



# Neural Networks

강필성

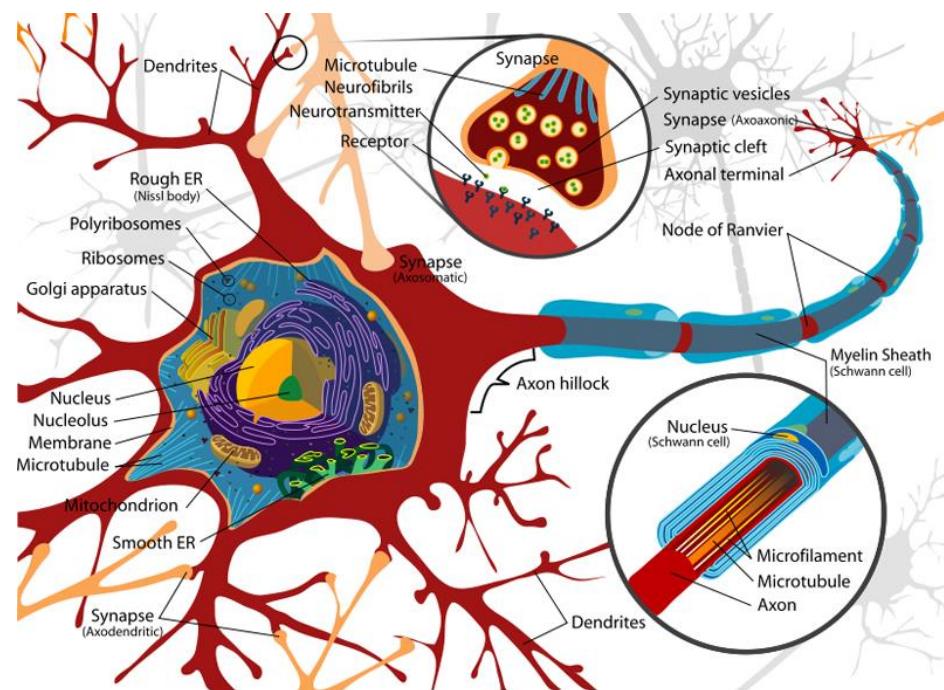
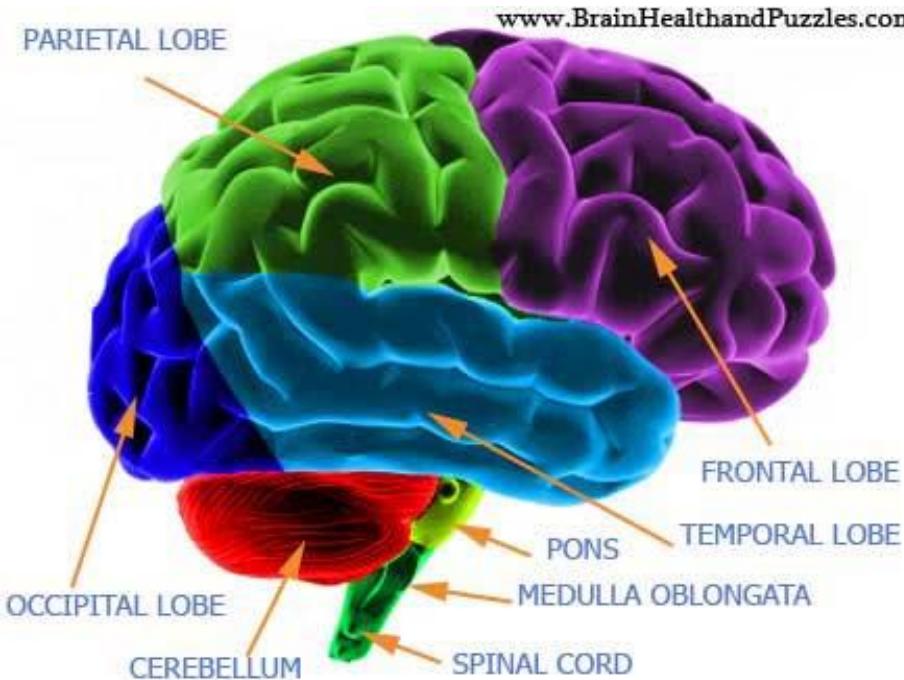
고려대학교 산업경영공학부

pilsung\_kang@korea.ac.kr

# Neural Networks

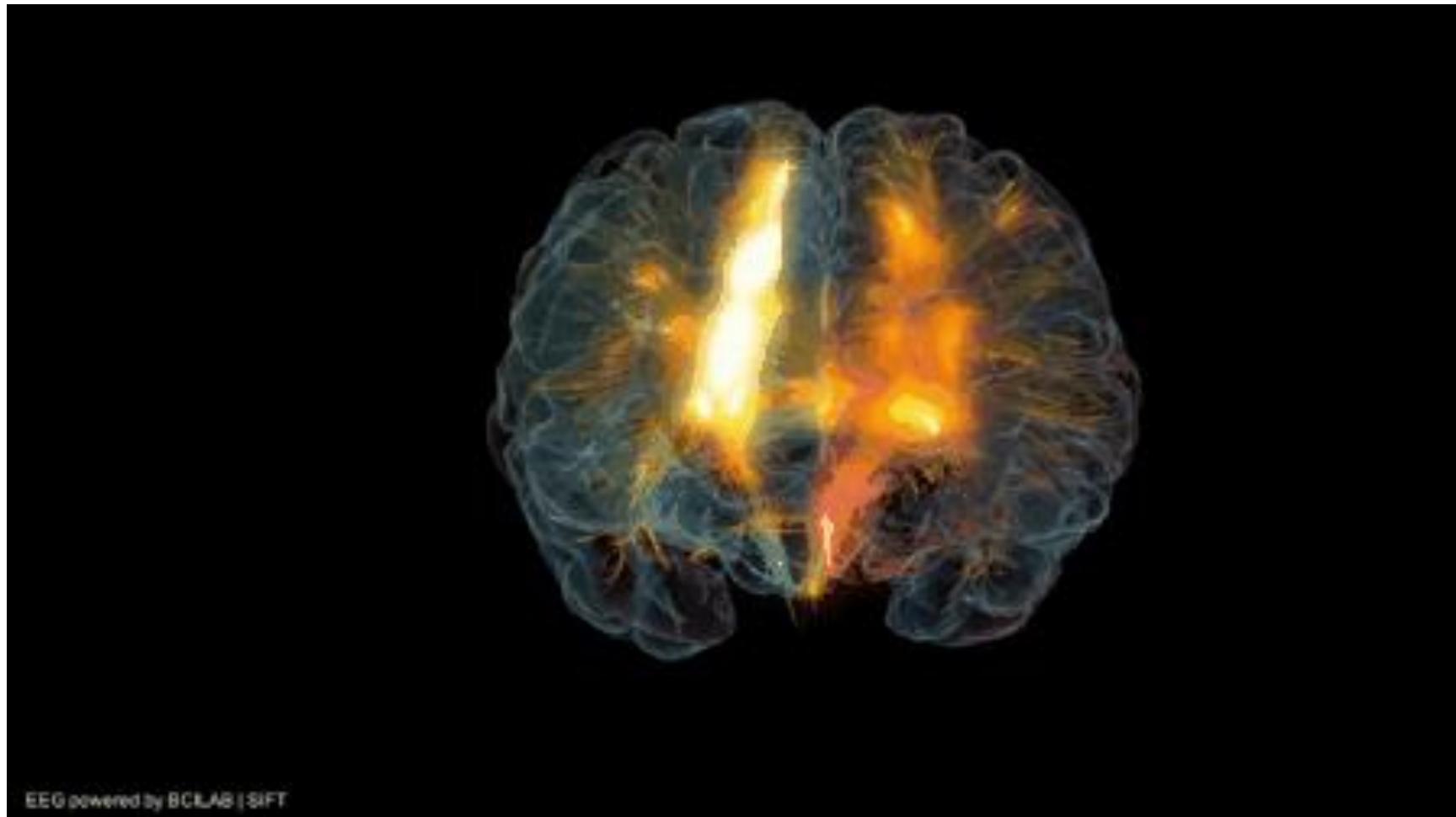
- 우리의 뇌가 작동하는 방식

- ✓ 외부의 자극이 감지되면 자극 감지 기관으로부터 뇌의 특정 부분까지의 신경계가 활성화되면서 해당 정보를 처리함
- ✓ 신경세포들은 전기적 신호를 이용하여 메시지를 주고받음



# Neural Networks

- Neuron Firing Off in Real-time

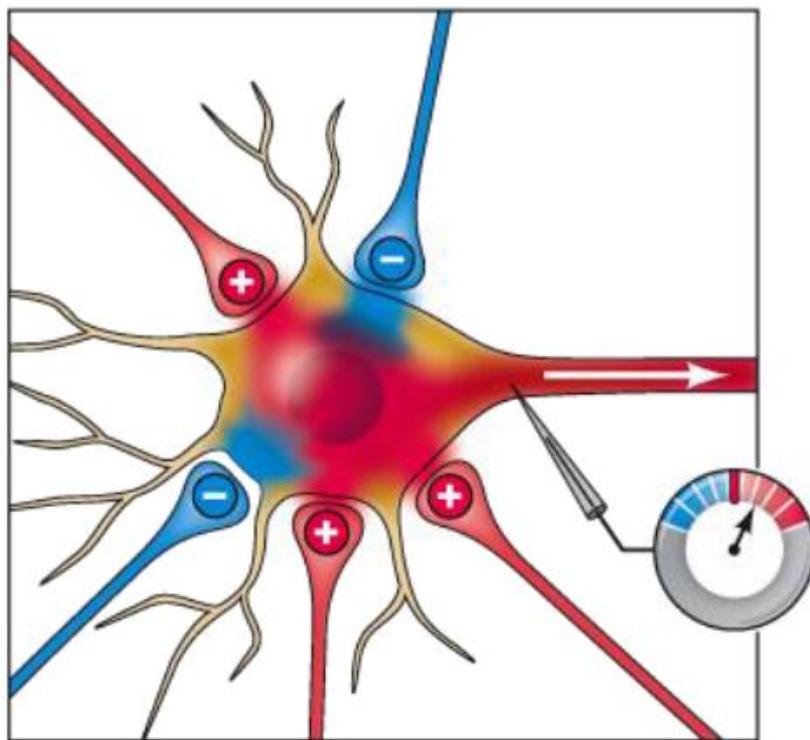
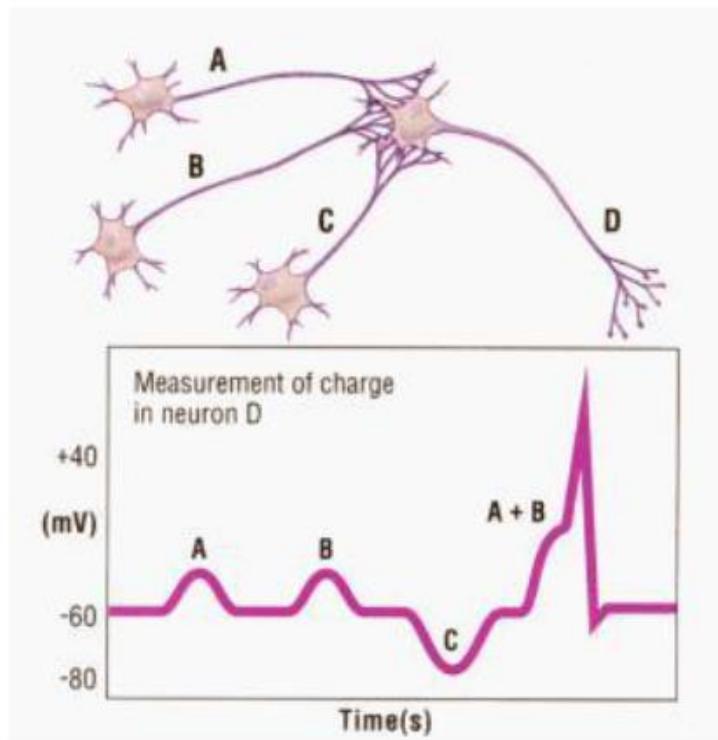


EEG powered by BCILAB | SIFT

<http://www.dailymail.co.uk/sciencetech/article-2581184/The-dynamic-mind-Stunning-3D-glass-brain-shows-neurons-firing-real-time.html>

# Neural Networks

- 뉴런의 작동 방식



뉴론은 계속해서 시그널을 받아

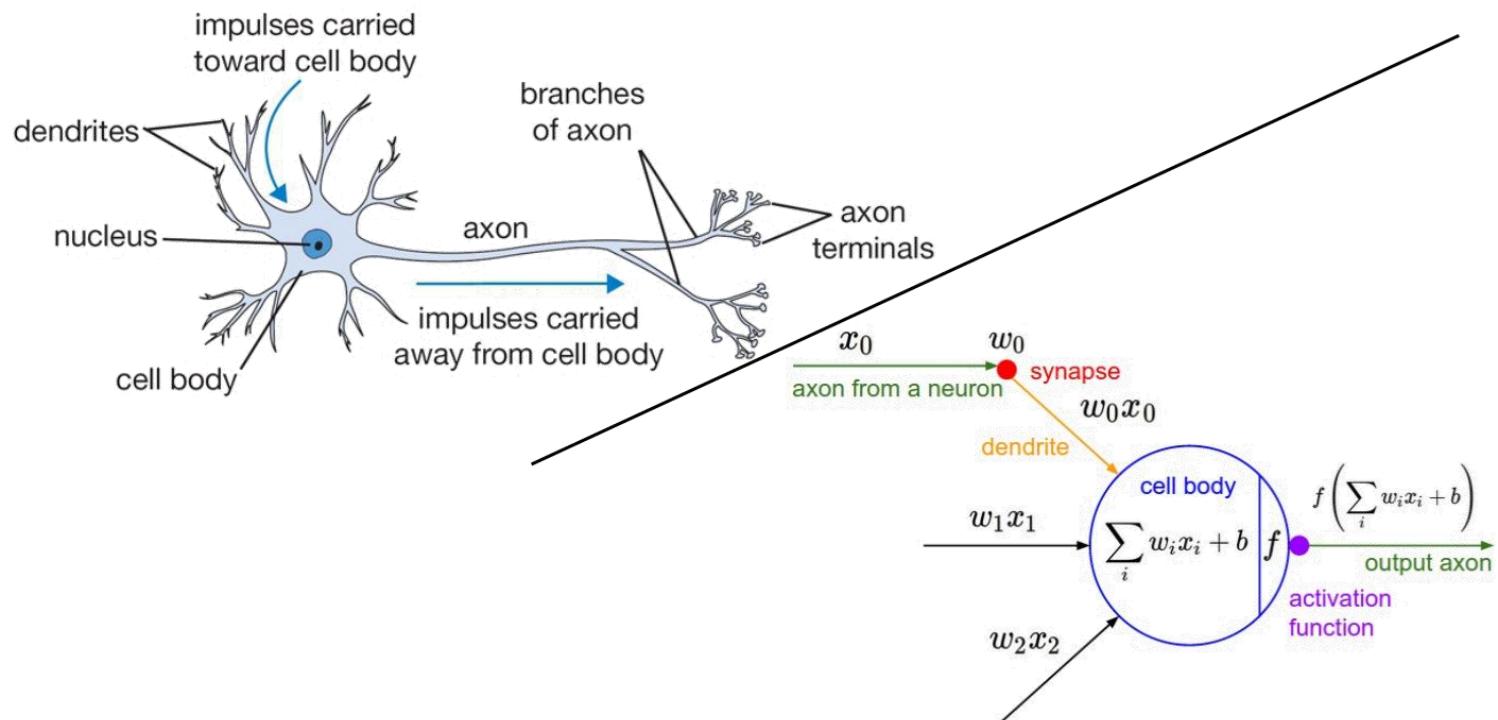
- 그 것을 조합- '*sum*' 하고,
- 특정 threshold 가 넘어서면 - '*fire*' 를 한다.

# Neural Networks: Perceptron

- 퍼셉트론

- ✓ 단일 뉴런의 작동 원리를 모사

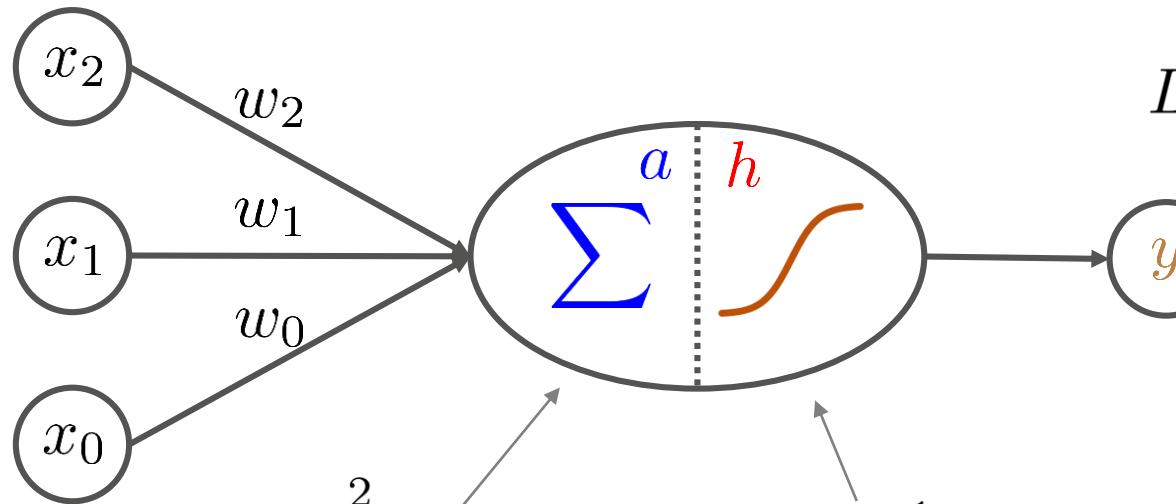
- 뉴런: 시냅스로부터 탐지된 자극을 수상돌기를 통해 세포핵에 전달 후 역치를 넘어서는 자극에 대해서는 축색돌기를 이용하여 다른 뉴런으로 정보를 전달
- 퍼셉트론: 입력변수의 값들에 대한 가중합에 대한 활성함수를 적용하여 최종 결과물 생성



# Neural Networks: Perceptron

- 퍼셉트론의 구조 및 역할

- ✓ 퍼셉트론은 뉴런이 하나인 단세포 생물!



