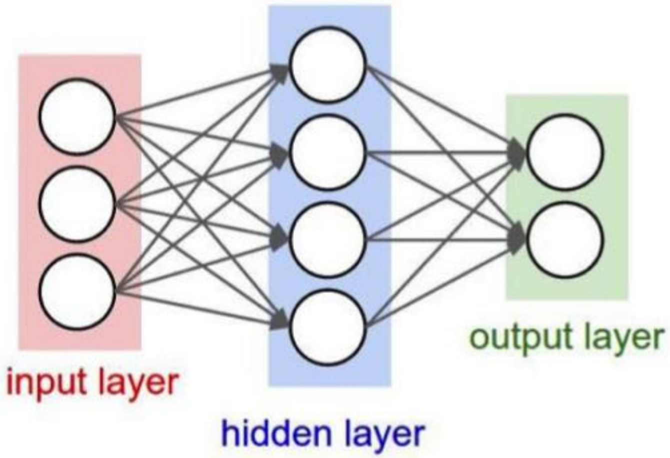


impulses carried
toward cell body
dendrites
branches
of axon
axon
terminals
nucleus
axon
impulses carried
away from cell body
cell body
x_0
w_0
synapse
axon from a neuron
w_0x_0
dendrite
cell body
$f\left(\sum_i w_ix_i + b\right)$
w_1x_1
$\sum_i w_ix_i + b$
f
output axon
activation
function
w_2x_2

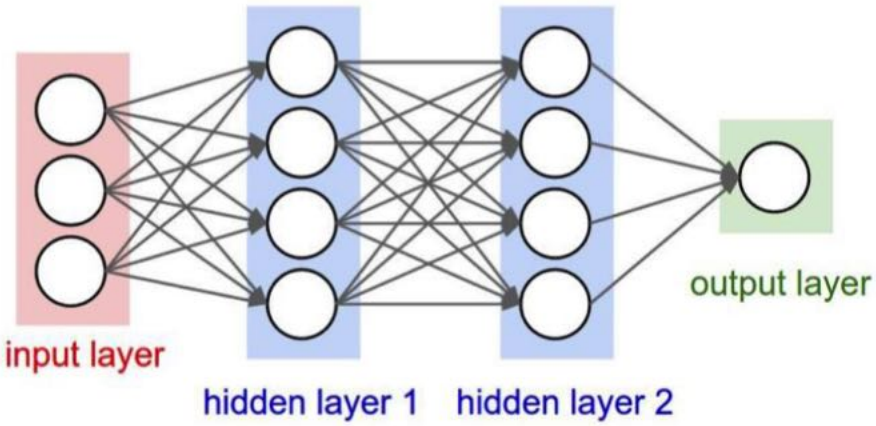


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body
x_0
w_0
synapse
axon from a neuron
w_0x_0
dendrite
cell body
$f\left(\sum_i w_ix_i + b\right)$
w_1x_1
$\sum_i w_ix_i + b$
f
output axon
activation function
w_2x_2

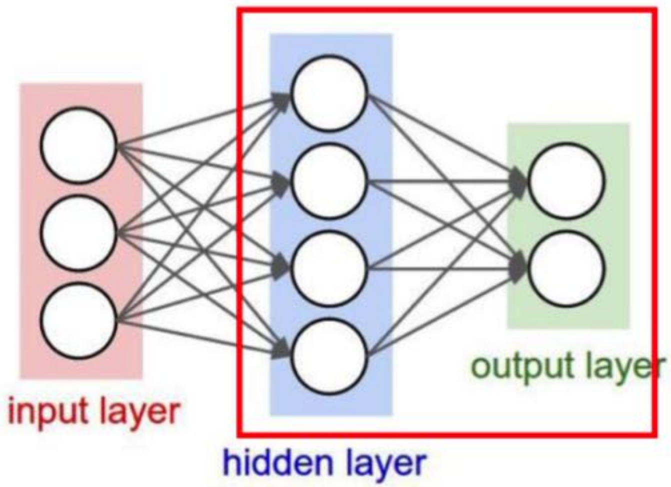

impulses carried
toward cell body
dendrites
nucleus
axon
branches
of axon
axon
terminals
impulses carried
away from cell body
cell body
x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation
function
$w_2 x_2$



