2024-2

Deep Learning Programming

Lecture 3.1: Neural network classifier

Sangryul Jeon
School of Computer Science and Engineering
srjeonn@pusan.ac.kr

1.1 Linear Model

이미지 x

$$f(x,W) = Wx + b$$

Linear Model

가중치 혹은 파라미터 w

편향

1.1 Linear Model

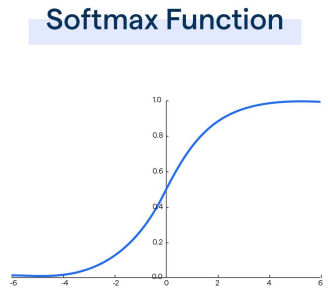
이미지 x

$$f(x, W) = Wx + b$$

Linear Model

$$p(y = c_i|x) = \frac{e^{s_i}}{\sum_j e^{s_j}}$$

Softmax Loss



Softmax 함수 식

1.1 Linear Model

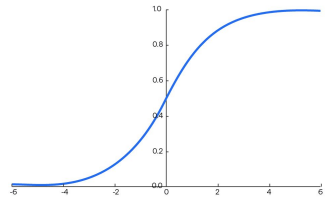
이미지 **x**

$$f(x,W) = Wx + b$$

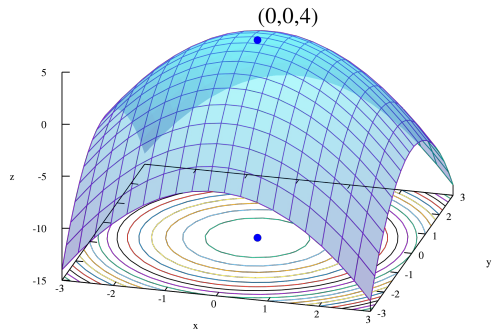
Linear Model

Softmax Function

$$p(y = c_i|x) = \frac{e^{s_i}}{\sum_j e^{s_j}}$$



Softmax Loss



https://en.wikipedia.org/wiki/File:Max_paraboloid.svg

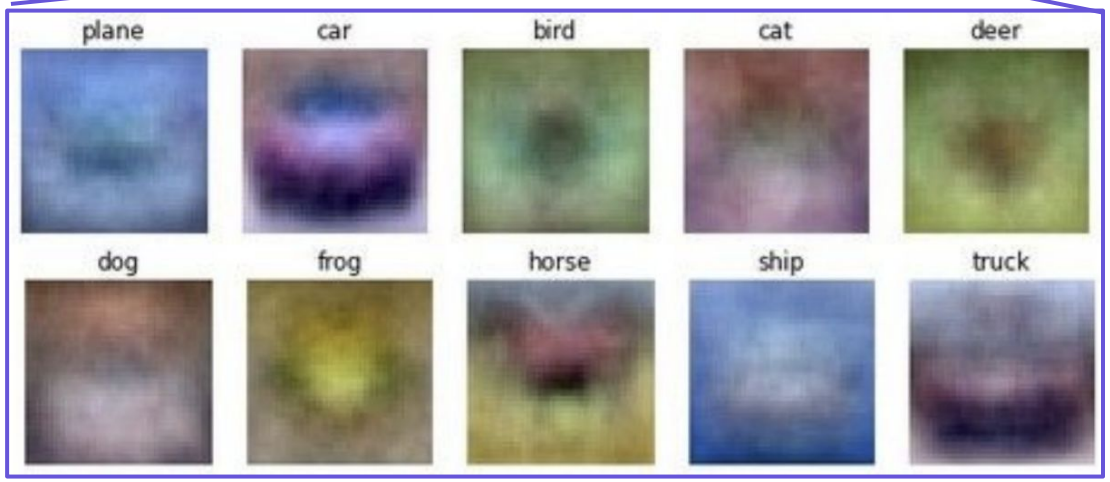
매개 변수들은 (Stochastic) Gradient Descent에 의해 조정됩니다

1.2 Issues with Linear Classifiers

Linear Classifier은 강력하지 않습니다. 왜냐하면

- 시각적으로) 각 클래스 당 하나의 template(카테고리)만 학습할 수 있습니다.

$$f(x) = Wx$$



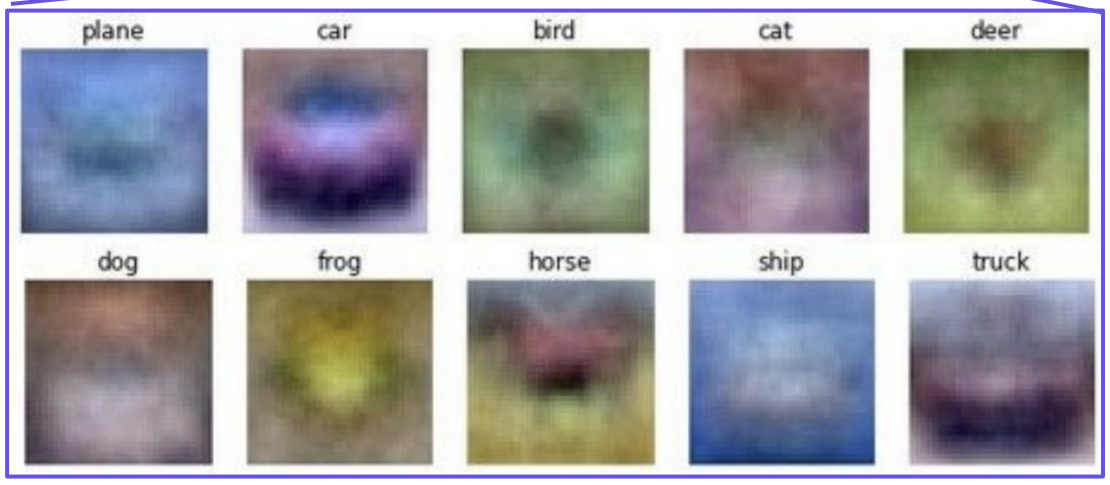
https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture2.pdf

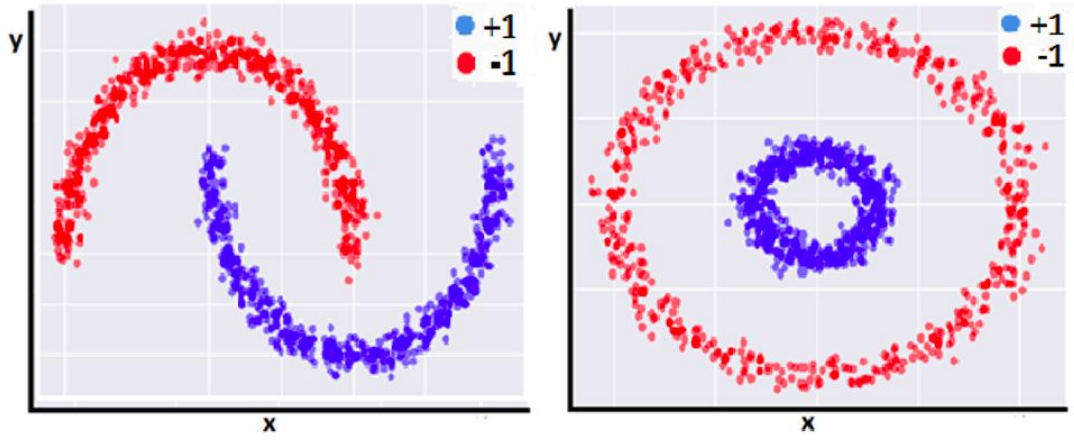
1.2 Issues with Linear Classifiers

Linear Classifier은 강력하지 않습니다. 왜냐하면

- 시각적으로) 각 클래스 당 하나의 template(카테고리)만 학습할 수 있습니다.
- 기하학적으로) 직선 형태의 decision boundary들만 그릴 수 있습니다.
    - 따라서 복잡한 관계로 이루어진 두 개의 클래스들은 완벽하게 분리할 수 없습니다.