원하는 값( $t$ )과 예측값( $y$ )의  
차이를 손실함수로 정의

$$L = \frac{1}{2}(t - y)^2$$

$$a = \sum_{i=0}^2 w_i x_i \quad h = \frac{1}{1 + \exp(-a)} \quad y = h$$

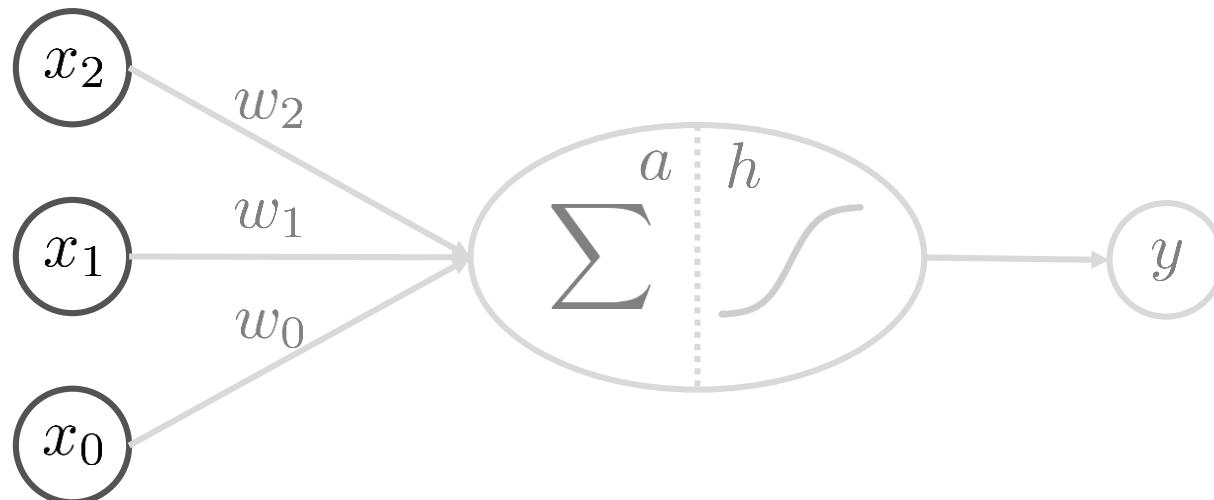
여러 변수들의 정보를  
나름대로 취합해서

얼마만큼 다음 단계로  
전달할지 결정한다  
(활성화)

# Neural Networks: Perceptron

- **입력 노드**<sup>input node</sup>

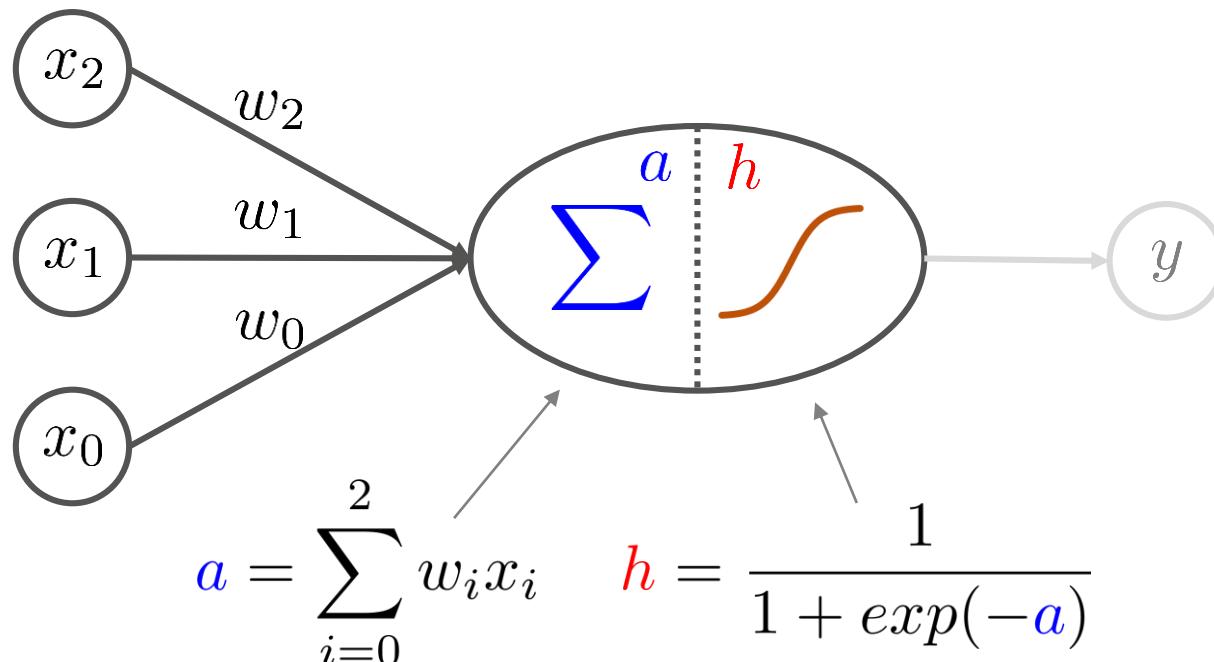
- ✓ 우리가 알고 있는 개별 설명 변수가 각각의 입력 노드에 해당



# Neural Networks: Perceptron

- 은닉 노드<sup>hidden node</sup>

✓ 개별 설명변수들의 값을 취합(선형 결합)하여 비선형 변환(활성화)을 수행



여러 변수들의 정보를  
나름대로 취합해서

얼마만큼 다음 단계로  
전달할지 결정한다  
(활성화)

# Neural Networks: Perceptron

- 활성함수의 역할

- ✓ 각 노드가 이전 노드들로부터 전달받은 정보를 다음 노드에 얼마만큼 전달해 줄 것인가를 결정

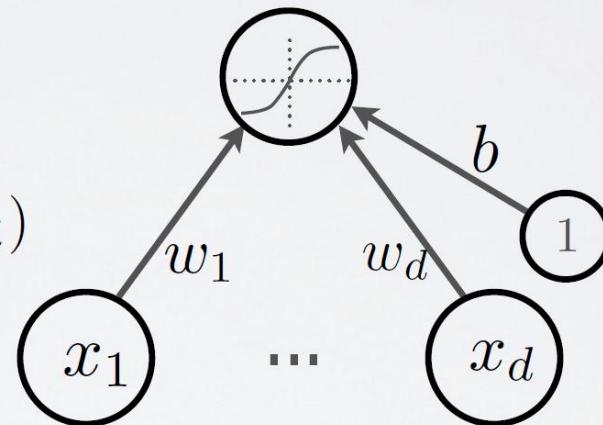
- Neuron pre-activation (or input activation):

$$a(\mathbf{x}) = b + \sum_i w_i x_i = b + \mathbf{w}^\top \mathbf{x}$$

- Neuron (output) activation

$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$

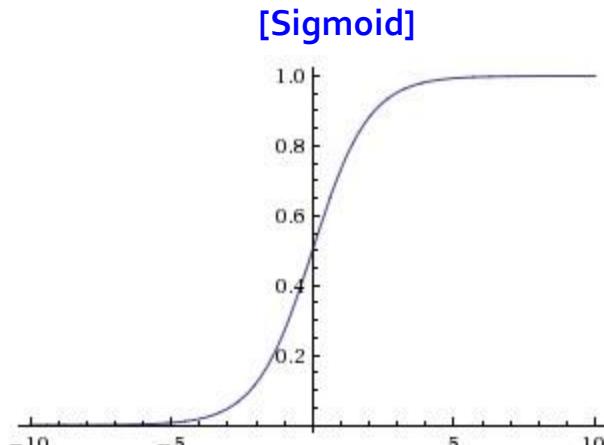
- $\mathbf{w}$  are the connection weights
- $b$  is the neuron bias
- $g(\cdot)$  is called the activation function



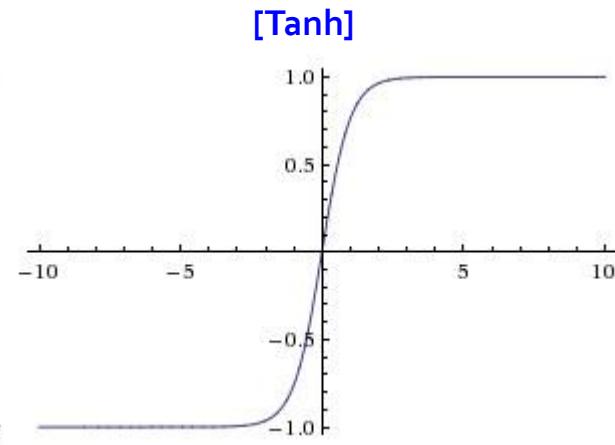
# Neural Networks: Perceptron

- 대표적 활성함수

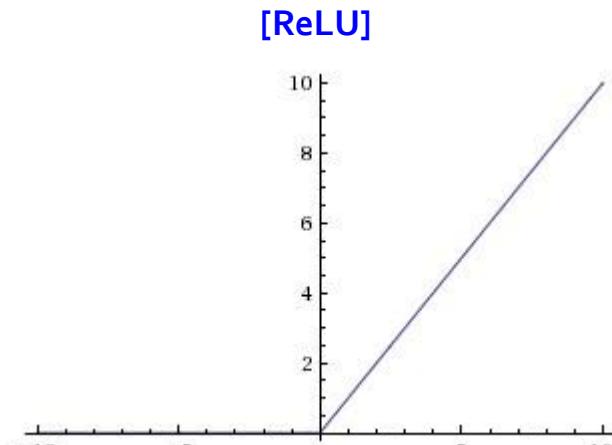
- ✓ Sigmoid: 가장 일반적으로 사용되는 활성함수,  $[0, 1]$ 의 범위를 가지며 학습 속도가 상대적으로 느림
- ✓ Tanh: Sigmoid 활성함수와 형태는 유사하나  $[-1, 1]$ 의 범위를 가져 학습 속도가 상대적으로 빠름
- ✓ ReLU(Rectified Linear Unit): 학습속도가 매우 빠르며 상대적으로 계산이 쉬움 (지수함수 형태를 사용하지 않으므로)



$$g(a) = \text{sigm}(a) = \frac{1}{1+\exp(-a)}$$



$$g(a) = \tanh(a) = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)} = \frac{\exp(2a)-1}{\exp(2a)+1}$$



$$g(a) = \text{reclin}(a) = \max(0, a)$$

# Neural Networks: Perceptron

- 퍼셉트론의 목적
  - ✓ 주어진 학습 데이터의 입력 정보( $x$ )와 출력 정보( $t$ )의 관계를 잘 찾도록 가중치  $w$ 를 조절하자!
- 관계가 잘 찾아졌는지는 어떻게 아는가?
  - ✓ 현재 퍼셉트론의 결과물  $y$ 가 실제 정답  $t$ 에 얼마나 가까운지를 손실 함수<sup>loss function</sup>를 통해 측정 (손실 함수는 객체별로 산출)
    - 회귀: 주로 squared loss 사용: 
$$L = \frac{1}{2}(t - y)^2$$
    - 분류: 주로 cross-entropy loss 사용 (퍼셉트론은 이범주 분류만 가능):
$$L = \sum_{i=1}^n \left( y_i \log p_i + (1 - y_i) \log(1 - p_i) \right)$$
  - ✓ 전체 데이터 셋에 대해서 현재 퍼셉트론이 얼마나 잘못하고 있는지는 비용 함수<sup>cost function</sup>을 사용하며, 이는 모든 객체의 손실함수에 대한 평균값을 주로 사용함

# 학습: 경사 하강법 (Gradient Descent)

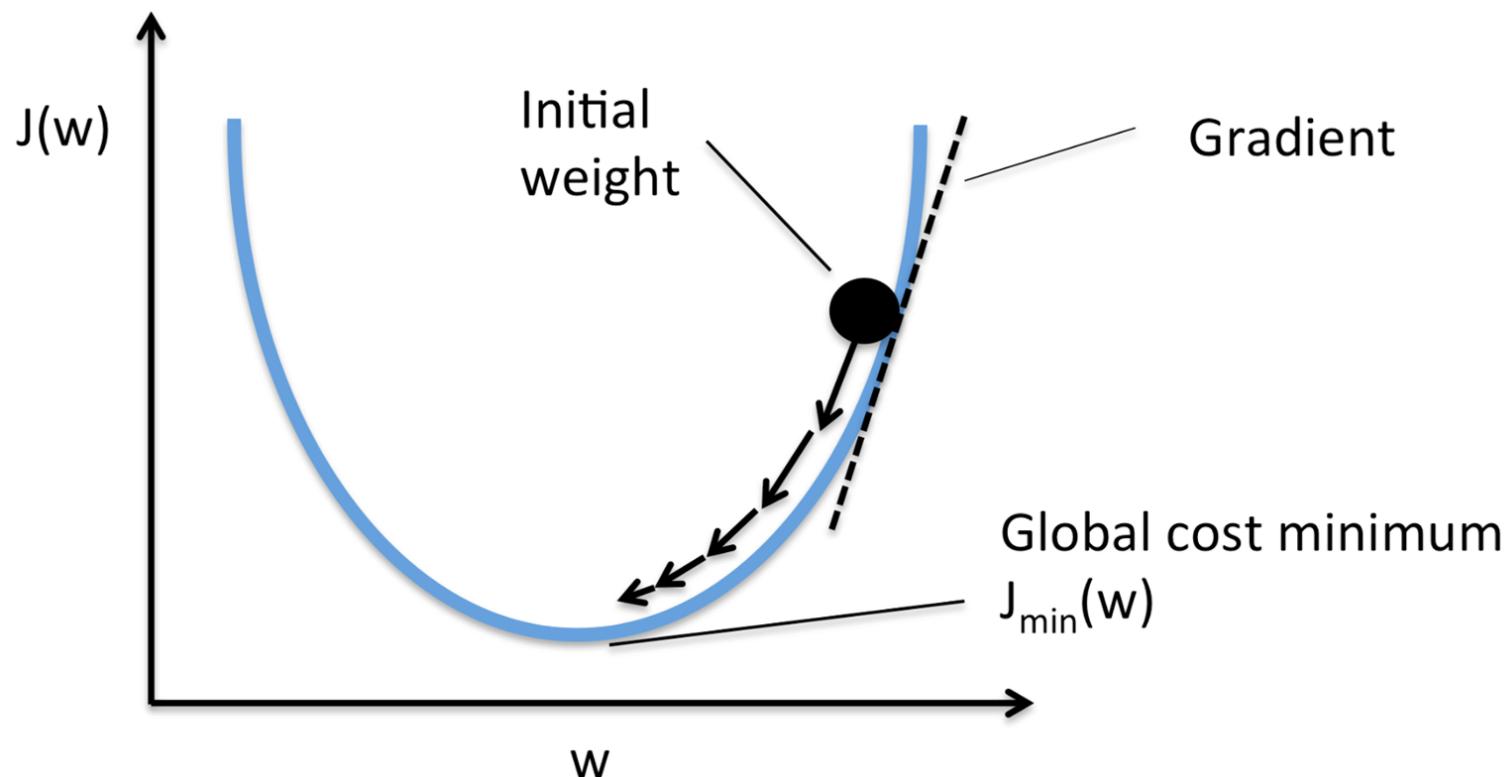
- 학습: 현재 퍼셉트론이 얼마나 틀렸는지를 알았으니 좀 더 잘 맞출 수 있도록 가중치  $w$ 를 조절하는 것
  - ✓ 기울기 하강: 눈을 가린 채로 산에서 가장 낮은 곳을 찾아가기



# 학습: 경사 하강법 (Gradient Descent)

- 경사 하강법

- ✓ 파란색 선: 가중치  $w$ 의 변화에 따른 비용함수 값의 변화
- ✓ 검은색 점: 현재 해의 위치
- ✓ 화살표: 비용함수를 최소화하기 위해 가중치  $w$ 가 이동해야 하는 방향



# 학습: 경사 하강법 (Gradient Descent)

- 비용함수를 현재의 가중치 값( $w$ )에 대해 1차 미분을 수행한 뒤 아래의 절차를 따름
  - ✓ 1차 미분 값 gradient가 0인가?
    - 그렇다: 현재의 가중치 값이 최적! → 학습 종료
    - 아니다: 현재의 가중치 값이 최적이 아님 → 좀 더 학습해봐
  - ✓ 1차 미분 값 gradient가 0이 아닐 경우 어떻게 해야 좀 더 잘하는 퍼셉트론을 만들 수 있는가?
    - 1차 미분 값(gradient)의 부호에 대한 반대 방향으로 가중치를 이동
  - ✓ 반대 방향으로 얼마나 움직여야 하는가?
    - 그건 잘 모름...
    - 조금씩 적당히(???) 움직여보고 그 다음에 다시 gradient를 구해보자
    - 하다 보면 되겠지...

# 학습: 경사 하강법 (Gradient Descent) (optional)

- 기울기 하강: Gradient descent algorithm

- ✓ 함수의 테일러 전개

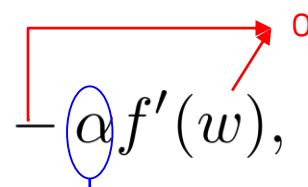
$$f(w + \Delta w) = f(w) + \frac{f'(w)}{1!} \Delta w + \frac{f''(w)}{2!} (\Delta w)^2 + \dots$$

- ✓ 목적함수가 최소화인 경우 함수의 1차 미분 값 gradient가 0이 아니면 gradient의 반대 방향으로 이동해야 목적함수의 값을 감소시킬 수 있음

$$w_{new} = w_{old} - \alpha f'(w), \quad \text{where } 0 < \alpha < 1.$$

어느 방향으로 갈 것인가?

얼마만큼 갈 것인가?

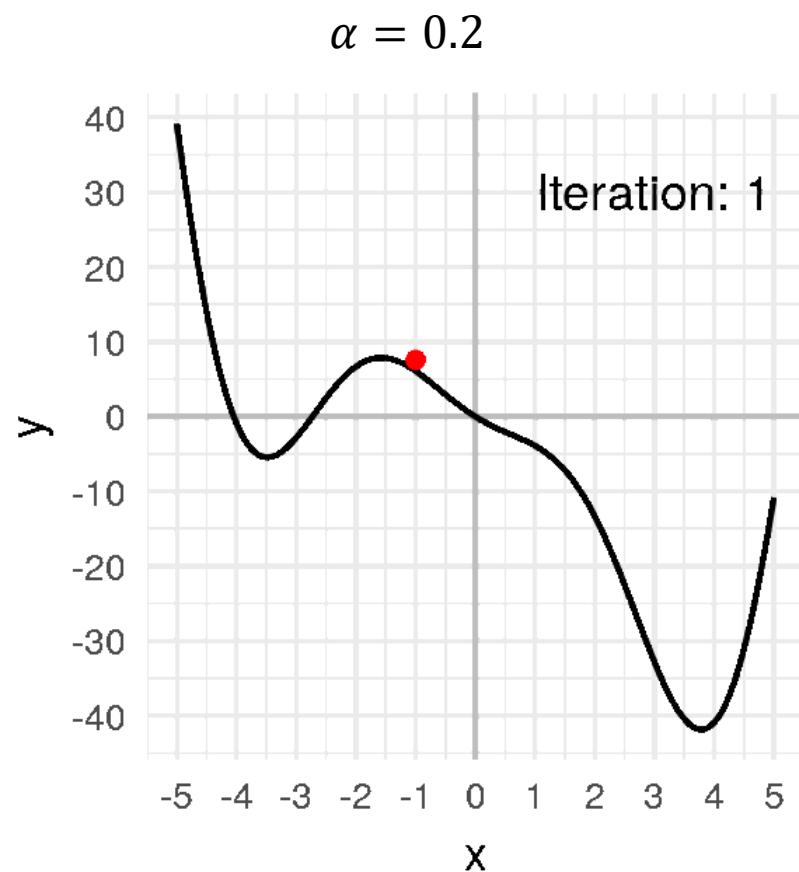
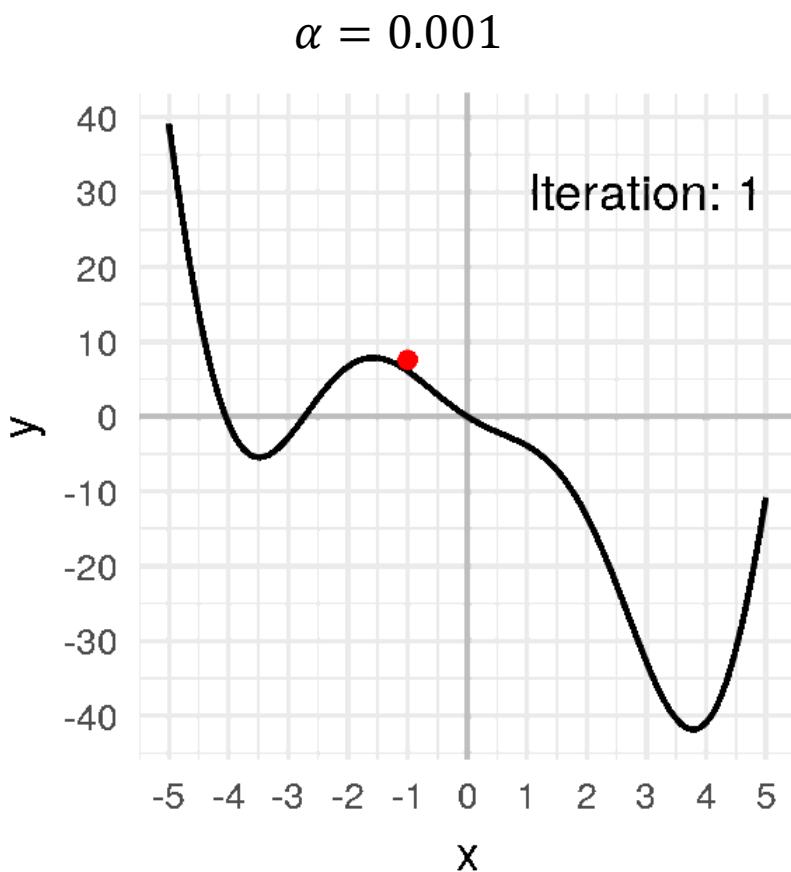