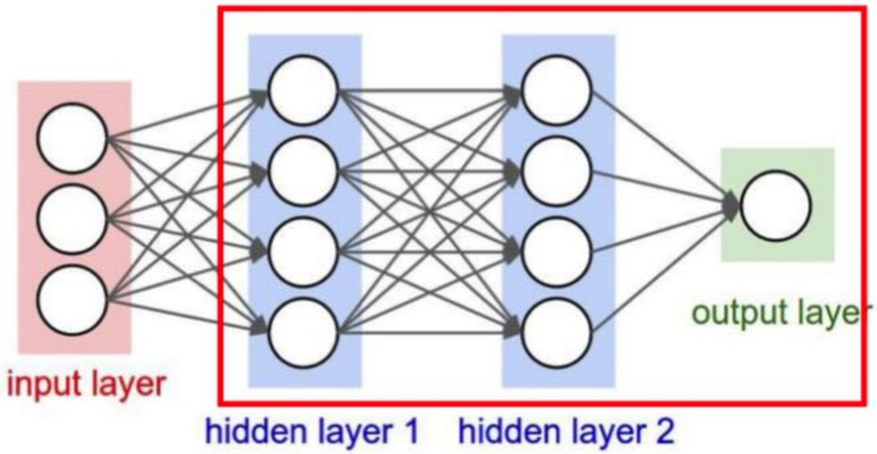
“2-layer Neural Network”
or
“1-hidden-layer Neural Network”



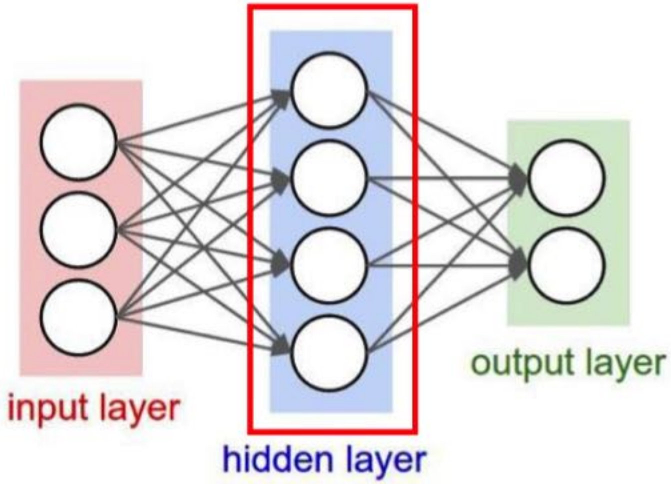
“3-layer Neural Network”
or
“2-hidden-layer Neural Network”



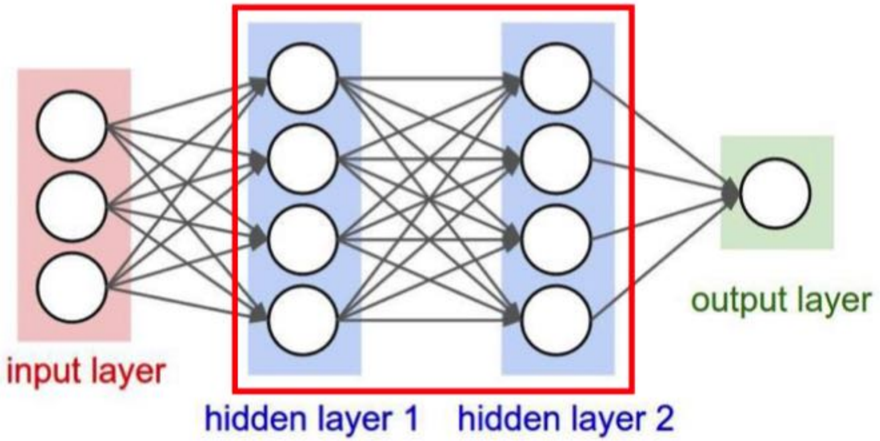
"2-layer Neural Network"
or
"1-hidden-layer Neural Network"