$$f(x) = Wx$$

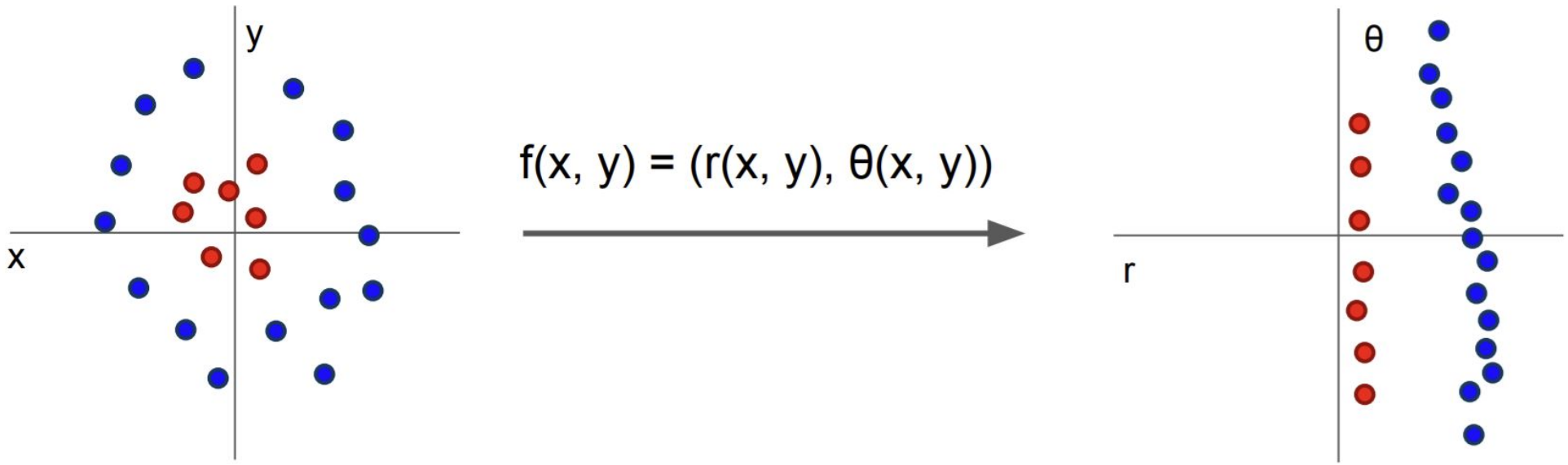




https://medium.com/@sachinkun21/using-a-linear-model-to-deal-with-nonlinear-dataset-c6ed0f7f3f51

1.3 Features

- 입출력 관계를 linear classifier로 직접 연결(mapping)하는 대신 입력을 표현하기 위해 몇 가지의 특징들을 추출할 수 있습니다.
- 만약 feature space에서 입력이 선형 분류가 가능하다면 linear classifier이 잘 작동할 수 있습니다!



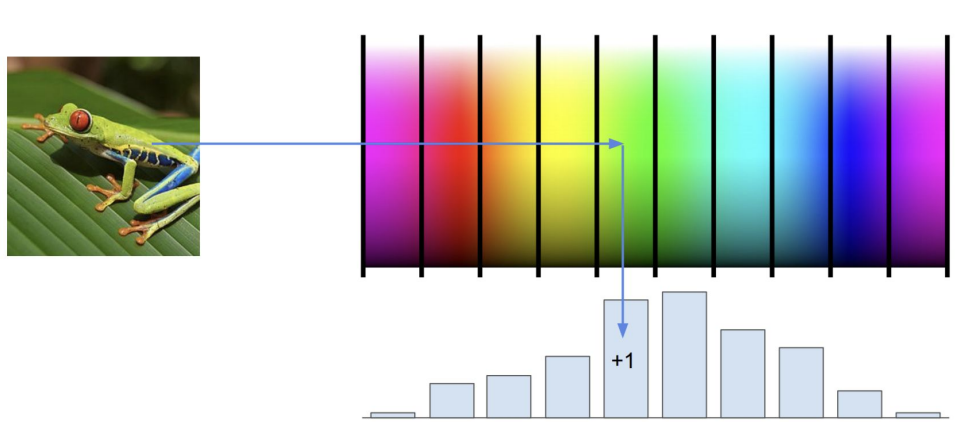
Original Space

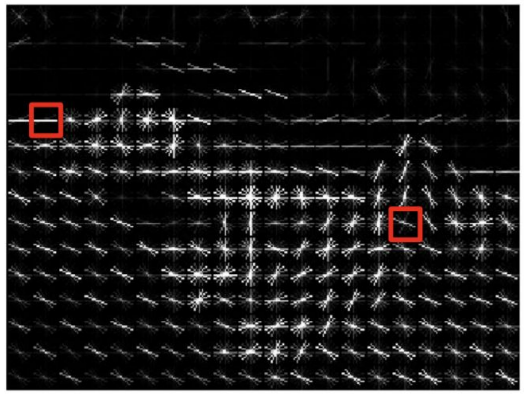
Feature Space

https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf

1.4 Image Features

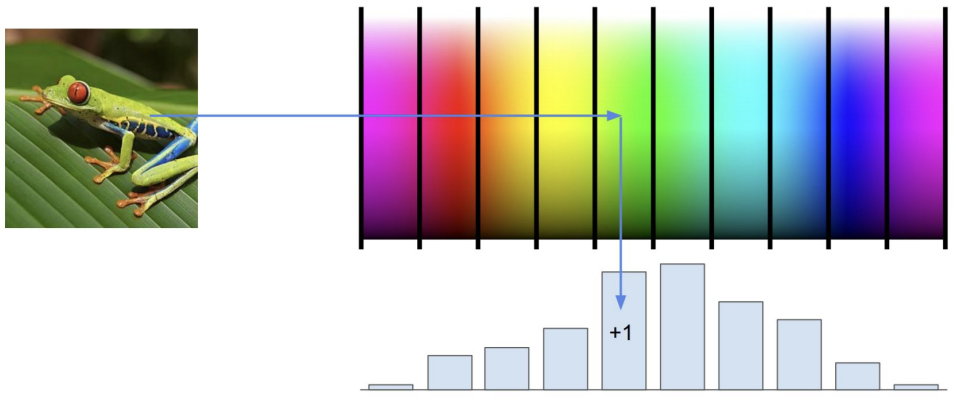
- 실제 이미지의 특징을 표현하는 방법은 복잡합니다
- 직접 입출력(pixel-class) 관계를 mapping하는 대신 몇가지 특징을 추출하여 입력(image)를 표현합니다.
- 예시
  - Color histogram
  - Histogram of oriented gradients (HoG)
  - Bag of words (BoW) with a pre-defined dictionary (codebook)



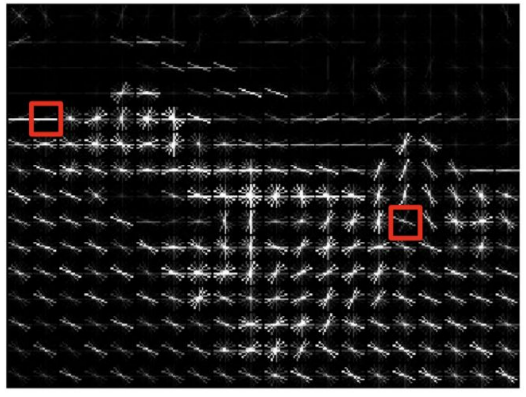


1.4 Image Features

- 예시
  - Color histogram
  - Histogram of oriented gradients (HoG)
  - Bag of words (BoW) with a pre-defined dictionary (codebook)



