이번주차: Chain rule → Neural network → Backpropagation.
= gradient descent + chain rule.

Inclass 17: Neural Network

다른수업. 딥러닝. 신경망.

[SCS4049] Machine Learning and Data Science

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

(shallow) neural network → Deep neural network

신경망 (NN)

Neural Networks and Perceptron

Biological neuron

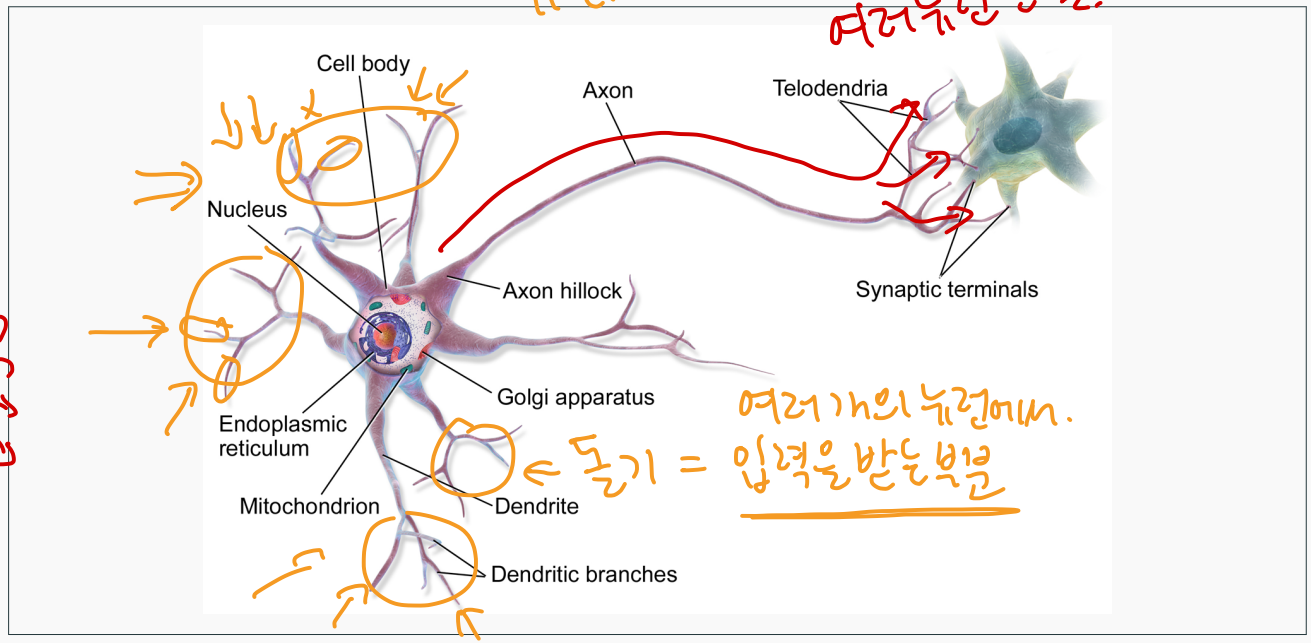


Figure 10-1. Biological neuron[3]

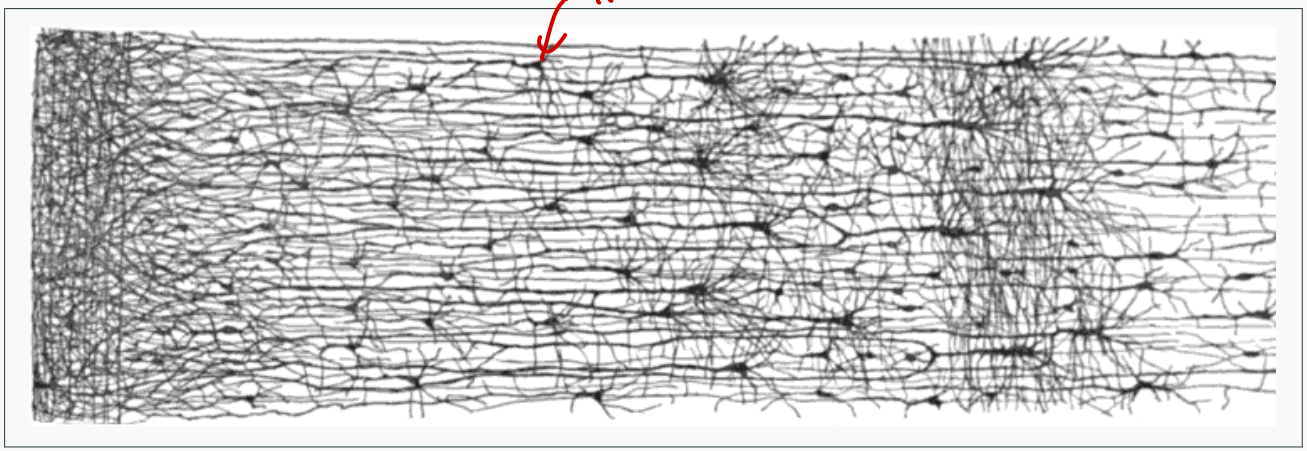


Figure 10-2. Multiple layers in a biological neural network (human cortex)[5]

Computing with the brain

An engineering perspective

- Compact
- Energy efficient (20 watts)
- 85 billion Glial cells (power, cooling, support)
- 86 billion Neurons (soma + wires)
- 69 billion Cerebellum neurons (soma + wires)
- 103~104 Connections (synapses) per neuron
- Volume = mostly wires

General computing machine?

