



Artificial Neural Network: MLP

Pilsung Kang

School of Industrial Management Engineering

Korea University

AGENDA

01 Artificial Neural Networks: Perceptron

02 Multi-layer Perceptron (MLP)

Perceptron: Limitation

- The Limitation of Linear Models

- ✓ **Classification:**

- Linear (Fisher) discriminant analysis, logistic regression, etc.
- Can only produce a linear class boundary

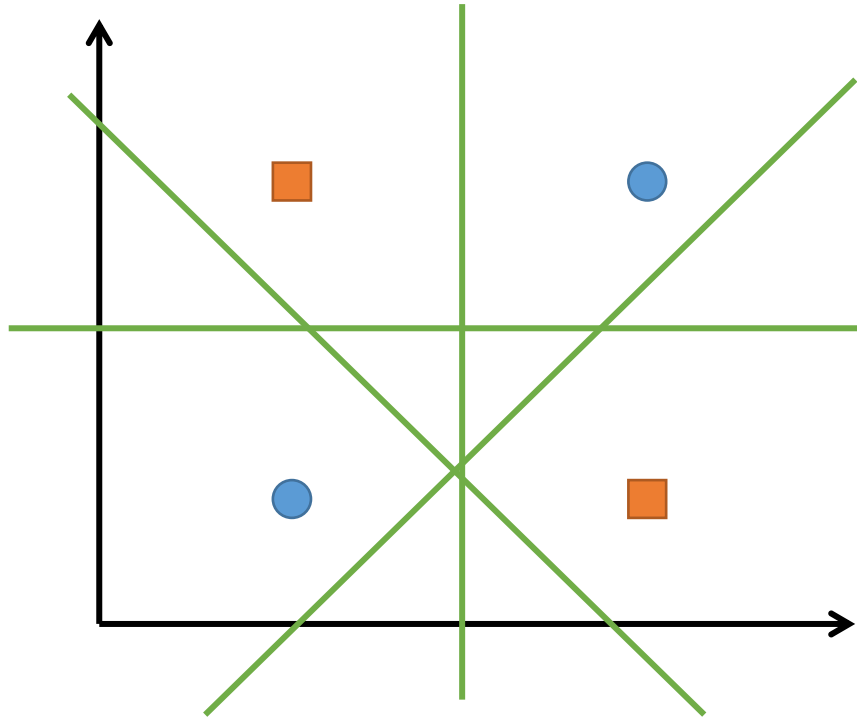
- ✓ **Regression:**

- Multiple linear regression
- Can only capture the linear relationship between the predictors and the outcome

- ✓ Cannot results in good prediction performance *when the classification boundary or the predictor/outcome relationship is not linear*

Perceptron: Limitation

- The Limitation of Linear Models
 - ✓ Draw a straight line that perfectly separates the circles and crosses (XOR)

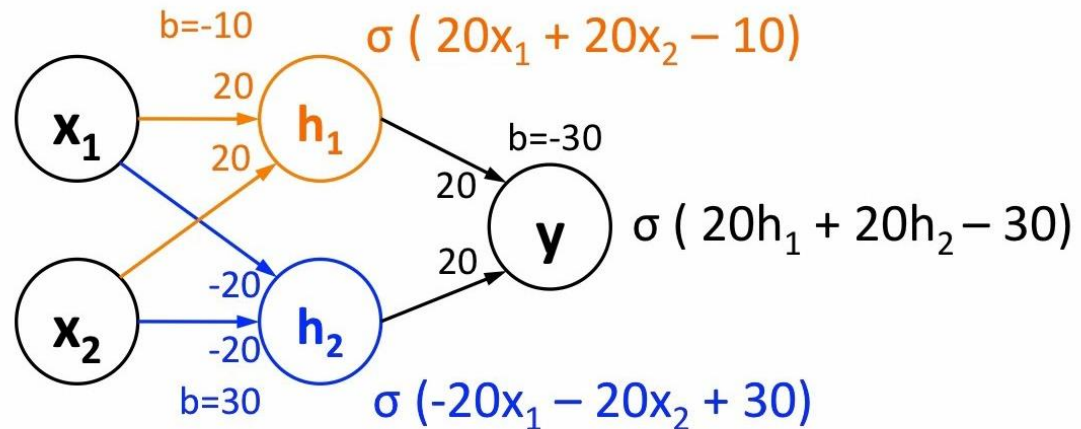
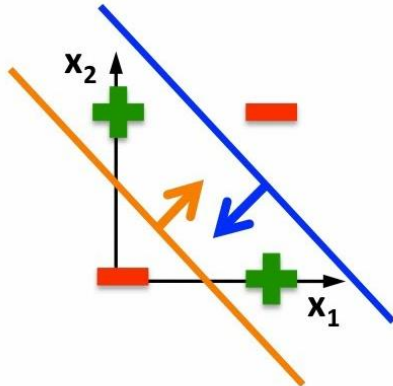


Multi-Layer Perceptron (MLP)

- Combine multiple perceptrons!

✓ If we cannot solve a complex problem directly, then it is better to **decompose** it into some small and simple problems that can be solved!

Linear classifiers cannot solve this



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

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

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

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

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

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

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

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

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

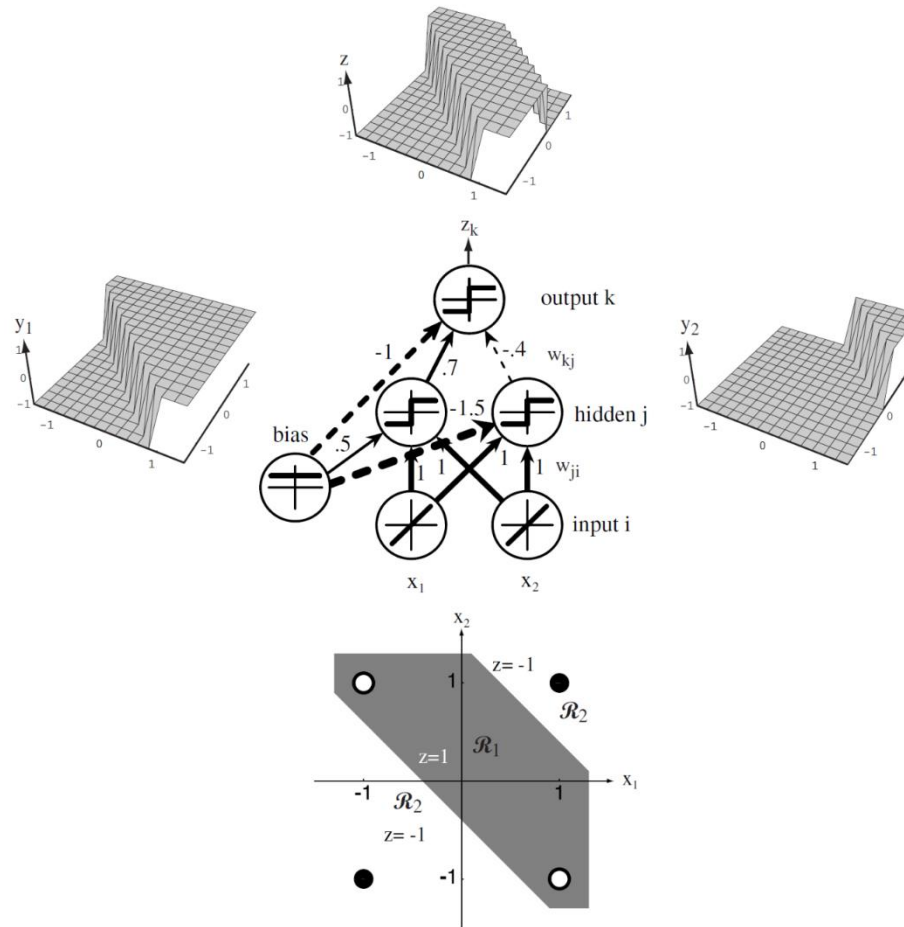
$$\sigma(20 \cdot 1 + 20 \cdot 0 - 30) \approx 0$$

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

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

Multi-Layer Perceptron (MLP)

- Non-linear model
 - ✓ Can find an arbitrary shape of class boundary or regression functions