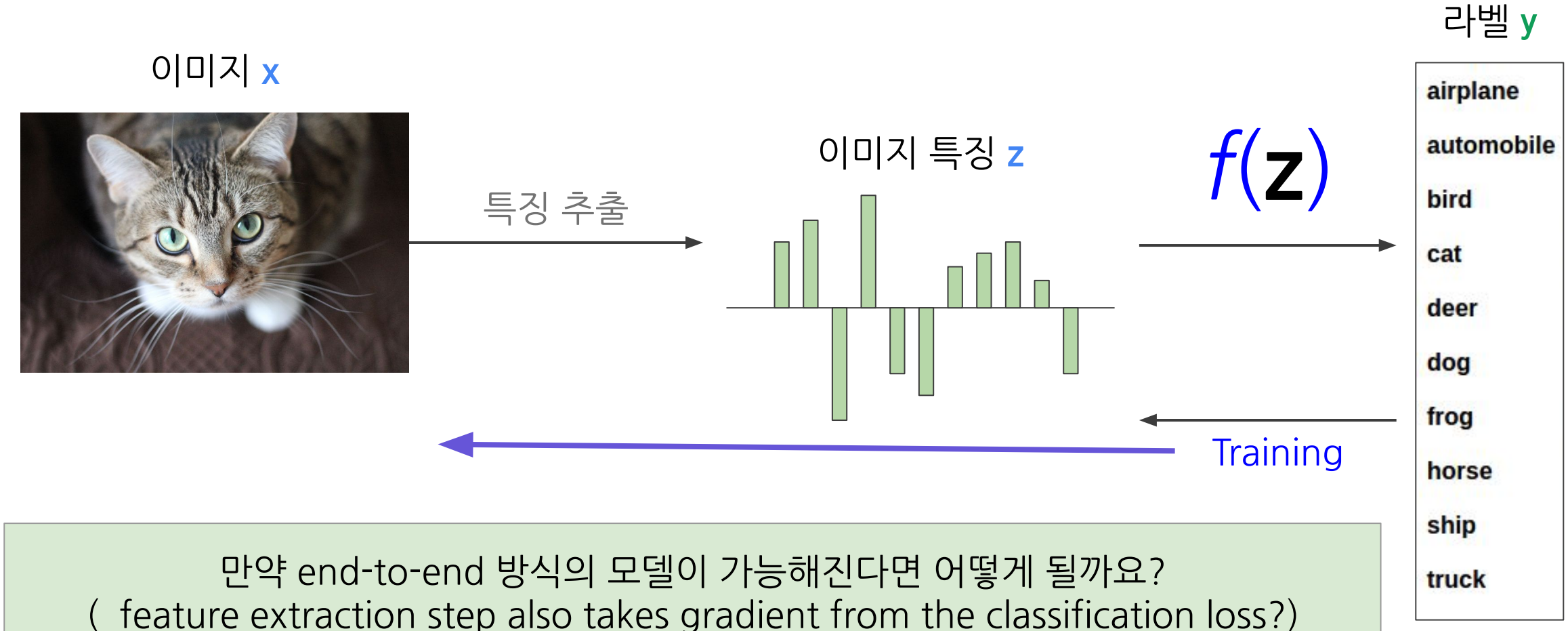
색상 정보만



패치(pixel)당 그래디언트 방향

Clustered codebook

1.5 Image Classifier with pre-extracted Features

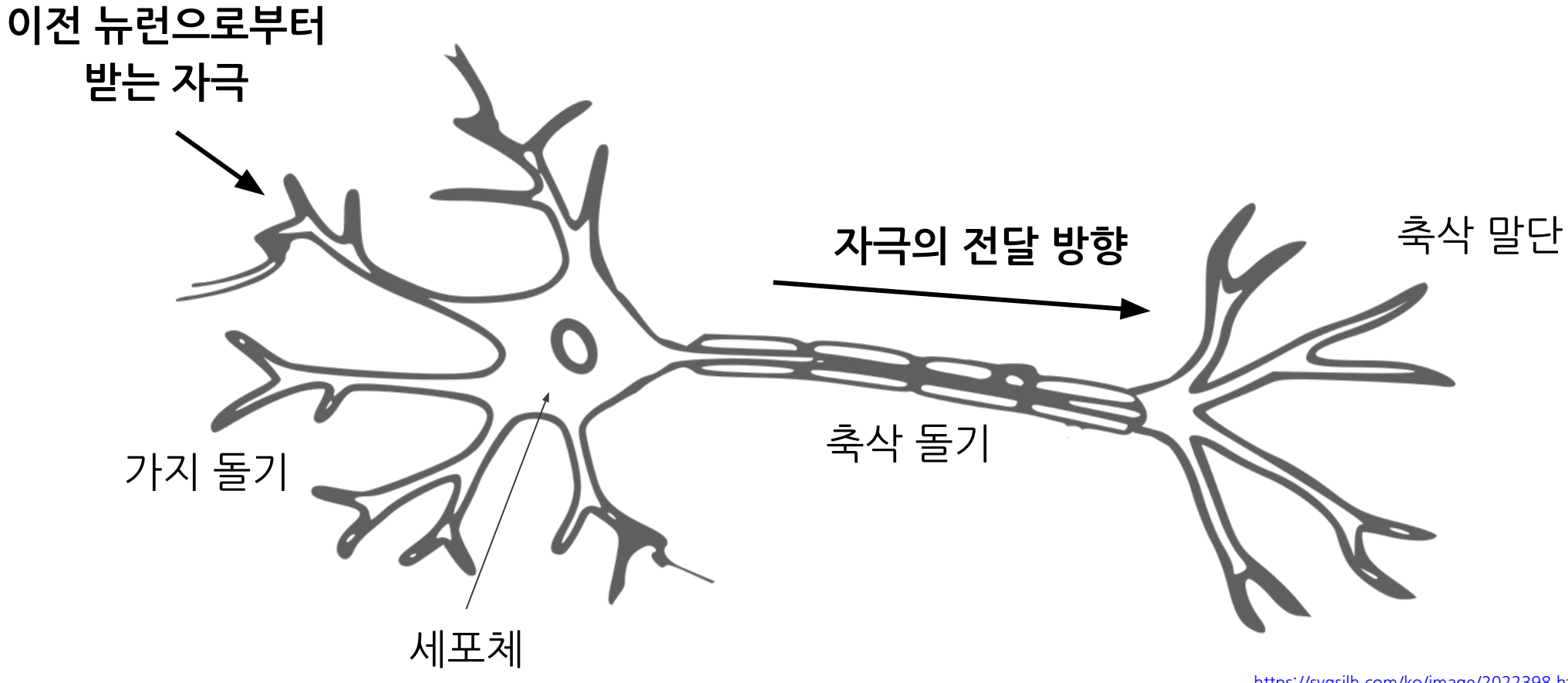


만약 end-to-end 방식의 모델이 가능해진다면 어떻게 될까요?
(feature extraction step also takes gradient from the classification loss?)

https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf

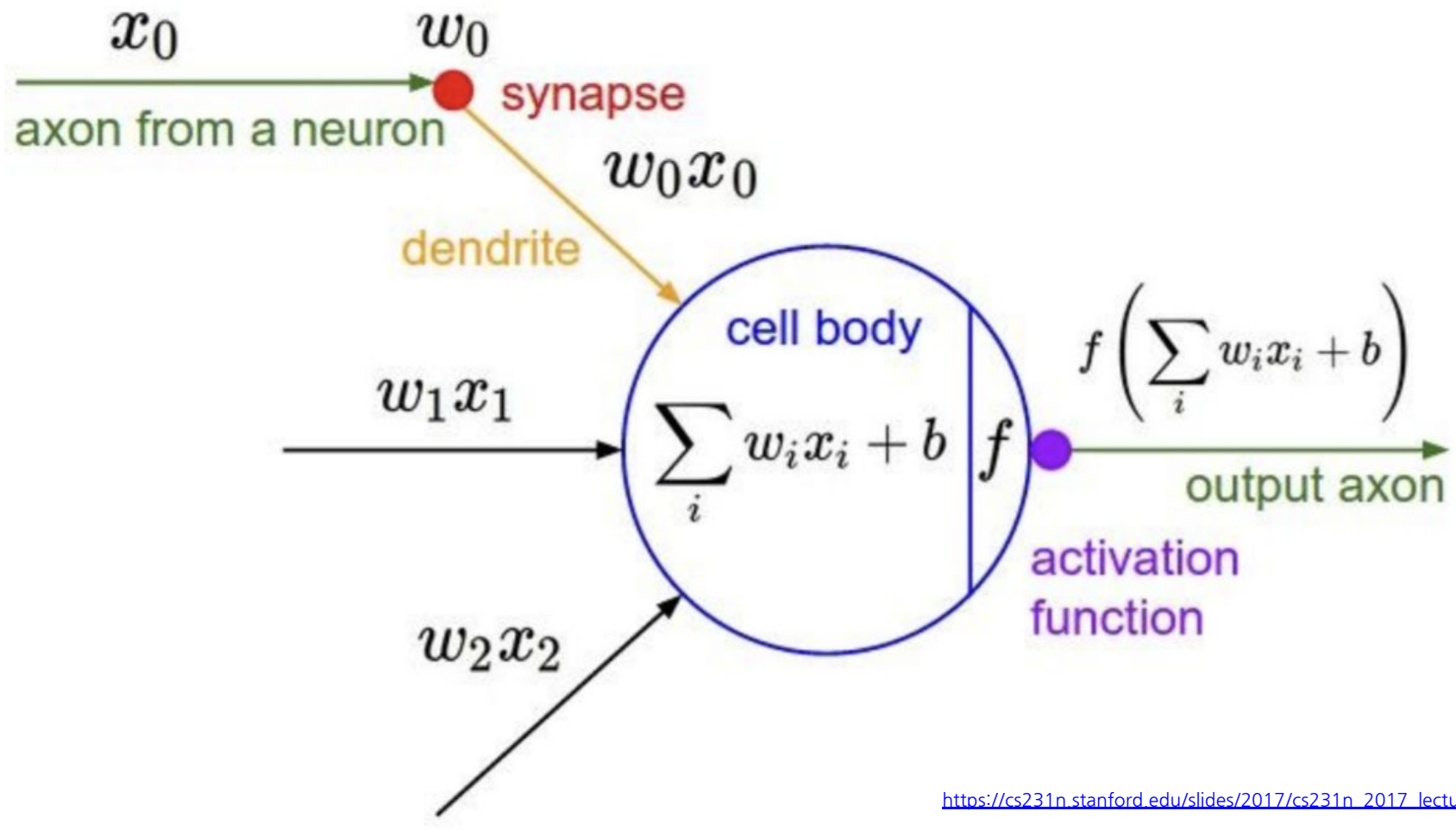
항상 좋은 성능을 가질까요?