- Slow for mathematical logic, arithmetic, etc
- Very fast for vision, speech, language, social interactions, etc
  - Evolution: vision $\Longrightarrow$ language $\Longrightarrow$ logic

Artificial neural network

Birds inspired us to fly, burdock plants inspired velcro, and countless more inventions were inspired by nature. It seems only logical, then, to look at the brain's architecture for inspiration on how to build an intelligent machine. This is the key idea that sparked artificial neural networks (ANNs). However, although planes were inspired by birds, they don't have to flap their wings. Similarly, ANNs have gradually become quite different from their biological cousins. Some researchers even argue that we should drop the biological analogy altogether (e.g., by saying "units" rather than "neurons"), lest we restrict our creativity to biologically plausible systems.

Artificial neural network

ANNs are at the very core of Deep Learning. They are versatile, powerful, and scalable, making them ideal to tackle large and highly complex Machine Learning tasks, such as classifying billions of images (e.g., Google Images), powering speech recognition services (e.g., Apple's Siri), recommending the best videos to watch to hundreds of millions of users every day (e.g., YouTube), or learning to beat the world champion at the game of Go by playing millions of games against itself (DeepMind's Alpha-Zero).

Threshold logic unit

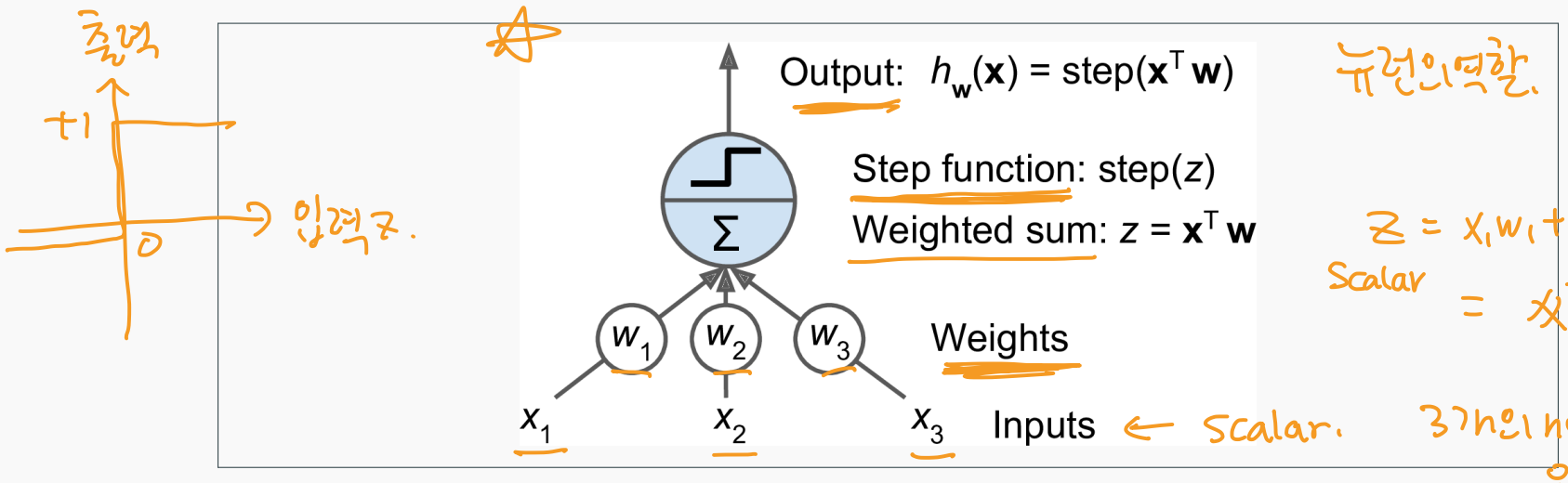


Figure 10-4. Threshold logic unit

Perceptron $\Longrightarrow$ binary classifier

$$z = w_1 x_1 + w_2 x_2 + \cdots + x_m x_m = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \mathbf{w}^T \mathbf{x} \qquad (1)$$

Threshold logic unit (TLU) or linear threshold unit (LTU)

$$h_{\mathbf{w}}(\mathbf{x}) = \text{step}(z) = \text{step}(\mathbf{w}^T \mathbf{x}) \qquad (2)$$