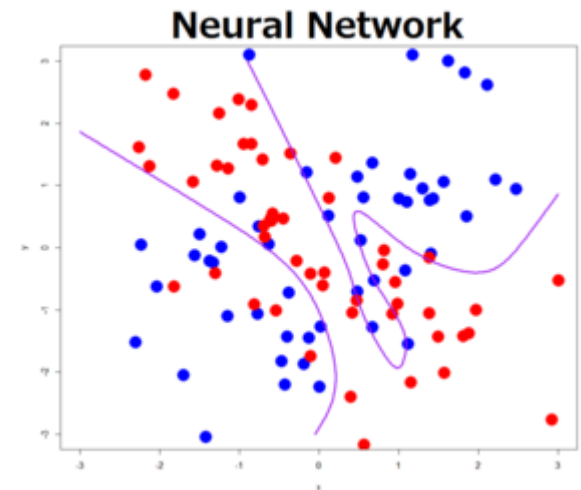
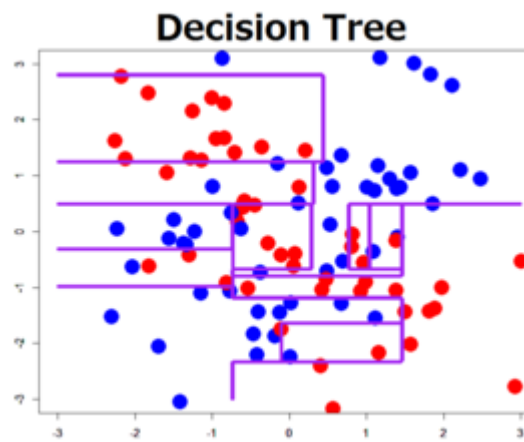
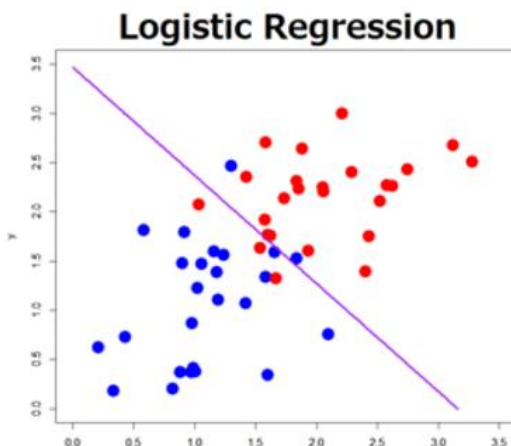
Multi-Layer Perceptron (MLP)

- Decision boundary of MLP

- ✓ Assume that the class decision boundary can be regarded as a combination of piecewise linear boundaries

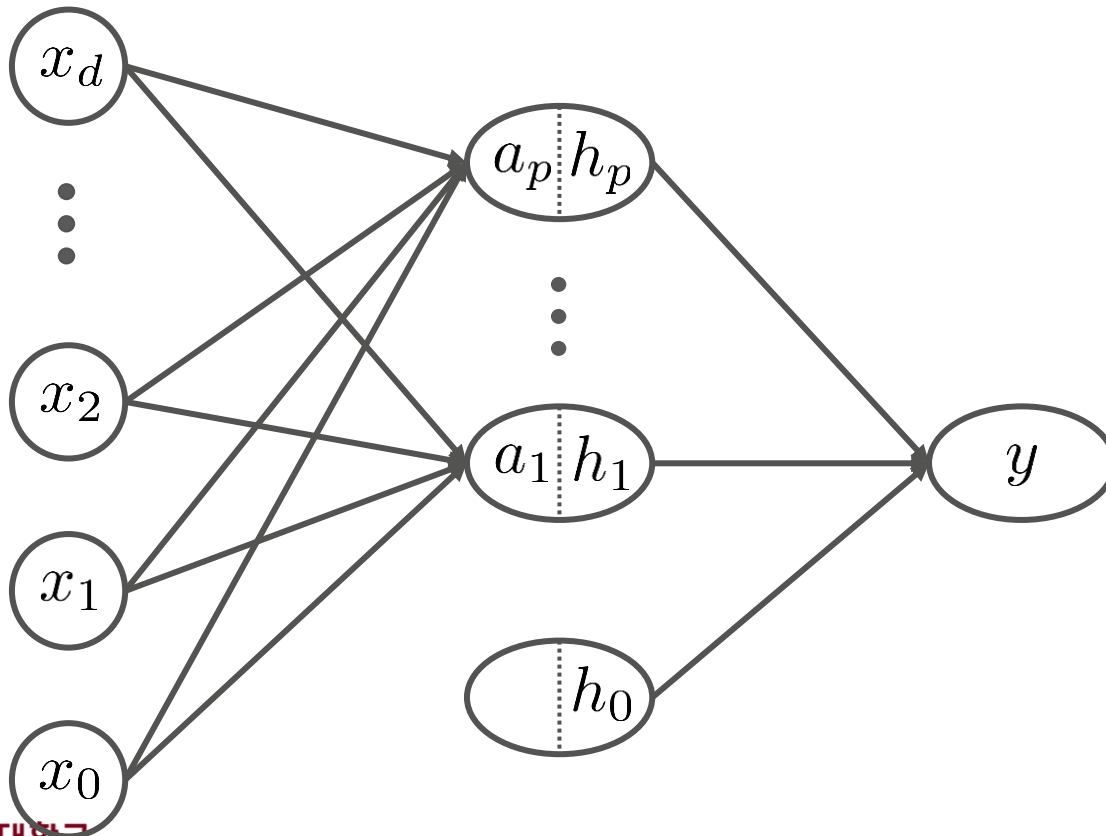
	Logistic Regression	Decision Tree	MLP
No. of lines	1	No restriction	User defined (No of hidden layers and hidden nodes)
Direction of lines	No restriction	Vertical to an axis	No restriction

- MLP has the highest degree of freedom to defined the decision boundary



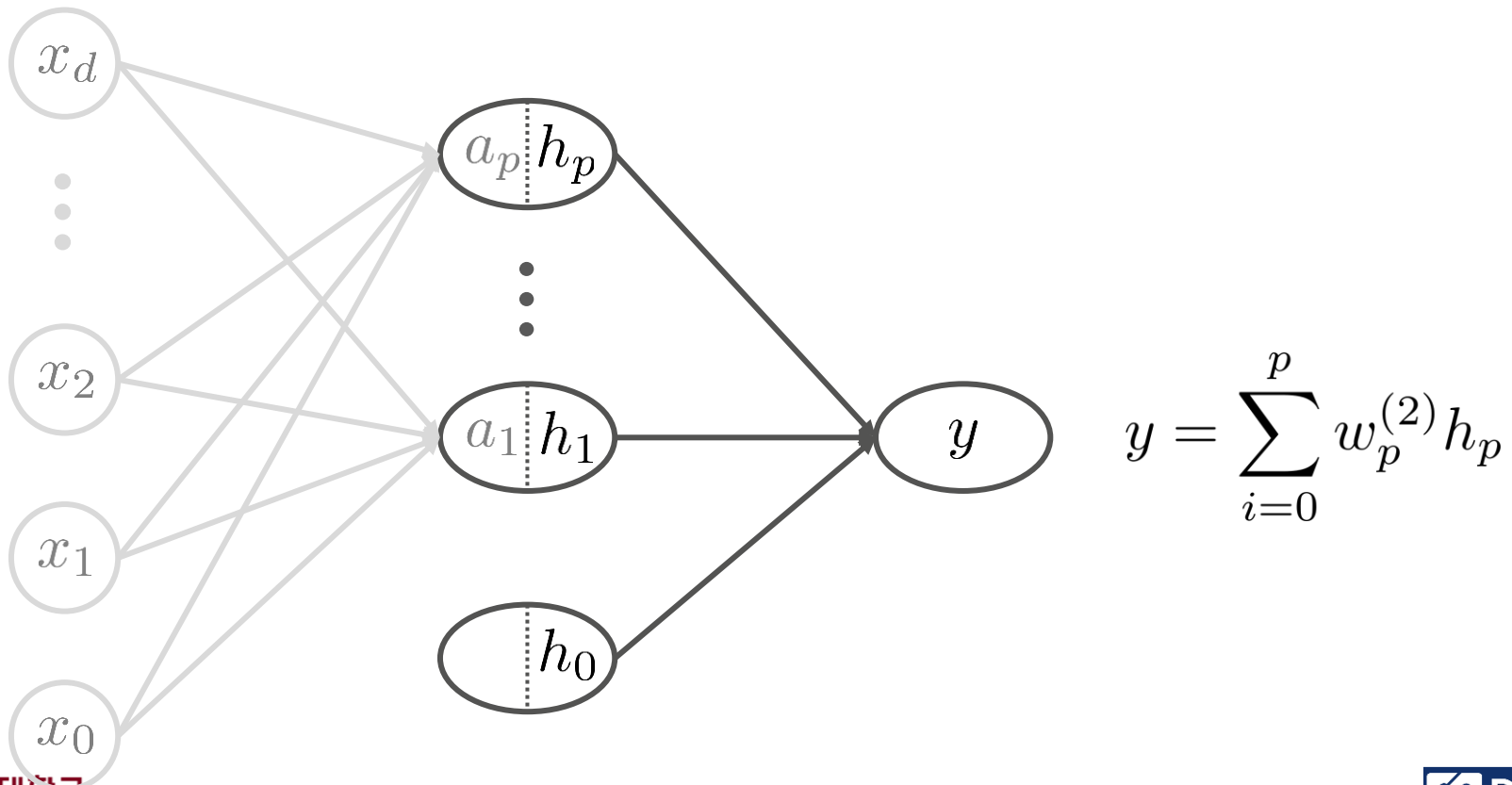
Multi-Layer Perceptron (MLP)

- Basic Structure: Feed-forward Neural Network with One Hidden Layer
 - ✓ Each hidden node can be considered as an independent perceptron
 - ✓ The output node is a combination of all perceptrons



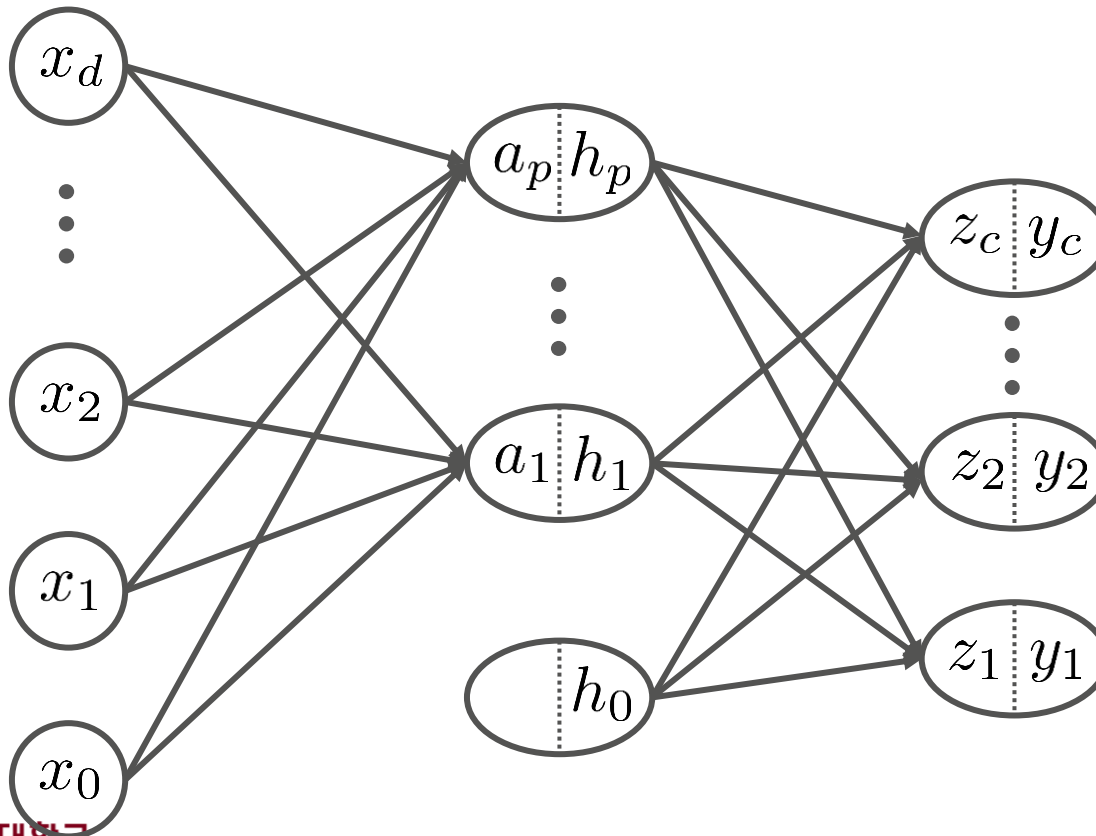
Multi-Layer Perceptron (MLP)

- Basic Structure: Feed-forward Neural Network with One Hidden Layer
 - ✓ Each hidden node can be considered as an independent perceptron
 - ✓ The output node is a linear combination of all perceptrons (regression)



Multi-Layer Perceptron (MLP)

- Basic Structure: Feed-forward Neural Network with One Hidden Layer
 - ✓ Each hidden node can be considered as an independent perceptron
 - ✓ The output node is a linear combination of all perceptrons (regression)



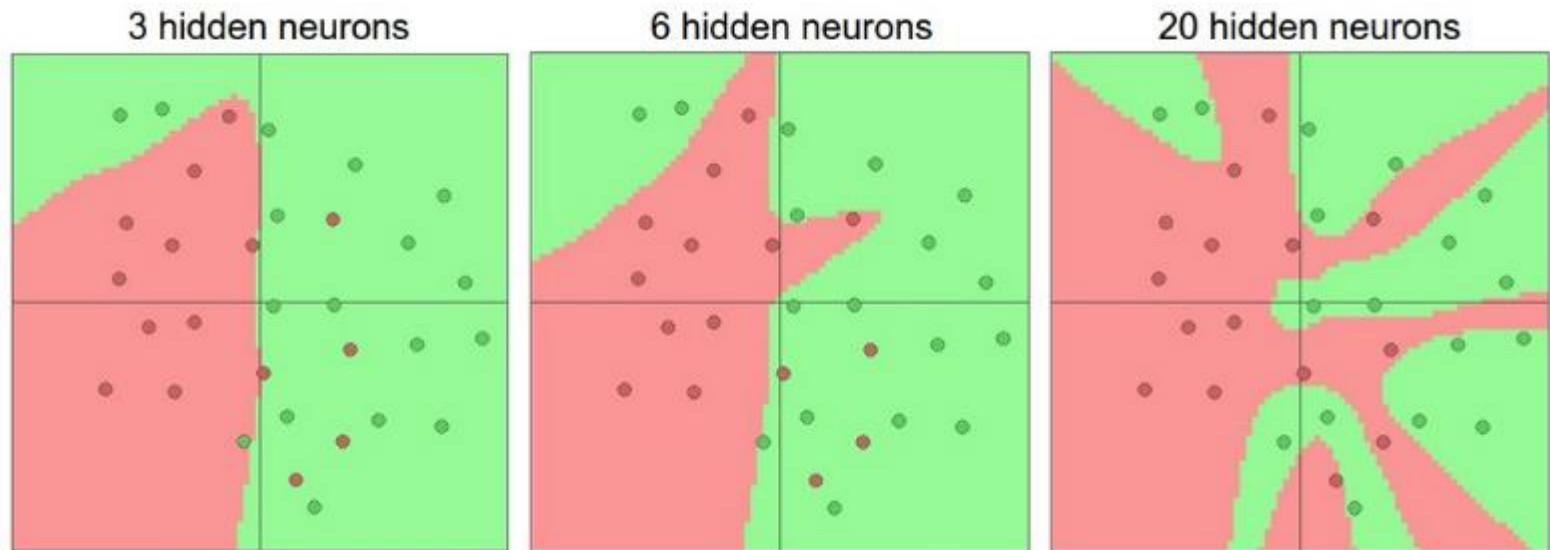
$$z_j = \sum_{i=0}^p w_{jp}^{(2)} h_p$$

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

↓
각 출력 노드의 출력값을 해당 범주에 속하는 확률로 해석할 수 있음

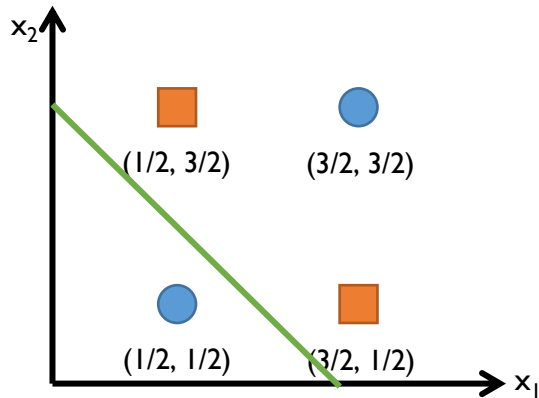
Multi-Layer Perceptron (MLP)

- The role of hidden nodes
 - ✓ Determines the complexity of ANN
 - ✓ If we use more number of hidden nodes, we can find a more sophisticated decision boundary (classification) or an arbitrary shape of function (regression)



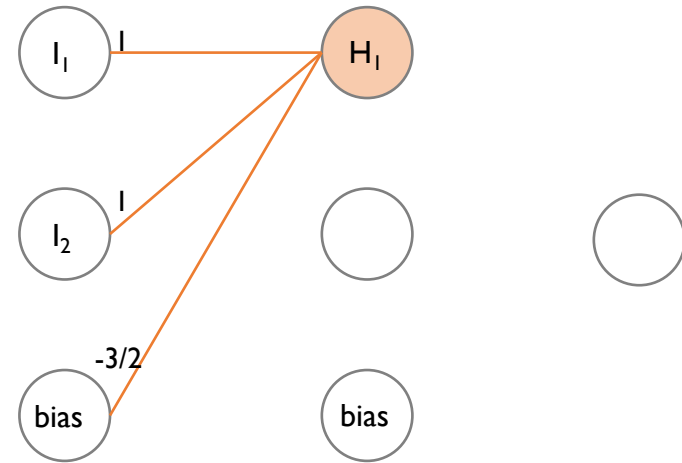
Multi-Layer Perceptron (MLP)

- XOR problem revisited



$$a_1 = x_1 + x_2 - \frac{3}{2}$$

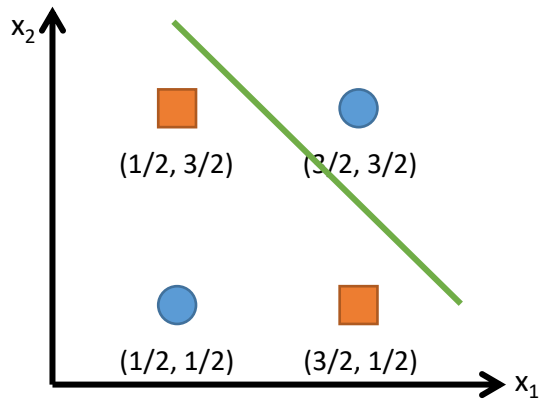
$$h_1 = g(a_1) = \begin{cases} 1 & \text{if } a_1 \geq 0 \\ -1 & \text{if } a_1 < 0 \end{cases}$$



	x_1	x_2	a_1	h_1
●	1/2	1/2	-1/2	-1
■	3/2	1/2	1/2	1
■	1/2	3/2	1/2	1
●	3/2	3/2	3/2	1

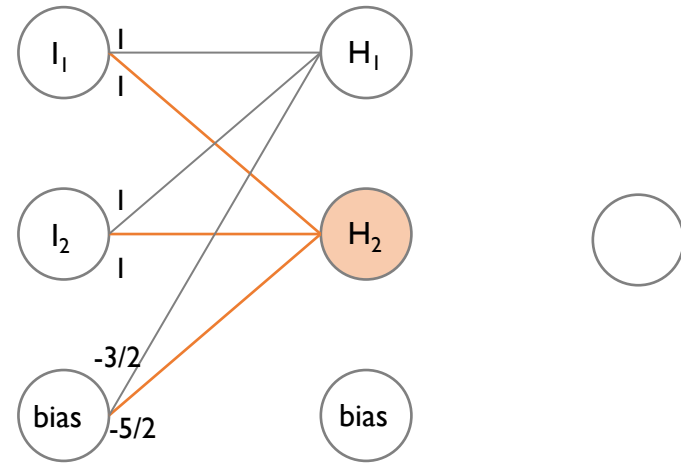
Multi-Layer Perceptron (MLP)

- XOR problem revisited (cont')



$$a_2 = x_1 + x_2 - \frac{5}{2}$$

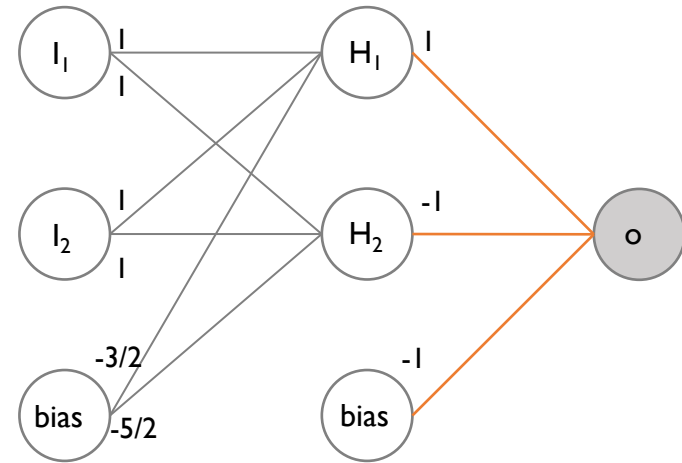
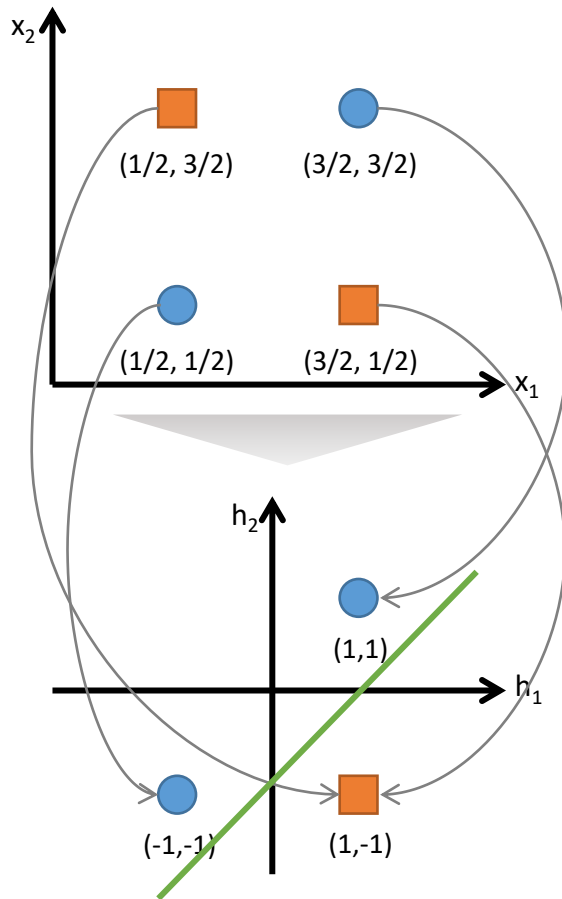
$$h_2 = g(a_2) = \begin{cases} 1 & \text{if } a_2 \geq 0 \\ -1 & \text{if } a_2 < 0 \end{cases}$$



x_1	x_2	a_2	h_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

Multi-Layer Perceptron (MLP)

- XOR problem revisited (cont')



	x_1	x_2	a_1	h_1	a_2	h_2	o	y
●	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

$$o = h_1 + h_2 - 1 \quad y = g(o) = \begin{cases} 1 & \text{if } o \geq 0 \\ -1 & \text{if } o < 0 \end{cases}$$

Multi-Layer Perceptron (MLP)

- General formulation

- ✓ The output of the hidden node j (when the activation function is sigmoid):

$$a_j = \sum_{i=0}^d w_{ji}^{(1)} x_i, \quad h_j = g(a_j) = \frac{1}{1 + \exp(-a_j)}$$

- ✓ The output of the output node (when the activation function is linear):

$$y = \sum_{j=0}^p w_j^{(2)} h_j$$

- ✓ The final outcome of the neural network:

$$y = \sum_{j=0}^p w_j^{(2)} \cdot g\left(\sum_{i=0}^d w_{ji}^{(1)} x_i\right)$$

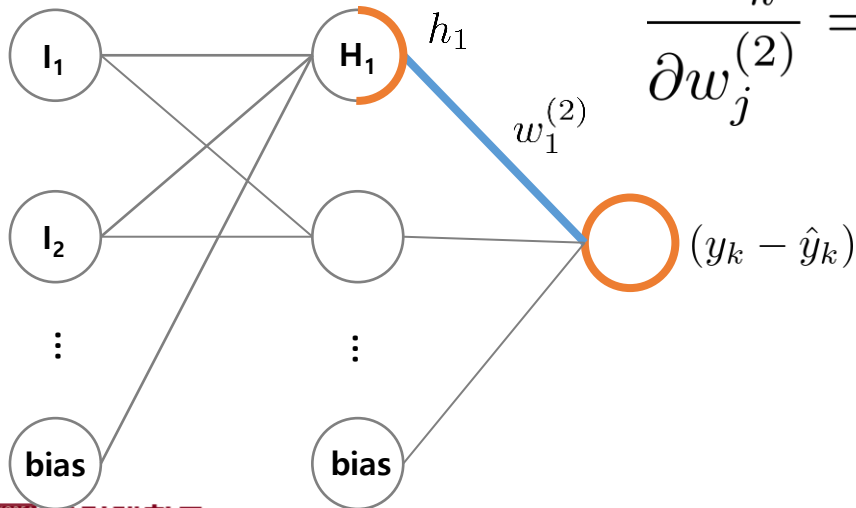
Multi-Layer Perceptron (MLP)

- Error Back-Propagation

✓ The loss of k^{th} observation

$$L_k = \frac{1}{2}(y_k - \hat{y}_k)^2 \quad , \quad y_k = \sum_{j=0}^p w_j^{(2)} \cdot g\left(\sum_{i=0}^d w_{ji}^{(1)} \mathbf{x}_{ki}\right)$$

✓ The weight $w_j^{(2)}$ which connects the j th hidden node and the output node will be updated by



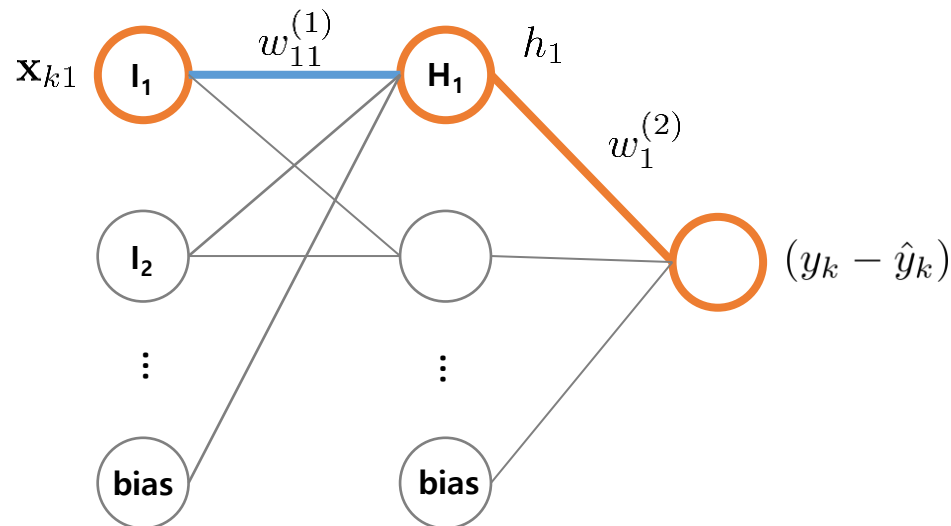
$$\frac{\partial L_k}{\partial w_j^{(2)}} = \frac{\partial L_k}{\partial y_k} \cdot \frac{\partial y_k}{\partial w_j^{(2)}} = (y_k - \hat{y}_k) \cdot h_j$$

Multi-Layer Perceptron (MLP)

- Error Back-Propagation

✓ The weight $w_{ji}^{(1)}$ which connects the i^{th} input node and j^{th} hidden node

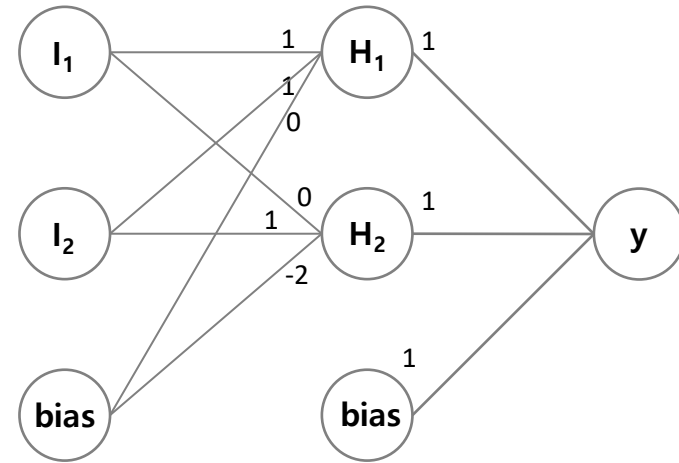
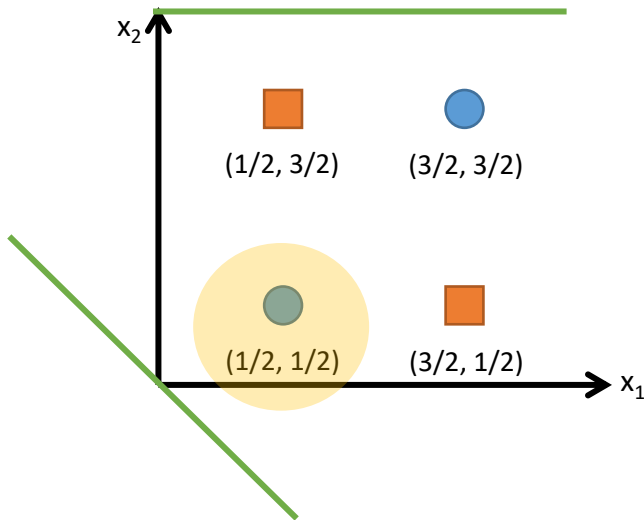
$$\begin{aligned}\frac{\partial L_k}{\partial w_{ji}^{(1)}} &= \frac{\partial L_k}{\partial y_k} \cdot \frac{\partial y_k}{\partial w_j^{(2)}} \cdot \frac{\partial h_j}{\partial a_j} \cdot \frac{\partial a_j}{\partial w_{ji}^{(1)}} \\ &= (y_k - \hat{y}_k) \cdot w_j^{(2)} \cdot a_j \cdot (1 - a_j) \cdot \mathbf{x}_{ki}\end{aligned}$$



MLP: Training

- Error Back-Propagation: Example

✓ Initial weight: Random generation



$$a_1 = \sum w_{1i}^{(1)} x_i = 1 \times 0.5 + 1 \times 0.5 + 0 \times 1 = 1$$

$$h_1 = \frac{1}{1 + \exp(1)} = 0.269$$

$$a_2 = \sum w_{2i}^{(1)} x_i = 0 \times 0.5 + 1 \times 0.5 + (-2) \times 1 = -1.5$$

$$h_2 = \frac{1}{1 + \exp(-1.5)} = 0.818$$

$$\hat{y} = \sum w_j^{(2)} h_j = 1 \times 0.269 + 1 \times 0.818 + 1 \times 1 = 2.087$$

MLP: Training

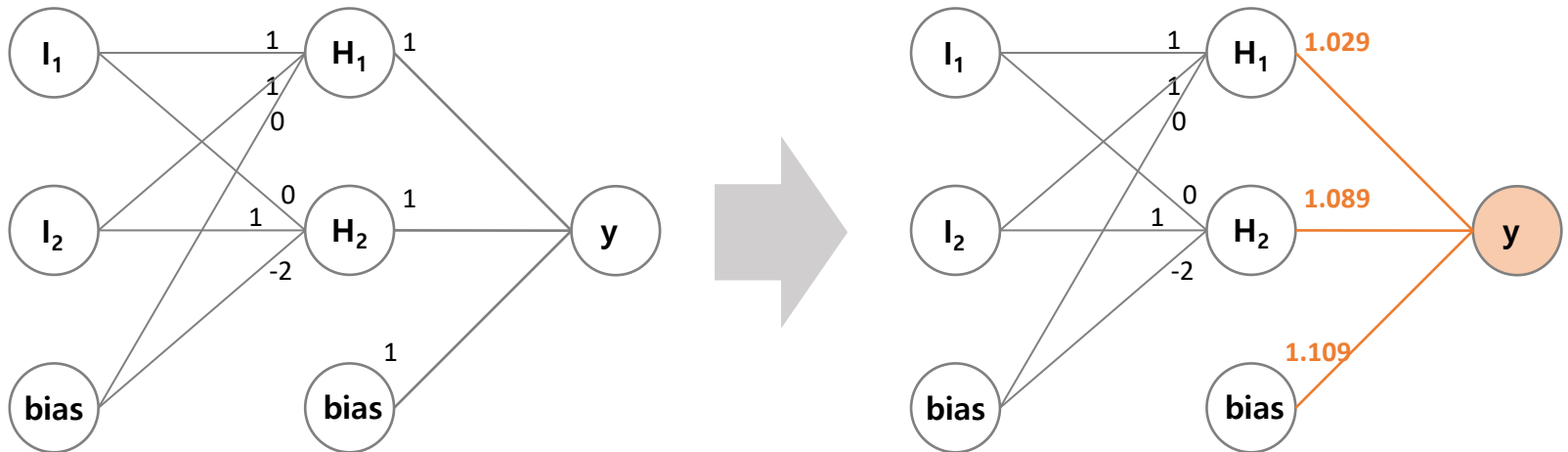
- Error Back-Propagation: Example

✓ Update the weights between the output and the hidden nodes

$$w_1^{(2)}(new) = w_1^{(2)}(old) - \eta \times (y - \hat{y}) \times h_1 = 1 - 0.1 \times (1 - 2.087) \times 0.269 = 1.029$$

$$w_2^{(2)}(new) = w_2^{(2)}(old) - \eta \times (y - \hat{y}) \times h_2 = 1 - 0.1 \times (1 - 2.087) \times 0.818 = 1.089$$

$$w_0^{(2)}(new) = w_0^{(2)}(old) - \eta \times (y - \hat{y}) \times b^{(2)} = 1 - 0.1 \times (1 - 2.087) \times 1 = 1.109$$



MLP: Training

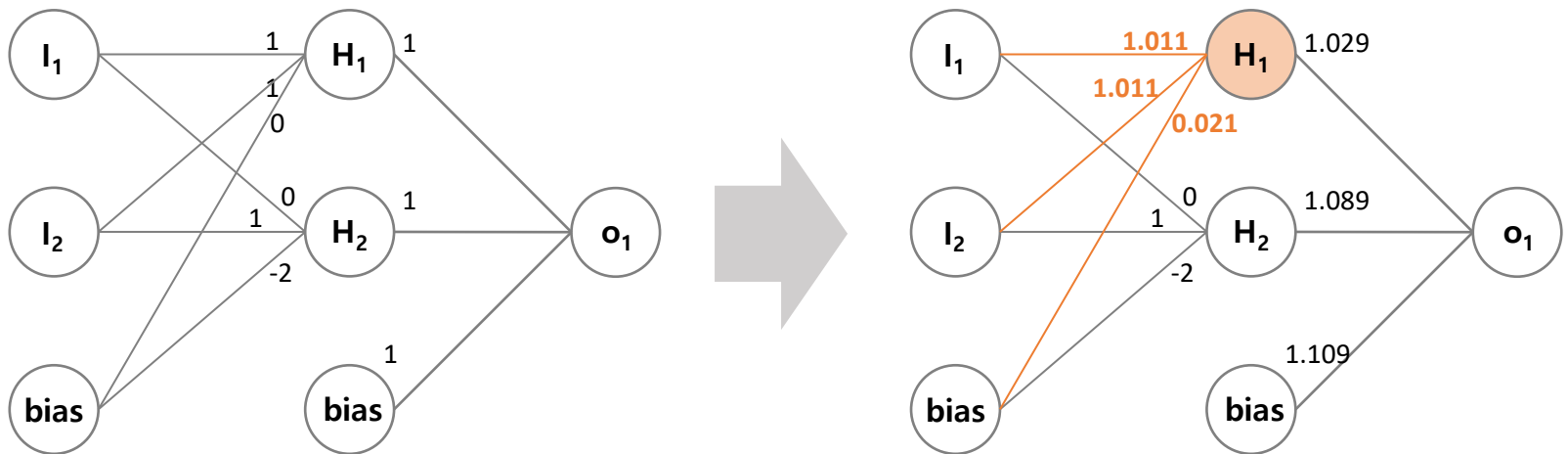
- Error Back-Propagation: Example

✓ Update the weights between the H_1 and the input nodes

$$w_{11}^{(1)}(new) = w_{11}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_1^{(2)} \times h_1 \times (1 - h_1) \times x_1 = 1.011$$

$$w_{12}^{(1)}(new) = w_{12}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_1^{(2)} \times h_1 \times (1 - h_1) \times x_2 = 1.011$$

$$w_{10}^{(1)}(new) = w_{10}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_1^{(2)} \times h_1 \times (1 - h_1) \times b^{(1)} = 0.021$$



MLP: Training

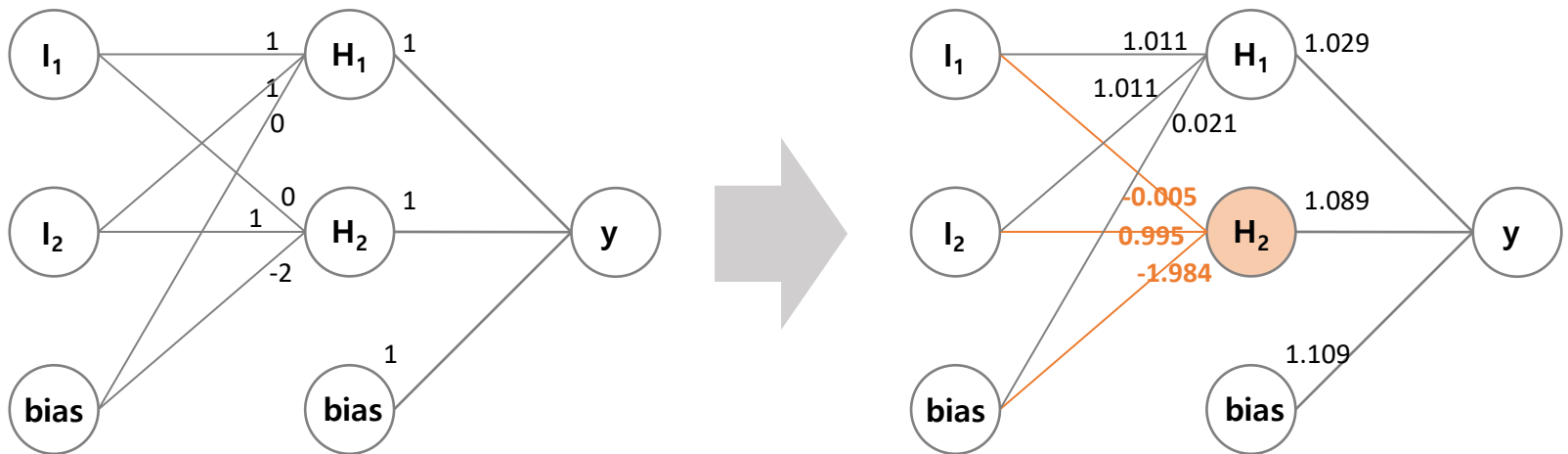
- Error Back-Propagation: Example

✓ Update the weights between the H_1 and the input nodes

$$w_{21}^{(1)}(new) = w_{21}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_2^{(2)} \times h_2 \times (1 - h_2) \times x_1 = -0.005$$

$$w_{22}^{(1)}(new) = w_{22}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_2^{(2)} \times h_2 \times (1 - h_2) \times x_2 = 0.995$$

$$w_{20}^{(1)}(new) = w_{20}^{(1)}(old) - \eta \times (y - \hat{y}) \times w_2^{(2)} \times h_2 \times (1 - h_2) \times b^{(1)} = -1.984$$



MLP: Training

- Goal
 - ✓ Find the weights that yield best predictions
- Features
 - ✓ The process described before is repeated for all records
 - ✓ At each record, compare the prediction to the actual target
 - ✓ Difference is the error for the output node
 - ✓ Error is propagated back and distributed to all the hidden nodes and used to update their weights

MLP: Training

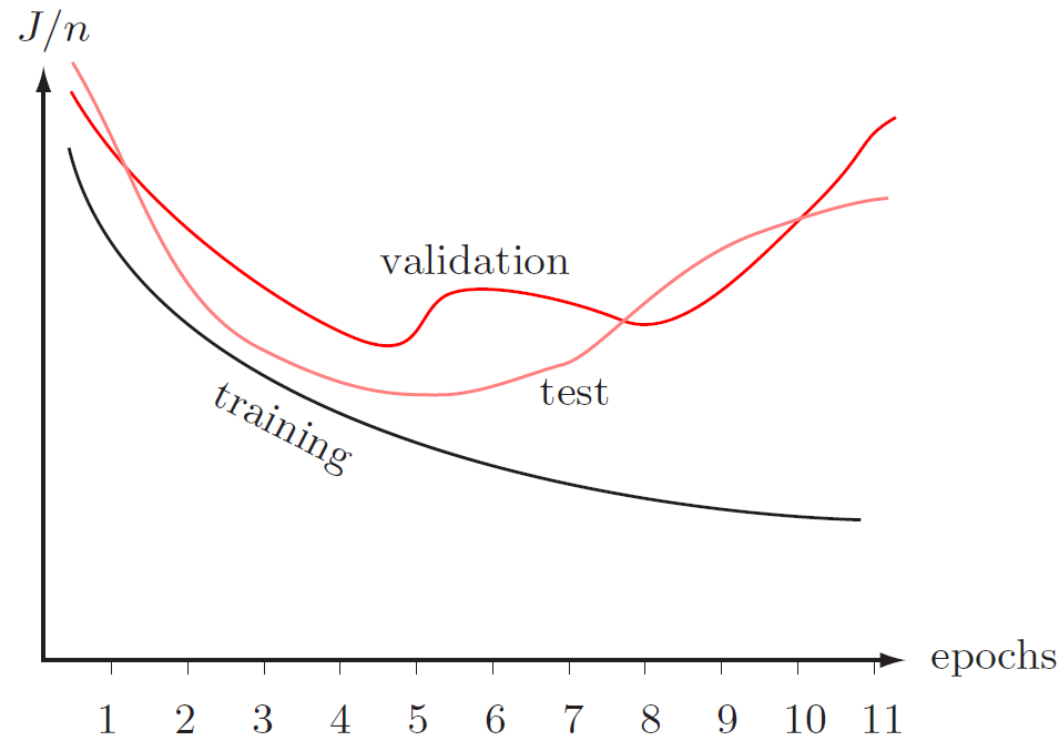
- Why it works
 - ✓ Big errors lead to big changes in weights
 - ✓ Small errors leave weights relatively unchanged
 - ✓ Over thousand of updates, a given weight keeps changing until the error associated with it is negligible
- Common criteria to stop updating
 - ✓ When weights change very little from one epoch to the next
 - ✓ When the misclassification rate reaches a required threshold
 - ✓ When a limit on runs is reached

MLP: Training

- Goal
 - ✓ Find the weights that yield best predictions
- Features
 - ✓ The process described before is repeated for all records
 - ✓ At each record, compare the prediction to the actual target
 - ✓ Difference is the error for the output node
 - ✓ Error is propagated back and distributed to all the hidden nodes and used to update their weights

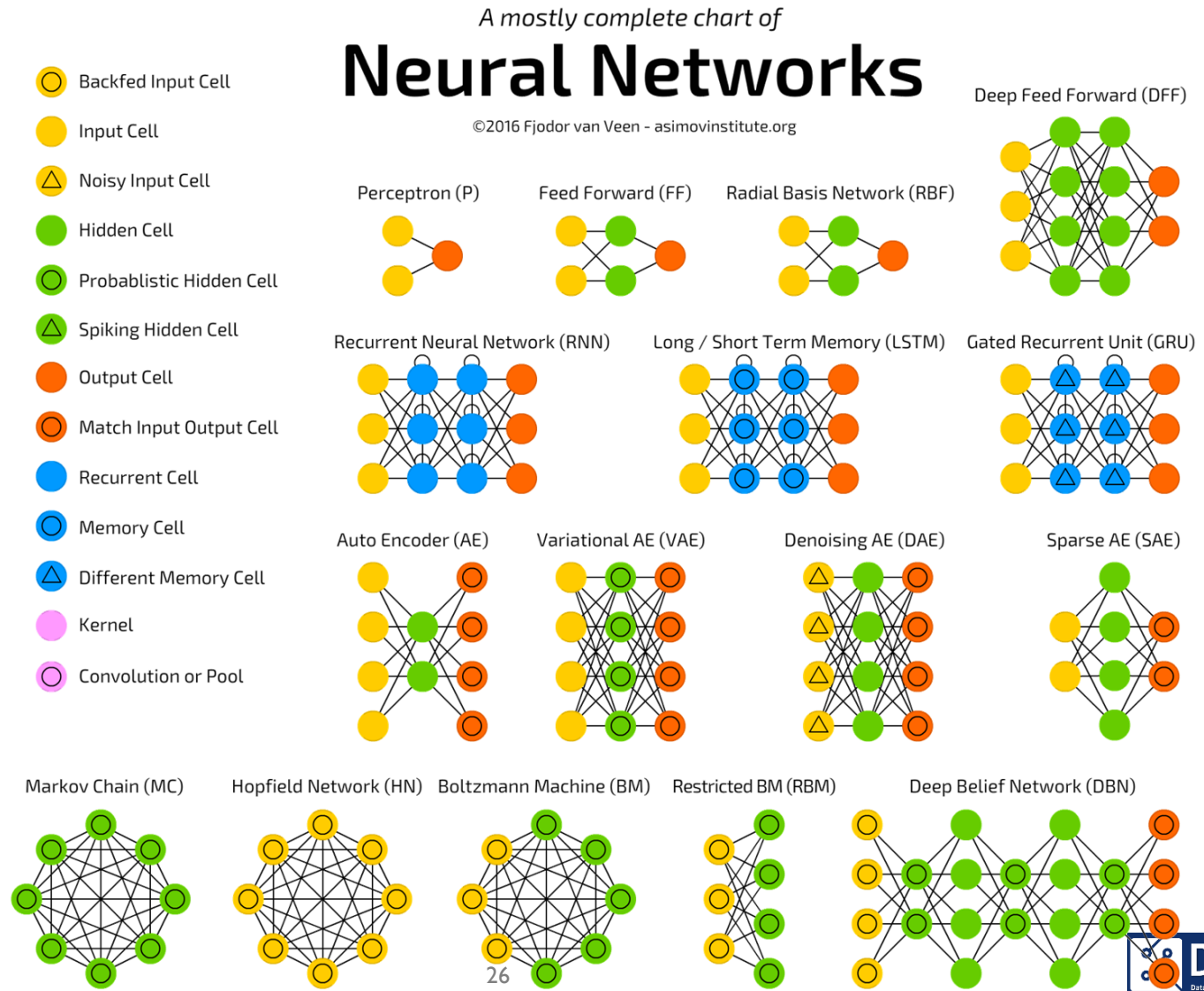
MLP: Training

- With sufficient iterations, neural networks can easily over-fit the data.
- To avoid over-fitting,
 - ✓ Track error in validation data
 - ✓ Limit iterations
 - ✓ Limit complexity of network
 - ✓ N. of hidden layers, nodes, etc.



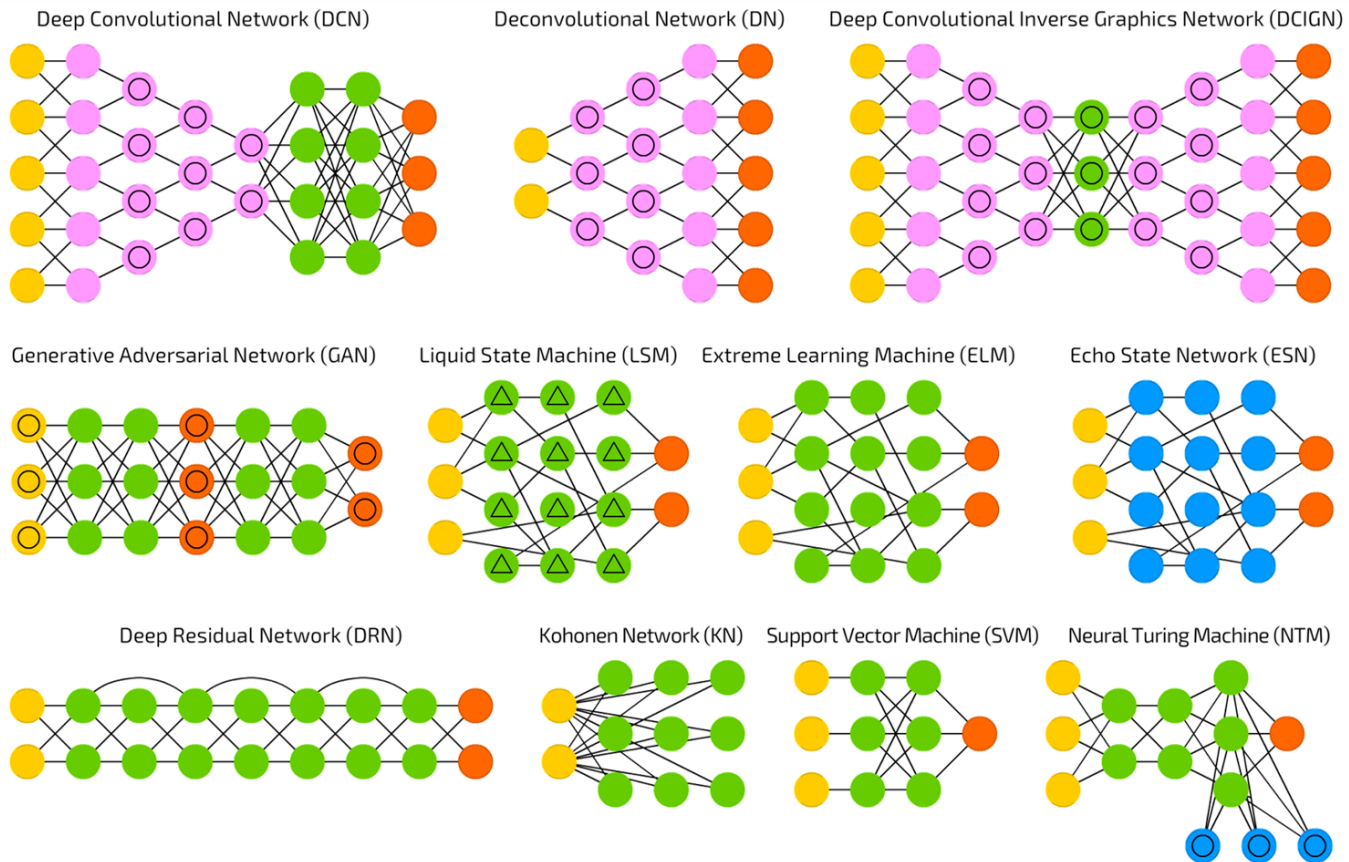
Multi-Layer Perceptron (MLP)

- Various Structure of Artificial Neural Networks



Multi-Layer Perceptron (MLP)

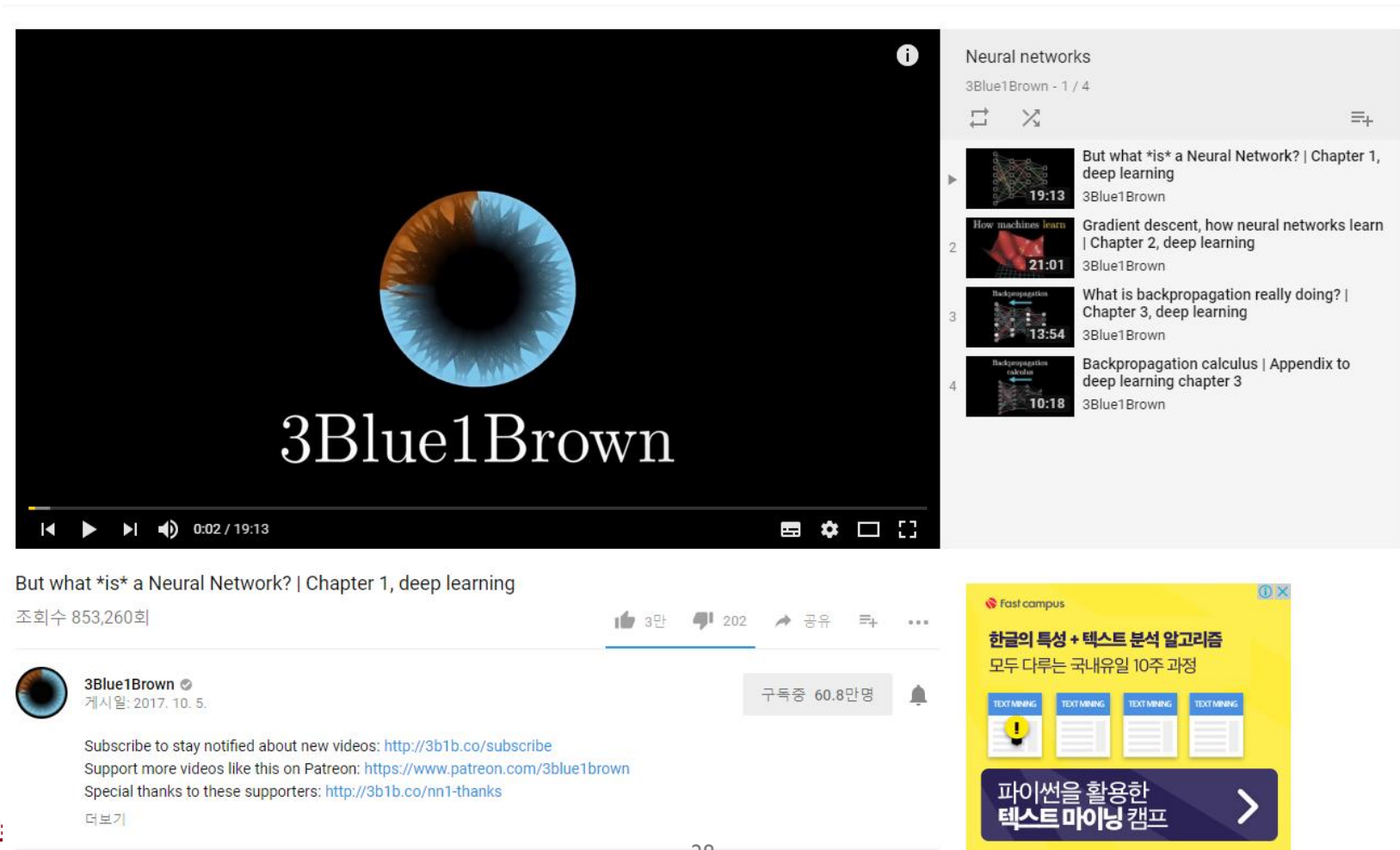
- Various Structure of Artificial Neural Networks



Recommended Video Lectures

- 유튜브 3Blue 1Brown Neural Network 강좌

✓ https://www.youtube.com/channel/UCYO_jab_esuFRV4b17AJtAw



Neural networks

3Blue1Brown - 1 / 4

But what *is* a Neural Network? | Chapter 1, deep learning

3Blue1Brown

19:13

How machines learn

21:01

Backpropagation

13:54

Backpropagation calculus

10:18

But what *is* a Neural Network? | Chapter 1, deep learning

조회수 853,260회

3Blue1Brown

게시일: 2017. 10. 5.

Subscribe to stay notified about new videos: <http://3b1b.co/subscribe>

Support more videos like this on Patreon: <https://www.patreon.com/3blue1brown>

Special thanks to these supporters: <http://3b1b.co/nn1-thanks>

더 보기

구독중 60.8만명

Fast campus

한글의 특성 + 텍스트 분석 알고리즘

모두 다루는 국내 유일 10주 과정

TEXT MINING

TEXT MINING

TEXT MINING

TEXT MINING

파이썬을 활용한 텍스트 마이닝 캠프

SBA

Data Science & Business Analytics

Recommended Video Lectures

- 유튜브 Brandon Rohrer 강좌

✓ <https://www.youtube.com/watch?v=ILsA4nyG7l0&list=PLVZqIMpoM6kbaeySxhdtgQPFEc5nV7Faa&index=2>

YouTube KR 검색

How neural networks work

Brandon Rohrer

0:02 / 24:37

How Deep Neural Networks Work

조회수 565,188회

1만 223 공유

Brandon Rohrer
게시일: 2017. 3. 2.

구독중 2.8만명

A gentle introduction to the principles behind neural networks, including backpropagation. Rated G for general audiences.

더보기

Talks

Brandon Rohrer - 2 / 11

- ▶ How Deep Neural Networks Work
Brandon Rohrer
24:38
- 3 How Convolutional Neural Networks work
Brandon Rohrer
26:14
- 4 How Data Science Works
Brandon Rohrer
49:48
- 5 Deep Learning Demystified
Brandon Rohrer
22:19

Neural Network 3D Simulation

Denis Dmitriev
조회수 9.8만회

Deep Learning Cars

Samuel Arzt
조회수 7만회

A friendly introduction to Deep Learning and Neural Networks