2.1 Perceptron



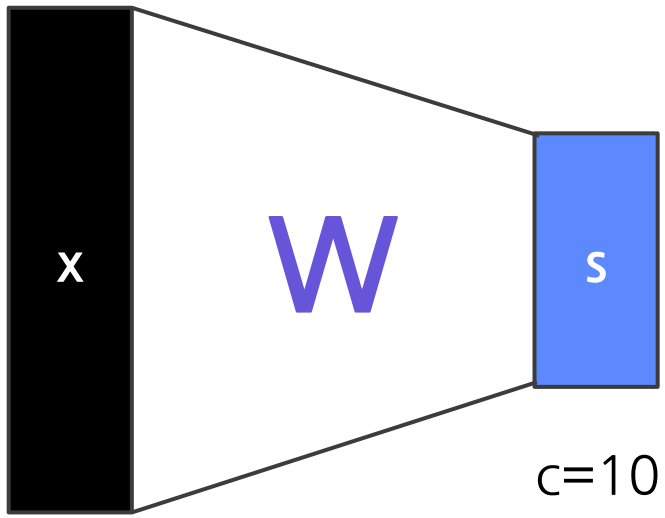
https://svgsilh.com/ko/image/2022398.html

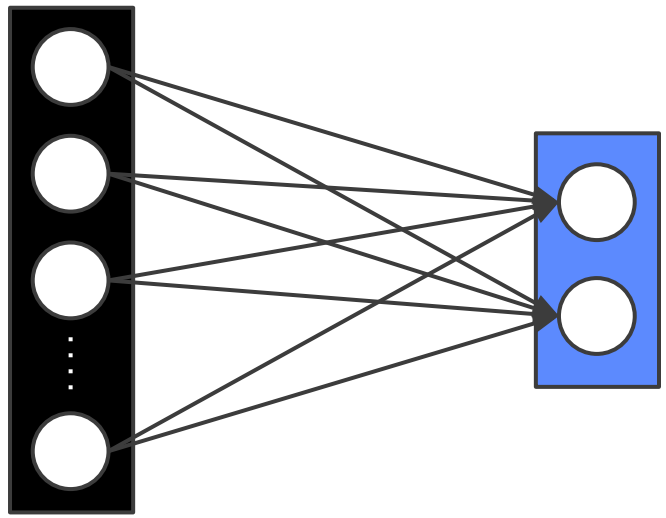
2.1 Perceptron



https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

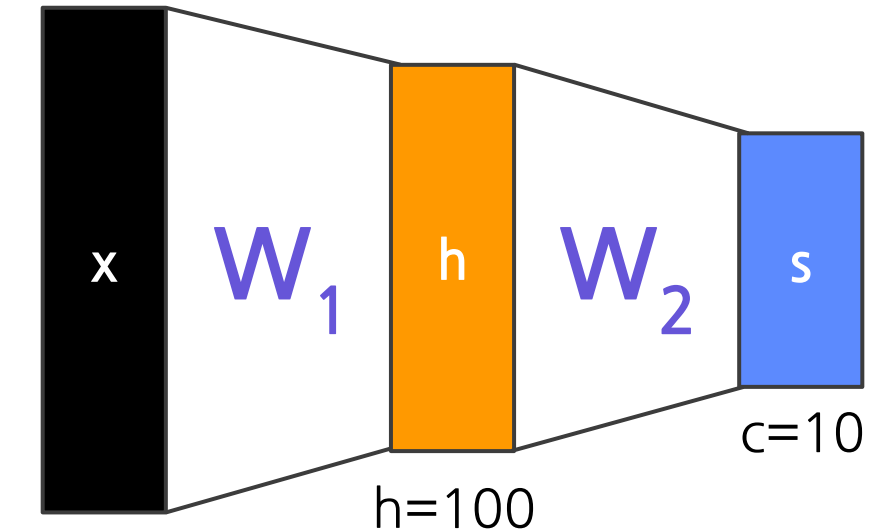
2.2 Neural Network with a Single Layer

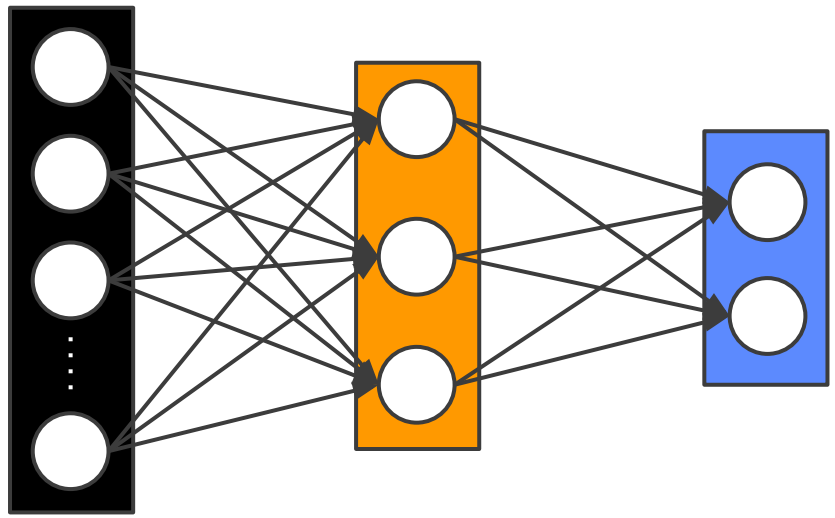




$$\mathbf{x} \in \mathbb{R}^d \qquad \mathbf{W} \in \mathbb{R}^{c \times d} \qquad \mathbf{s} \in \mathbb{R}^c$$

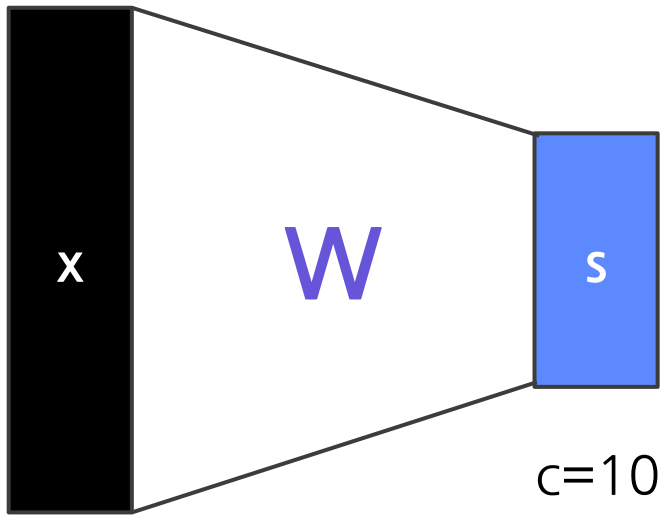
2.3 Multilayer Perceptron (MLP)



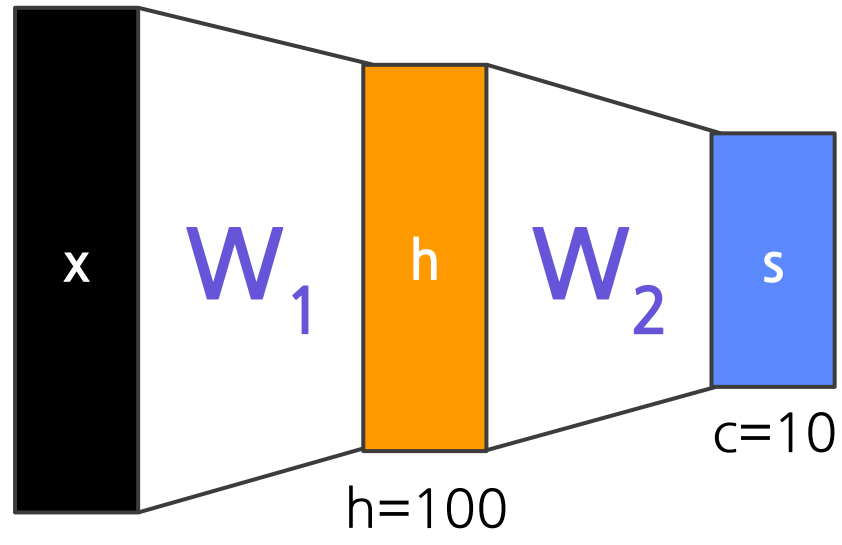


$$\mathbf{x} \in \mathbb{R}^d \quad \mathbf{W}_1 \in \mathbb{R}^{h \times d} \quad \mathbf{W}_2 \in \mathbb{R}^{c \times h} \quad \mathbf{s} \in \mathbb{R}^c$$

2.3 Multilayer Perceptron (MLP)



$$f(\mathbf{x}) = \mathbf{W}\mathbf{x}$$



$$f(\mathbf{x}) = \mathbf{W}_2(\mathbf{W}_1\mathbf{x})$$

2.3 Multilayer Perceptron (MLP)

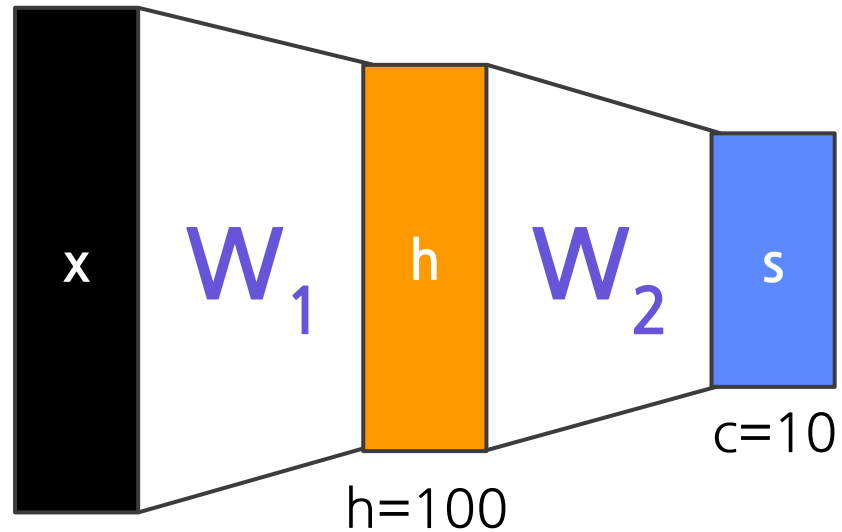
Multi-linear layers은 여전히 linear합니다.