- ✓ 이동 후의 새로운 함수 값은 이동 전의 함수 값보다 작음

$$f(w_{new}) = f(w_{old} - \alpha f'(w_{old})) \cong f(w_{old}) - \alpha |f'(w)|^2 < f(w_{old})$$

# 학습: 경사 하강법 (Gradient Descent)

- $\alpha$ 의 크기에 따른 학습 과정



# 학습: 경사 하강법 (Gradient Descent)

- Use chain rule

$$\frac{\partial L}{\partial y} = y - t \quad \frac{\partial y}{\partial h} = 1$$

$$\frac{\partial h}{\partial a} = \frac{exp(-a)}{(1 + exp(-a))^2} = \frac{1}{1 + exp(-a)} \cdot \frac{exp(-a)}{1 + exp(-a)} = h(1 - h)$$

$$\frac{\partial a}{\partial w_i} = x_i$$

- Gradients for  $w$  and  $x$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h} \cdot \frac{\partial h}{\partial a} \cdot \frac{\partial a}{\partial w_i} = (y - t) \cdot 1 \cdot h(1 - h) \cdot x_i$$

- Update  $w$

$$w_i^{new} = w_i^{old} - \alpha \times \frac{L}{\partial w_i} = w_i^{old} - \alpha \times (y - t) \cdot 1 \cdot h(1 - h) \cdot x_i$$

# 학습: 경사 하강법 (Gradient Descent)

- Gradient Descent에서 weight가 업데이트 되는 원리

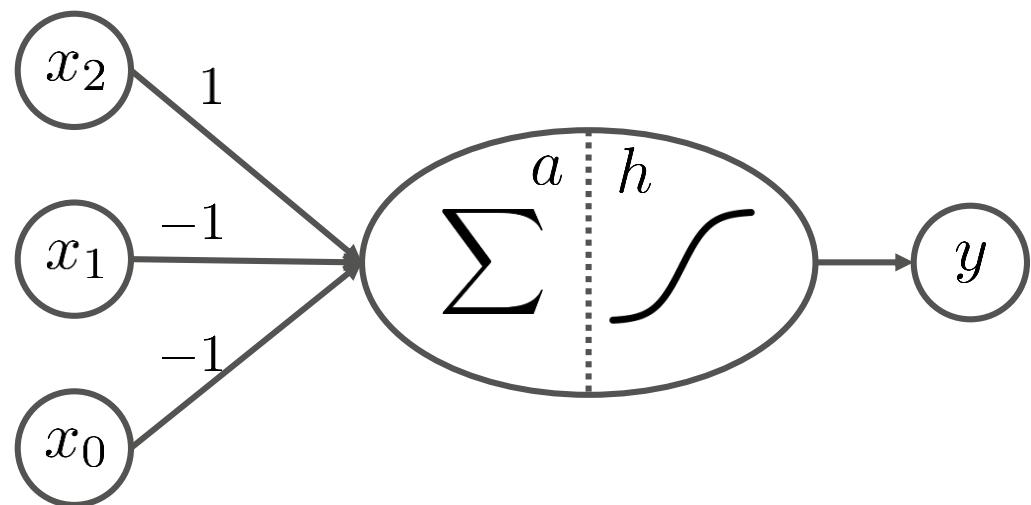
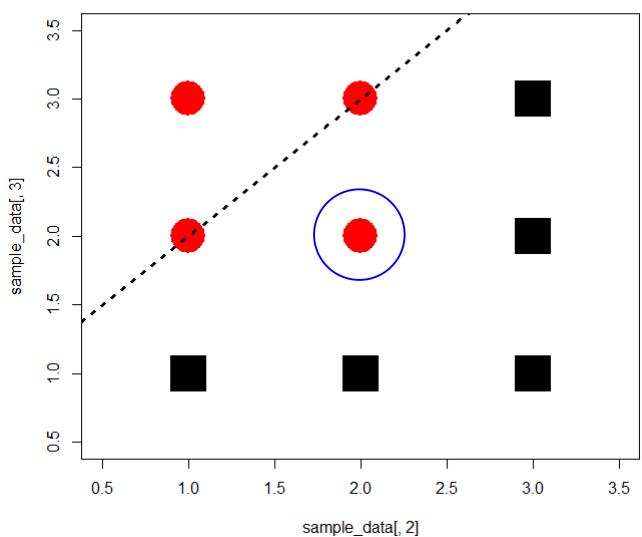
$$w_i^{new} = w_i^{old} - \alpha \times (y - t) \cdot 1 \cdot h(1 - h) \cdot x_i$$


현재의 출력값(y)과 정답(t)이  
차이가 많이 날 수록  
가중치를 많이 업데이트 하라

대상 가중치와 연결된 입력  
변수의 값이 클 수록  
가중치를 많이 업데이트 하라

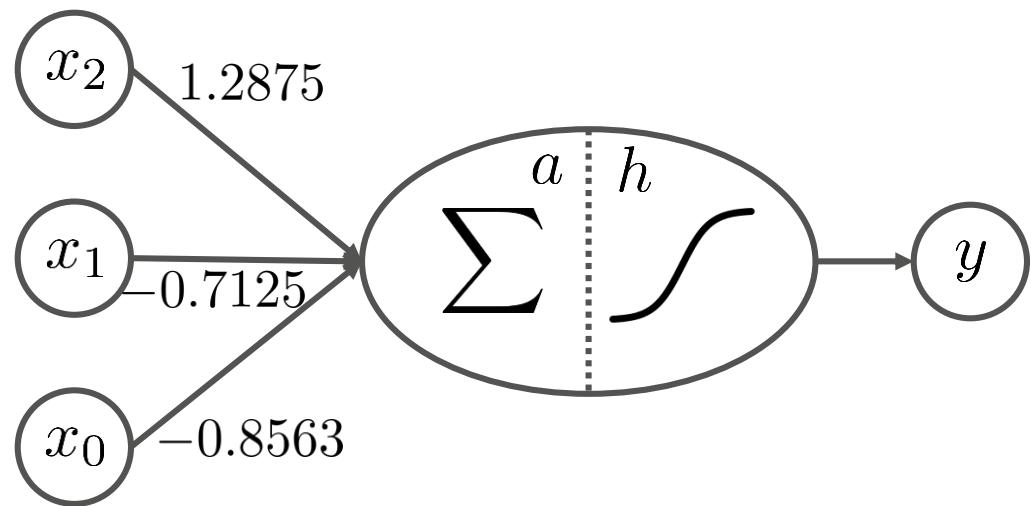
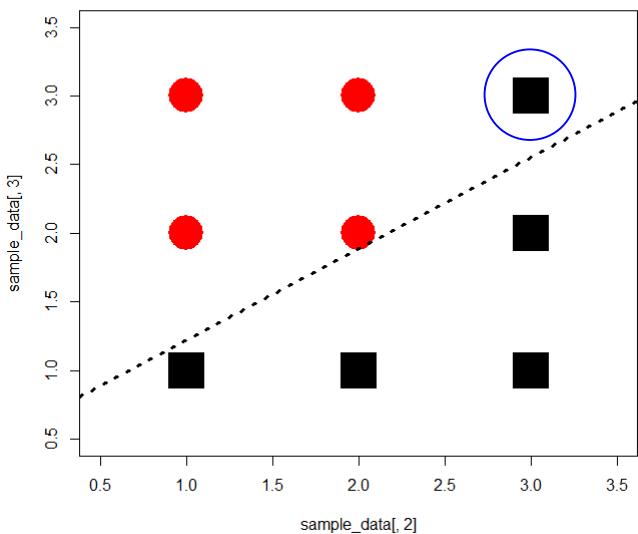
# 퍼셉트론의 학습: 예시 ( $\alpha = 1$ )

- 초기화 및 첫 번째 학습 example 선택 ( $x_1=2, x_2=2, t=1$ )



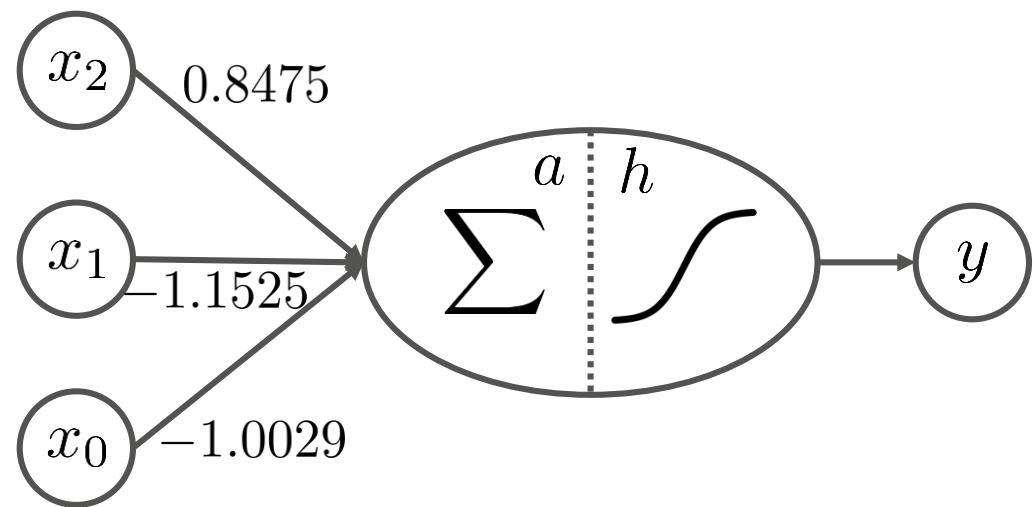
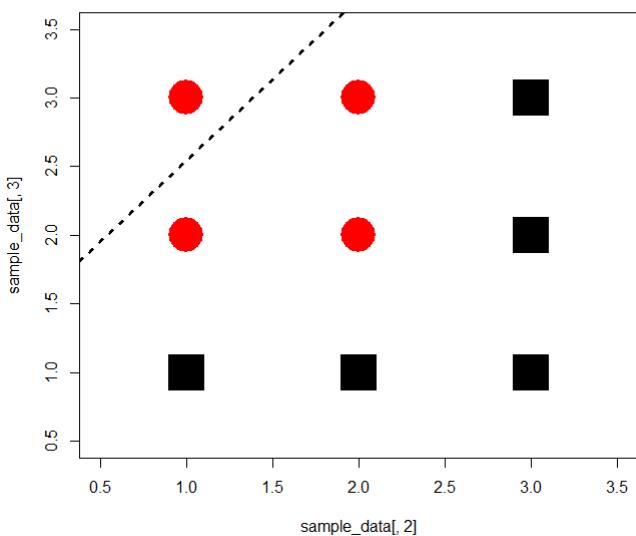
# 퍼셉트론의 학습: 예시 ( $\alpha = 1$ )

- 학습 결과 및 두 번째 학습 example 선택 ( $x_1=3, x_2=3, t=0$ )



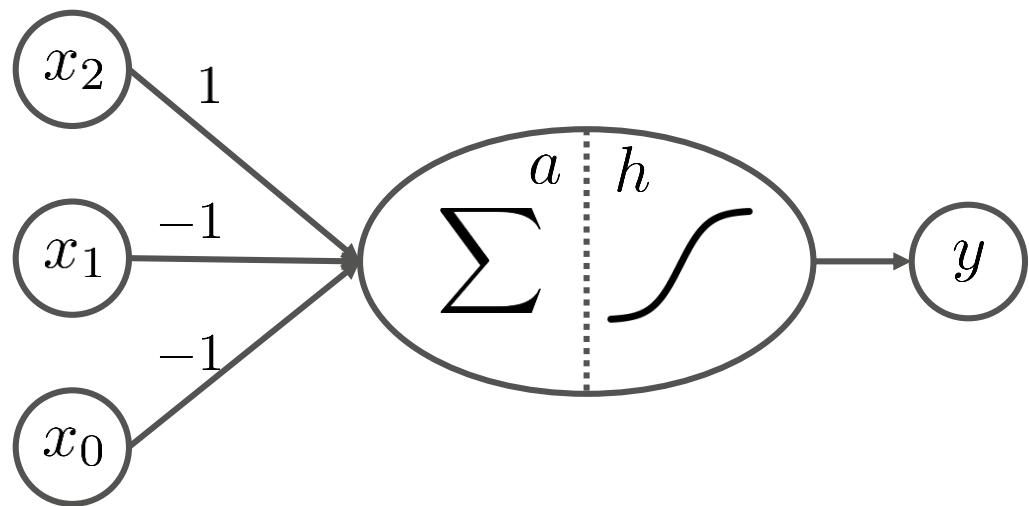
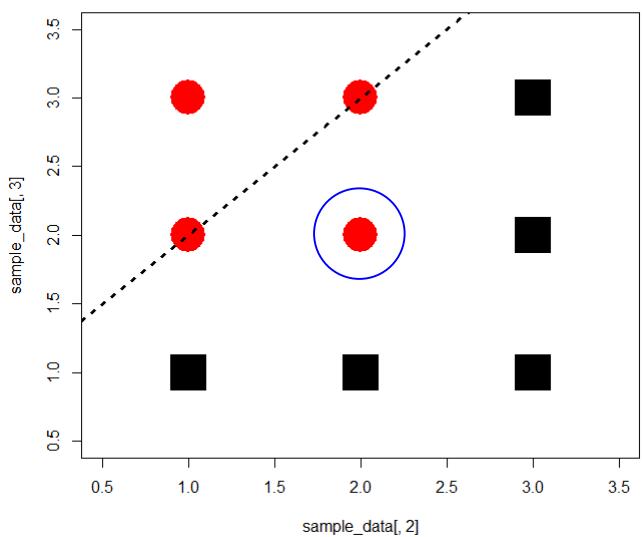
# 퍼셉트론의 학습: 예시 ( $\alpha = 1$ )

- 학습 결과



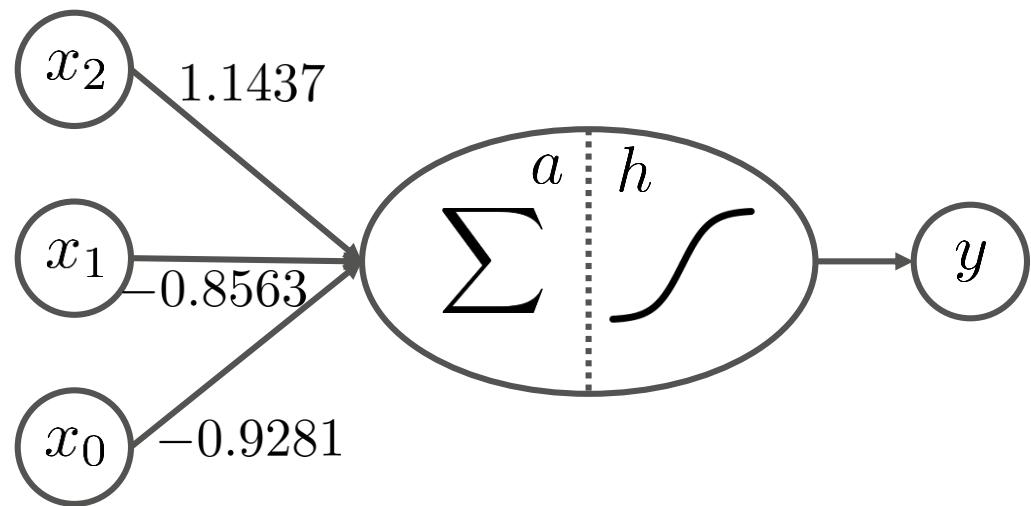
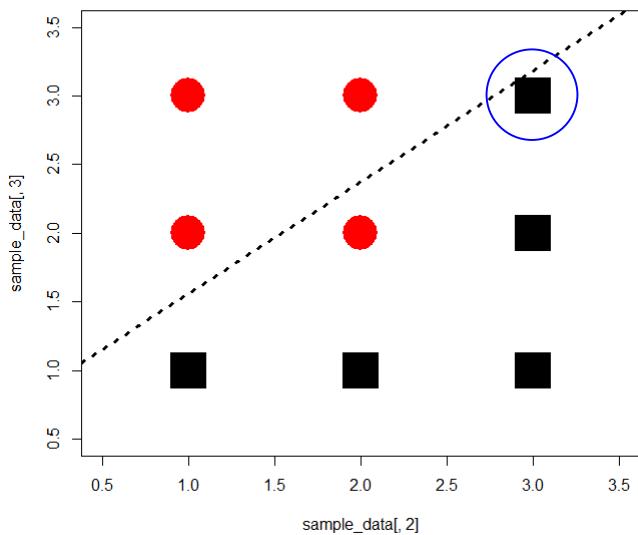
# 퍼셉트론의 학습: 예시 ( $\alpha = 0.5$ )

- 초기화 및 첫 번째 학습 example 선택 ( $x_1=2, x_2=2, t=1$ )