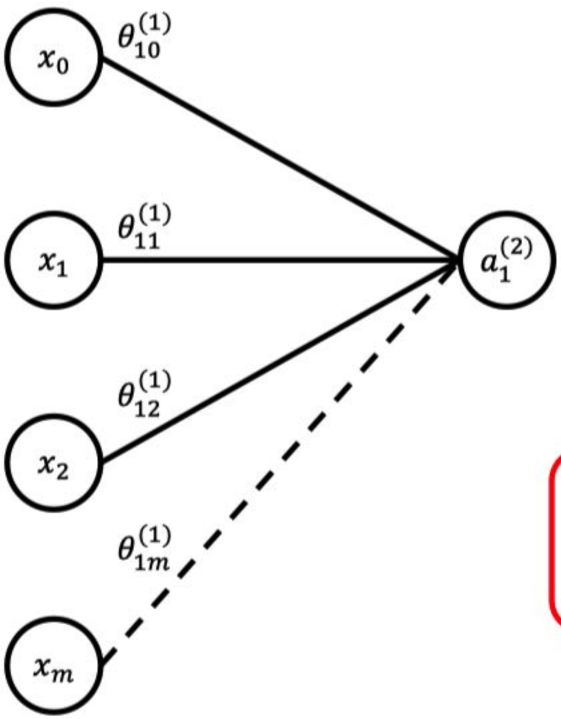
"3-layer Neural Network"
or
"2-hidden-layer Neural Network"



"2-layer Neural Network"
or
"1-hidden-layer Neural Network"



"3-layer Neural Network"
or
"2-hidden-layer Neural Network"

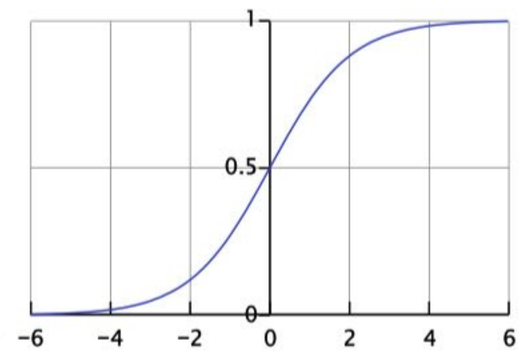


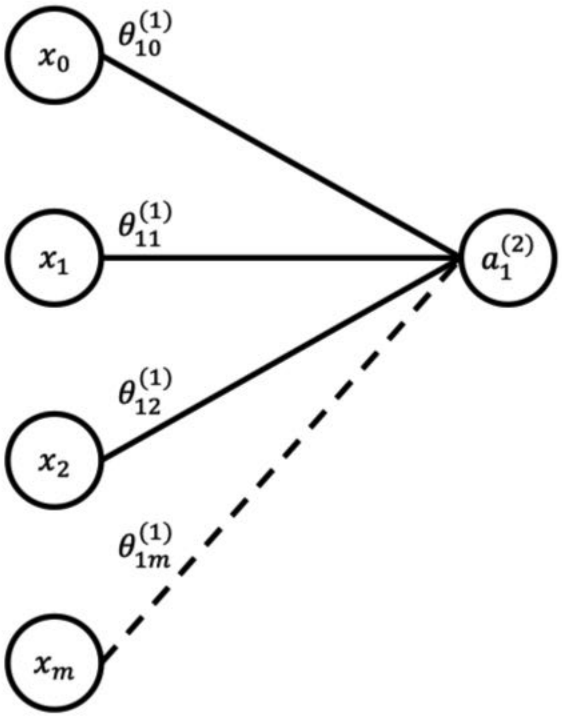
$$z_1^{(2)} = \theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \cdots + \theta_{1m}^{(1)} x_m$$
$$= \sum_{i=0}^{m} \theta_{1i}^{(1)} x_i$$

$$a_1^{(2)} = \sigma\left(z_1^{(2)}\right)$$

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

Logistic sigmoid →





$$z_1^{(2)} = \theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \cdots + \theta_{1m}^{(1)} x_m$$

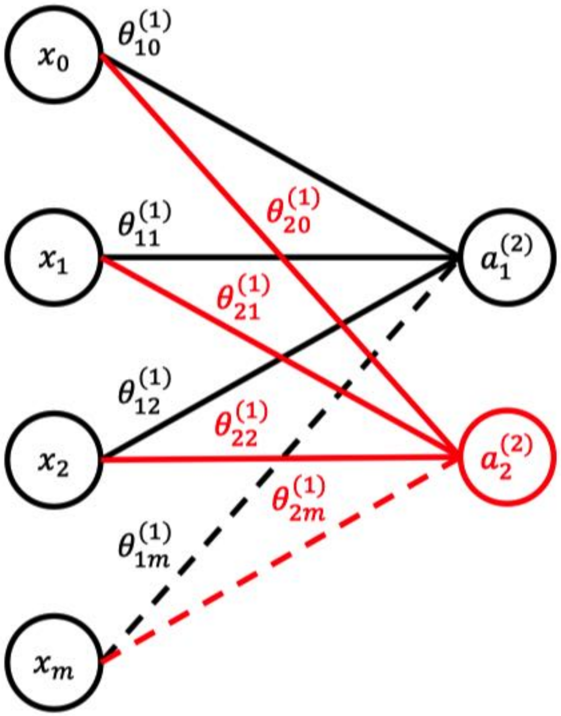
$$\boxed{= \sum_{i=0}^{m} \theta_{1i}^{(1)} x_i}$$