- $W = W_2 W_1$

어떻게 하면 non-linear하게 만들 수 있을까요?

→ Activation functions!

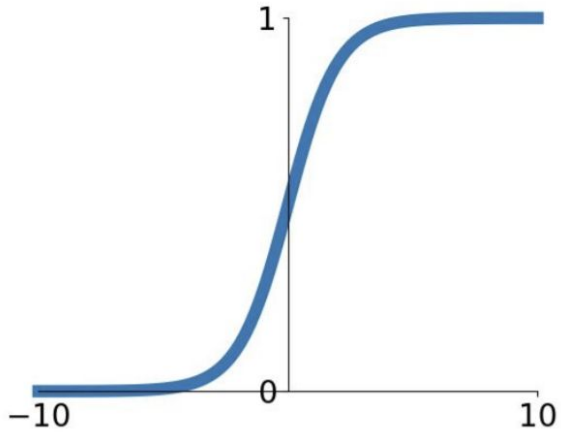
$$f(\mathbf{x}) = a_2(\mathbf{W}_2\, a_1(\mathbf{W}_1\mathbf{x}))$$



$$f(\mathbf{x}) = \mathbf{W}_2(\mathbf{W}_1\mathbf{x})$$

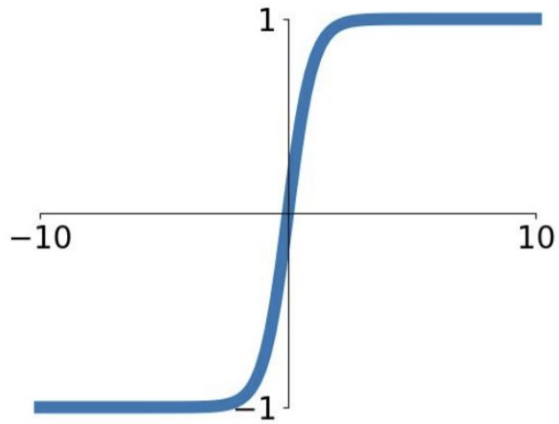
2.4 Activation Functions

- Sigmoid



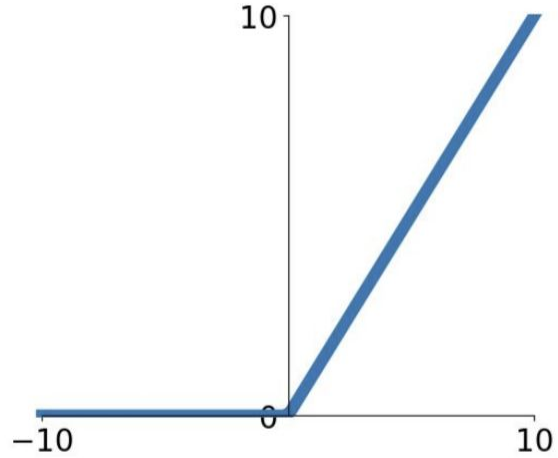
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

- tanh



$$\tanh(x)$$

- ReLU



$$\max(0, x)$$

https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf

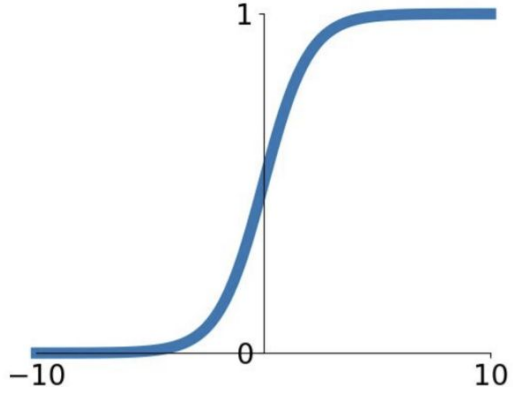
2.4 Sigmoid Functions

특징

- 출력 값의 범위: [0, 1]
- 가장 많이 사용되었던 함수

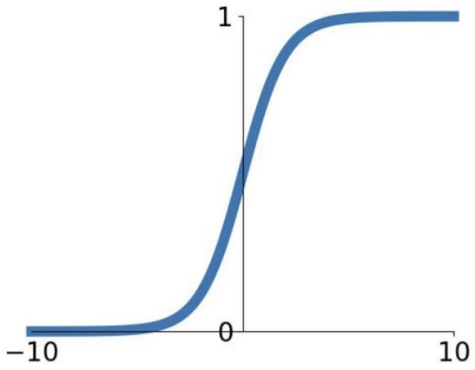
단점

- Vanishing Gradient: input이 크거나 작을때 기울기가 0에 가까워집니다.
- **출력 값들이 zero-centered** 하지 않습니다.
- exp() 의 연산이 비쌉니다.



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

2.4 Sigmoid Functions: Killed Gradients



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

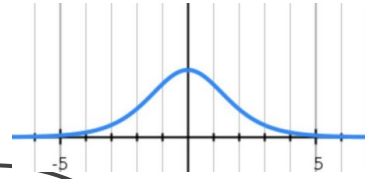
$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1-\sigma(x))$$

$$\begin{aligned}\sigma'(x) &= \frac{\partial}{\partial x}\frac{1}{1+e^{-x}} \\ &= \frac{e^{-x}}{(1+e^{-x})^2} \\ &= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\ &= \sigma(x)(1-\sigma(x))\end{aligned}$$

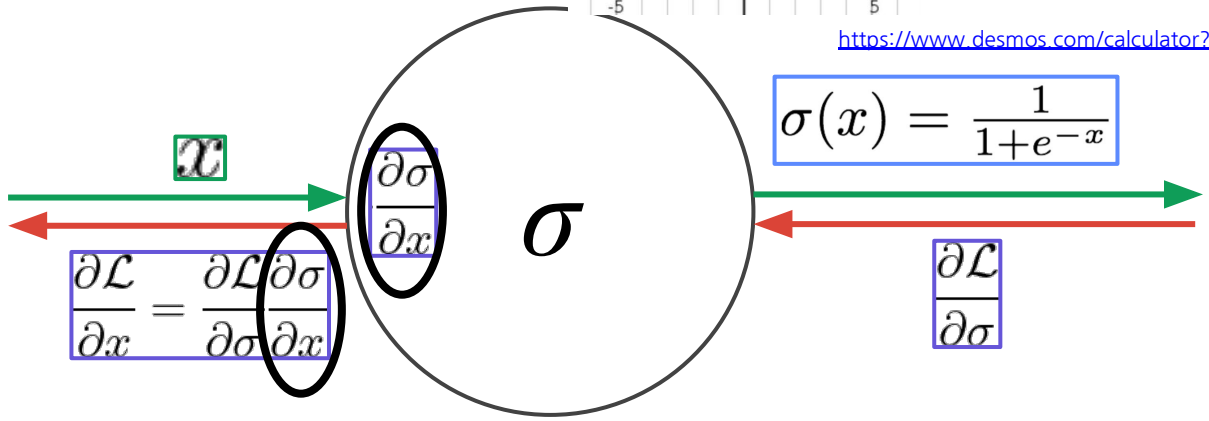
- x가 10일 때

  $\sigma(x) \approx 1$ 이기 때문에 $1 - \sigma(x) \approx 0$, 그레디언트는 0에 수렴합니다.
- x가 -10일 때

  $\sigma(x) \approx 0$ 이기 때문에 그레디언트는 0에 수렴합니다.
- 입력 x에 대한 그레디언트 그래프?



https://www.desmos.com/calculator?lang=ko

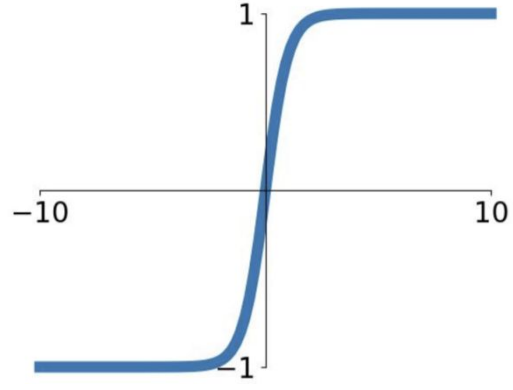


따라서 만약 입력 x가 크다면 이 노드를 통과했을 때 기울기에 $\approx$ 0이 곱해진다.