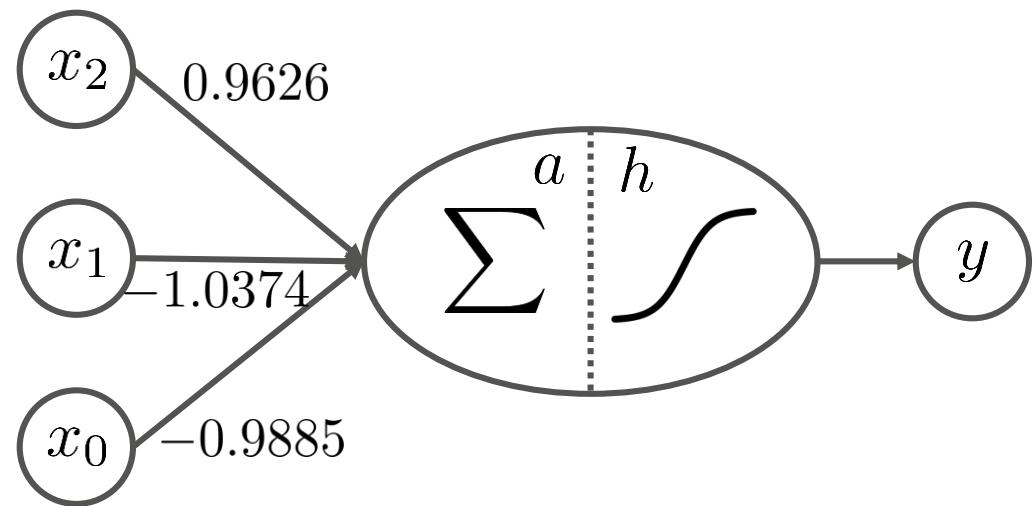
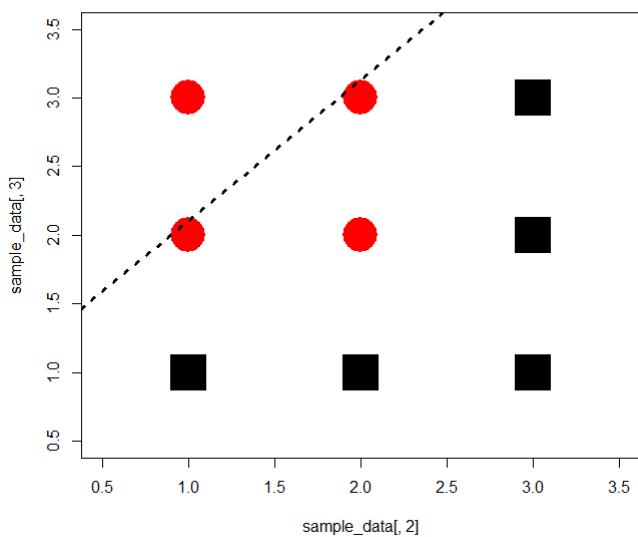
# 퍼셉트론의 학습: 예시 ( $\alpha = 0.5$ )

- 학습 결과 및 두 번째 학습 example 선택 ( $x_1=3, x_2=3, t=0$ )



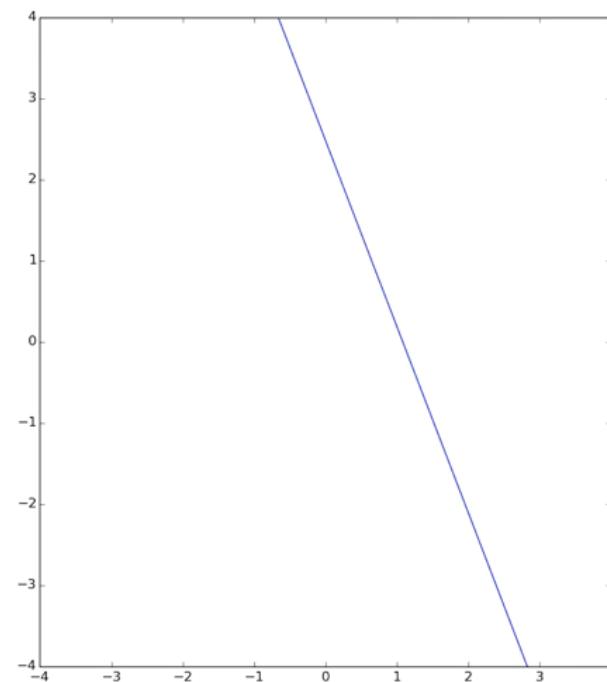
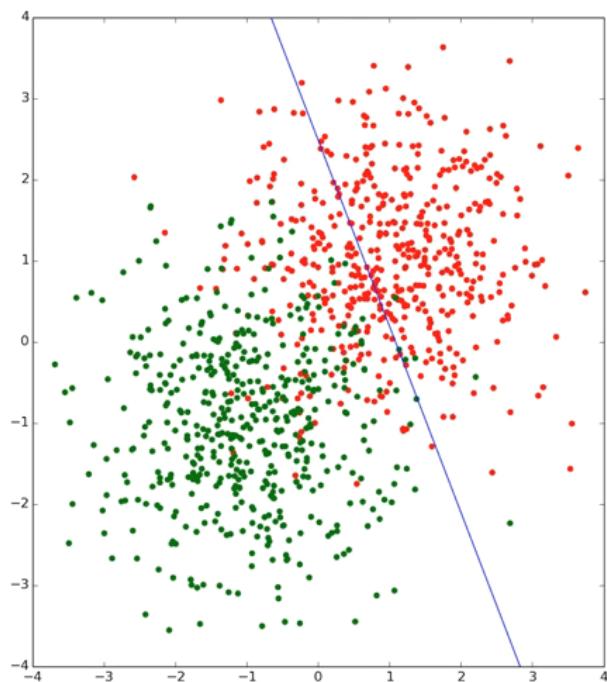
# 퍼셉트론의 학습: 예시 ( $\alpha = 0.5$ )

- 학습 결과



# 퍼셉트론 (Perceptron): 학습 예시

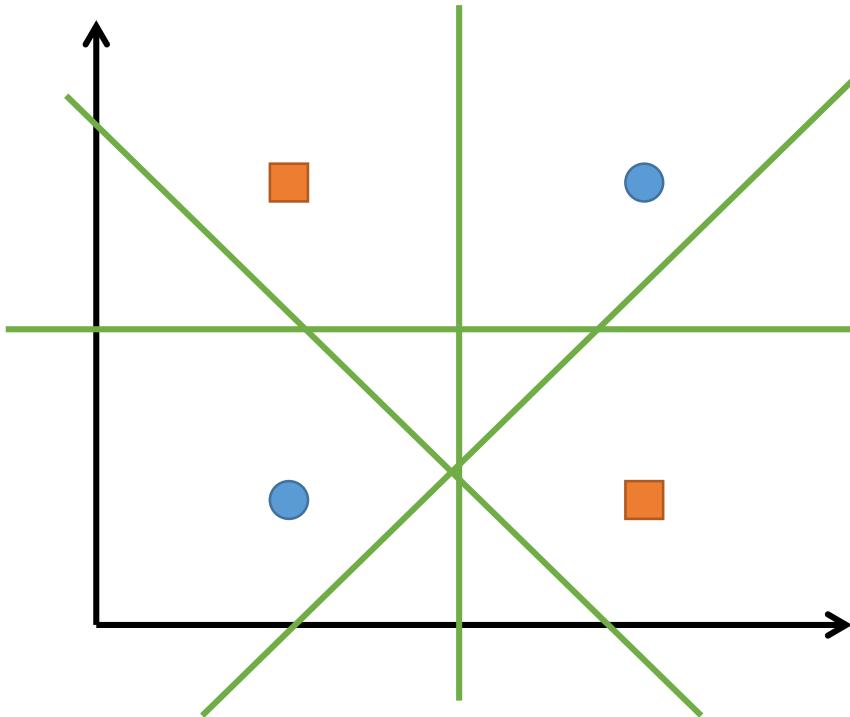
- 퍼셉트론의 학습



# Neural Networks: Multi-Layer Perceptron (MLP)

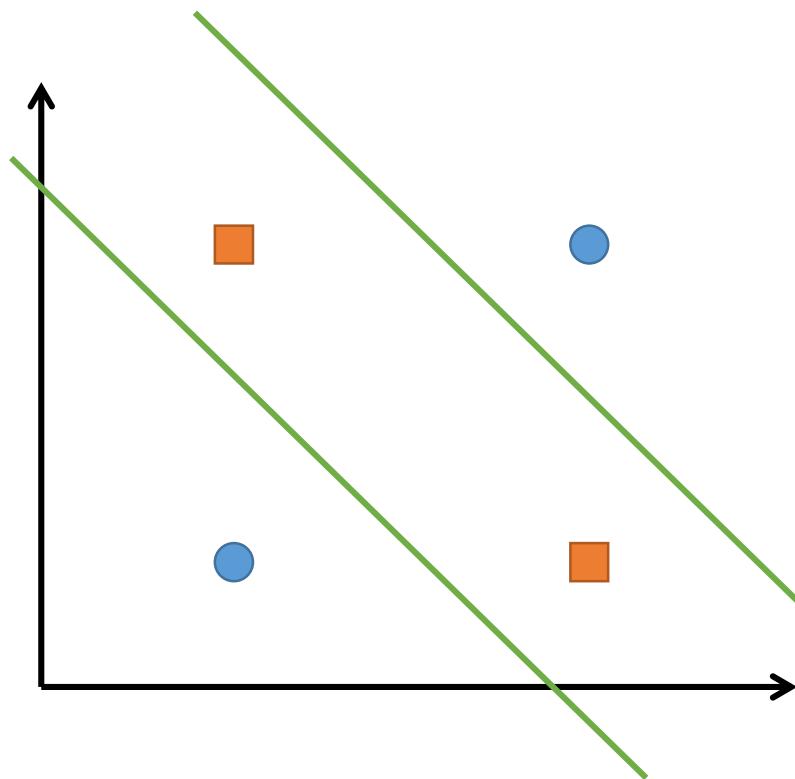
- 퍼셉트론(선형 모델)의 한계

✓ 단 하나의 직선을 그어 파란색 원과 브은색 사각형을 분류할 수 있는가?



# Neural Networks: Multi-Layer Perceptron (MLP)

- **다층 퍼셉트론: 여러 개의 퍼셉트론을 결합하여 해결!**
  - ✓ 문제를 한꺼번에 풀지 말고 풀 수 있는 형태의 문제 여러 개로 나누어서 풀자
    - 두 직선 사이에 있으면 네모로 판정하고 바깥쪽에 있으면 원으로 판별하자
    - 각각의 직선은 퍼셉트론으로 만들어 낼 수 있으니 두 개의 퍼셉트론으로 해결 가능

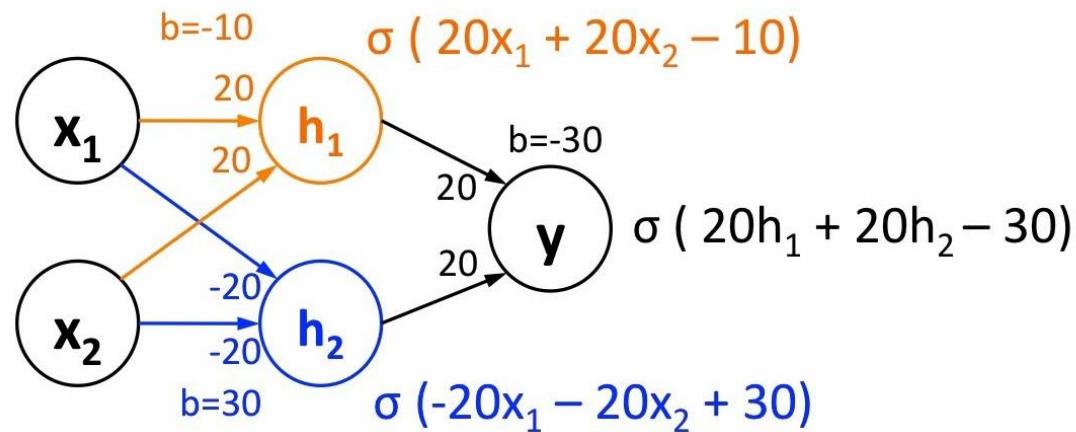
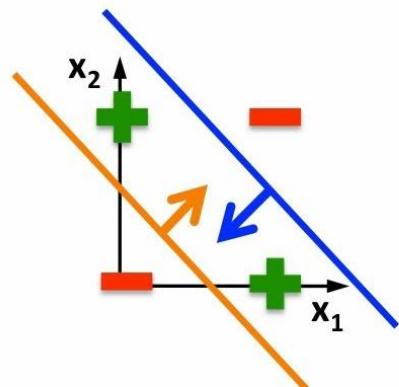


# Neural Networks: Multi-Layer Perceptron (MLP)

- 다층 퍼셉트론: 여러 개의 퍼셉트론을 결합하여 해결!

- ✓ 문제를 한꺼번에 풀지 말고 풀 수 있는 형태의 문제 여러 개로 나누어서 풀자
  - 두 직선 사이에 있으면 네모로 판정하고 바깥쪽에 있으면 원으로 판별하자

Linear classifiers  
cannot solve this



$$\sigma(20*0 + 20*0 - 10) \approx 0$$

$$\sigma(20*1 + 20*1 - 10) \approx 1$$

$$\sigma(20*0 + 20*1 - 10) \approx 1$$

$$\sigma(20*1 + 20*0 - 10) \approx 1$$

$$\sigma(-20*0 - 20*0 + 30) \approx 1$$

$$\sigma(-20*1 - 20*1 + 30) \approx 0$$

$$\sigma(-20*0 - 20*1 + 30) \approx 1$$

$$\sigma(-20*1 - 20*0 + 30) \approx 1$$

$$\sigma(20*0 + 20*1 - 30) \approx 0$$

$$\sigma(20*1 + 20*0 - 30) \approx 0$$

$$\sigma(20*1 + 20*1 - 30) \approx 1$$

$$\sigma(20*1 + 20*1 - 30) \approx 1$$

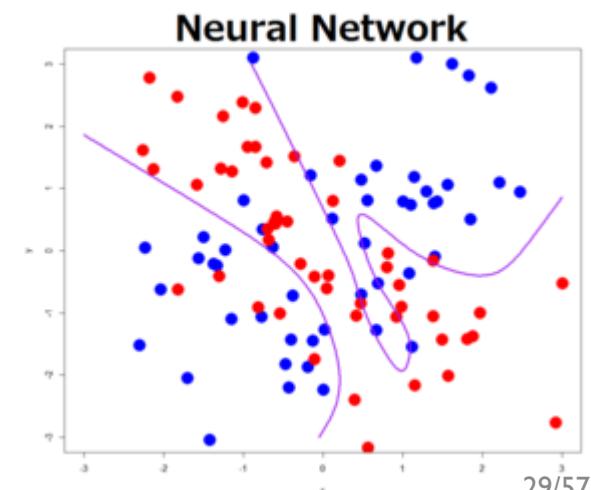
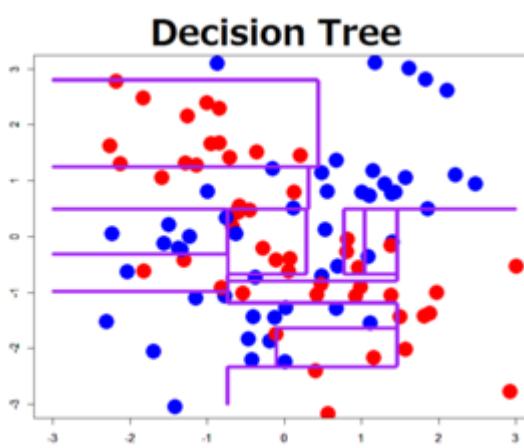
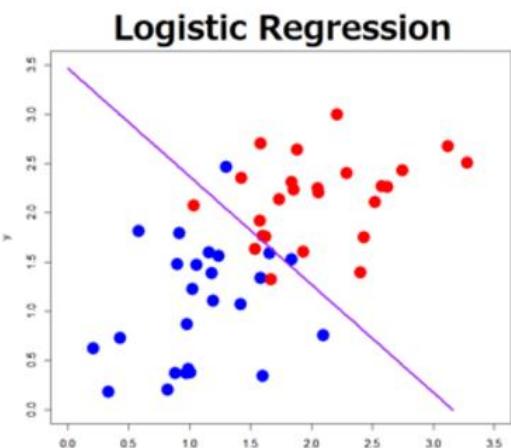
# 인공 신경망: Multi-Layer Perceptron (MLP)

- 다층 퍼셉트론의 예측력이 우수한 이유

- ✓ 분류 경계면을 여러 직선들의 집합으로 가정하면 로지스틱 회귀분석, 의사결정나무, 인공신경망은 다음과 같은 특징을 가짐

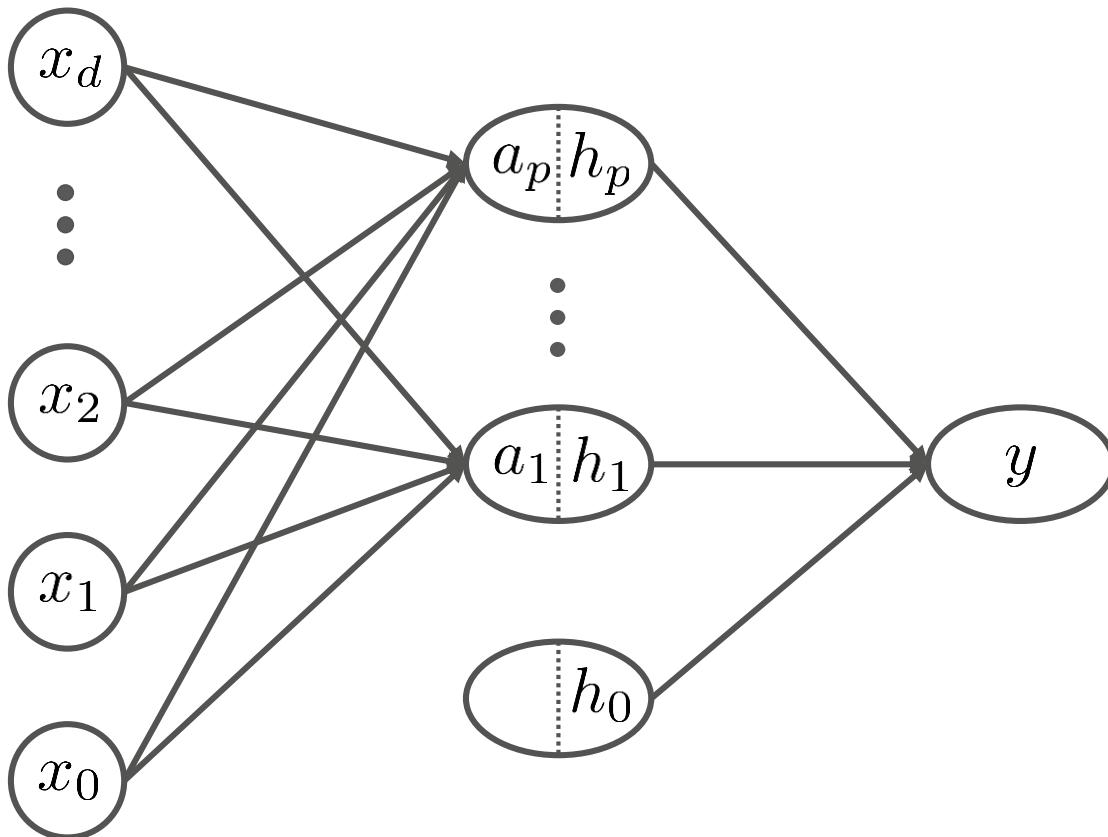
	로지스틱 회귀분석	의사결정나무	인공신경망
선의 수	1개	제한 없음	사용자 지정 (은닉 층 및 노드의 수)
선의 방향	제약 없음	축에 수직	제약 없음