Linear combination

$$\begin{bmatrix} \theta_{10}^{(1)} & \theta_{11}^{(1)} & \theta_{12}^{(1)} & \ldots & \theta_{1m}^{(1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

$$a_1^{(2)} = \sigma\left(z_1^{(2)}\right)$$

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



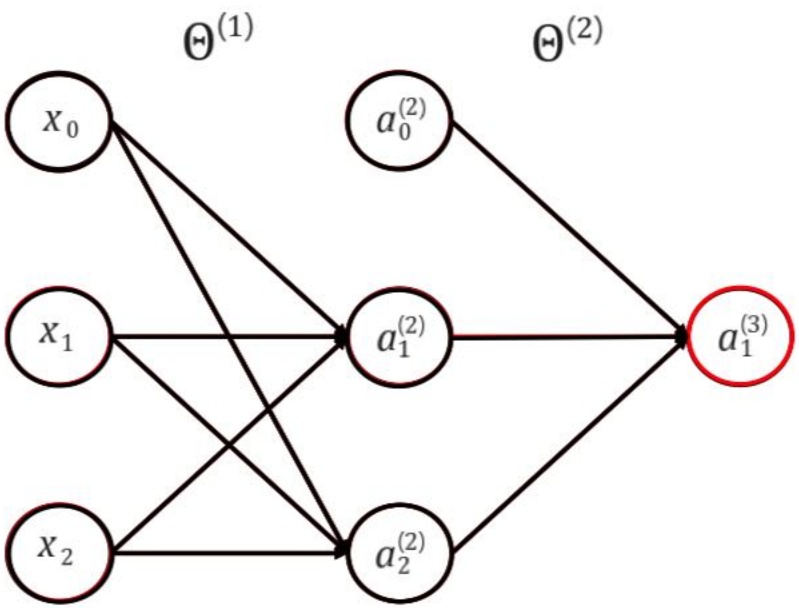
$$z_1^{(2)} = \theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \cdots + \theta_{1m}^{(1)}x_m$$
$$= \sum_{i=0}^{m} \theta_{1i}^{(1)}x_i$$

$$z_2^{(2)} = \theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \cdots + \theta_{2m}^{(1)}x_m$$
$$= \sum_{i=0}^{m} \theta_{2i}^{(1)}x_i$$

$$\begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \end{bmatrix} = \begin{bmatrix} \theta_{10}^{(1)} & \theta_{11}^{(1)} & \theta_{12}^{(1)} & \dots & \theta_{1m}^{(1)} \\ \theta_{20}^{(1)} & \theta_{21}^{(1)} & \theta_{22}^{(1)} & \dots & \theta_{2m}^{(1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

$$(2 \times m) \times (m \times 1)$$

Feed Forward

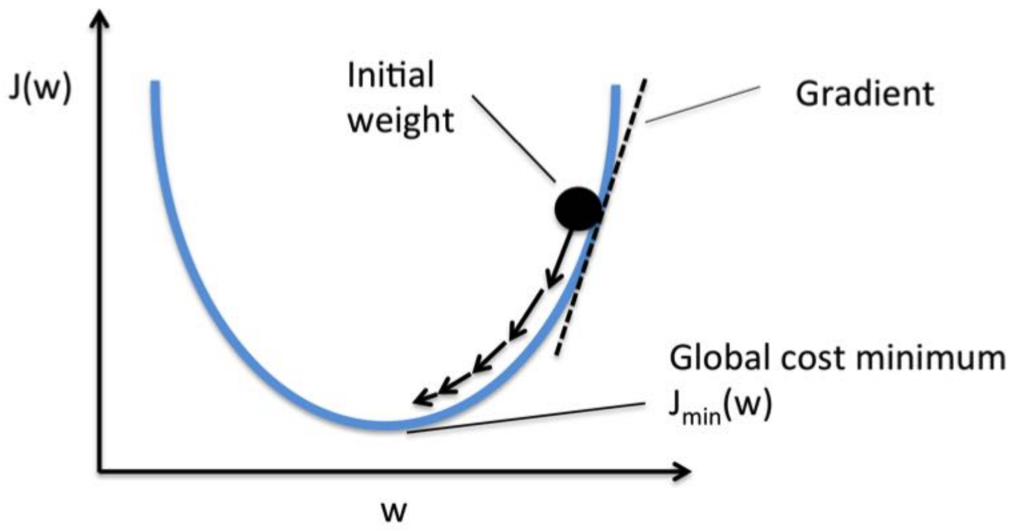


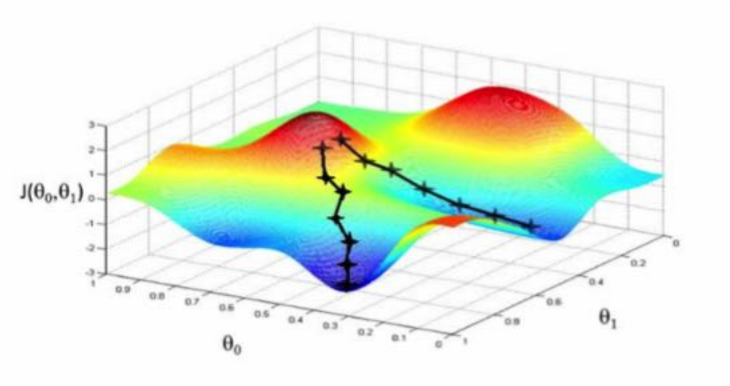
$$\begin{pmatrix} z_1^{(2)} \\ z_2^{(2)} \end{pmatrix} = \begin{pmatrix} \Theta_{10}^{(1)} & \Theta_{11}^{(1)} & \Theta_{12}^{(1)} \\ \Theta_{20}^{(1)} & \Theta_{21}^{(1)} & \Theta_{22}^{(1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

$$\begin{pmatrix} a_1^{(2)} \\ a_2^{(2)} \end{pmatrix} = \begin{pmatrix} g(z_1^{(2)}) \\ g(z_2^{(2)}) \end{pmatrix}$$

$$z_1^{(3)} = \begin{pmatrix} \Theta_{20}^{(1)} & \Theta_{21}^{(1)} & \Theta_{22}^{(1)} \end{pmatrix} \begin{pmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \end{pmatrix}$$

$$\boxed{a_1^{(3)} = g(z_1^{(3)})}$$

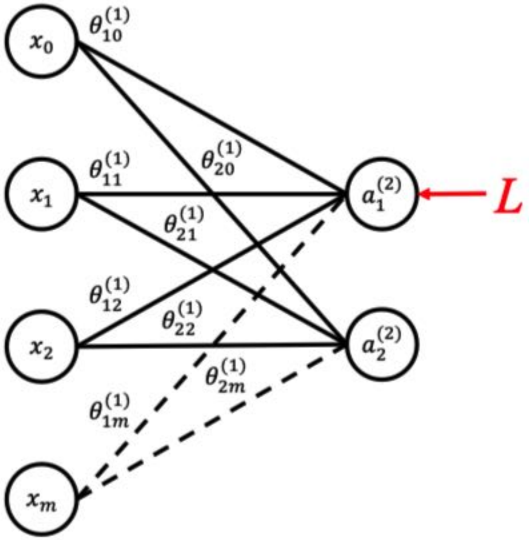

J(w)
Initial
weight
Gradient
Global cost minimum
$J_{min}(w)$
w



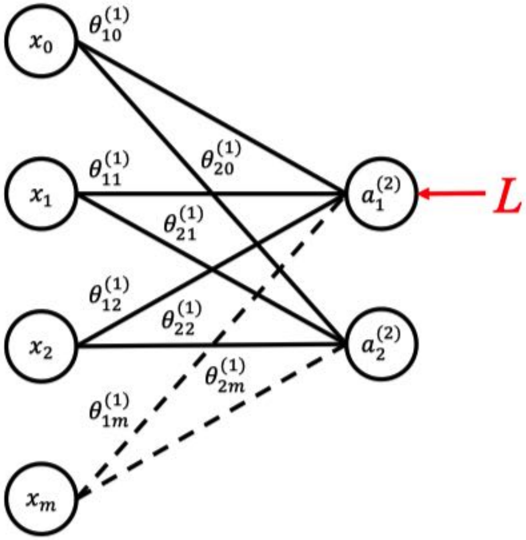
- Compute $\frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}$
- Update weights with