Threshold logic unit

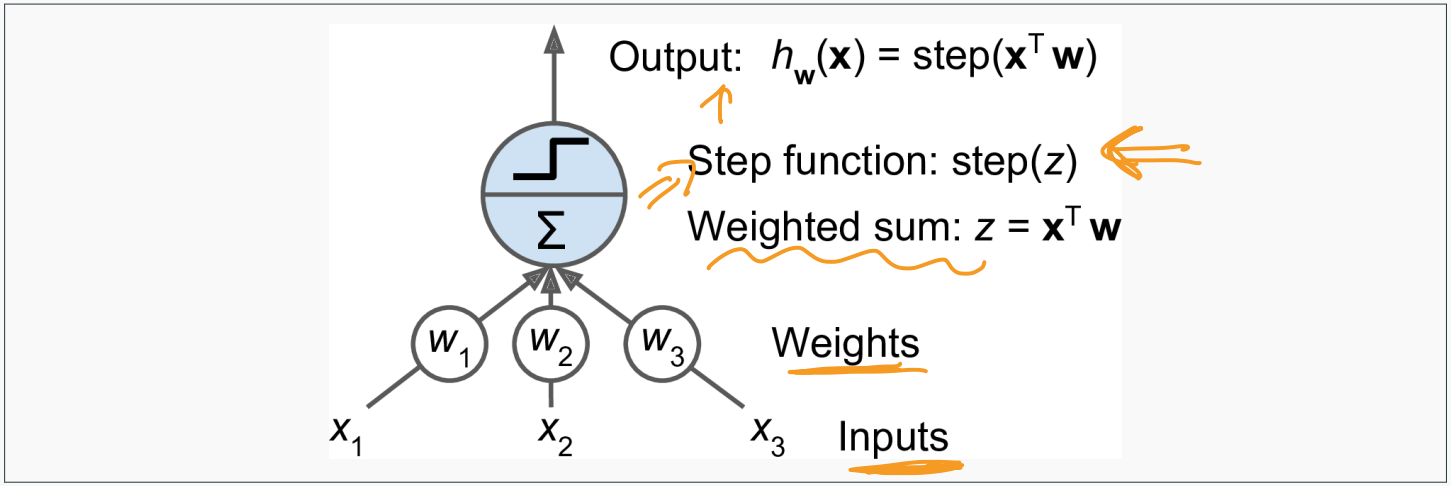
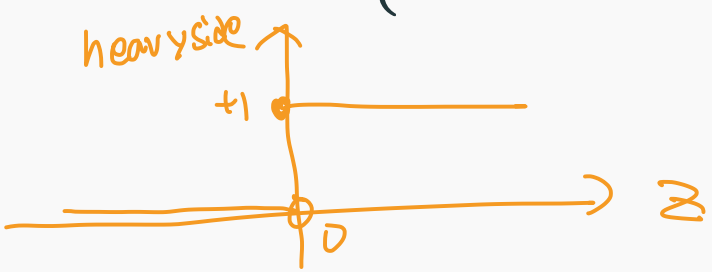


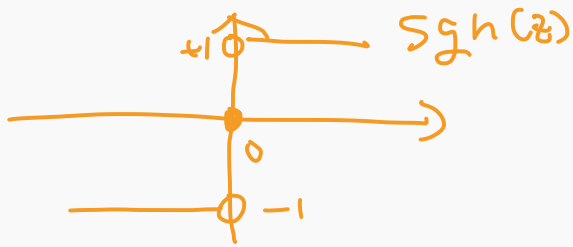
Figure 10-4. Threshold logic unit

Common step functions used in Perceptrons

discrete

$$\text{heavyside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{otherwise} \end{cases} \qquad \text{sign}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{otherwise} \end{cases} \tag{3}$$





Perceptron algorithm

Computing the outputs of a fully connected layer

$$h_{\mathbf{W},\mathbf{b}} = \phi(\mathbf{XW} + \mathbf{b}) \tag{4}$$

Perceptron learning ruel (weight update)