2.4 Tanh Functions

특징

- 출력 값의 범위: [-1, 1].
- Zero-centered.



$\tanh(x)$

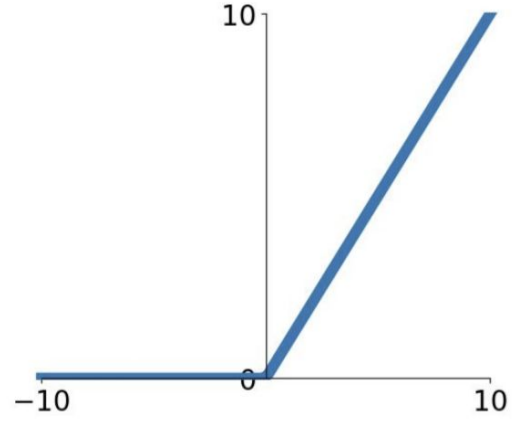
단점

- Vanishing Gradient 문제가 여전히 발생합니다.
  - 커진 sigmoid 뉴런과 비슷하다.
    : tanh(x) = 2σ(2x) - 1

2.4 ReLU (Rectified Linear Unit)

특징

- + 영역에서 saturate 되지 않습니다.
- 연산이 효율적입니다.
- sigmoid나 tanh 보다 빨리 수렴합니다.



단점

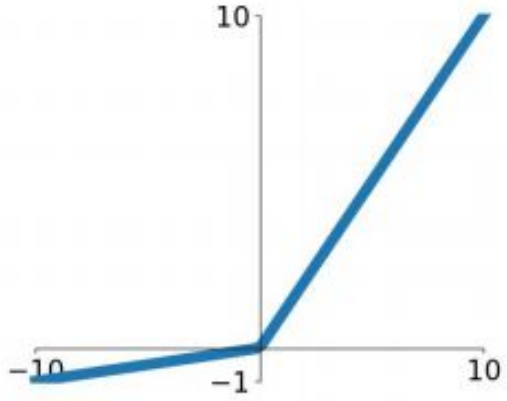
- 출력 값이 zero-centered 되지 않습니다.
- **Dead ReLU problem**: 출력 값이 음수라면 saturated 되는 문제가 발생합니다.
- **x=o일 때 미분 불가능합니다.** 0.01과 같은 약간의 기울기는 ReLU 뉴런들을 초기화하는데 도움을 줍니다.

$$\max(0, x)$$

2.4 Leaky ReLU (Rectified Linear Unit)

특징

- 모든 영역에서 saturate되지 않습니다.
- 연산에 효율적입니다.
- sigmoid나 tanh 보다 수렴 속도가 빠릅니다.
- No Dead ReLU 문제: gradient vanishing 되지 않습니다.



$$\max(0.01x, x)$$

단점

- 추가적인 하이퍼파라미터(x가 0 미만일 때의 기울기)

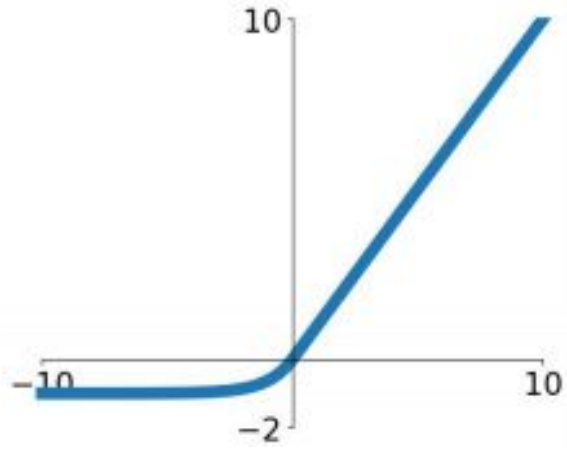
2.4 ELU (Exponential Linear Unit)

특징

- ReLU의 모든 장점
- (Leaky) ReLU에 비해 saturated된 음수 지역은 견고성을 더합니다.

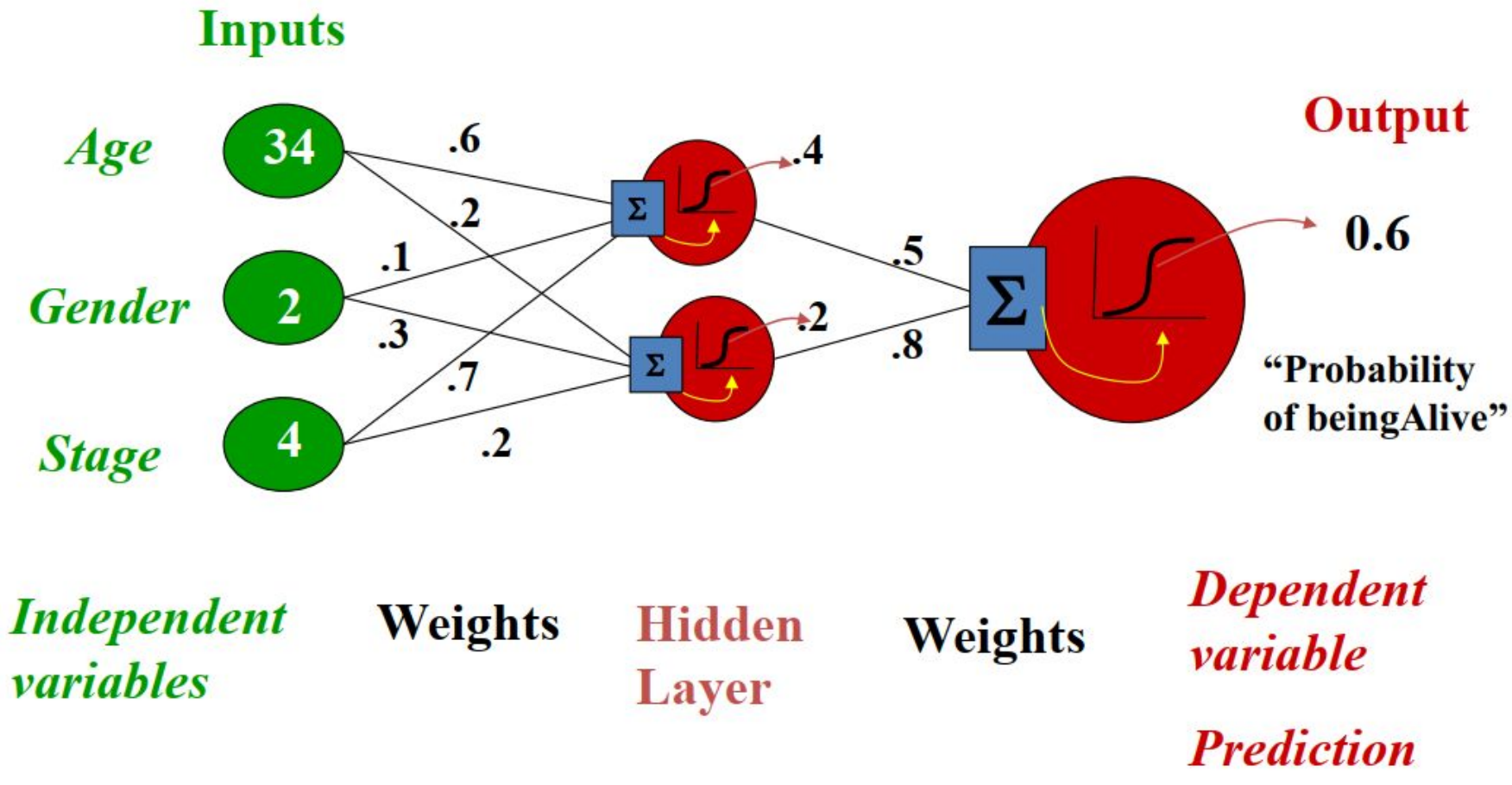
단점

- exp()의 연산이 비쌉니다.



$$\begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$$

2.5 An Example of Neural Network



https://courses.grainger.illinois.edu/cs440/fa2023/lectures/neural-nets.html

2.6 Computing Gradients

(Stochastic) GD에서는 **classification loss에 대한 그래디언트**를 필요로 한다.