$$\theta_0 := \theta_0 - \alpha \frac{\partial J}{\partial \theta_0}$$

$$\theta_1 := \theta_1 - \alpha \frac{\partial J}{\partial \theta_1}$$



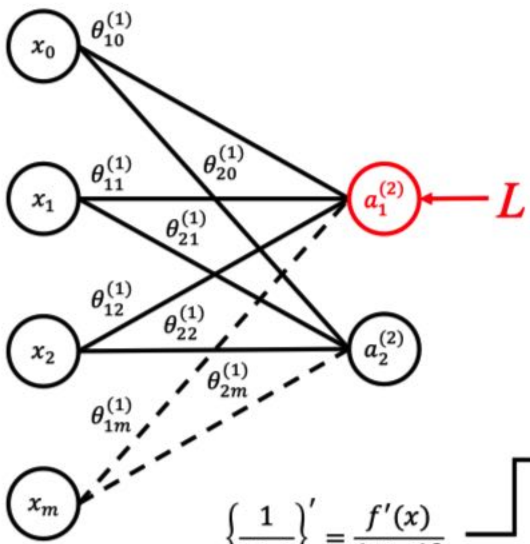
- Use chain rule (for MSE loss)



- Use chain rule (for MSE loss)

$$\frac{\partial L}{\partial a_1^{(2)}}$$

$$\frac{\partial L}{\partial a_1^{(2)}} = \frac{\partial (y_1 - a_1^{(2)})^2}{\partial a_1^{(2)}} = -2(y_1 - a_1^{(2)})$$

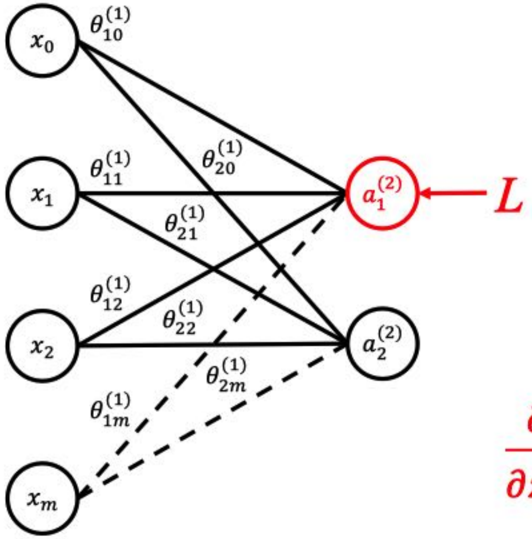


- Use chain rule (for MSE loss)

$$\frac{\partial L}{\partial z_1^{(2)}} = \frac{\partial L}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}}$$

$$\left\{\frac{1}{f(x)}\right\}' = \frac{f'(x)}{\{f(x)\}^2}$$

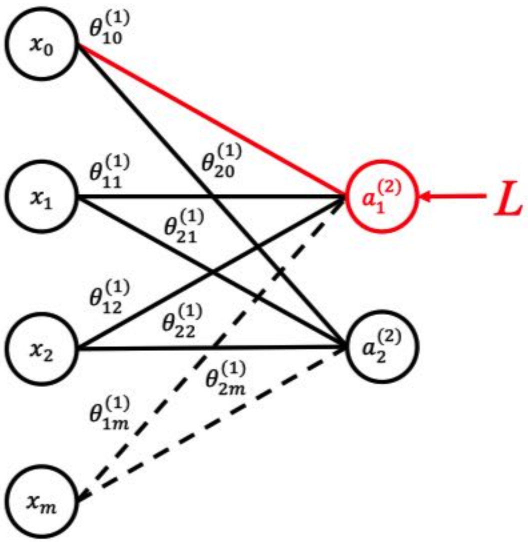
$$\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} = \frac{\partial \sigma(z_1^{(2)})}{\partial z_1^{(2)}} = \sigma(z_1^{(2)})(1 - \sigma(z_1^{(2)}))$$



- Use chain rule (for MSE loss)

$$\frac{\partial L}{\partial z_1^{(2)}} = \frac{\partial L}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}}$$