$$w_{i,j} \leftarrow w_{i,j} + \eta(y_j - \hat{y}_j)x_i \tag{5}$$

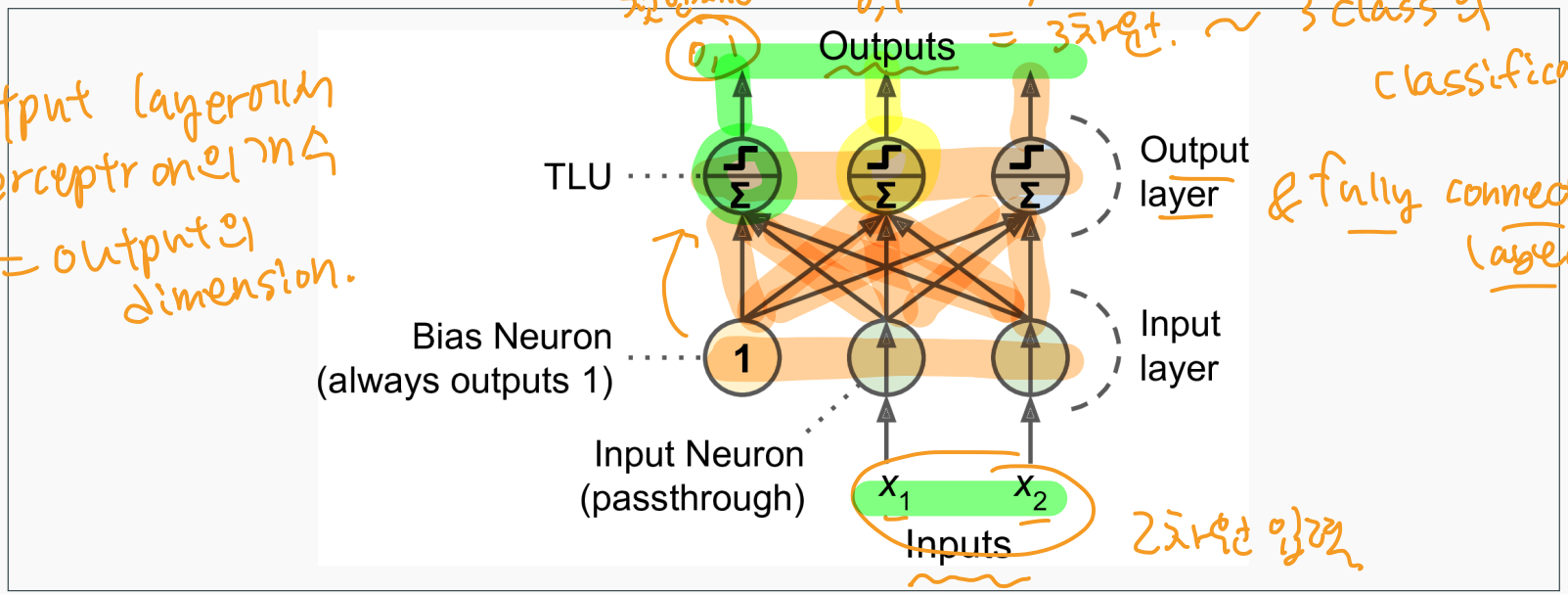


Figure 10-5. Perceptron diagram

Deep Feedforward Networks

Multi-layer perceptron algorithm