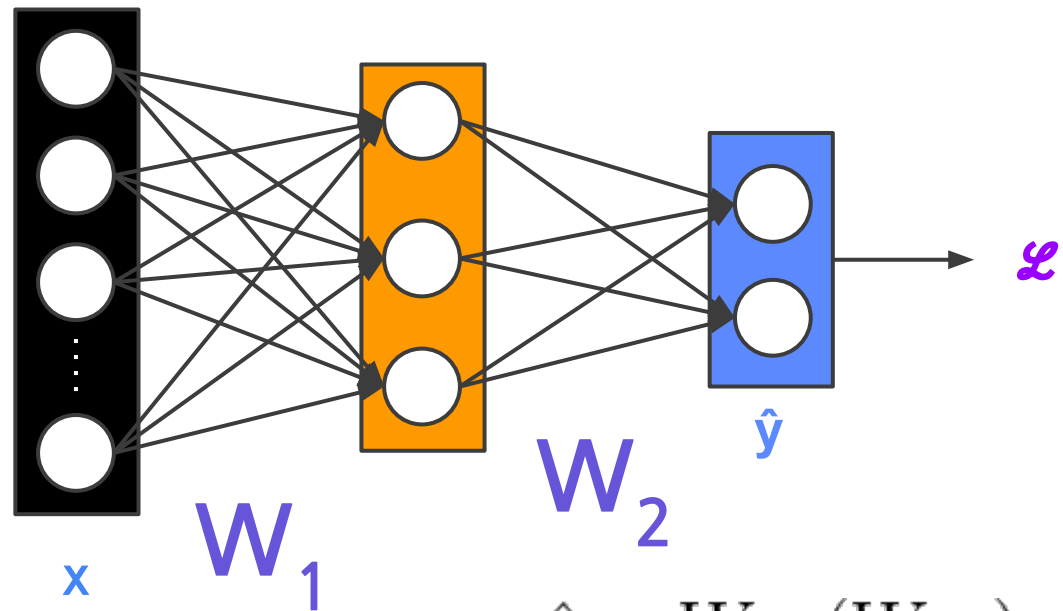
- *w.r.t.* **각 파라미터 (weight)**

$$\frac{\partial \mathcal{L}}{\partial W_1}, \frac{\partial \mathcal{L}}{\partial W_2}$$

- 각각 파라미터를 업데이트 하기 위해 변수가 loss에 얼마나 큰 영향을 미치는지를 나타냅니다.

$$\Theta^{new} = \Theta^{old} - \alpha \nabla_{\Theta} J(\Theta)$$

$$\theta_i^{new} = \theta_i^{old} - \alpha \frac{\partial}{\partial \theta_i^{old}} J(\Theta)$$



$$\hat{y} = W_2 \sigma(W_1 \mathbf{x})$$

2.6 Computing Gradients

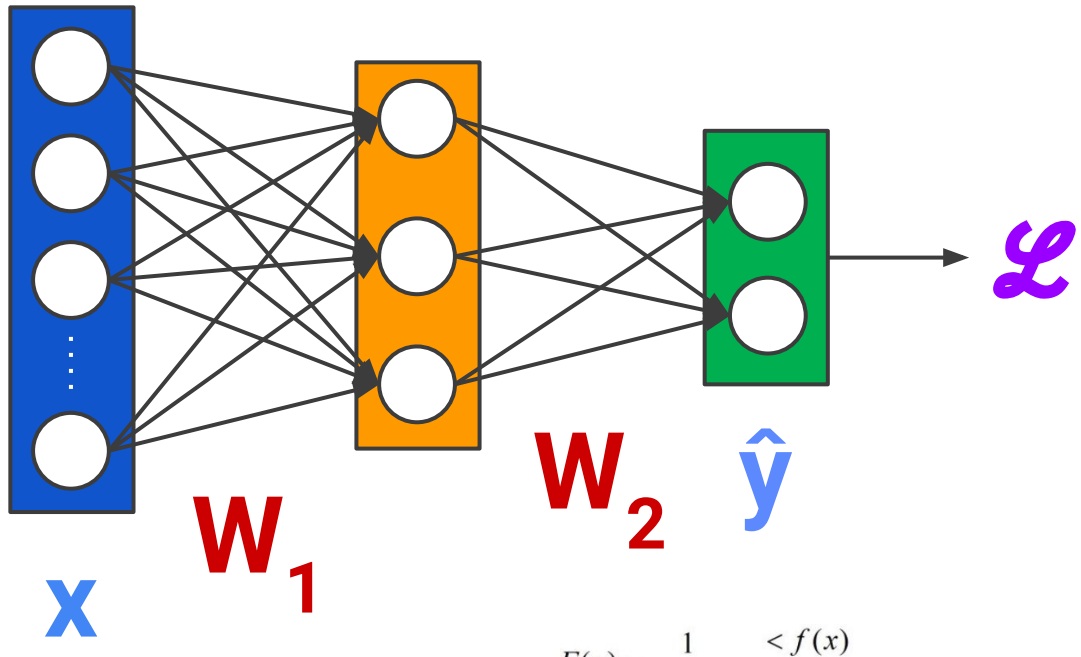
$$\mathcal{L} = (\hat{y} - y)^2 \qquad \hat{y} = W_2\sigma(W_1\mathbf{x})$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = 2(\hat{y} - y)$$

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial W_2} = 2(\hat{y} - y) \cdot \sigma(W_1\mathbf{x})$$

$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial h}\frac{\partial h}{\partial W_1} \qquad h = \sigma(W_1\mathbf{x})$$

$$= 2(\hat{y} - y) \cdot W_2 \cdot h(1 - h) \cdot \mathbf{x}$$



$$F(x) = \frac{1}{1+e^{-x}} \quad \begin{matrix} < f(x) \\ < g(x) \end{matrix}$$

$$F'(x) = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$$

$$= \frac{0-(-e^{-x})}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}}$$

$$= F(x)\frac{1+e^{-x}-1}{1+e^{-x}} = F(x)\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right)$$

$$= F(x)(1-F(x))$$

2.6 Implementation: 2-layer MLP


```
import numpy as np
from numpy.random import randn

n, d, h, c = 64, 1000, 100, 10
x, y = randn(n, d), randn(n, c)
w1, w2 = randn(d, h), randn(h, c)
learning_rate = 1e-4

for t in range(1000):
  y_0= x.dot(w1)
  h_0 = 1 / (1 + np.exp(-y_0))
  y_pred = h.dot(w2)
  loss = np.square(y_pred - y).sum()
  print(t, loss)

  grad_y_pred = 2.0 * (y_pred - y)
  grad_w2 = h.T.dot(grad_y_pred)
  grad_h = grad_y_pred.dot(w2.T)
  grad_w1 = x.T.dot(grad_h * h * (1 - h))

  w1 -= learning_rate * grad_w1
  w2 -= learning_rate * grad_w2
```

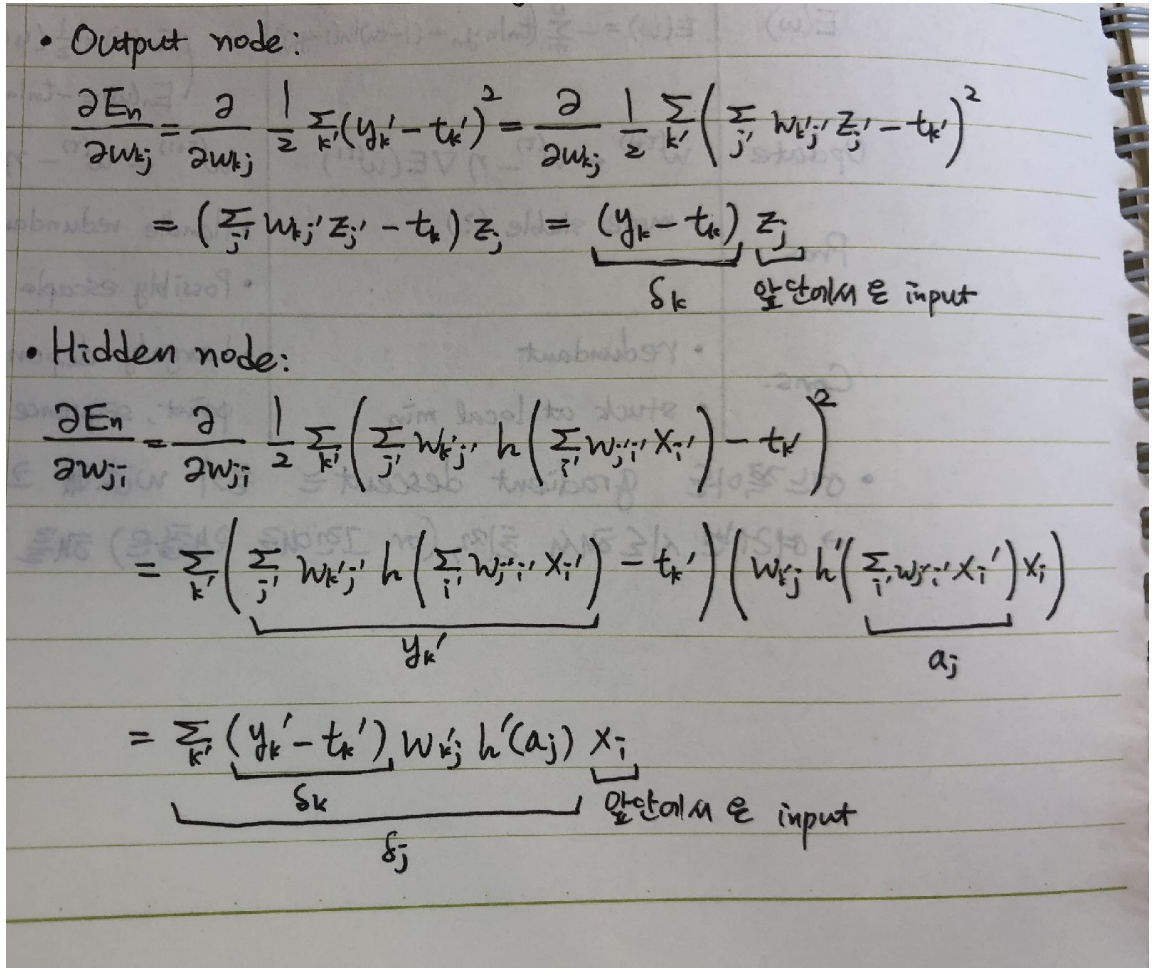
모델 정의
(n: #examples, d: input dim,
d: hidden dim, c: #classes)

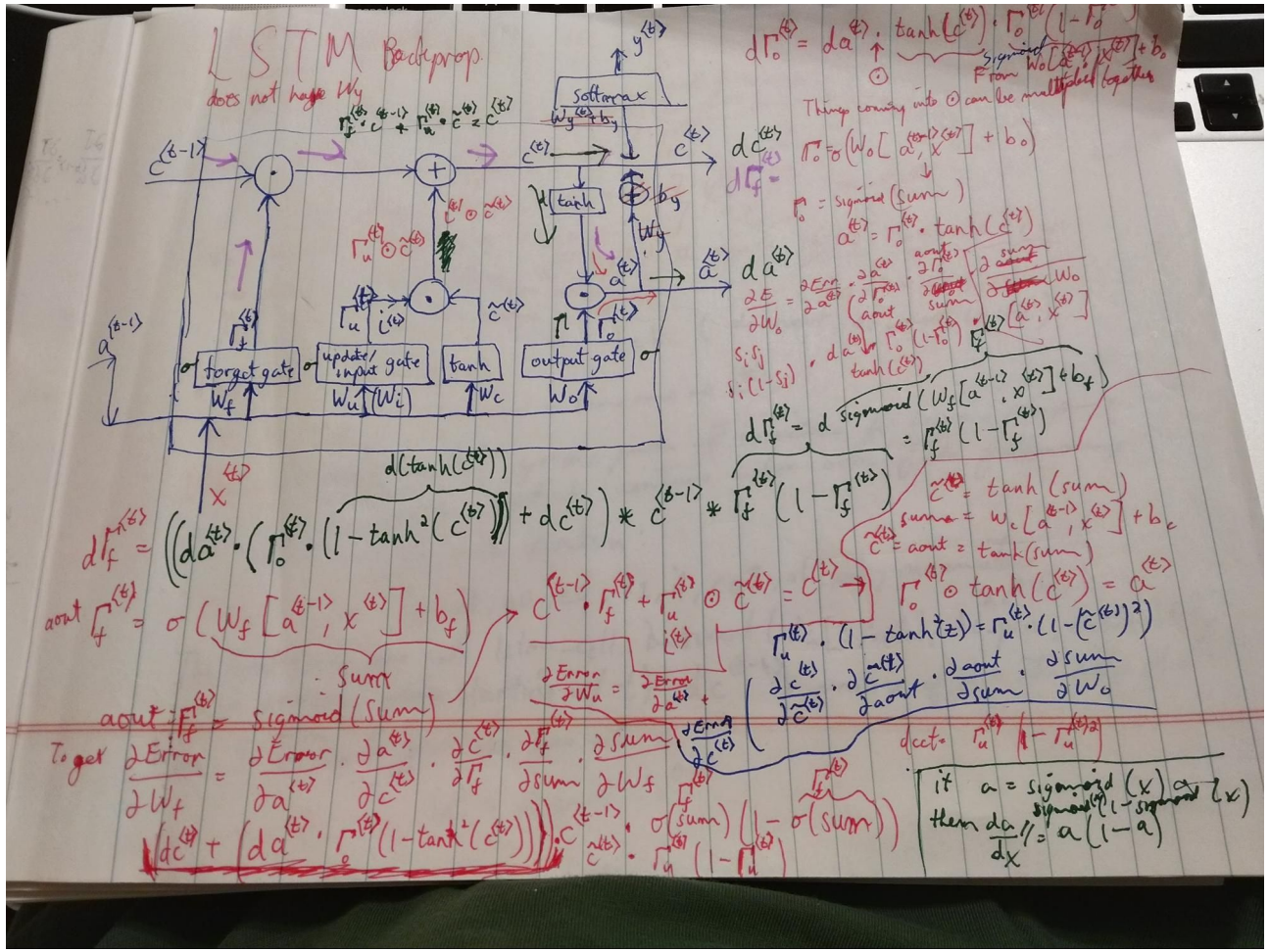
Forward Pass: 현재 모델을 이용하여 예측합니다.

해석적 **그래디언트**들을 계산합니다.

Gradient Descent

2.6 Computing Gradients

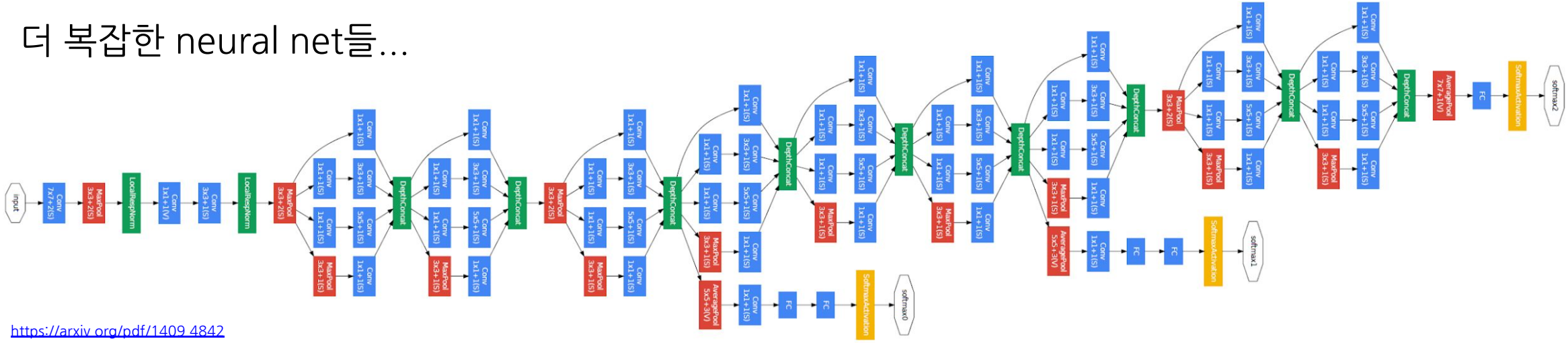




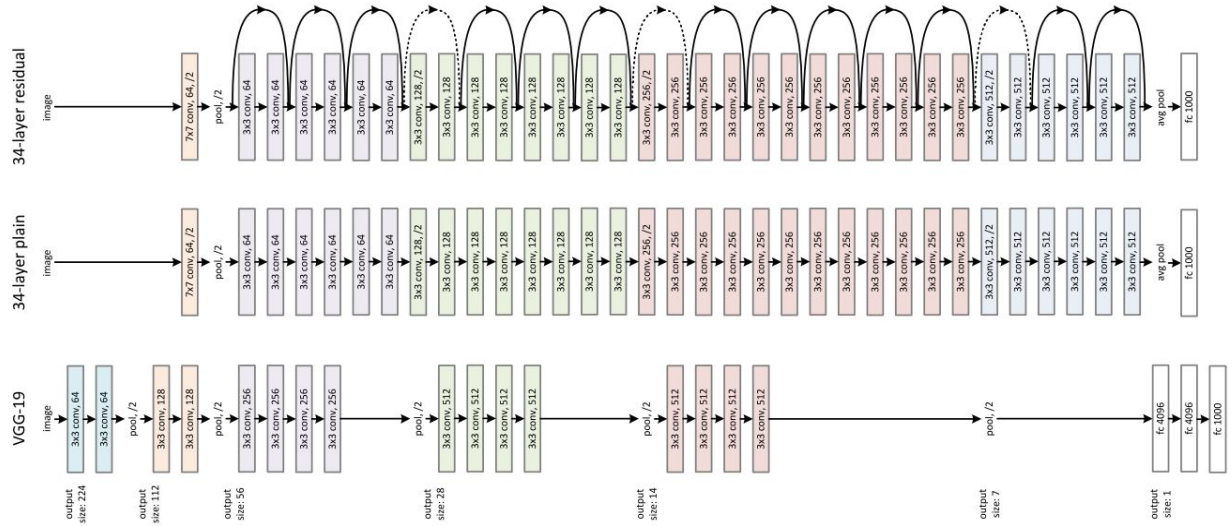
https://www.kdnuggets.com/

2.6 Computing Gradients

더 복잡한 neural net들...



https://arxiv.org/pdf/1409.4842



https://arxiv.org/pdf/1512.03385

Thank you

Prof. Jeon, Sangryul

Computer Vision Lab.

Pusan National University, Korea

Tel: +82-51-510-2423

Web: http://sr-jeon.github.io/

E-mail: srjeonn@pusan.ac.kr