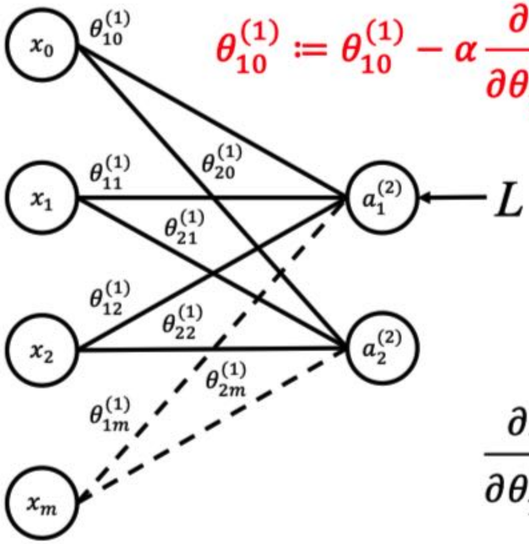
$$\frac{\partial L}{\partial z_1^{(2)}} = -2(y_1 - a_1^{(2)})\sigma(z_1^{(2)})(1 - \sigma(z_1^{(2)}))$$



- Use chain rule (for MSE loss)

$$\frac{\partial L}{\partial \theta_{10}^{(1)}} = \frac{\partial L}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial \theta_{10}^{(1)}}$$

$$\frac{\partial z_1^{(2)}}{\partial \theta_{10}^{(1)}} = \frac{\partial \sum_{i=0}^{m} \theta_{1i}^{(1)} x_i}{\partial \theta_{10}^{(1)}} = x_0$$

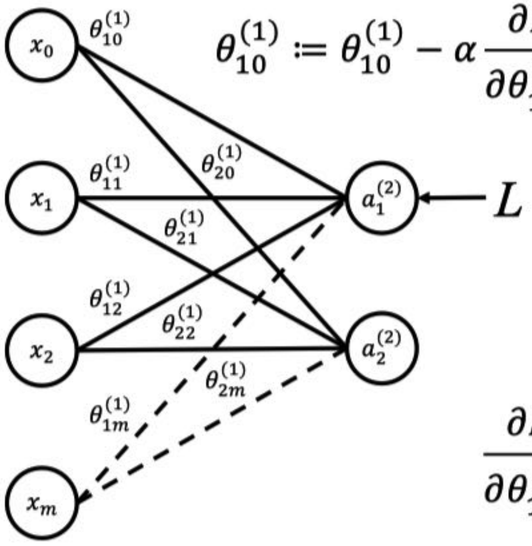


$$\theta_{10}^{(1)} := \theta_{10}^{(1)} - \alpha \frac{\partial L}{\partial \theta_{10}^{(1)}}$$

- Use chain rule (for MSE loss)

$$\frac{\partial L}{\partial \theta_{10}^{(1)}} = \frac{\partial L}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial \theta_{10}^{(1)}}$$

$$\frac{\partial L}{\partial \theta_{10}^{(1)}} = -2(y_1 - a_1^{(2)})\sigma(z_1^{(2)})(1 - \sigma(z_1^{(2)}))x_0$$



$$\theta_{10}^{(1)} := \theta_{10}^{(1)} - \alpha \frac{\partial L}{\partial \theta_{10}^{(1)}}$$

- Use chain rule (for MSE loss)

$$\frac{\partial L}{\partial \theta_{10}^{(1)}} = \frac{\partial L}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial \theta_{10}^{(1)}}$$

$$\frac{\partial L}{\partial \theta_{10}^{(1)}} = -2(y_1 - a_1^{(2)})\sigma(z_1^{(2)})(1 - \sigma(z_1^{(2)}))x_0$$

I don't like math ㅠㅠ

전제

신경망에는 적응 가능한 가중치와 편향이 있고, 이 가중치와 편향을 훈련 데이터에 적응하도록 조정하는 과정을 '학습'이라 합니다. 신경망 학습은 다음과 같이 4단계로 수행합니다.

1단계 – 미니배치

훈련 데이터 중 일부를 무작위로 가져옵니다. 이렇게 선별한 데이터를 미니배치라 하며, 그 미니배치의 손실 함수 값을 줄이는 것이 목표입니다.