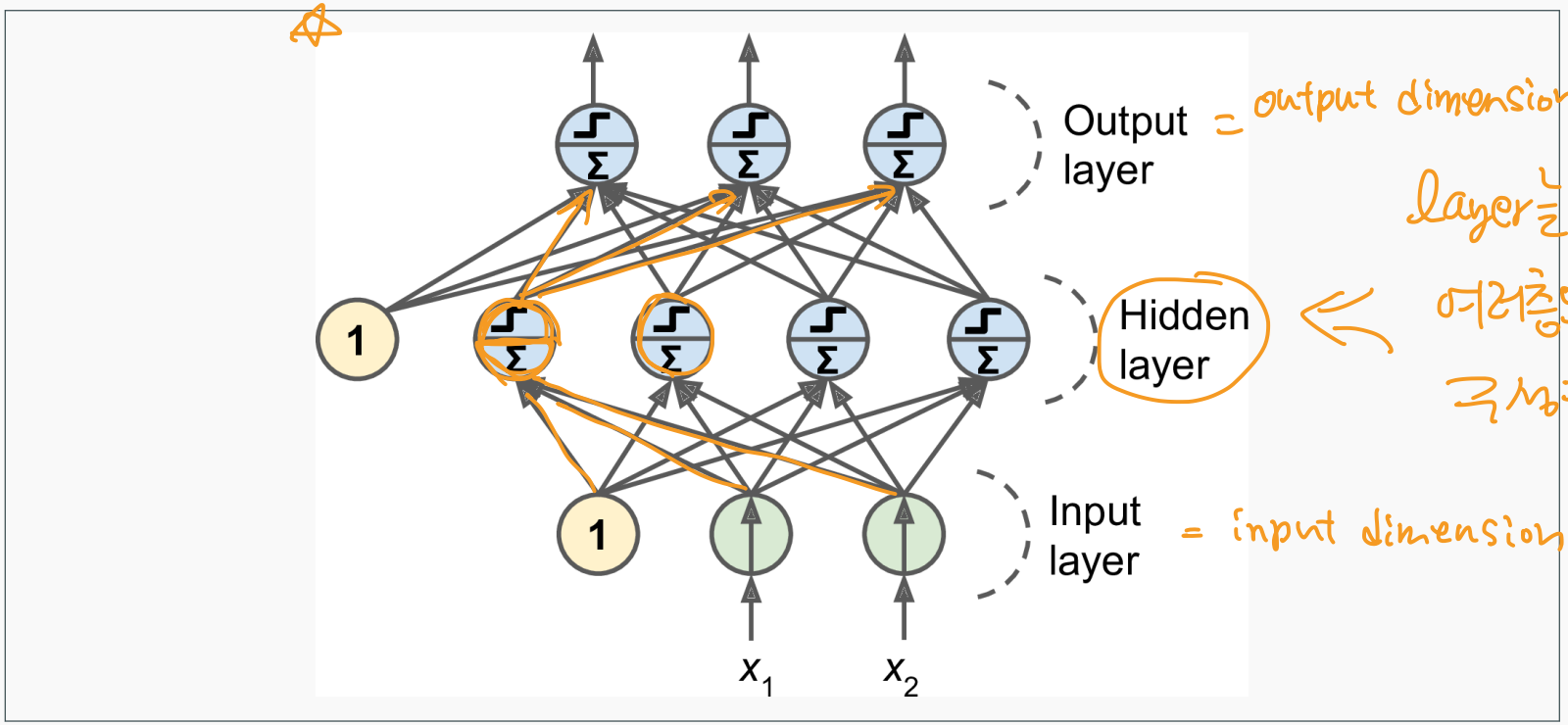


Figure 10-7. Multi-Layer Perceptron

Activation functions and their derivatives

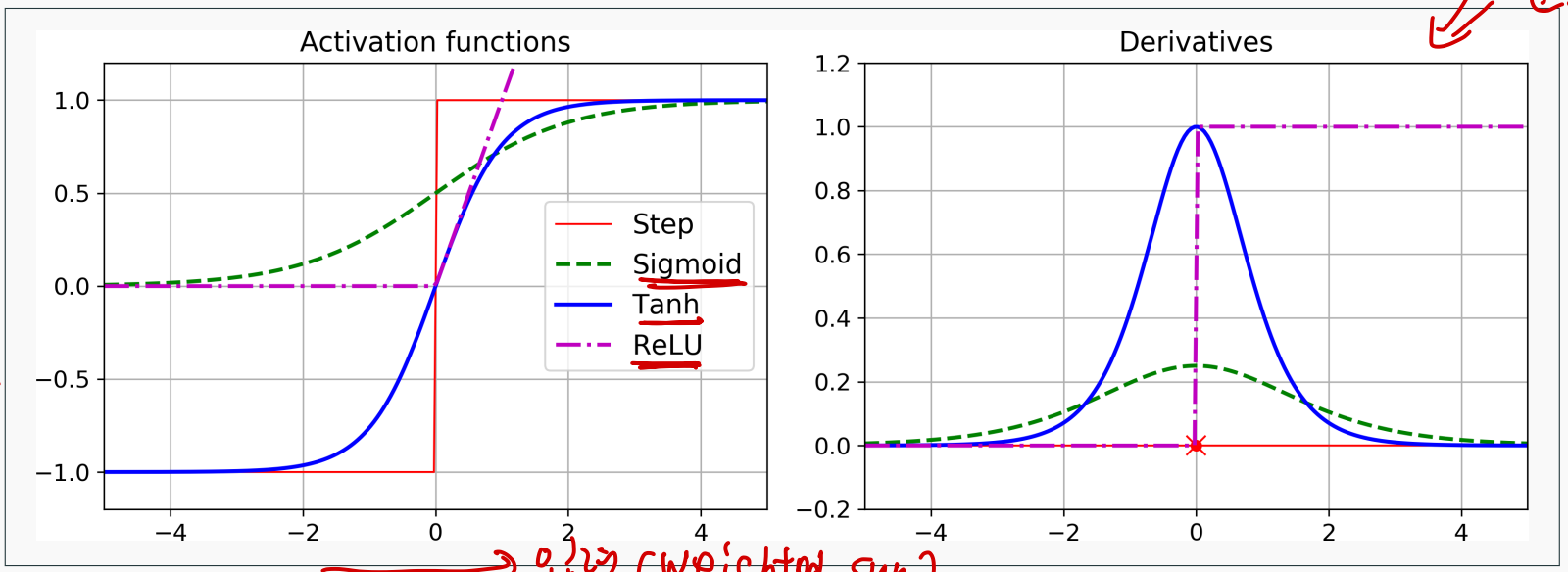


Figure 10-8. Activation functions and their derivatives