- ✓ 분류경계면 생성에 있어 가장 인공신경망의 자유도가 가장 높음



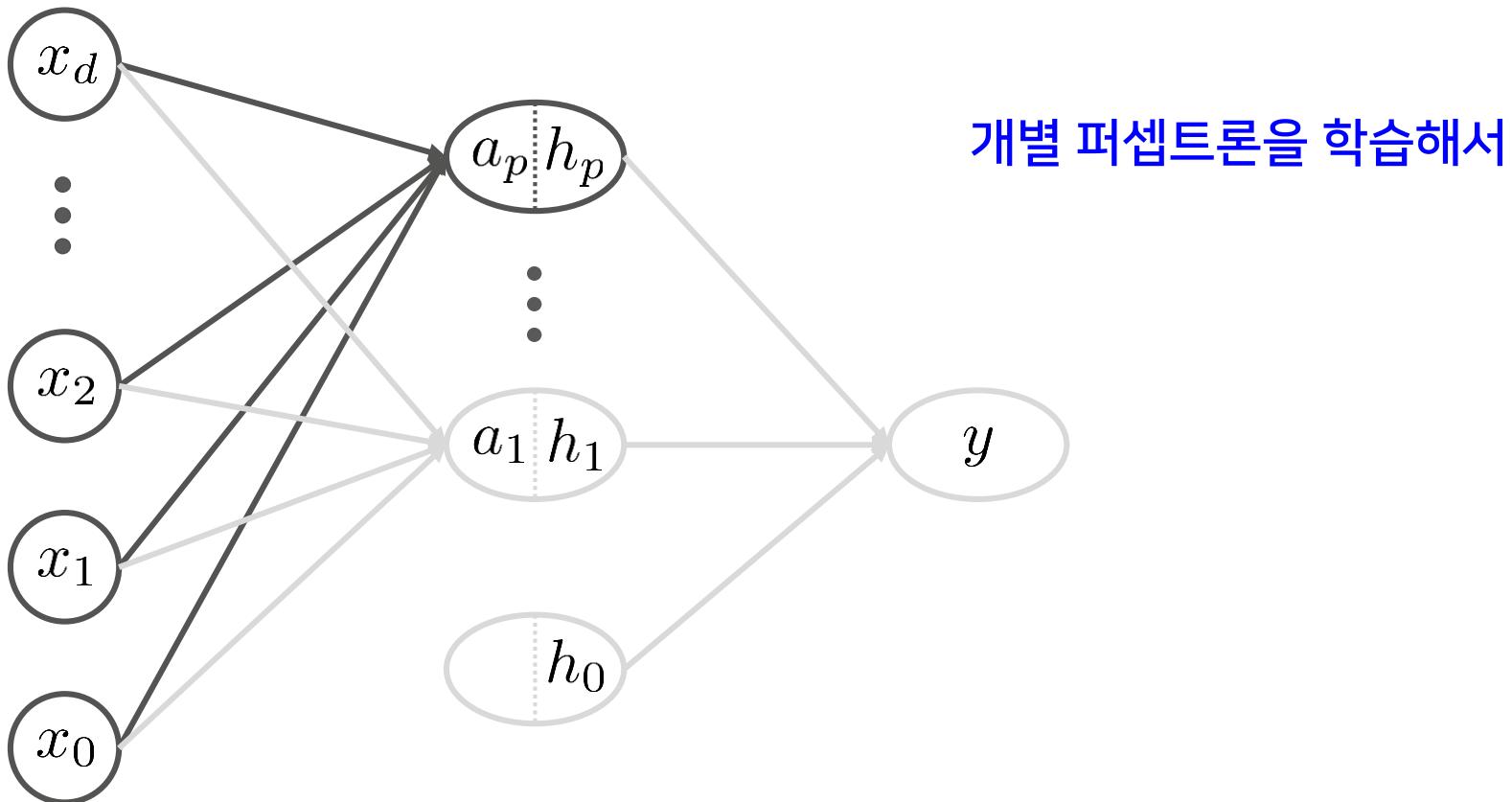
# Neural Networks: Multi-Layer Perceptron (MLP)

- 기초 구조: 1개의 은닉층을 가진 Feed-forward Neural Network (회귀)
  - ✓ 개별 은닉노드가 하는 역할은 퍼셉트론과 동일하며 출력노드의 결과값은 은닉노드의 결과값을 선형결합하여 생성



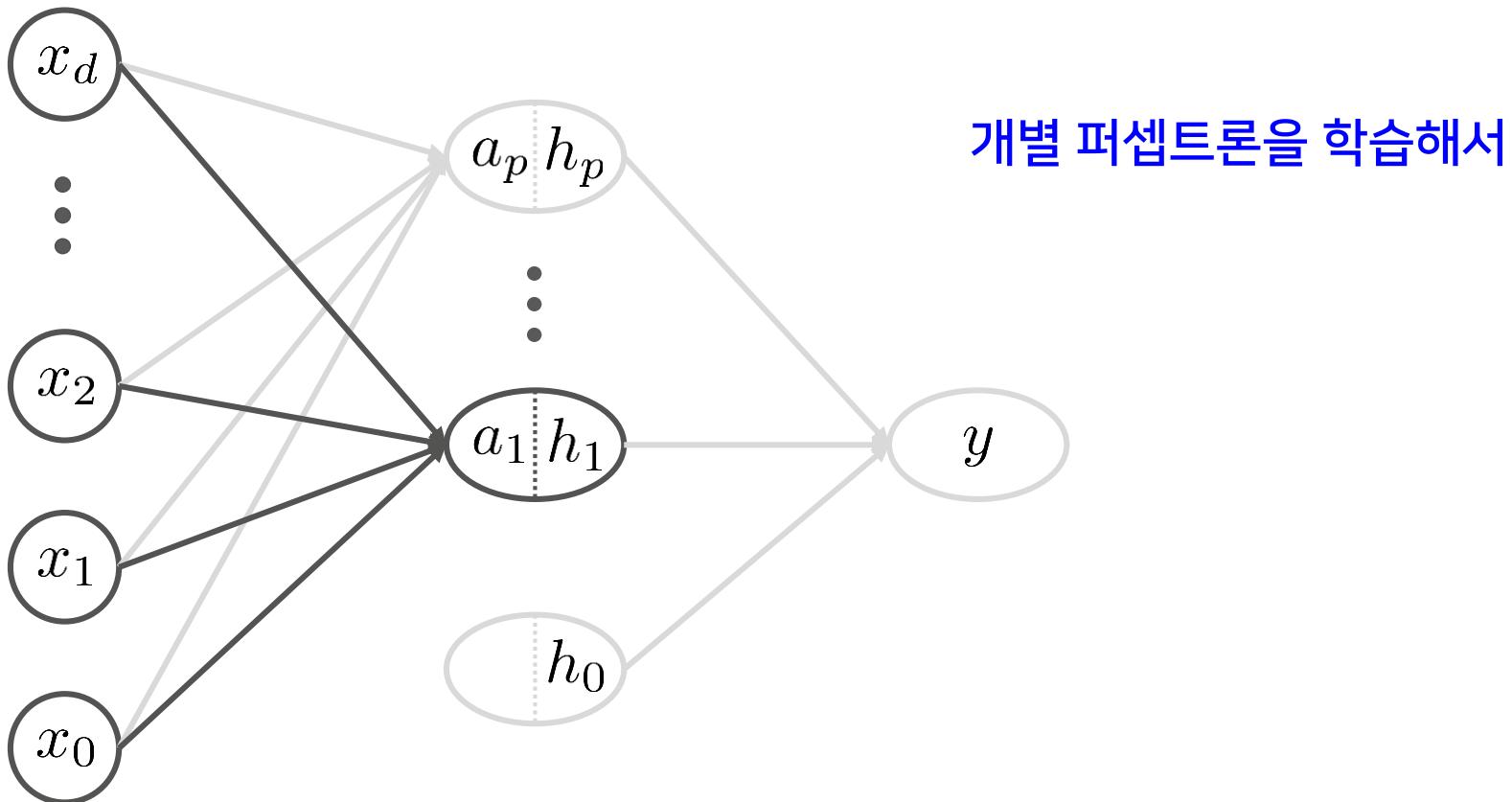
# Neural Networks: Multi-Layer Perceptron (MLP)

- 기초 구조: 1개의 은닉층을 가진 Feed-forward Neural Network (회귀)
  - ✓ 개별 은닉노드가 하는 역할은 퍼셉트론과 동일하며 출력노드의 결과값은 은닉노드의 결과값을 선형결합하여 생성



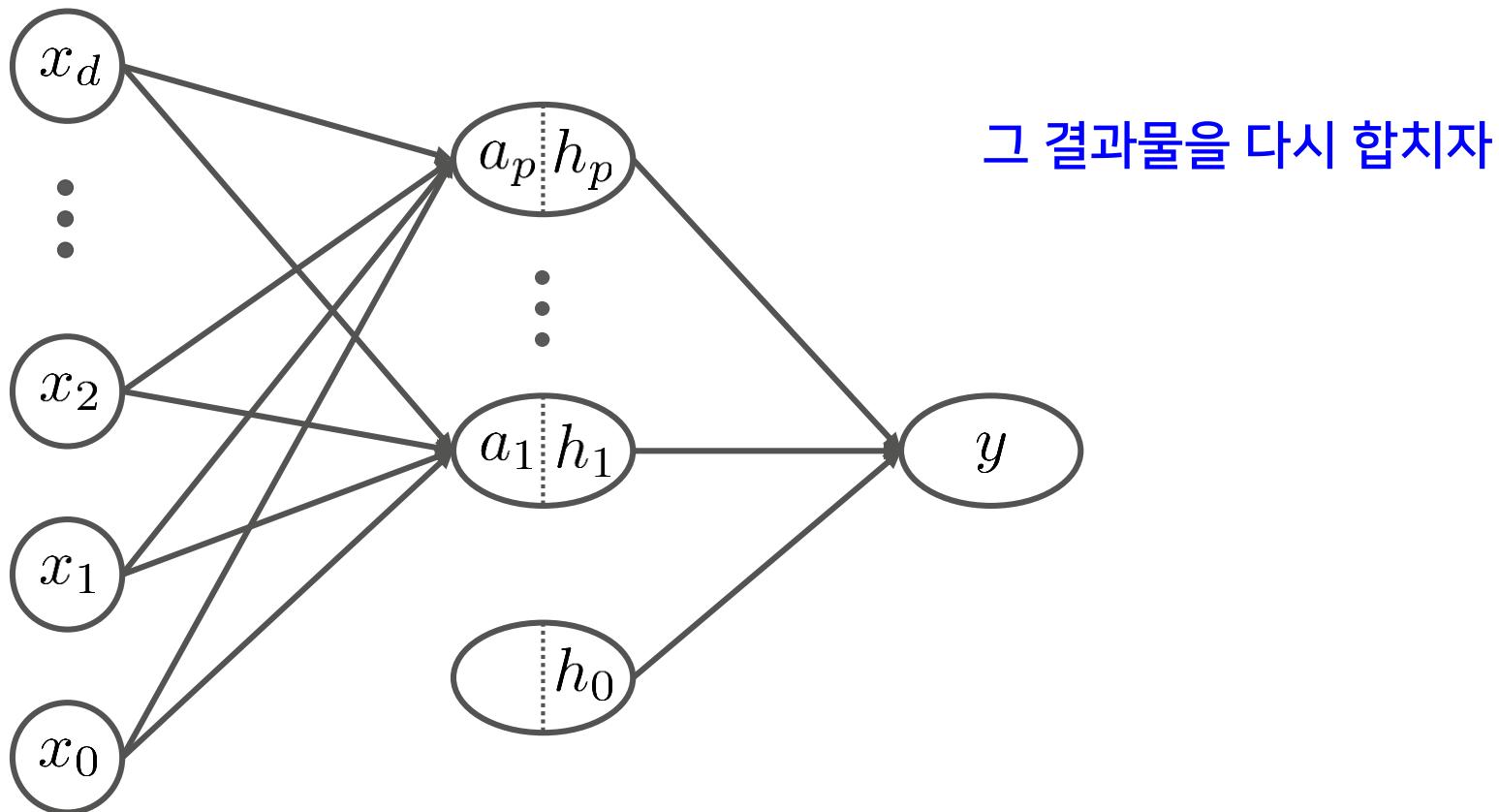
# Neural Networks: Multi-Layer Perceptron (MLP)

- 기초 구조: 1개의 은닉층을 가진 Feed-forward Neural Network (회귀)
  - ✓ 개별 은닉노드가 하는 역할은 퍼셉트론과 동일하며 출력노드의 결과값은 은닉노드의 결과값을 선형결합하여 생성



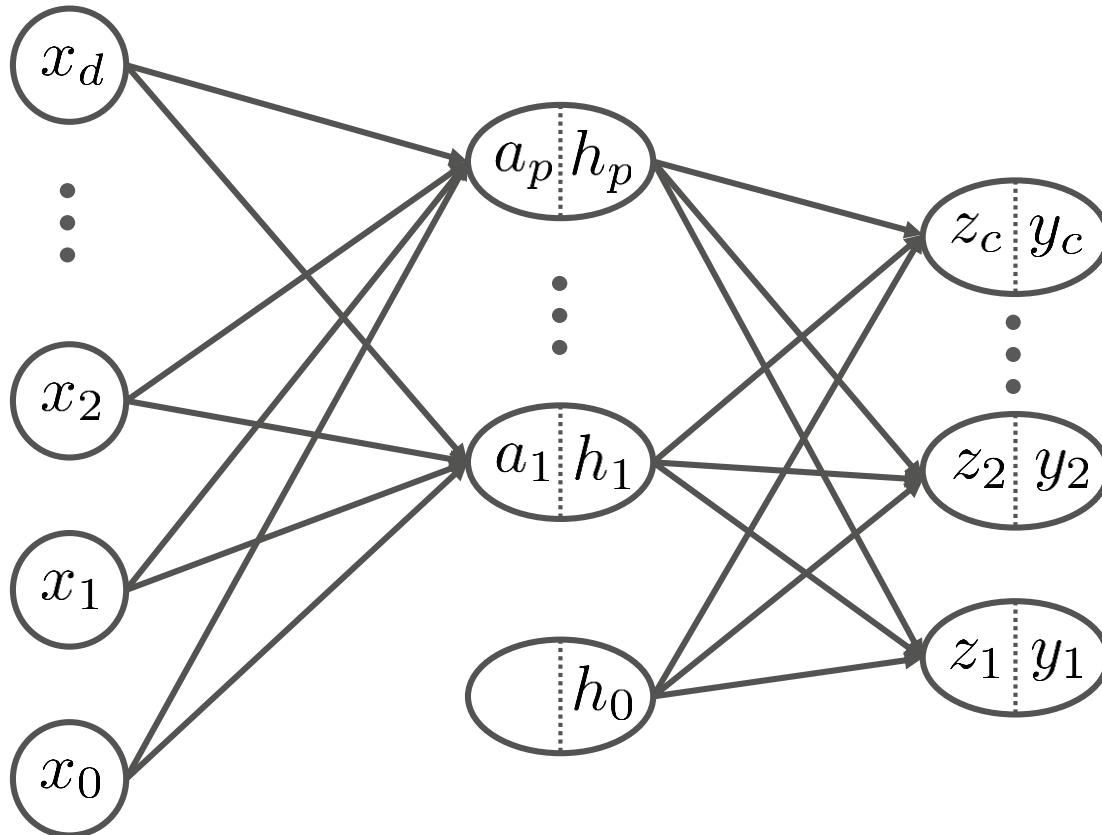
# Neural Networks: Multi-Layer Perceptron (MLP)

- 기초 구조: 1개의 은닉층을 가진 Feed-forward Neural Network (회귀)
  - ✓ 개별 은닉노드가 하는 역할은 퍼셉트론과 동일하며 출력노드의 결과값은 은닉노드의 결과값을 선형결합하여 생성



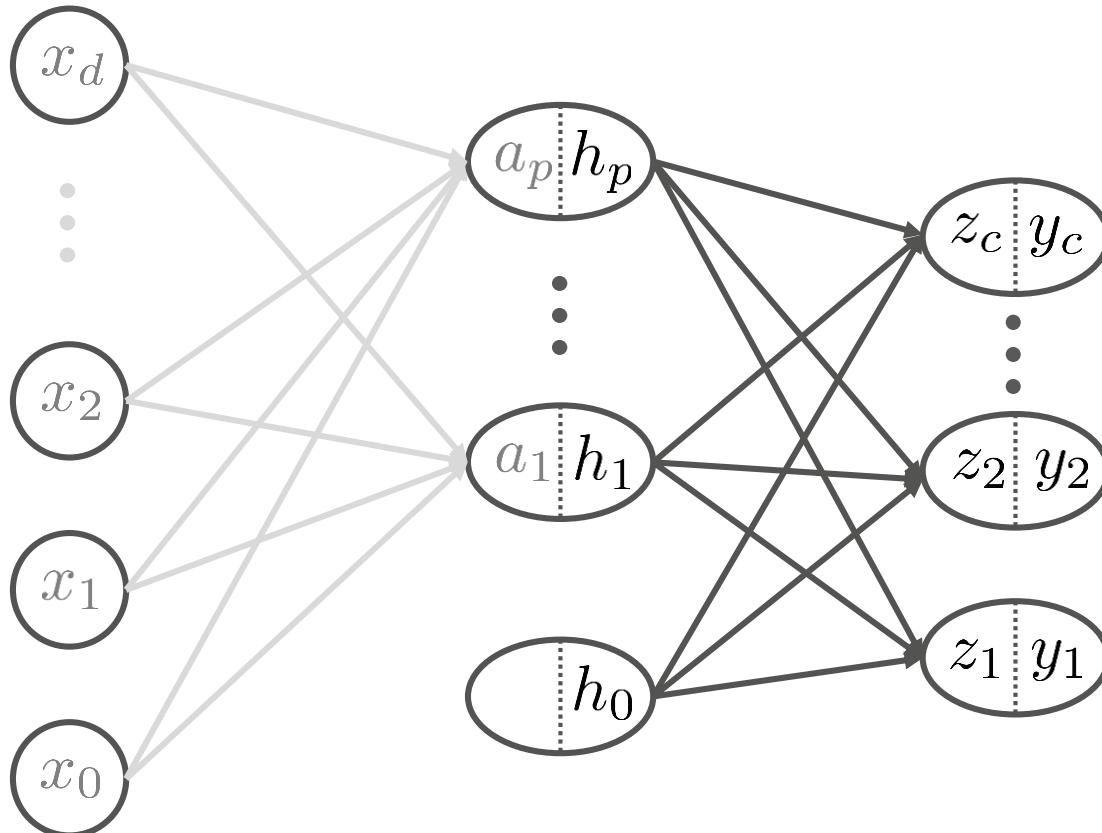
# Neural Networks: Multi-Layer Perceptron (MLP)

- 기초 구조: 1개의 은닉층을 가진 Feed-forward Neural Network (분류)
  - ✓ 개별 은닉노드가 하는 역할은 퍼셉트론과 동일하며 출력노드들은 은닉노드의 정보를 취합한 뒤 비선형 변환을 수행