2단계 – 기울기 산출

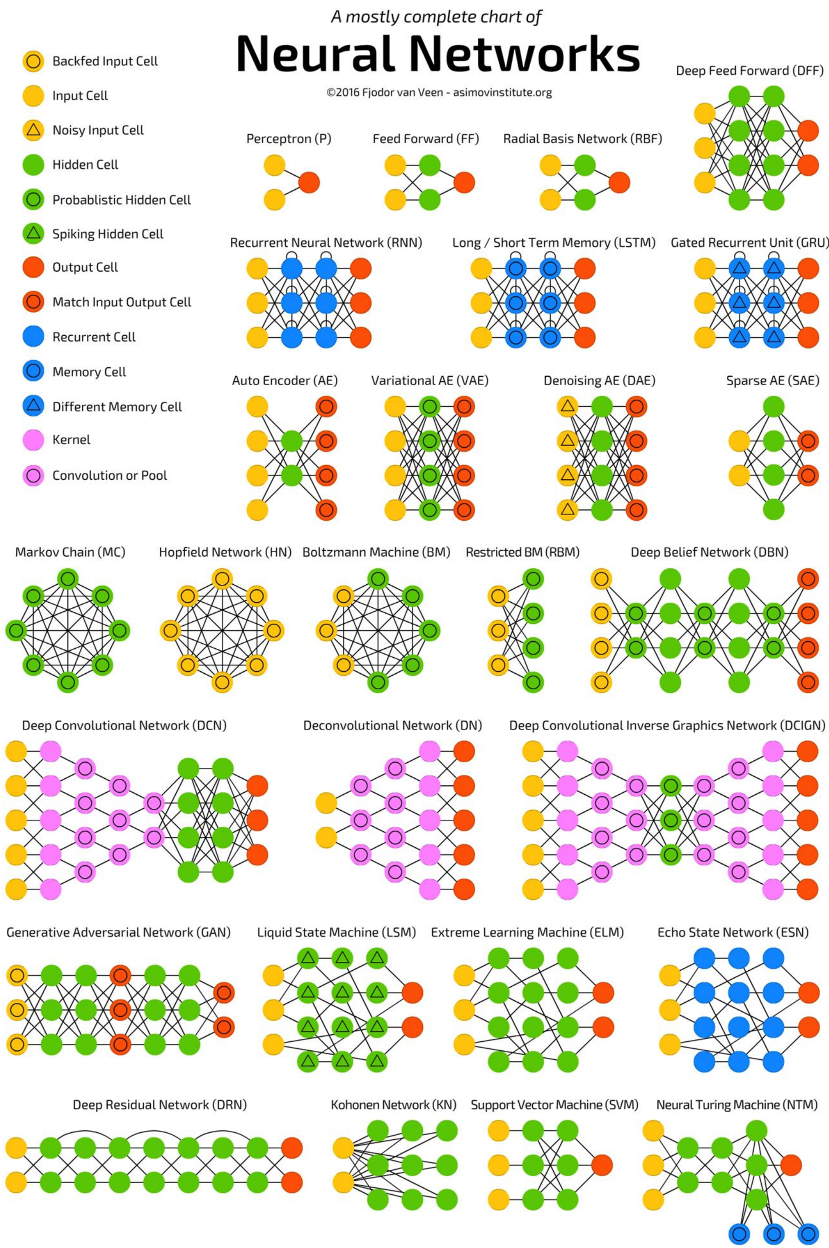
미니배치의 손실 함수 값을 줄이기 위해 각 가중치 매개변수의 기울기를 구합니다. 기울기는 손실 함수의 값을 가장 작게 하는 방향을 제시합니다.

3단계 – 매개변수 갱신

가중치 매개변수를 기울기 방향으로 아주 조금 갱신합니다.

4단계 – 반복

1~3단계를 반복합니다.



Difficulties		해결 방안
학습	DNN 학습이 잘 안 됨. Vanishing Gradient	Unsupervised Pre-Training를 통한 해결 ReLU(Rectified Linear Unit Function)
계산량	학습이 많은 계산이 필요함	H/W의 발전 및 GPU 활용
성능	다른 Machine Learning Algorithm의 높은 성능	Dropout 알고리즘 등으로 Machine Learning 대비 월등한 성능

학습

신경망에서 원하는 결과를 얻기 위해
뉴런 사이의 적당한 가중치를 알아내는 것

a. 훈련 데이터 (Training Set) 준비 : 입력 데이터와 출력 데이터

b. 신경망에 데이터 훈련 : 출력층 값을 확인

c. 지도학습 데이터와 차이 (오차) 계산

d. 오차가 최대한 작도록 가중치를 최적화
(Gradient Method(경사하강법)를 이용)
(Back Propagation(오류 역전파)