A modern MLP (including ReLU and softmax) for classification

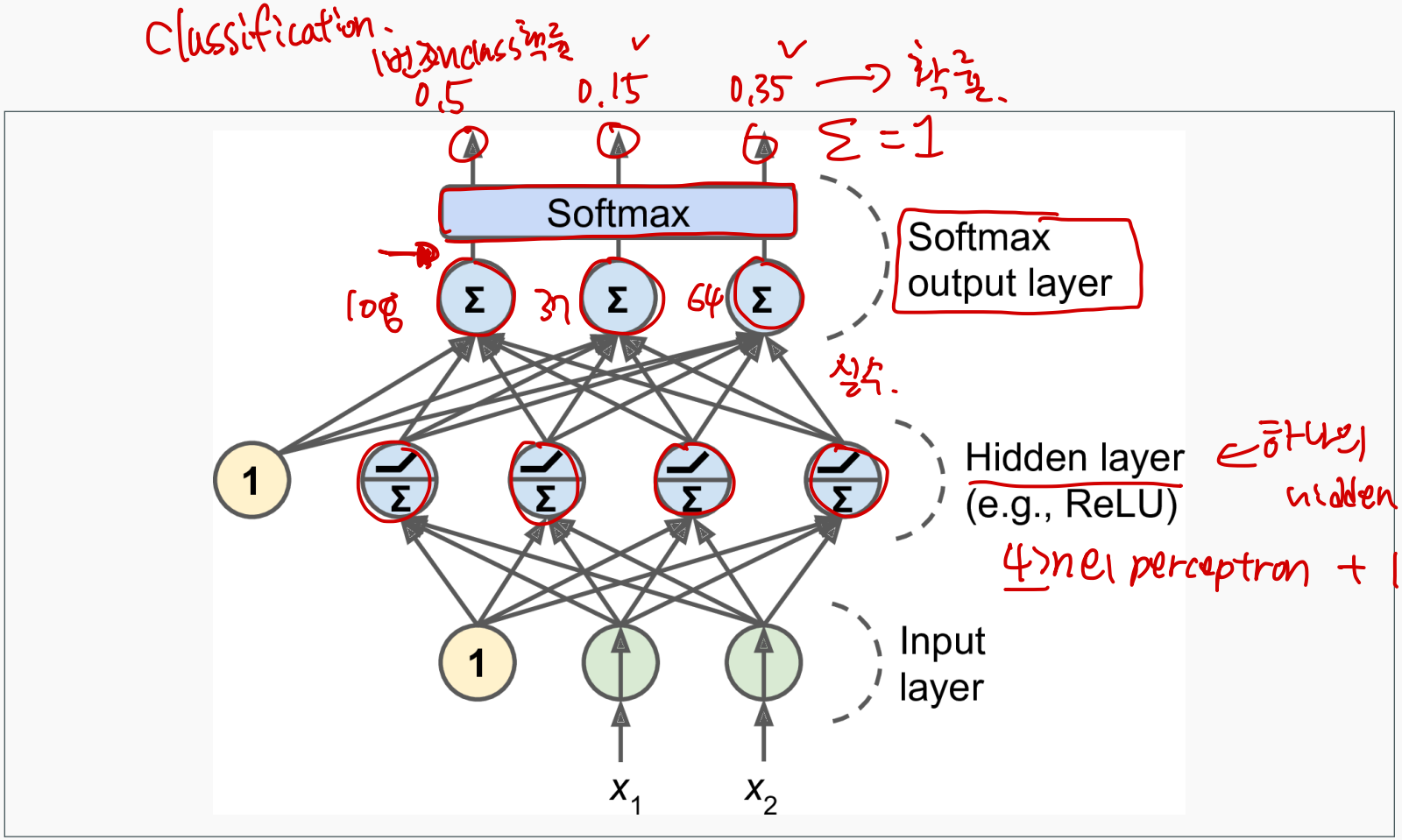


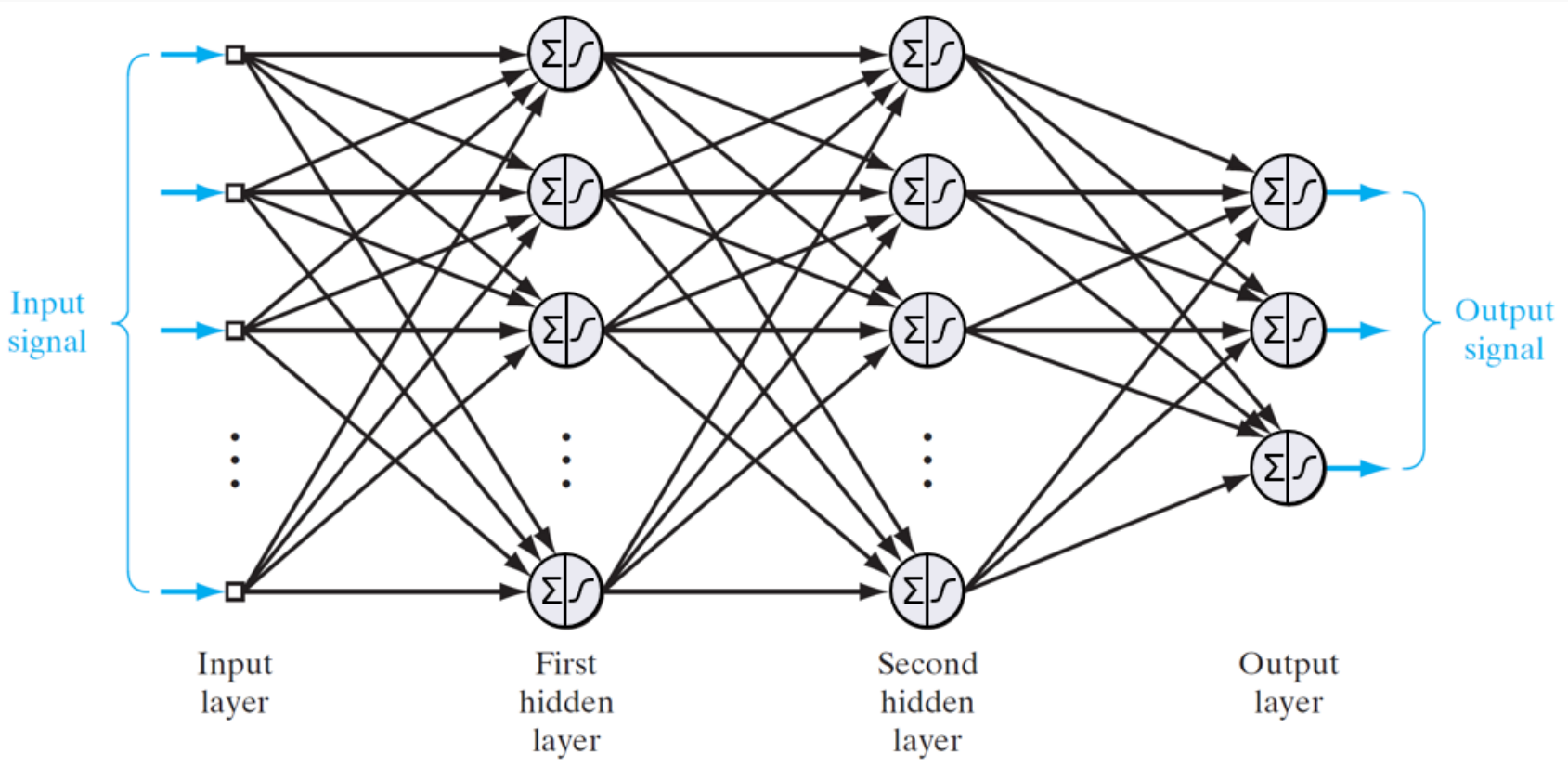
Figure 10-9. A modern MLP (including ReLU and softmax) for classification