# Neural Networks: Multi-Layer Perceptron (MLP)

- 기초 구조: 1개의 은닉층을 가진 Feed-forward Neural Network (분류)
  - ✓ 개별 은닉노드가 하는 역할은 퍼셉트론과 동일하며 출력노드들은 은닉노드의 정보를 취합한 뒤 비선형 변환을 수행 (주로 softmax 함수 사용)



$$z_j = \sum_{i=0}^p w_{jp}^{(2)} h_p$$

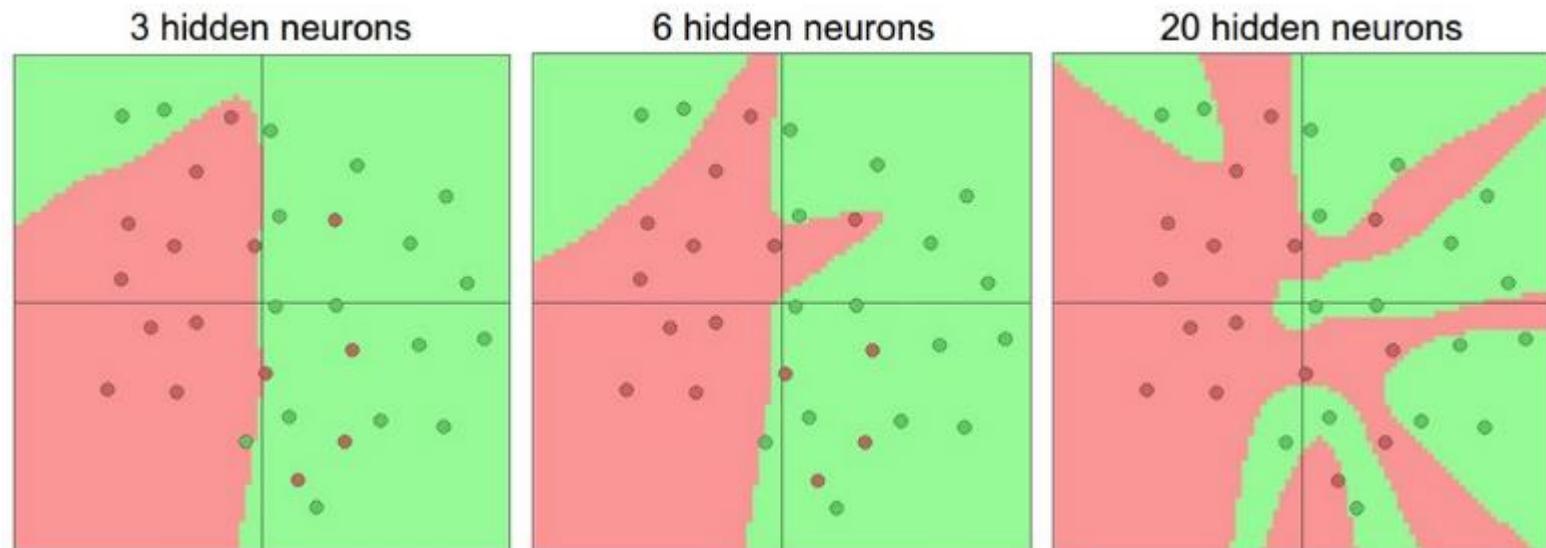
$$y_j = \frac{e^{z_j}}{\sum_{k=1}^c e^{z_k}}$$

각 출력 노드의 출력값을 해당 범주에 속하는 확률로 해석할 수 있음

# 1개의 은닉층을 가진 MLP: 은닉 노드

- **은닉노드의 역할**

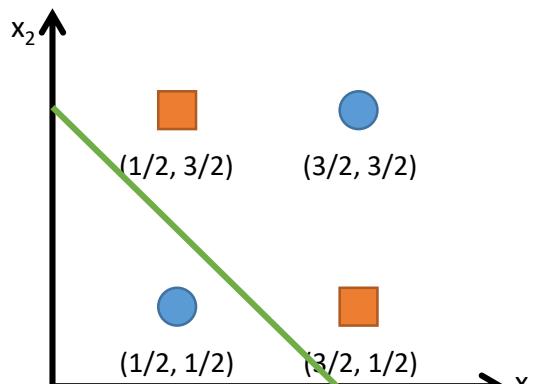
- ✓ 은닉노드의 수가 인공신경망의 복잡도(complexity)를 결정
- ✓ 은닉노드가 많을수록 임의의 분류 경계면을 찾거나(분류 문제) 굴곡이 많은 함수를 추정(회귀 문제)할 수 있음
- ✓ 아래 예시는 동일한 데이터셋에 은닉노드의 수를 다르게 하여 찾은 분류 경계면임



# 1개의 은닉층을 가진 MLP: 작동원리

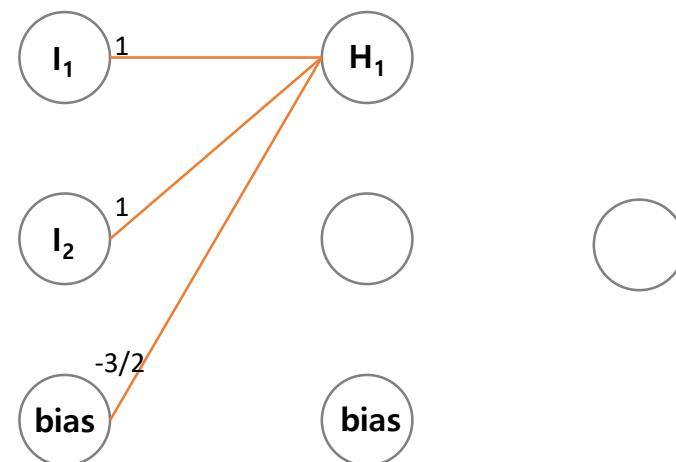
- XOR 예제를 이용한 MLP 작동원리

- ✓ XOR 문제는 두 개의 분류 경계면을 결합하여 풀어낼 수 있음 → 두 개의 은닉노드 필요



$$h_1 = x_1 + x_2 - \frac{3}{2}$$

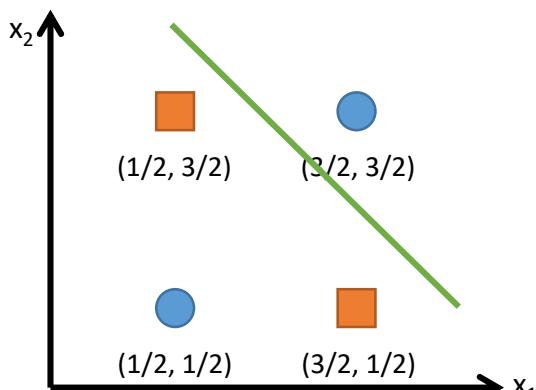
$$z_1 = g(h_1) = \begin{cases} 1 & \text{if } h_1 \geq 0 \\ -1 & \text{if } h_1 < 0 \end{cases}$$



	$x_1$	$x_2$	$h_1$	$z_1$
Blue Circle (Class 1)	1/2	1/2	-1/2	-1
Orange Square (Class 0)	3/2	1/2	1/2	1
Orange Square (Class 0)	1/2	3/2	1/2	1
Blue Circle (Class 1)	3/2	3/2	3/2	1

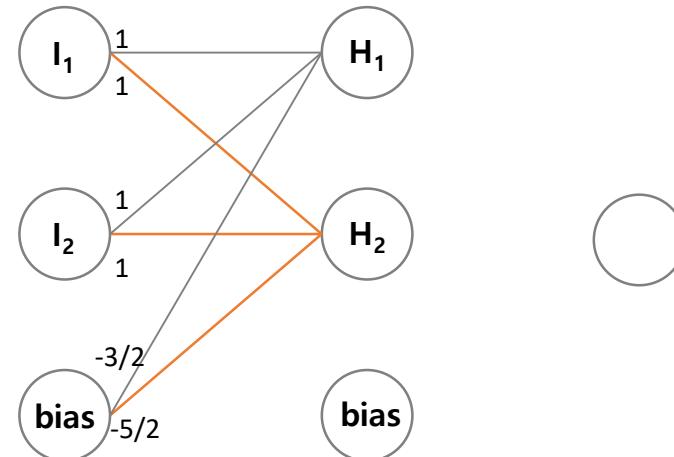
# 1개의 은닉층을 가진 MLP: 작동원리

- XOR 예제를 이용한 MLP 작동원리



$$h_2 = x_1 + x_2 - \frac{5}{2}$$

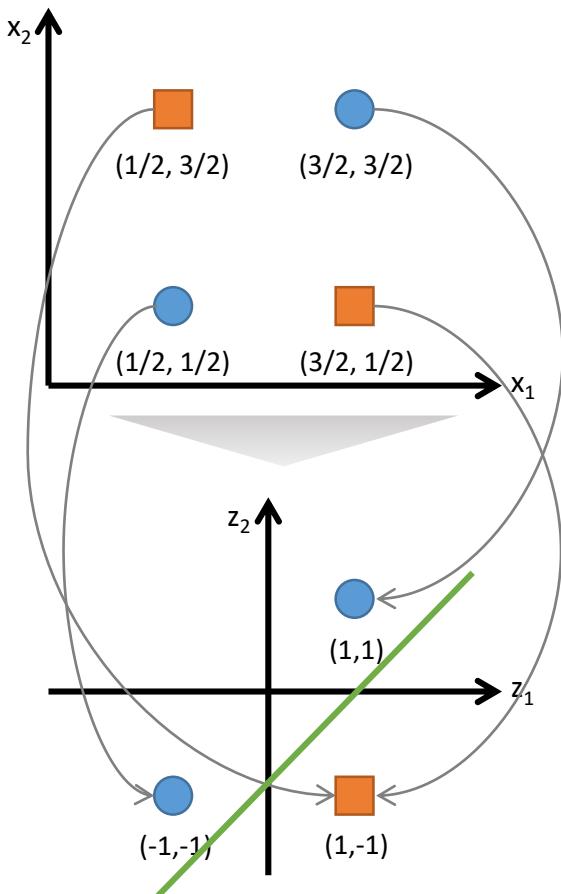
$$z_2 = g(h_2) = \begin{cases} 1 & \text{if } h_2 \geq 0 \\ -1 & \text{if } h_2 < 0 \end{cases}$$



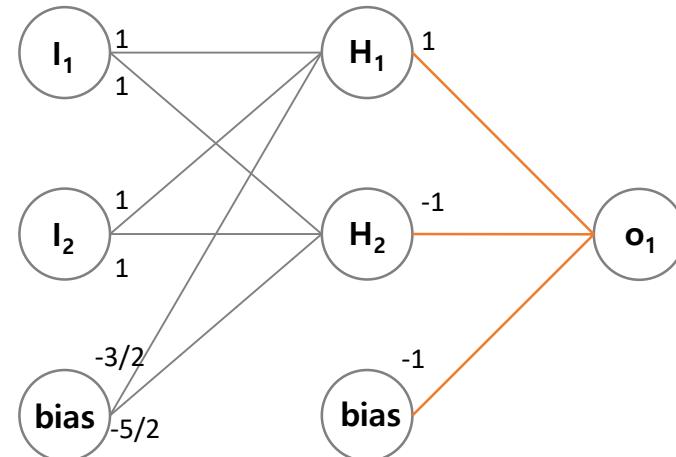
$x_1$	$x_2$	$h_2$	$z_2$
1/2	1/2	-3/2	-1
3/2	1/2	-1/2	-1
1/2	3/2	-1/2	-1
3/2	3/2	1/2	1

# 1개의 은닉층을 가진 MLP: 작동원리

- XOR 예제를 이용한 MLP 작동원리



$$o_1 = z_1 - z_2 - 1$$



	$x_1$	$x_2$	$h_1$	$z_1$	$h_2$	$z_2$	$o_1$	$z$
●	1/2	1/2	-1/2	-1	-3/2	-1	-1	-1
■	3/2	1/2	1/2	1	-1/2	-1	1	1
■	1/2	3/2	1/2	1	-1/2	-1	1	1
●	3/2	3/2	3/2	1	1/2	1	-1	-1

$$z = g(o_1) = \begin{cases} 1 & \text{if } o_1 \geq 0 \\ -1 & \text{if } o_1 < 0 \end{cases}$$

# I 개의 은닉층을 가진 MLP: Formulation

- General formulation (회귀를 기준으로 설명)

- ✓ j번째 은닉 노드는 모든 입력 변수의 값과 연결된 가중치의 결합으로 표현됨 (시그모이드 활성함수)

$$h_j = \sum_{i=1}^{d+1} w_{ji}^{(1)} x_i, \quad z_j = g(h_j) = \frac{1}{1 + \exp(-h_j)}$$

- ✓ 출력노드는 모든 은닉노드의 값과 연결된 가중치의 선형 결합으로 표현됨

$$o = \sum_{i=1}^{p+1} w_i^{(2)} z_i$$

- ✓ 따라서, 종속변수는 입력변수들의 선형/비선형 결합의 조합으로 표현됨

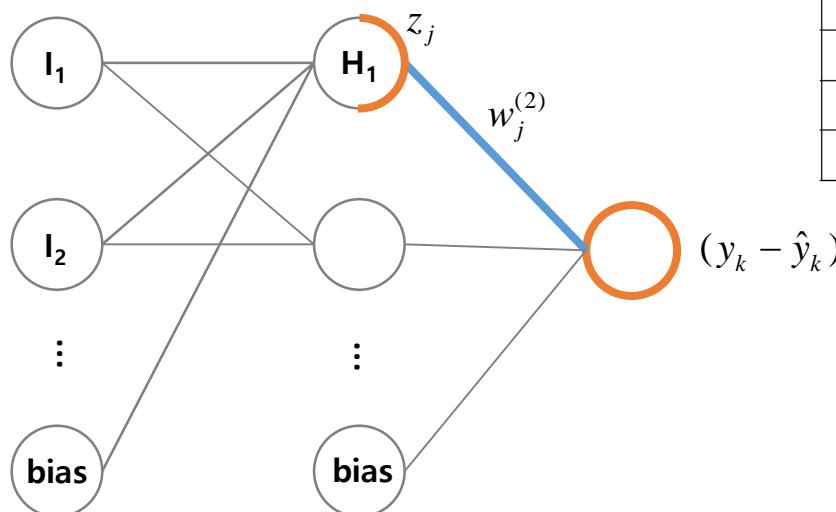
$$\hat{y} = \sum_{j=1}^{p+1} w_j^{(2)} g\left(\sum_{i=1}^{d+1} w_{ji}^{(1)} x_i\right)$$

# 1개의 은닉층을 가진 MLP: 학습

- 오류 역전파 알고리즘: Error Back-propagation by Gradient Descent
  - ✓ k번째 관측치의 오차
  - ✓ j번째 은닉 노드와 출력노드를 연결하는 가중치  $w_j^{(2)}$ 의 변화량:

$$Err_k = \frac{1}{2} (y_k - \hat{y}_k)^2, \quad \hat{y}_k = \sum_{j=1}^{p+1} w_j^{(2)} g\left(\sum_{i=1}^{d+1} w_{ji}^{(1)} x_i\right)$$

$$\frac{\partial Err_k}{\partial w_j^{(2)}} = \frac{\partial Err_k}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial w_j^{(2)}} = (y_k - \hat{y}_k) \cdot z_j$$

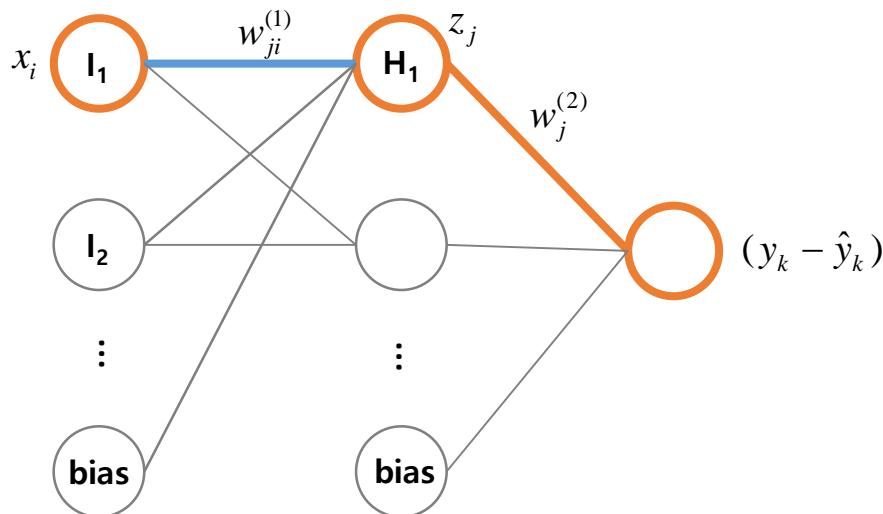


최종 결과물을 얻고	Feed Forward and Prediction
그 결과물과 우리가 원하는 결과물과의 차이점을 찾은 후	Cost Function
그 차이가 무엇으로 인해 생기는지	Differentiation (미분)
역으로 내려가면서 추정하여	Back Propagation
새로운 Parameter 값을 배움	Weight Update

# 1개의 은닉층을 가진 MLP: 학습

- 오류 역전파 알고리즘: Error Back-propagation by Gradient Descent
  - ✓ j번째 은닉노드와 i번째 입력노드를 연결하는 가중치  $w_{ji}^{(1)}$ 의 변화량

$$\frac{\partial Err_k}{\partial w_{ji}^{(1)}} = \frac{\partial Err_k}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial z_j} \cdot \frac{\partial z_j}{\partial h_j} \cdot \frac{\partial h_j}{\partial w_{ji}^{(1)}} = (y_k - \hat{y}_k) \cdot w_j^{(2)} \cdot z_j \cdot (1 - z_j) \cdot x_i$$



# 1개의 은닉층을 가진 MLP: 학습

- 역전파 알고리즘이 작동하는 이유

- ✓ 출력층-은닉층 가중치 Gradient

$$\frac{\partial Err_k}{\partial w_j^{(2)}} = \frac{\partial Err_k}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial w_j^{(2)}} = (y_k - \hat{y}_k) \cdot z_j$$

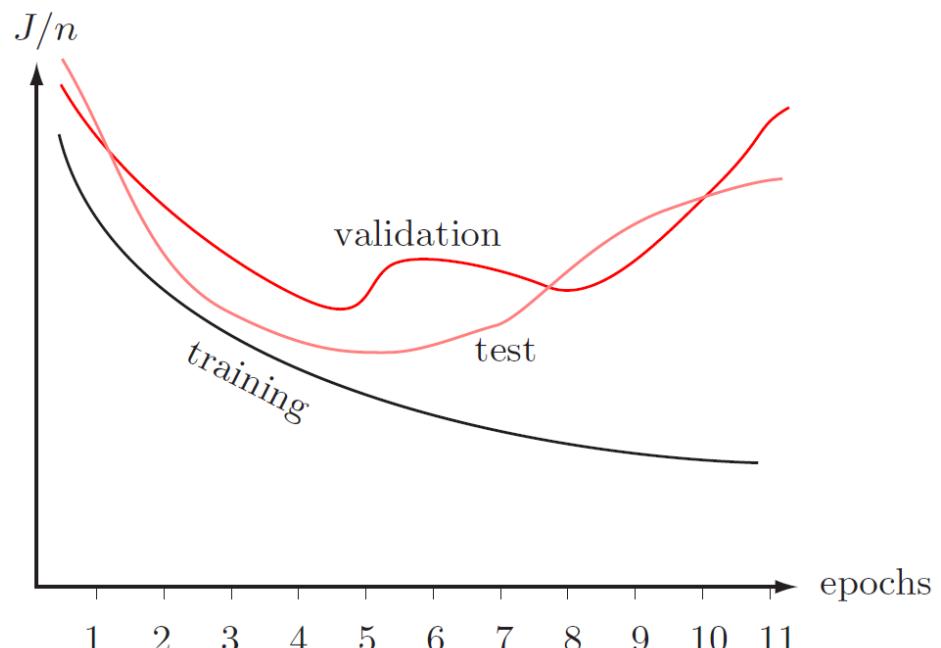
- ✓ 은닉층-입력층 가중치 Gradient

$$\frac{\partial Err_k}{\partial w_{ji}^{(1)}} = \frac{\partial Err_k}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial z_j} \cdot \frac{\partial z_j}{\partial h_j} \cdot \frac{\partial h_j}{\partial w_{ji}^{(1)}} = (y_k - \hat{y}_k) \cdot w_j^{(2)} \cdot z_j \cdot (1 - z_j) \cdot x_i$$

- ✓ 큰 오차는 가중치를 크게 변화시키고 작은 오차는 가중치를 작게 변화시킴
  - ✓ 이러한 가중치의 변화를 지속적으로 반복하면, 가중치는 점진적으로 특정 값으로 수렴하게 됨

# 인공 신경망: 과적합 문제

- 은닉 노드의 수를 많이 사용하면서 충분히 많은 수의 epochs을 수행할 경우 인공 신경망은 과적합할 가능성이 높음
- 과적합을 방지하기 위한 장치
  - ✓ 검증 데이터 오류 확인
  - ✓ Epoch 수 제한
  - ✓ 인공신경망의 복잡도 제한
    - 은닉층/은닉노드의 수 제한
  - ✓ 가중치 변화량이 일정 수준 이하일 경우 학습 종료

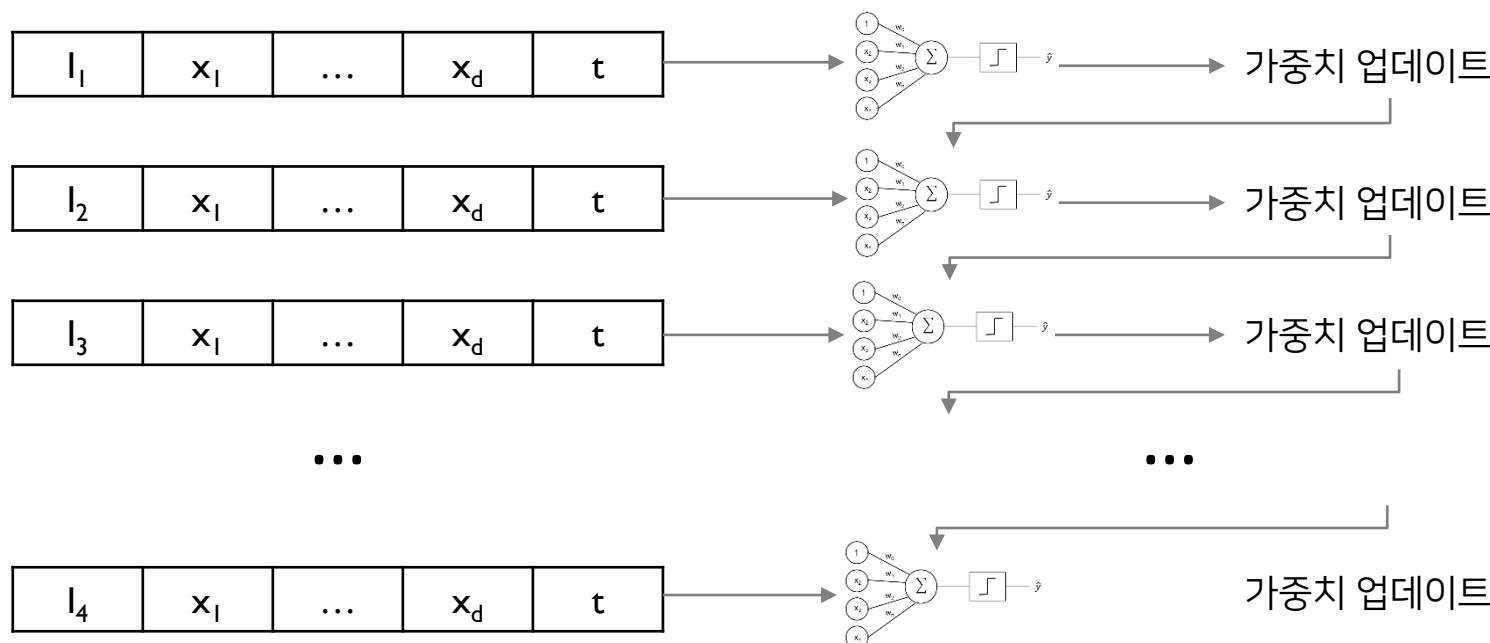


# 인공 신경망 학습 시 고려사항

- 고려사항 I: 가중치를 얼마나 자주 업데이트해야 하는가?

- ✓ Stochastic Gradient Descent (SGD)

- 개체 하나마다 손실 함수를 계산하고 해당 손실함수로부터 Gradient를 계산하여 가중치 업데이트 (학습 데이터가 N개 있으면 1회 iteration에 가중치 업데이트는 N번 수행)

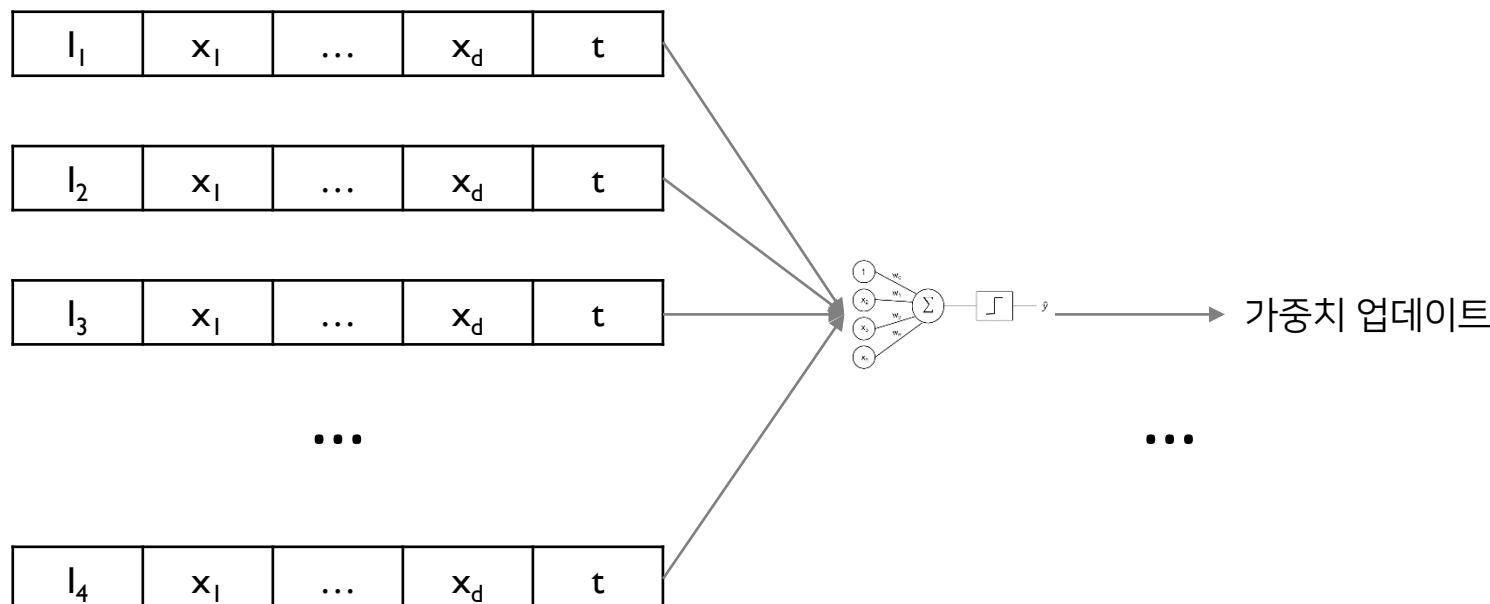


# 인공 신경망 학습 시 고려사항

- 고려사항 I: 가중치를 얼마나 자주 업데이트해야 하는가?

- ✓ Batch Gradient Descent (BGD)

- 학습 데이터셋의 모든 개체들에 대해서 네트워크를 고정한 상태로 비용함수를 계산하고 해당 비용함수로부터 Gradient를 계산하여 가중치 업데이트 (학습 데이터가 N개 있으면 1회 iteration에 가중치 업데이트는 1번 수행)

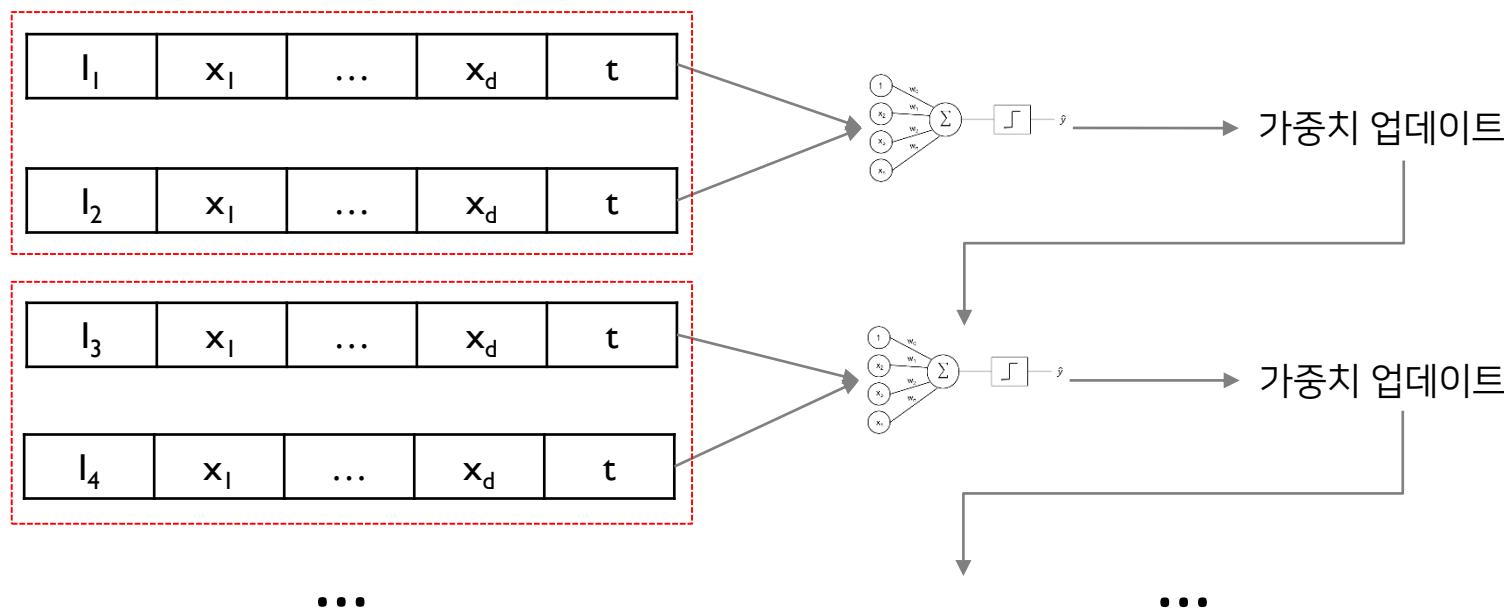


# 인공 신경망 학습 시 고려사항

- 고려사항 I: 가중치를 얼마나 자주 업데이트해야 하는가?

- ✓ Mini-Batch Gradient Descent

- SGD와 BGD의 중간 단계로  $n$ 개의 개체로 구성된 mini-batch마다 gradient를 계산한 뒤 가중치를 업데이트 (아래 예시는 batch size = 2로 설정한 경우)

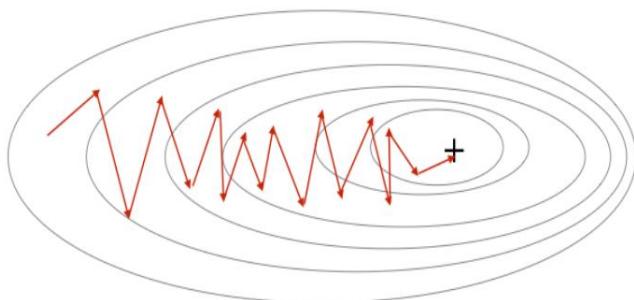


# 인공 신경망 학습 시 고려사항

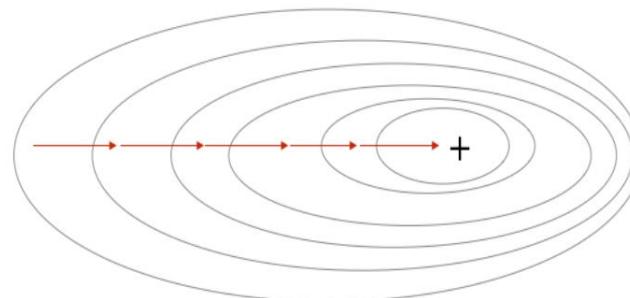
- 고려사항 I: 가중치를 얼마나 자주 업데이트해야 하는가?

- ✓ 가중치 업데이트 방식에 따른 가중치 변화 예시

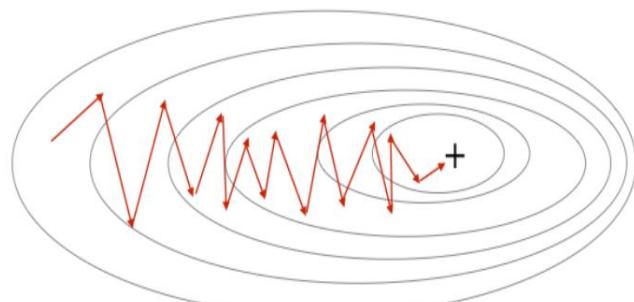
Stochastic Gradient Descent



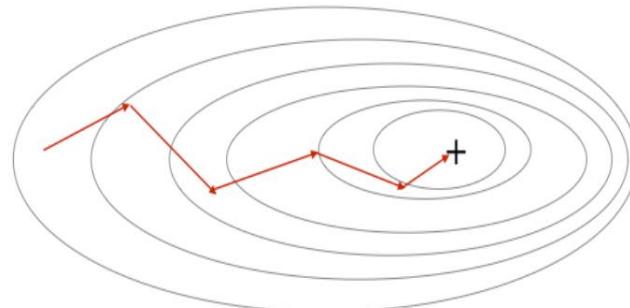
Gradient Descent



Stochastic Gradient Descent



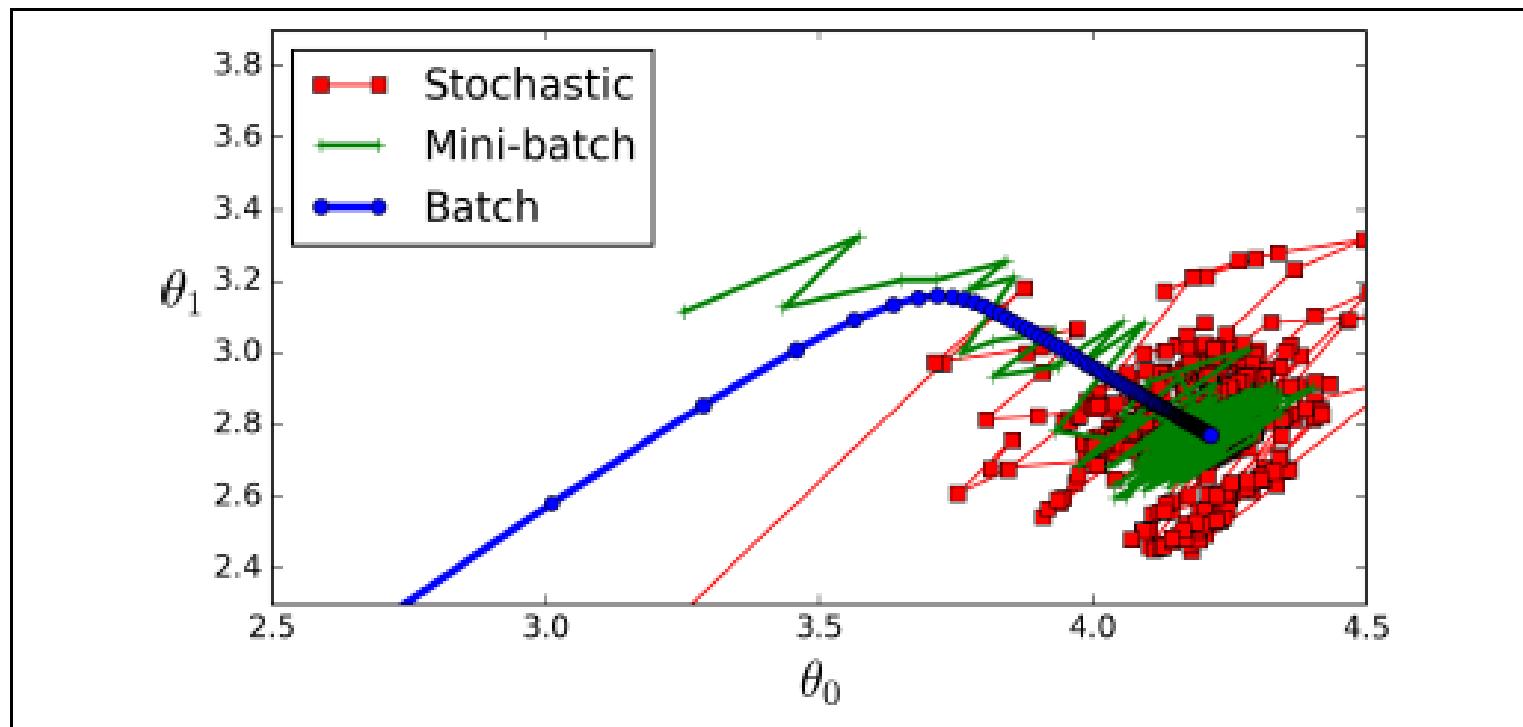
Mini-Batch Gradient Descent



# 인공 신경망 학습 시 고려사항

- 고려사항 I: 가중치를 얼마나 자주 업데이트해야 하는가?

- ✓ 가중치 업데이트 방식에 따른 가중치 변화 예시

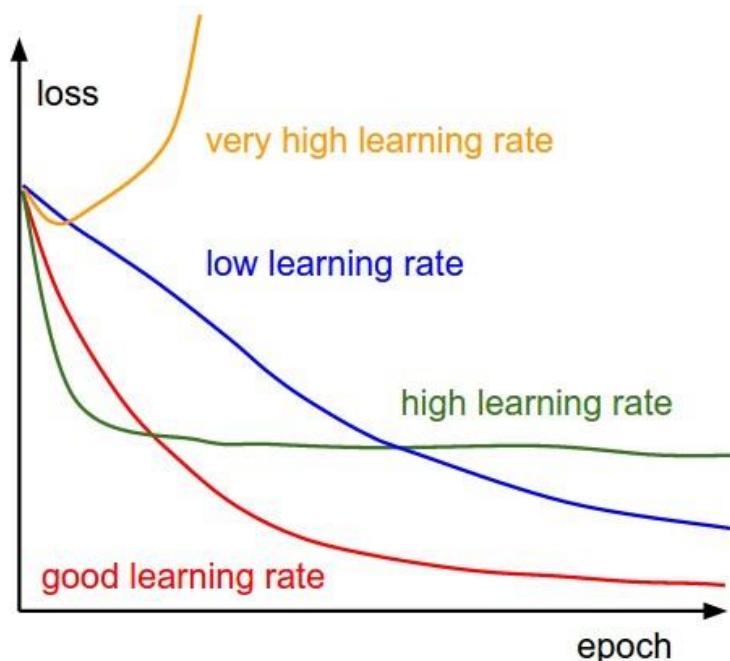


# 인공 신경망 학습 시 고려사항

- 고려사항 2: 가중치를 한번 업데이트 할 때 얼마만큼 변화시켜야 하는가?
  - ✓ 학습률(learning rate) 결정의 문제

$$w_{new} = w_{old} - \alpha f'(w)$$

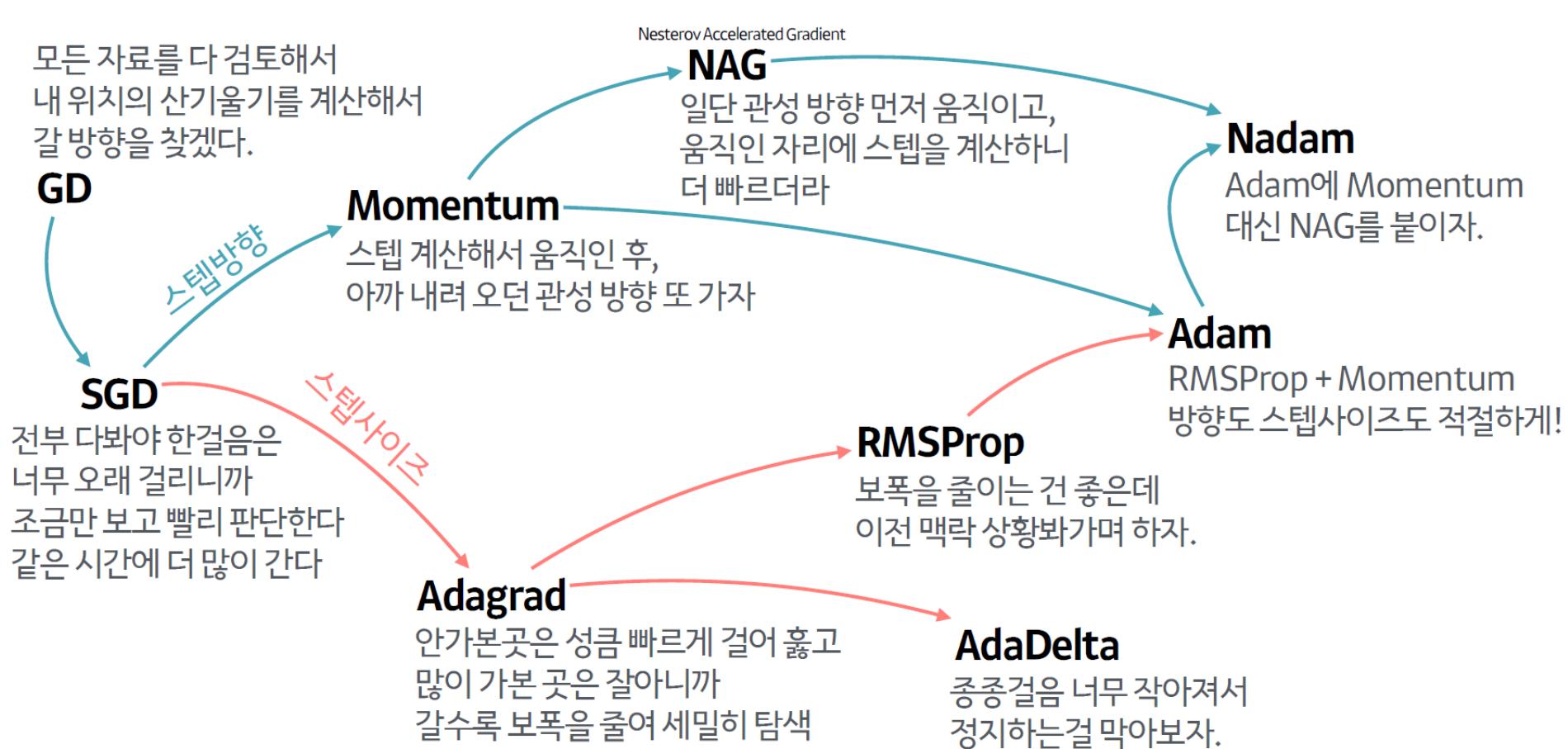
- ✓ 너무 크면 학습이 안되고, 너무 작으면 시간이 오래 걸리고...
- ✓ 적당한 학습률이 필요한데 그 적당함(?)을 찾는 것이 어려움



# 인공 신경망 학습 시 고려사항

## • 고려사항 2: 가중치를 한번 업데이트 할 때 얼마만큼 변화시켜야 하는가?

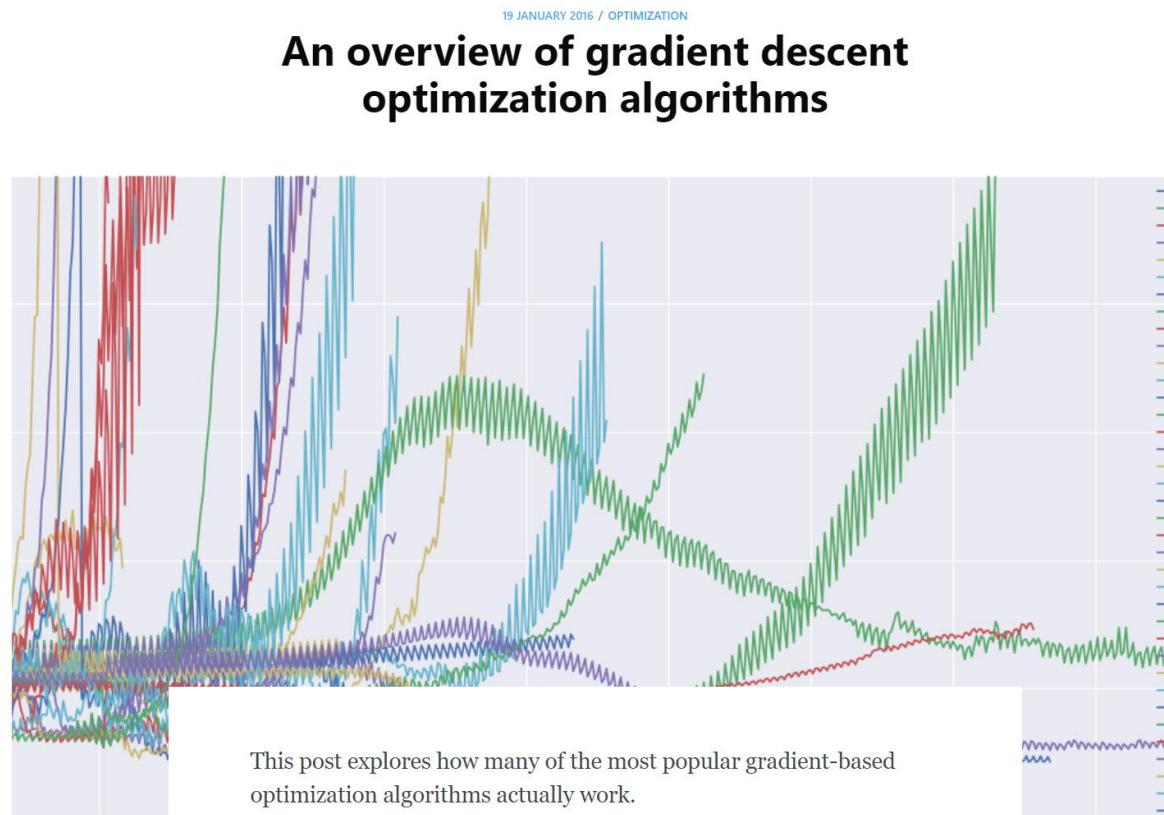
- ✓ 하용호. 자습해도 모르겠던 딥러닝, 머리속에 인스톨 시켜드립니다. (<https://www.slideshare.net/yongho/ss-79607172>)



# 인공 신경망 학습 시 고려사항

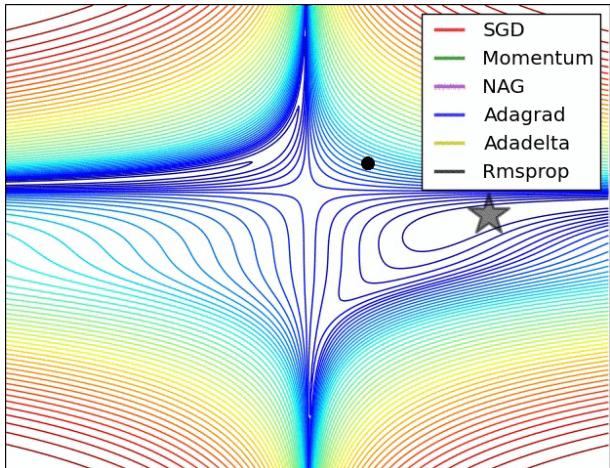
- 고려사항 2: 가중치를 한번 업데이트 할 때 얼마만큼 변화시켜야 하는가?

- ✓ 참고 자료: An overview of gradient descent optimization algorithms by Sebastian Ruder
- ✓ <http://ruder.io/optimizing-gradient-descent/>

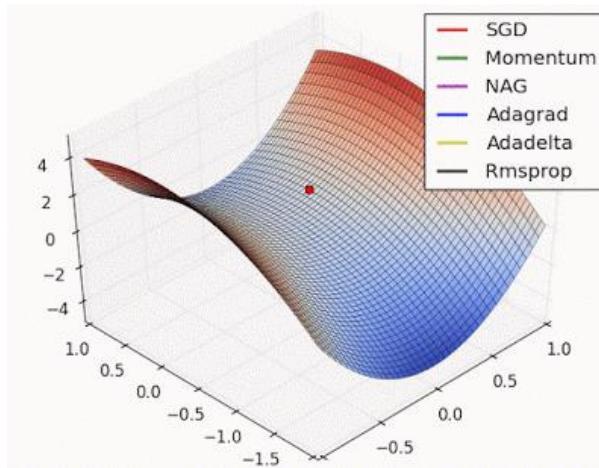


# 인공 신경망 학습 시 고려사항

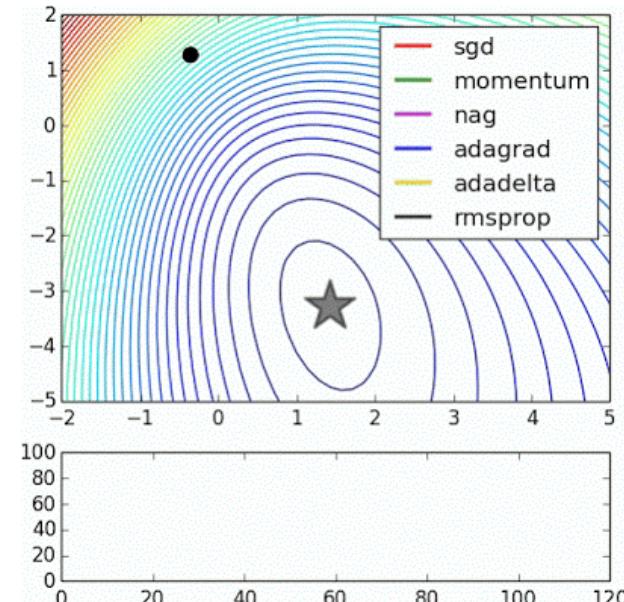
- 학습 방식에 따른 해의 수렴 예시



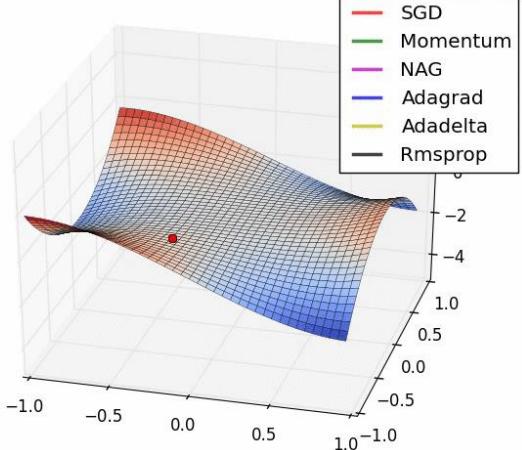
Beale's function



Long valley



Noisy moons

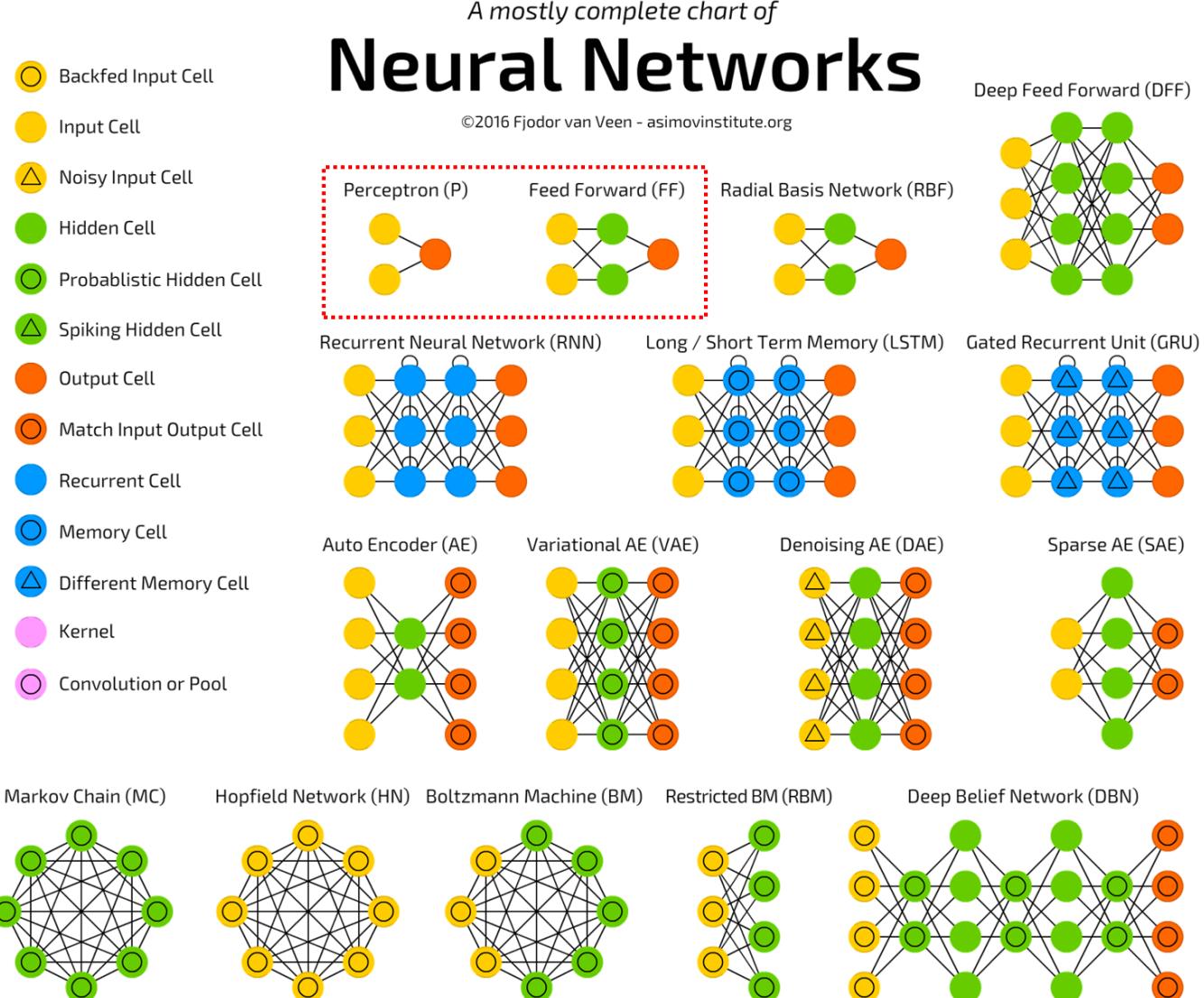


Saddle point



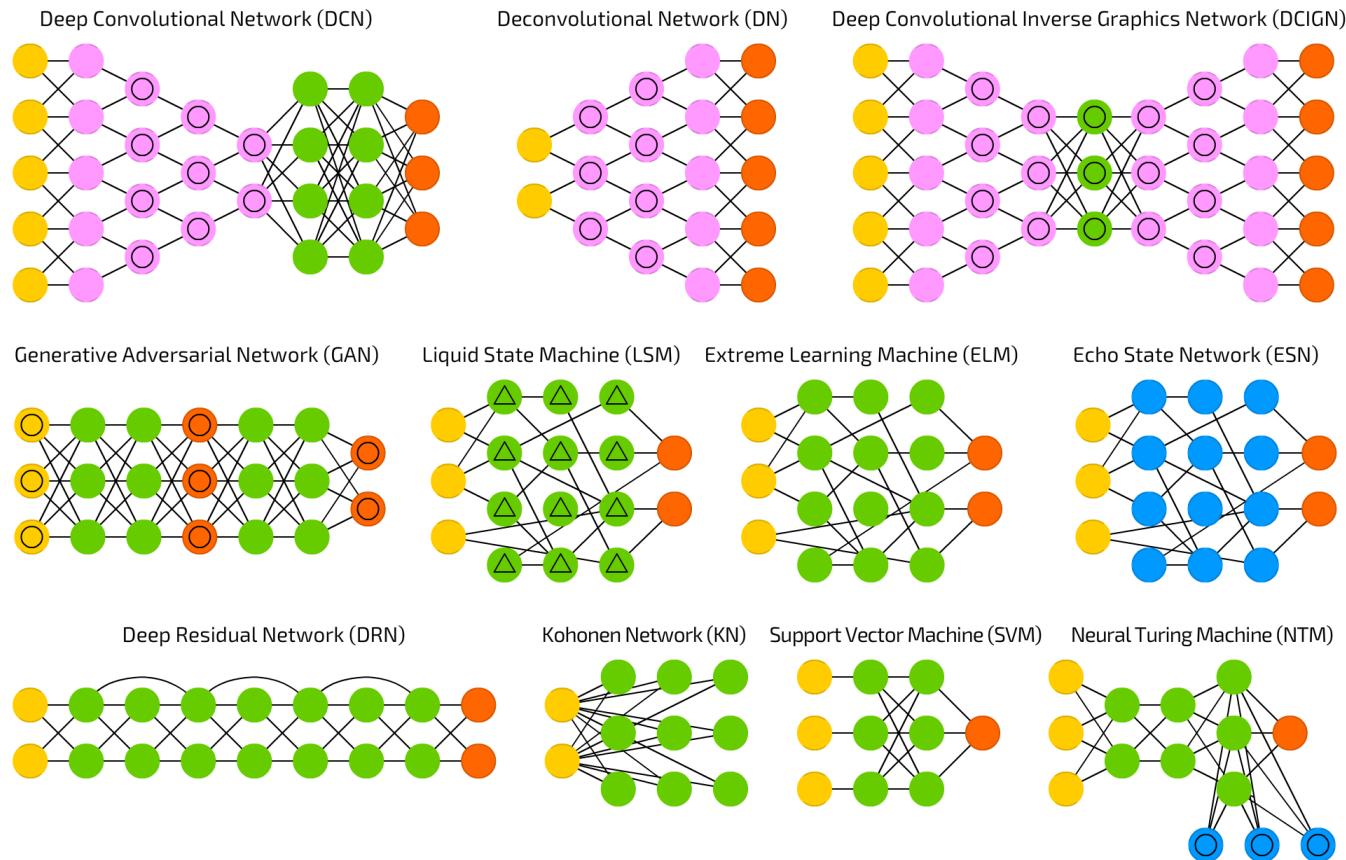
# 참고: 다양한 인공신경망 구조

- Neural Network Zoo



# 참고: 다양한 인공신경망 구조

- Neural Network Zoo



# 인공신경망: 요약

- 인공신경망 장점
  - ✓ 우수한 예측 성능
  - ✓ 복잡한 입력변수-출력변수 관계를 규명할 수 있음
- 인공신경망 단점
  - ✓ 입력변수와 출력변수 간의 인과관계를 규명할 수 없음 (black-box model)
  - ✓ 변수선택 매커니즘 부재, 변수의 수가 증가하면 과적합 위험 증가
  - ✓ 높은 수준의 계산복잡도 → 학습에 오랜 시간이 소요
- Deep Learning (2회차 교육에서 중점적으로 다룸)
  - ✓ 최근 연산속도의 비약적인 발전으로 인해 다시 각광받고 있는 여러 개의 은닉층을 가진 인공신경망



ANY  
questions?