Deep feedforward networks

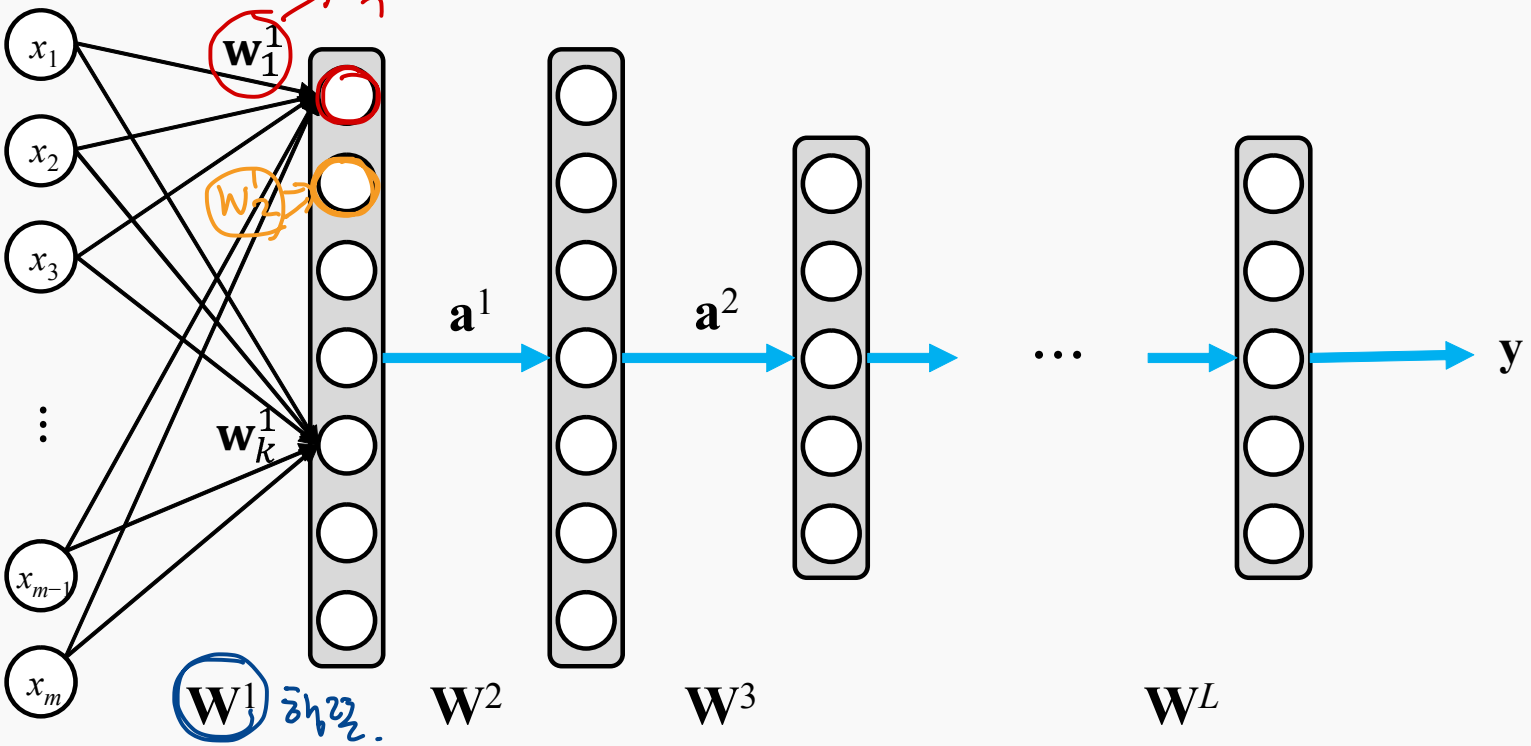
Deep Feedforward Networks, Multi-Layer Perceptron, or Feedforward Neural Networks

- Fully Connected Multi-layer
- Feedforward $\Longrightarrow$ Feedback이 없음
  (Feedback 이 있는 신경망: Recurrent Neural Networks)
- Nonlinear Activation Functions
- Rumelhart (1986) 의 Backpropagation (역전파) 알고리즘 이후 집중적 연구
- Reverse-mode Autodiff 를 사용한 Gradient-based learning 방법

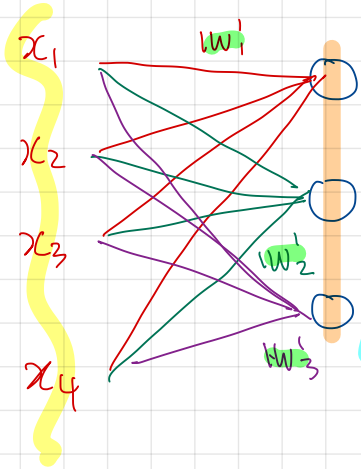
Deep feedforward networks



Goal of deep feedforward networks



13/100



$y_1^1 = \varphi^1(w_1^{1T} x)$ → scalar

$y_2^1 = \varphi^1(w_2^{1T} x)$ → scalar

$y_3^1 = \varphi^1(w_3^{1T} x)$ → scalar.

각 layer의 input

$$\begin{bmatrix} y_1^1 \\ y_2^1 \\ y_3^1 \end{bmatrix} = \varphi^1 \left[W^1 x \right]$$

$$W^1 = \begin{bmatrix} w_1^{1T} \\ w_2^{1T} \\ w_3^{1T} \end{bmatrix}$$

$$\varphi(W^1 \cdot x) = \begin{bmatrix} w_1^{1T} \\ w_2^{1T} \\ w_3^{1T} \end{bmatrix} x.$$

$$= \varphi\left(\begin{bmatrix} w_1^{1T} x \\ w_2^{1T} x \\ w_3^{1T} x \end{bmatrix} \right)$$

Goal of deep feedforward networks

Let $\mathbf{W}^l = \begin{bmatrix} \mathbf{w}_1^l & \mathbf{w}_2^l & \cdots & \mathbf{w}_{m_l}^l \end{bmatrix}^T$ and φ^l be the activation function for layer l, then

$$\mathbf{y}^1 = \varphi^1(\mathbf{W}^1\mathbf{x}) \tag{6}$$
$$\mathbf{y}^2 = \varphi^2(\mathbf{W}^2\mathbf{y}^1) \tag{7}$$
$$= \varphi^2(\mathbf{W}^2\varphi^1(\mathbf{W}^1\mathbf{x})) \tag{8}$$
$$\mathbf{y}^3 = \varphi^3(\mathbf{W}^3\mathbf{y}^2) \tag{9}$$
$$= \varphi^3(\mathbf{W}^3\varphi^2(\mathbf{W}^2\mathbf{y}^1)) \tag{10}$$
$$= \varphi^3(\mathbf{W}^3\varphi^2(\mathbf{W}^2\varphi^1(\mathbf{W}^1\mathbf{x}))) \tag{11}$$
$$\cdots$$
$$\mathbf{y} = \varphi^L(\mathbf{W}^L\varphi^{L-1}(\mathbf{W}^{L-1}\varphi^{L-2}(\cdots\varphi^2(\mathbf{W}^2\varphi^1(\mathbf{W}^1\mathbf{x}))\cdots))) \tag{12}$$

Very much complex nonlinear function

$\Longrightarrow$ require universal function approximator

Nonlinear activation functions

Logistic sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{13}$$

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) \tag{14}$$

Hyperbolic tangent function

$$\varphi(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{15}$$

$$\varphi'(z) = 2\varphi(2z) - 1 \tag{16}$$

Rectified linear unit function

$$\mathrm{ReLU}(z) = \max(0, z) \tag{17}$$

$$\mathrm{ReLU}'(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{18}$$

Nonlinear activation functions

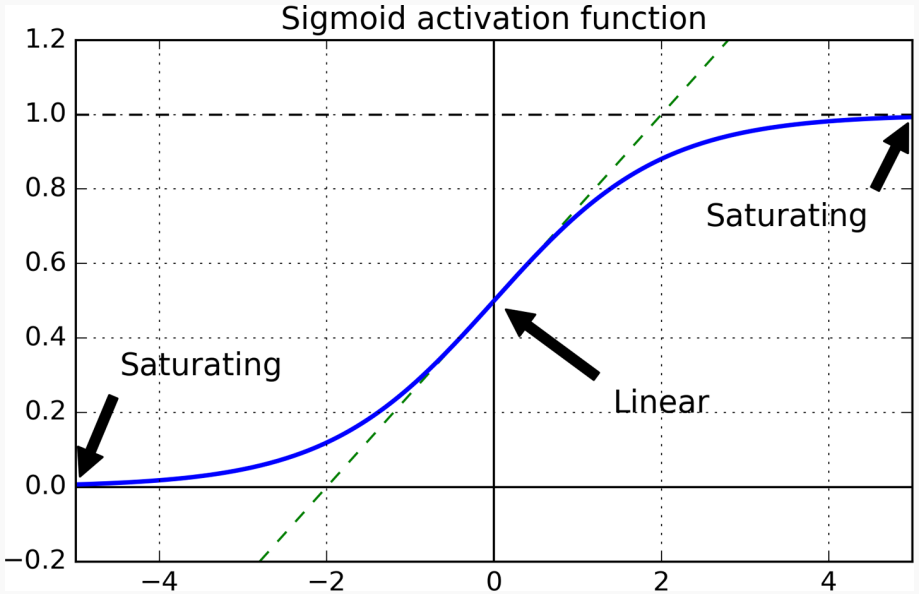


Figure 11-1. Logistic activation function saturation

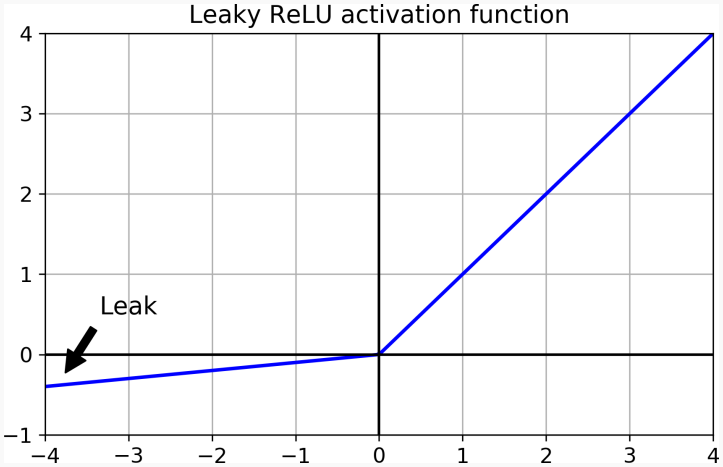


Figure 11-2. Leaky ReLU

Appendix

Reference and further reading

- “Chap 10 | Introduction to Artificial Neural Network with Keras” and “ Chap 11 | Training Deep Neural Networks” of A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow