ACADENTIAL

Section 7. 활성 함수 (Activation Function)

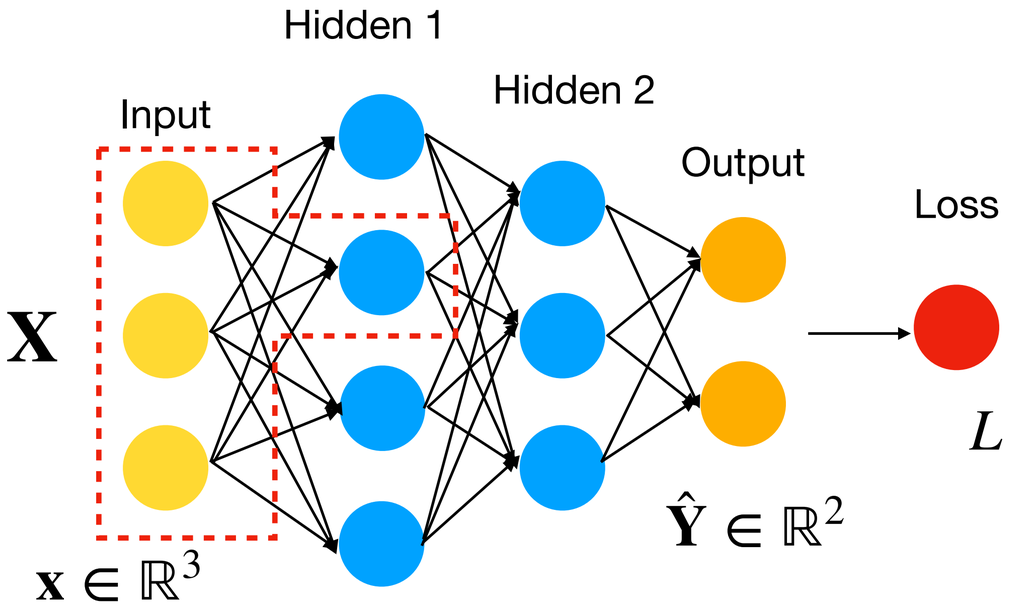
변정현

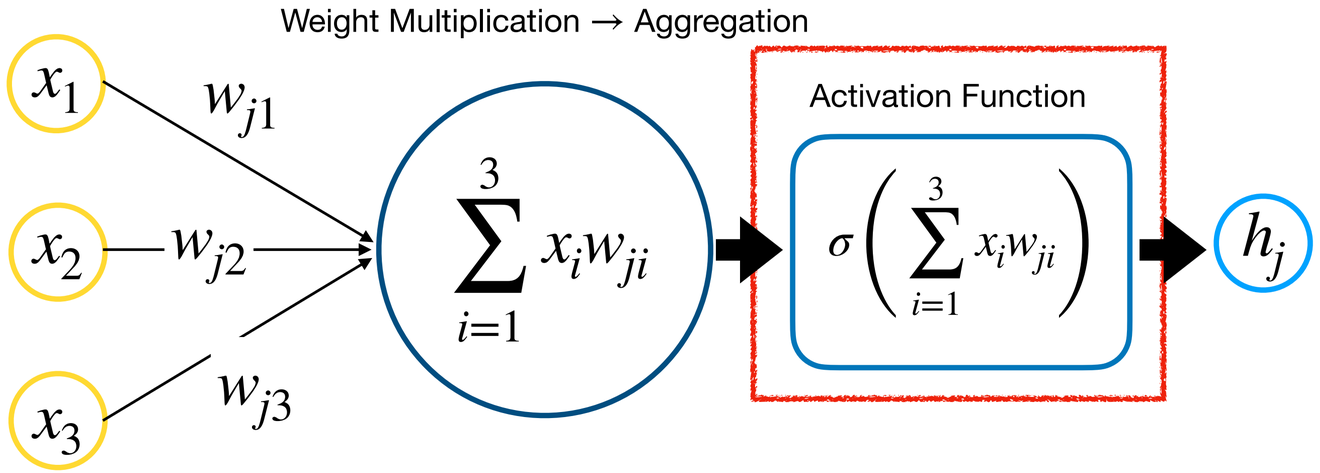
ACADENTIAL

목차

ACADENTIAL

Recap from previous Chapters





이번 강의에서는 “Activation Function” (활성화 함수)가 무엇인지 살펴보자!

ACADENTIAL

Objective
학습 목표

- Activation function의 정의와 역할 이해
- Activation function이 Non-linear decision boundary을 학습하는데 필요한 이유
- Activation function의 종류들
  - Sigmoid
  - Tanh
  - ReLU
  - Leaky ReLU
  - ELU
  - Softmax
- 각 Activation function 종류의 장단점

ACADENTIAL

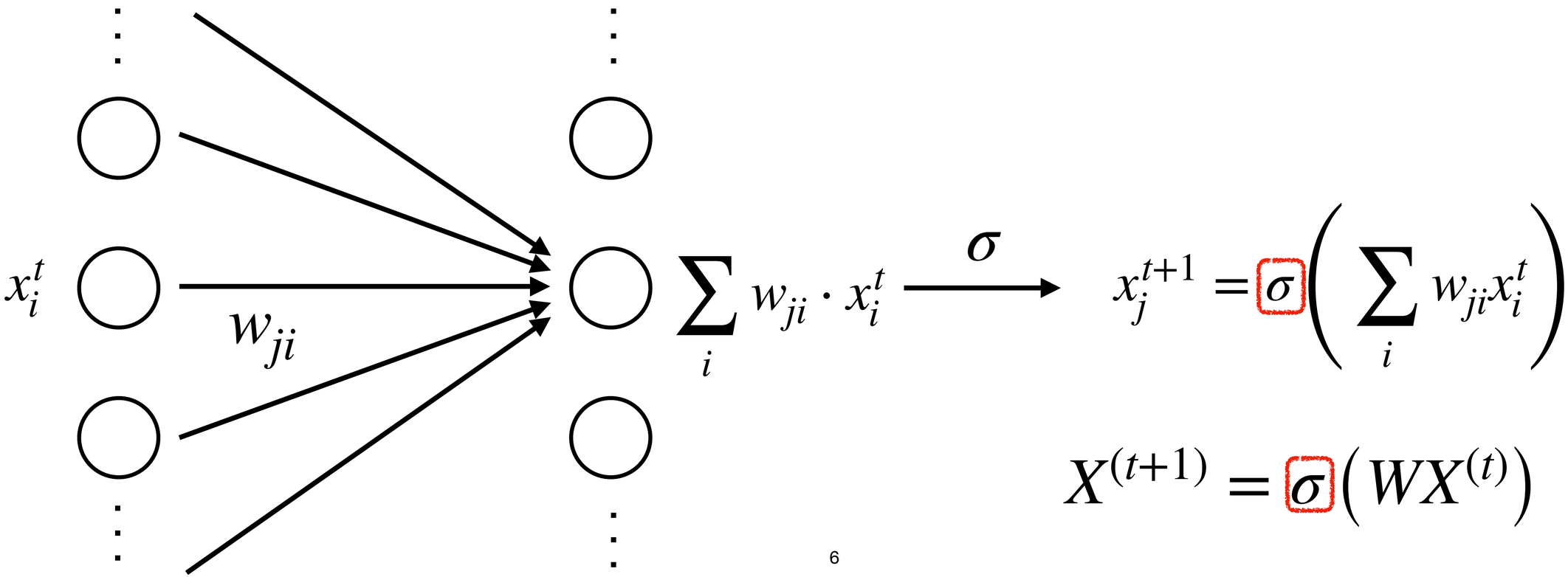
7-1. Non-linear한 Activation Function이 필요한 이유

ACADENTIAL

Activation Functions

What are Activation Functions?

Activation Function σ = Layer와 layer 사이에 위치한 **non-linear function** (비선형 함수)



ACADENTIAL

Activation Functions

Activation function은 왜 필요한가?

ACADENTIAL

Activation Functions

Activation function은 왜 필요한가?

왜냐하면 **non-linear한 activation function을 사용**함으로서 뉴럴넷은 **non-linear한 decision boundary**을 그릴 수 있기 때문이다!

ACADENTIAL

Activation Functions

Activation function은 왜 필요한가?

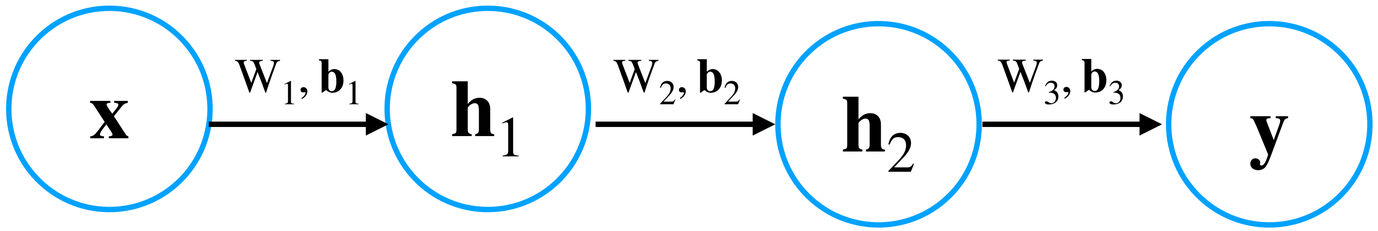
왜냐하면 **non-linear한 activation function을 사용**함으로서 뉴럴넷은 **non-linear한 decision boundary**을 그릴 수 있기 때문이다!

그리고 **linear한 activation function을 사용**하면 아무리 많은 Layer들을 쌓아도 **Single Layer Neural Network에 불과**하다.

ACADENTIAL

Activation Functions

Activation function은 왜 필요한가?

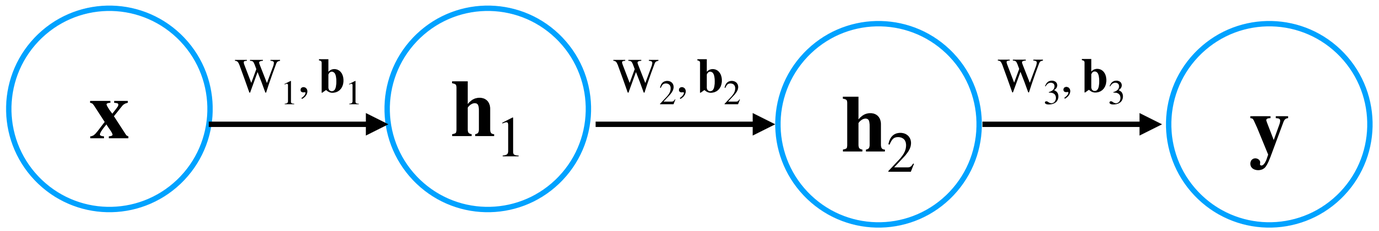


$$\mathbf{h_1} = W_1\mathbf{x} + \mathbf{b}_1 \quad \mathbf{h}_2 = W_2\mathbf{h}_1 + \mathbf{b}_2 \quad \mathbf{y} = W_3\mathbf{h}_2 + \mathbf{b}_3$$

Activation Functions

Activation function은 왜 필요한가?

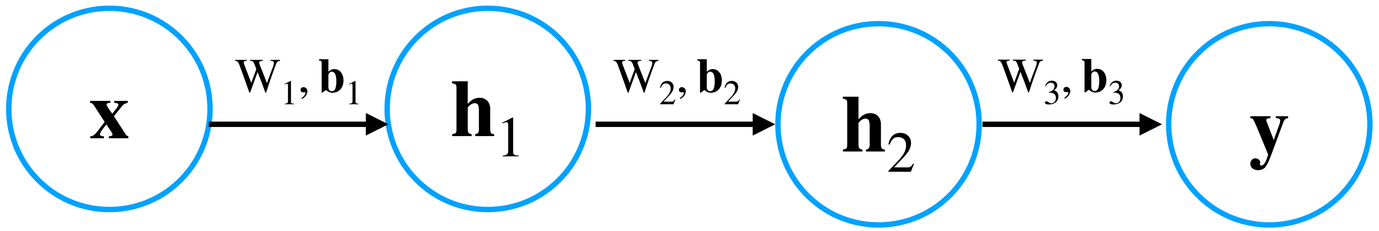
$$\mathbf{y} = W_3\mathbf{h}_2 + \mathbf{b}_3$$



$$\mathbf{h_1} = W_1\mathbf{x} + \mathbf{b}_1 \quad \mathbf{h}_2 = W_2\mathbf{h}_1 + \mathbf{b}_2 \quad \mathbf{y} = W_3\mathbf{h}_2 + \mathbf{b}_3$$

Activation Functions

Activation function은 왜 필요한가?



$$\mathbf{h_1} = W_1\mathbf{x} + \mathbf{b}_1 \qquad \mathbf{h}_2 = W_2\mathbf{h}_1 + \mathbf{b}_2 \qquad \mathbf{y} = W_3\mathbf{h}_2 + \mathbf{b}_3$$

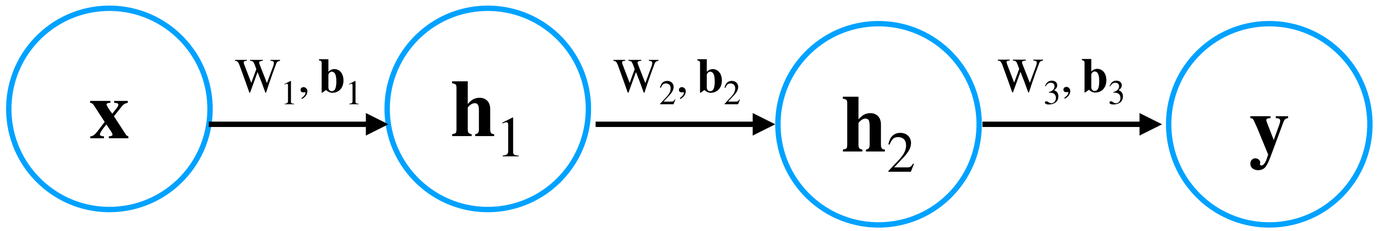
$$\mathbf{y} = W_3\mathbf{h}_2 + \mathbf{b}_3$$

$$= W_3(W_2\mathbf{h}_1 + \mathbf{b}_2) + \mathbf{b}_3$$

ACADENTIAL

Activation Functions

Activation function은 왜 필요한가?



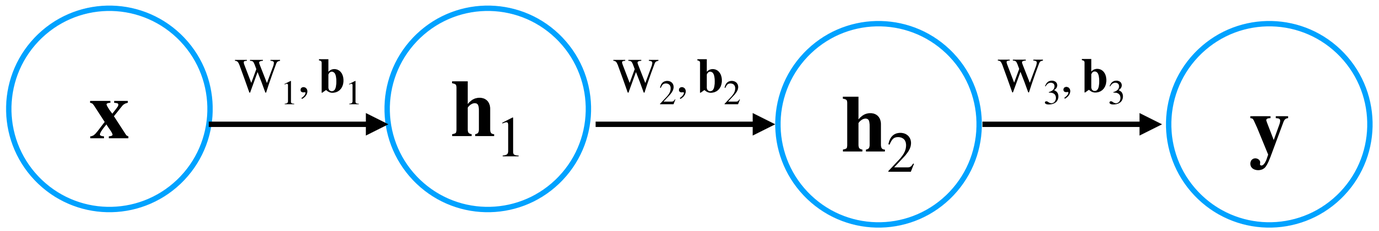
$$\mathbf{h_1} = W_1\mathbf{x} + \mathbf{b}_1 \qquad \mathbf{h}_2 = W_2\mathbf{h}_1 + \mathbf{b}_2 \qquad \mathbf{y} = W_3\mathbf{h}_2 + \mathbf{b}_3$$

$$\begin{aligned}\mathbf{y} &= W_3\mathbf{h}_2 + \mathbf{b}_3 \\ &= W_3(W_2\mathbf{h}_1 + \mathbf{b}_2) + \mathbf{b}_3 \\ &= W_3(W_2(W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3\end{aligned}$$

ACADENTIAL

Activation Functions

Activation function은 왜 필요한가?



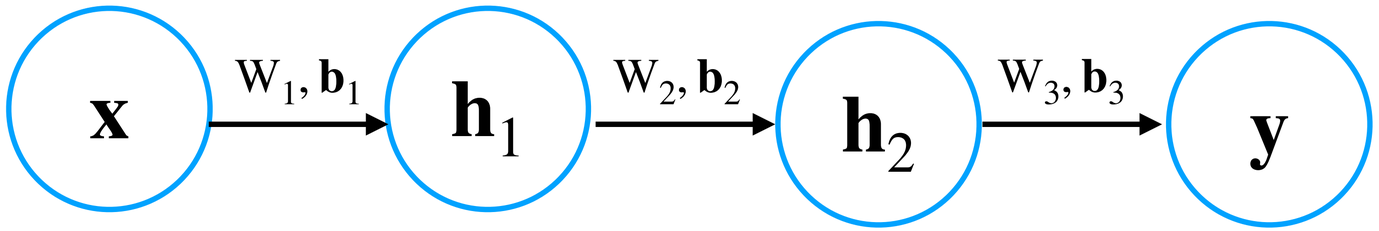
$$\mathbf{h_1} = W_1\mathbf{x} + \mathbf{b}_1 \qquad \mathbf{h}_2 = W_2\mathbf{h}_1 + \mathbf{b}_2 \qquad \mathbf{y} = W_3\mathbf{h}_2 + \mathbf{b}_3$$

$$\begin{aligned}
\mathbf{y} &= W_3\mathbf{h}_2 + \mathbf{b}_3 \\
&= W_3(W_2\mathbf{h}_1 + \mathbf{b}_2) + \mathbf{b}_3 \\
&= W_3(W_2(W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3 \\
&= W_3W_2W_1\mathbf{x} + \left(W_3W_2\mathbf{b}_1 + W_3\mathbf{b}_2 + \mathbf{b}_3\right)
\end{aligned}$$

ACADENTIAL

Activation Functions

Activation function은 왜 필요한가?



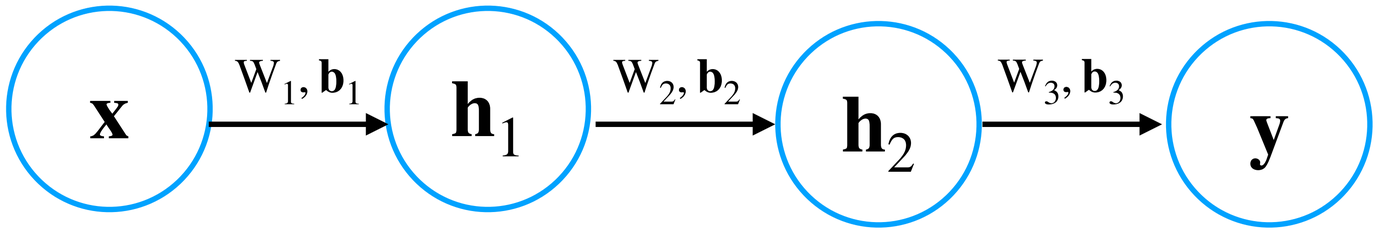
$$\mathbf{h_1} = W_1\mathbf{x} + \mathbf{b}_1 \qquad \mathbf{h}_2 = W_2\mathbf{h}_1 + \mathbf{b}_2 \qquad \mathbf{y} = W_3\mathbf{h}_2 + \mathbf{b}_3$$

$$\begin{aligned}
\mathbf{y} &= W_3\mathbf{h}_2 + \mathbf{b}_3 \\
&= W_3(W_2\mathbf{h}_1 + \mathbf{b}_2) + \mathbf{b}_3 \\
&= W_3(W_2(W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3 \\
&= W_3W_2W_1\mathbf{x} + \left(W_3W_2\mathbf{b}_1 + W_3\mathbf{b}_2 + \mathbf{b}_3\right) \\
&= W'\mathbf{x} + \mathbf{b}'
\end{aligned}$$

ACADENTIAL

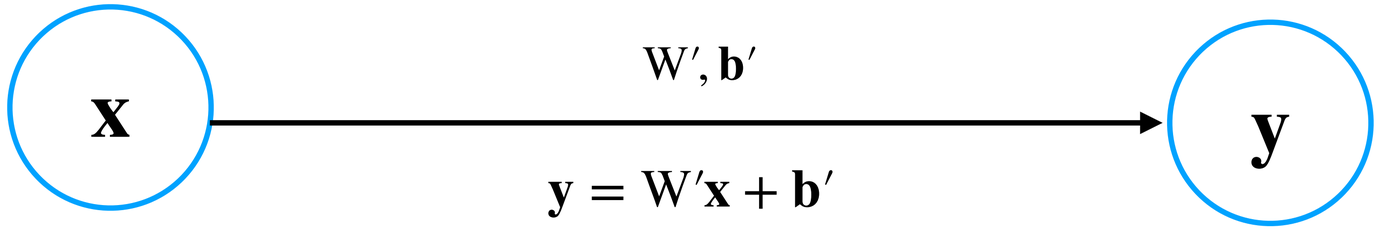
Activation Functions

Activation function은 왜 필요한가?



$$\mathbf{h_1} = W_1\mathbf{x} + \mathbf{b}_1 \qquad \mathbf{h}_2 = W_2\mathbf{h}_1 + \mathbf{b}_2 \qquad \mathbf{y} = W_3\mathbf{h}_2 + \mathbf{b}_3$$

=



$$\begin{aligned}
\mathbf{y} &= W_3\mathbf{h}_2 + \mathbf{b}_3 \\
&= W_3(W_2\mathbf{h}_1 + \mathbf{b}_2) + \mathbf{b}_3 \\
&= W_3(W_2(W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3 \\
&= W_3W_2W_1\mathbf{x} + \left(W_3W_2\mathbf{b}_1 + W_3\mathbf{b}_2 + \mathbf{b}_3\right) \\
&= W'\mathbf{x} + \mathbf{b}'
\end{aligned}$$

ACADENTIAL

Activation Functions

Activation function은 왜 필요한가?

"**linear한 activation function을 사용**하면 아무리 많은 Layer들을 쌓아도 **Single Layer Neural Network에 불과**하다."

ACADENTIAL

Activation Functions

Activation function은 왜 필요한가?

왜냐하면 **non-linear한 activation function을 사용**함으로서 뉴럴넷은 **non-linear한 decision boundary**을 그릴 수 있기 때문이다!

But HOW?

ACADENTIAL

Neural Network은 Non-linear Decision Boundary을 어떻게 학습할까?

ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

먼저 Neural Network Layer을 구성하는 각 요소들이 어떤 역할과 의미를 가지는지 이해해보자.

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

- $\mathbf{W}\mathbf{x}$
  - 행렬 (Matrix)와 벡터 (vector)을 서로 곱하는 것
  - 의미 = input vector $\mathbf{x}$에 **Affine transformation**을 적용하는 것.
  - Affine transformation = input space 상에서의 **격자의 크기가 일정하거나 균일하게 바뀌는 변환**

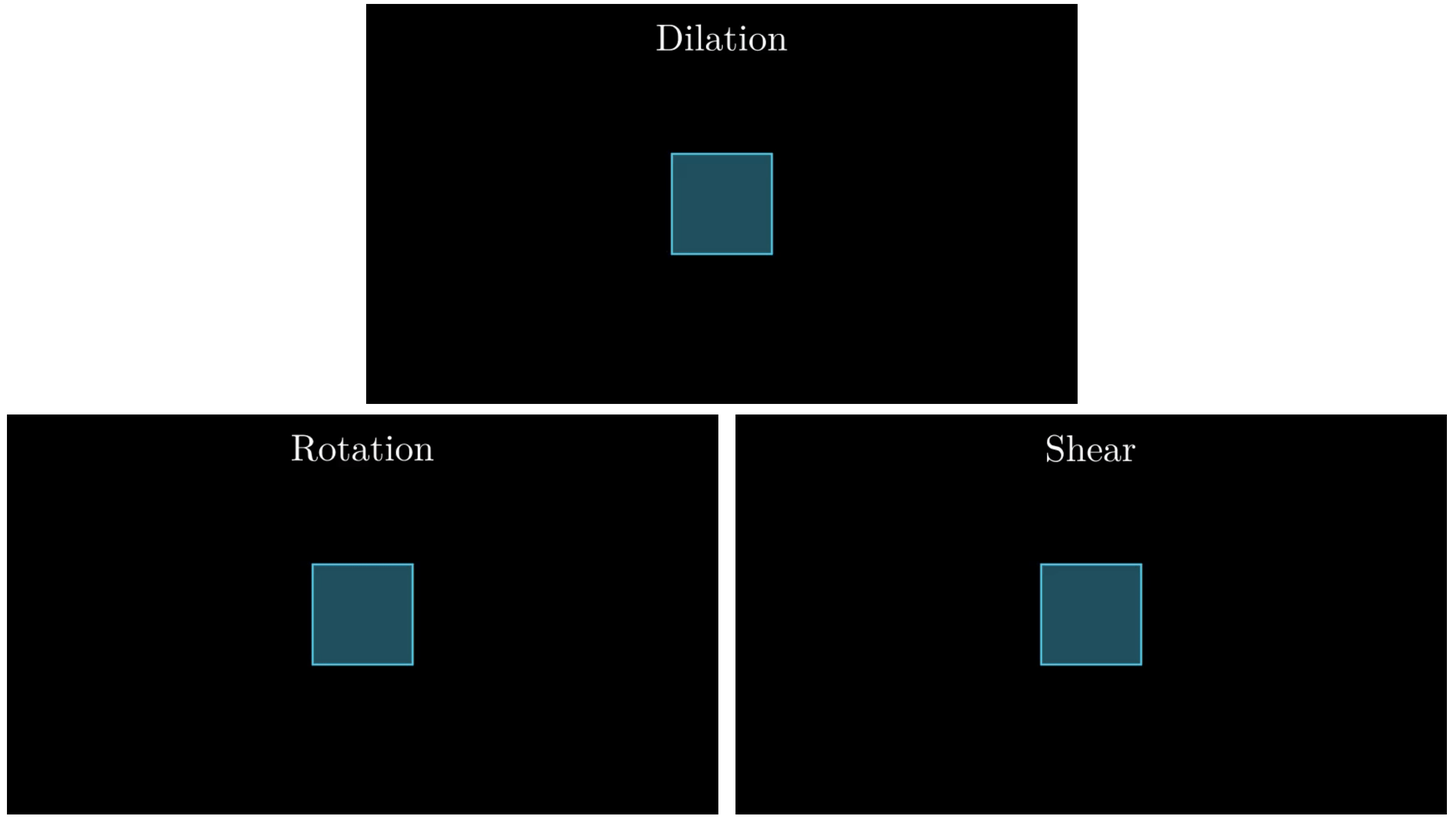
ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

$\mathbf{Wx}$

- input vector $\mathbf{x}$에 **Affine transformation**을 적용하는 것.
- input space 상에서의 **격자가 일정하거나 균일하게 바뀌는** 변환
- **예시**:
  - Rotation (회전)
  - Dilation (확대)
  - Shear (전단)



gif 출처: Khan academy

ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

- $\cdot + \mathbf{b}$
  - bias vector $\mathbf{b}$을 더해주는 것.
  - 의미 = **평행 이동 (translation)** 시키는 것.

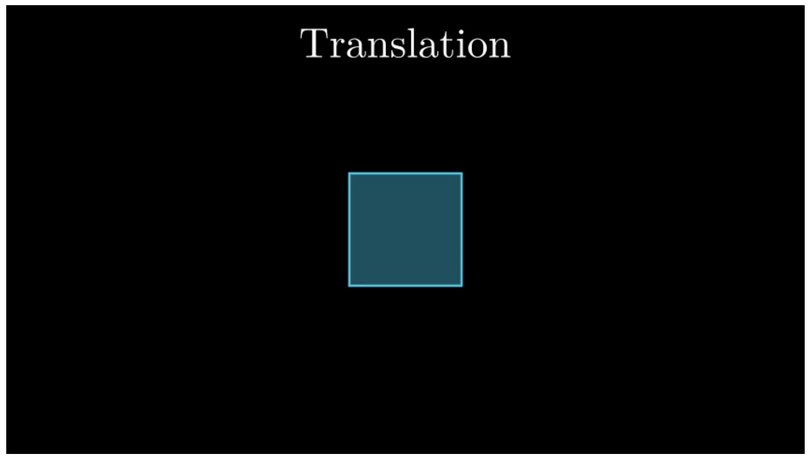
ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

$\cdot + \mathbf{b}$

- bias vector $\mathbf{b}$을 더해주는 것.
- 의미 = **평행 이동 (translation)** 시키는 것.



gif 출처: Khan academy

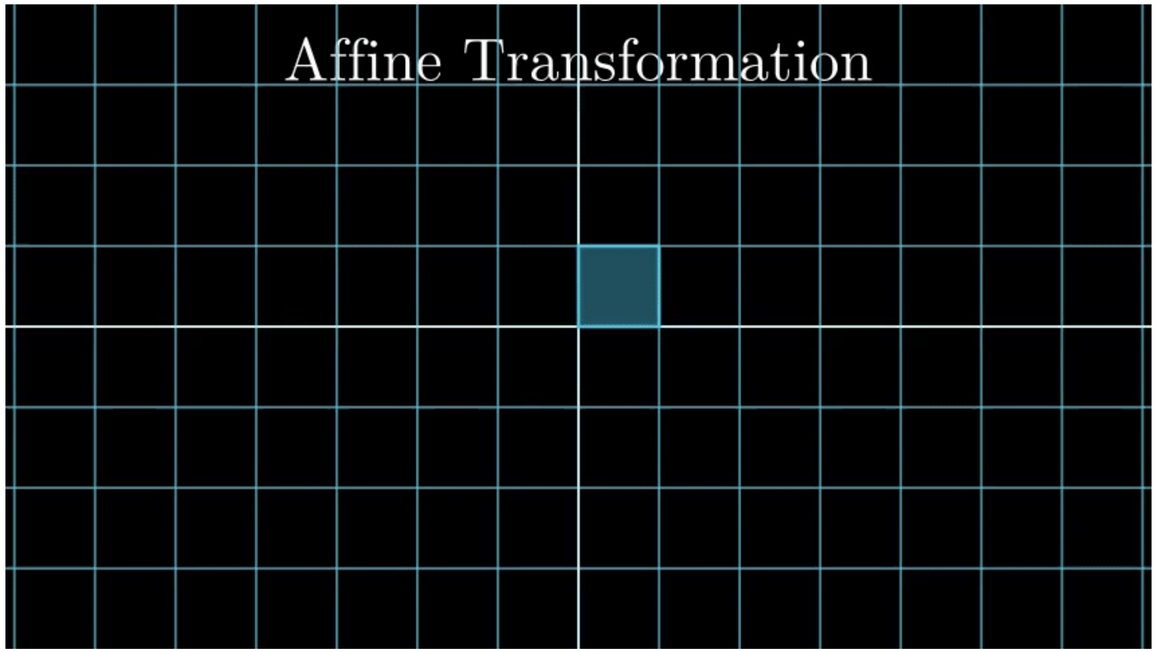
ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

$$\mathbf{W}\mathbf{x} + \mathbf{b}$$

- 종합적으로 보았을때, 다음과 같다!



gif 출처: Khan academy

ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

- $\sigma(\cdot)$
  - non-linear한 activation function을 적용하는 것.
  - 의미
    - **격자가 휘거나 뒤틀리는 변환**을 적용하는 것.
    - **격자가 일정하지 않은** 변환

ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

- $\sigma(\cdot)$
  - non-linear한 activation function을 적용하는 것.
  - 의미
    - **격자가 휘거나 뒤틀리는 변환**을 적용하는 것.
    - **격자가 일정하지 않은** 변환

어느 지점에서는 공간이 상대적으로 **더 많이 "왜곡 / 뒤틀리고"**

혹은 어느 지점에서는 상대적으로 **덜 "왜곡 / 뒤틀리는 변환"**

ACADENTIAL

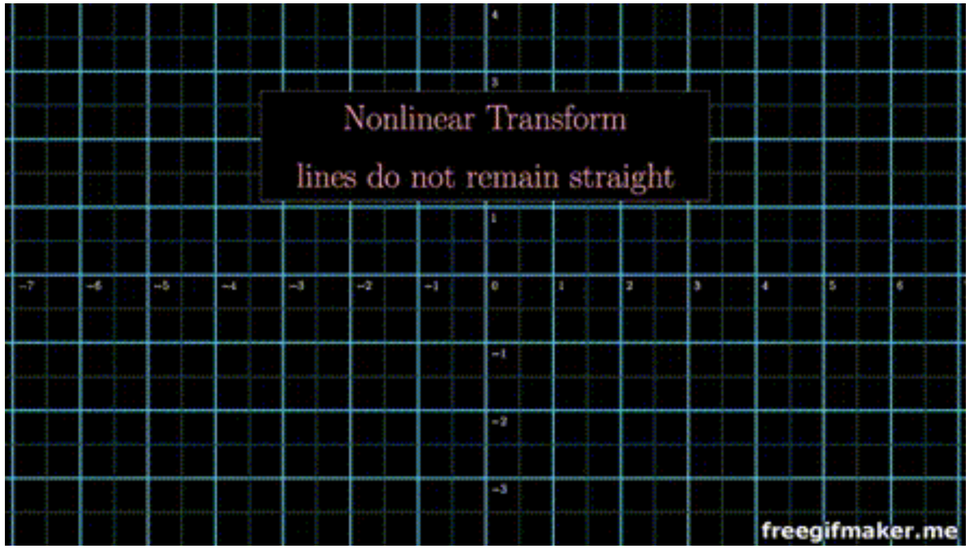
Activation Functions

How does NN learn non-linear decision boundary?

$\sigma(\cdot)$

- non-linear한 activation function 을 적용하는 것.

- 의미 = **input space의 격자가 휘거나 뒤틀리는 변환**을 적용하는 것.



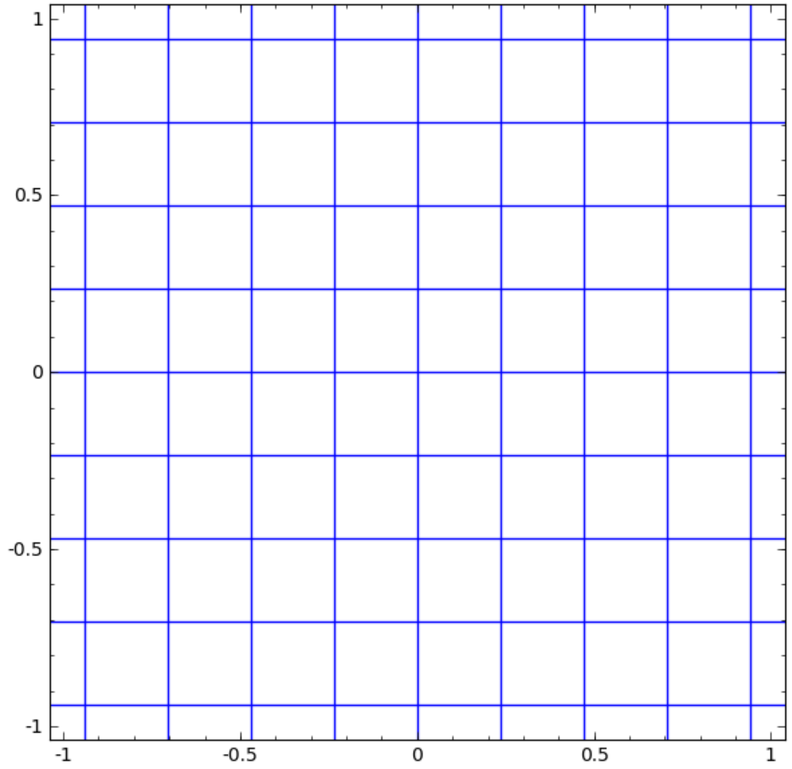
ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

위 변환을 종합해서 visualize해 보면 예시로 오른쪽과 같을 수 있다!



gif 출처: srome.github.io

ACADENTIAL

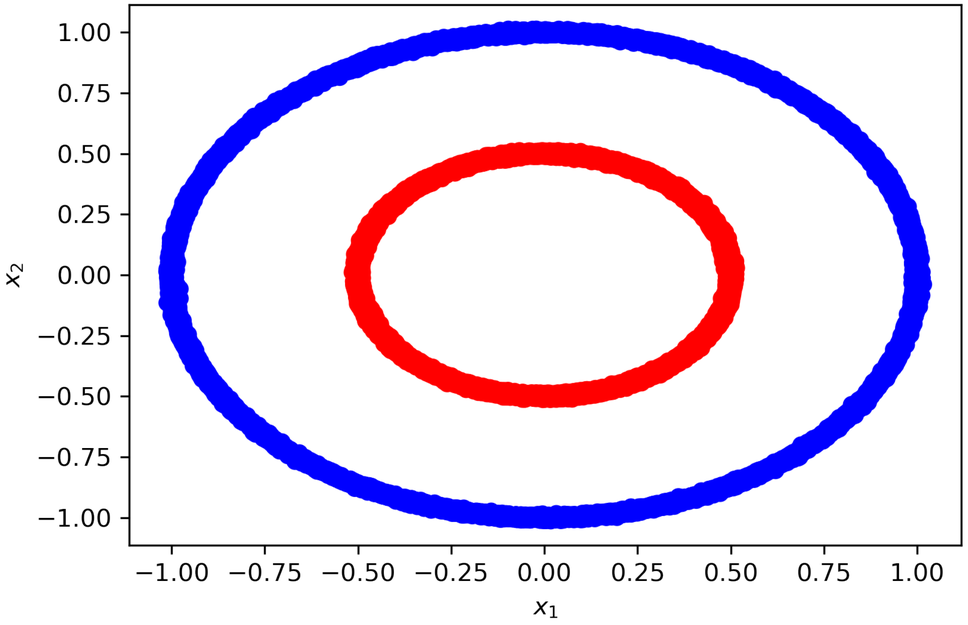
Activation Functions

How does NN learn non-linear decision boundary?

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

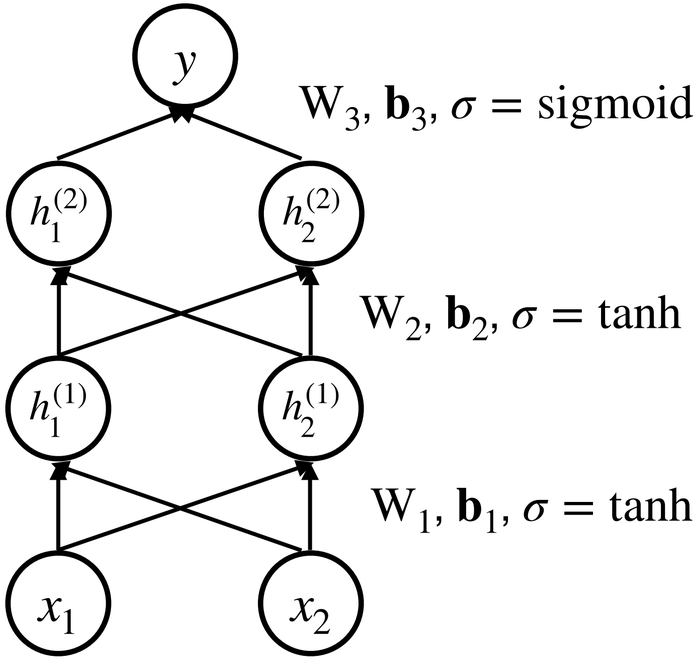
위 식이 $\mathbf{x}$을 어떻게 변환 하는지 이해했다.

그렇다면 Neural Network은 어떻게 빨간색과 파란색의 label을 구분하는 **Decision boundary을 어떻게 학습하는 것일까?**



Activation Functions

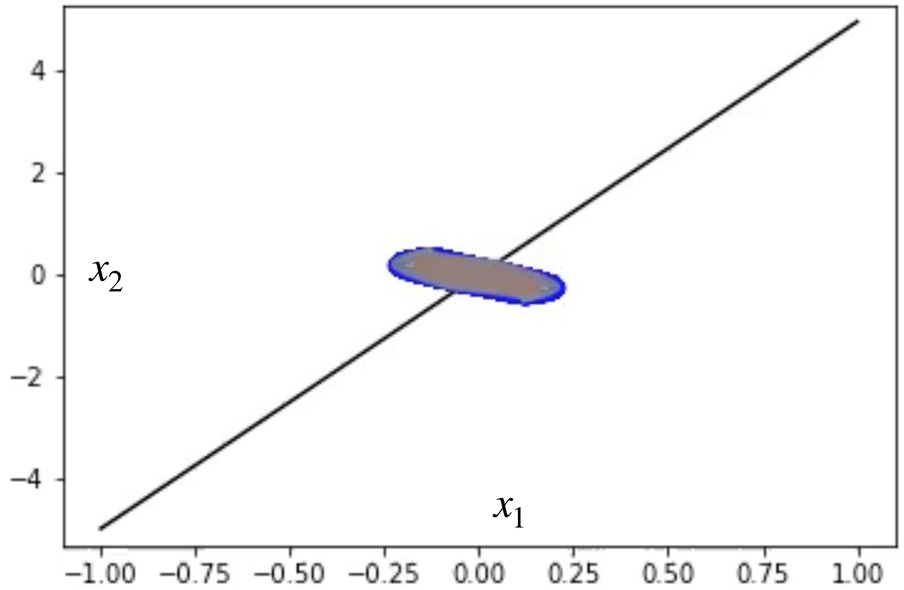
How does NN learn non-linear decision boundary?



2차원 공간에서의 변환

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(2, activation='tanh', input_dim=2))
model.add(Dense(2, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
sgd = keras.optimizers.SGD(lr=0.001)
model.compile(optimizer=sgd,
              loss='mse',
              metrics=['accuracy'])
```



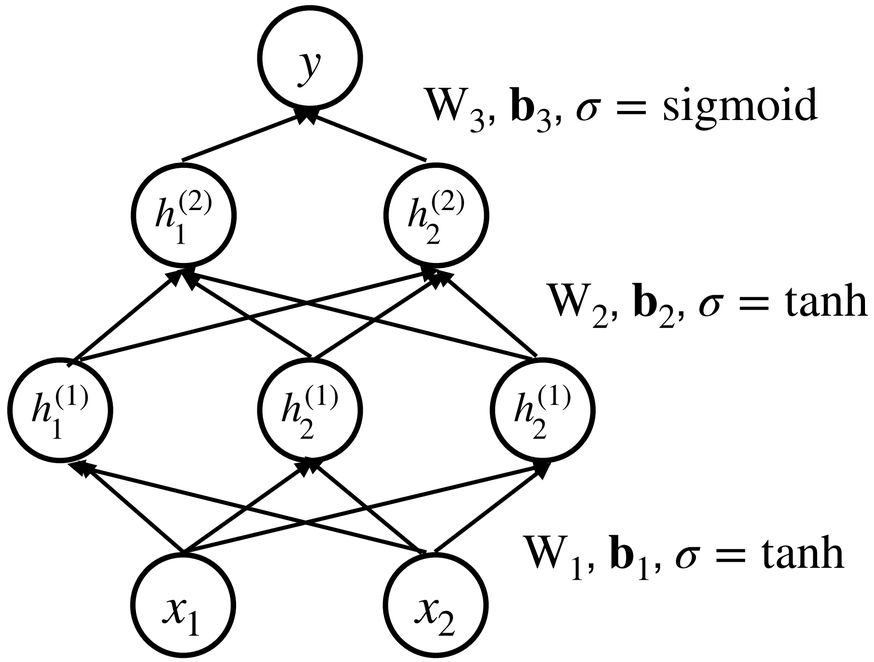
gif 출처: srome.github.io

ACADENTIAL

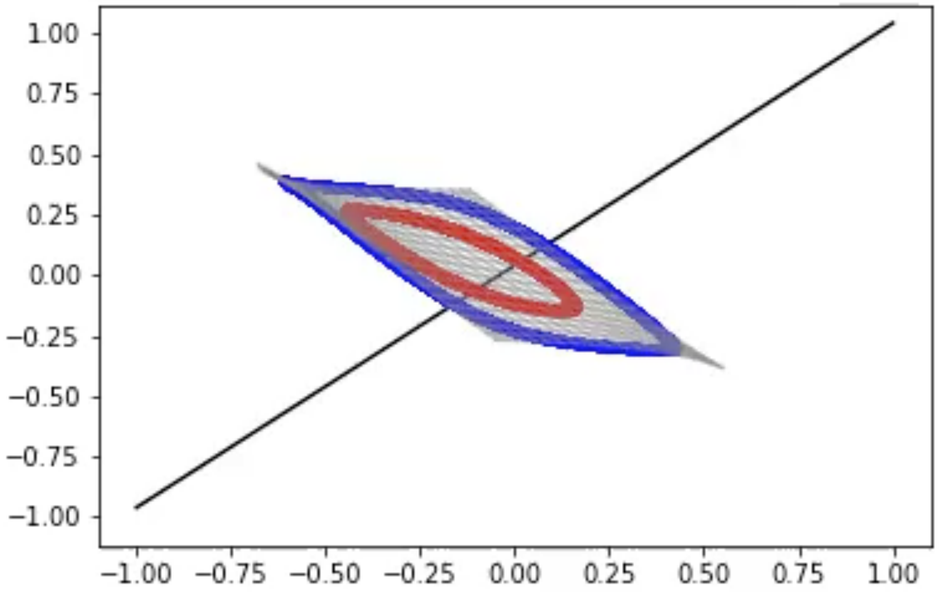
Activation Functions

How does NN learn non-linear decision boundary?

3차원 공간에서의 변환




```
model = Sequential()
model.add(Dense(3, activation='tanh', input_dim=2
model.add(Dense(2, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
sgd = keras.optimizers.SGD(lr=0.005)
model.compile(optimizer=sgd,
              loss='mse',
              metrics=['accuracy'])
```



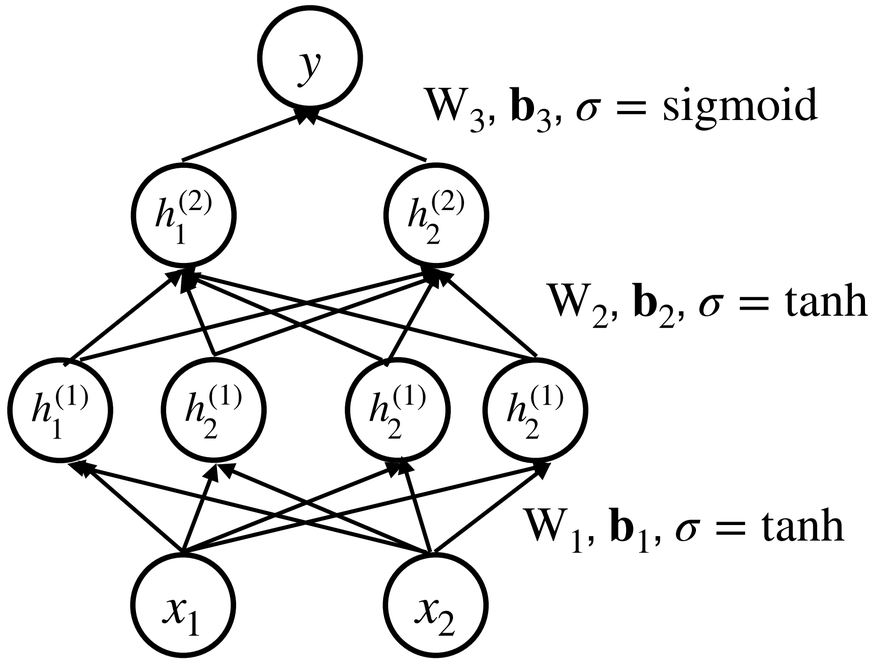
gif 출처: srome.github.io

ACADENTIAL

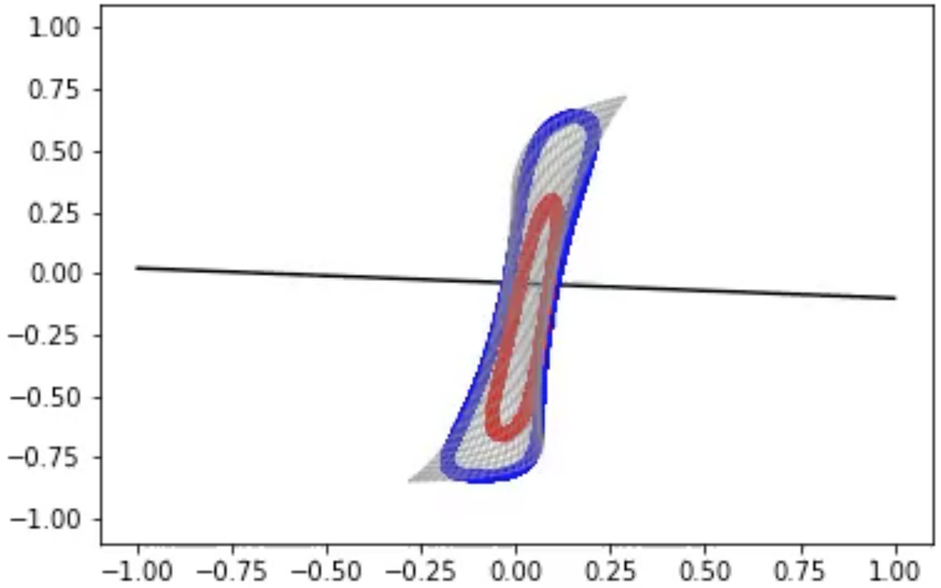
Activation Functions

How does NN learn non-linear decision boundary?

4차원 공간에서의 변환




```
model = Sequential()
model.add(Dense(4, activation='tanh', input_di
model.add(Dense(2, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
sgd = keras.optimizers.SGD(lr=0.005)
model.compile(optimizer=sgd,
              loss='mse',
              metrics=['accuracy'])
```



gif 출처: srome.github.io

ACADENTIAL

Activation Functions

How does NN learn non-linear decision boundary?

Summary:

- $\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$에서 일련의 **"$\mathbf{W}\mathbf{x}$의 affine transformation"** (확장, 회전, 전단)와 **"$\cdot + \mathbf{b}$의 평행 이동"**, **"$\sigma(\cdot)$의 비선형 변환"**을 수행함.
- 그리고
  1. **더 많은 layer 들을 쌓고** ("공간을 여러번 접고 펼치고 당기는 것")
  2. **더 많은 neuron** (즉, 더 큰 dimensional한 space 상에서 변환하는 것)
- 더 복잡한 $\mathbf{x} \rightarrow y$의 mapping을 학습할 수 있다.

ACADENTIAL

7-2. Activation Function의 종류 및 역할

ACADENTIAL

Activation Function

Activation function의 example

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU
- ELU
- Softmax

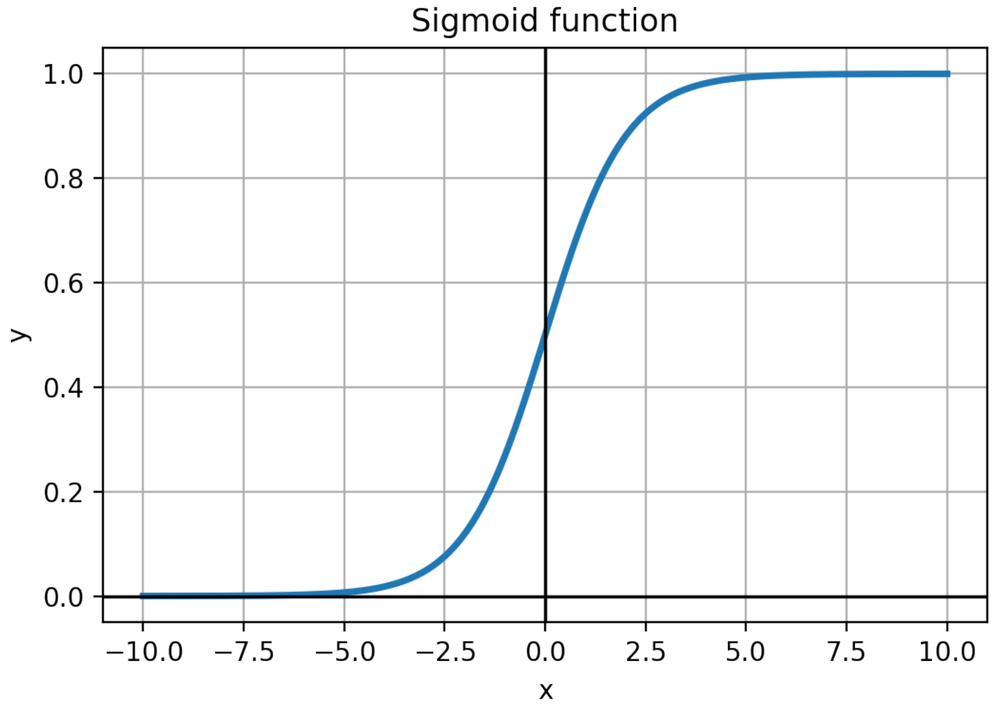
ACADENTIAL

Activation Function

Sigmoid

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

- (-∞, ∞) 범위의 인풋값 x을 0~1 사이의 값으로 변환해준다.
- 뇌의 뉴론과 비교를 해보면
  - $\sigma(x) = 1$: neuron이 다음 뉴론에 전기 신호를 보내는 것 (activated)
  - $\sigma(x) = 0$: 보내지 않는 것 (not activated)

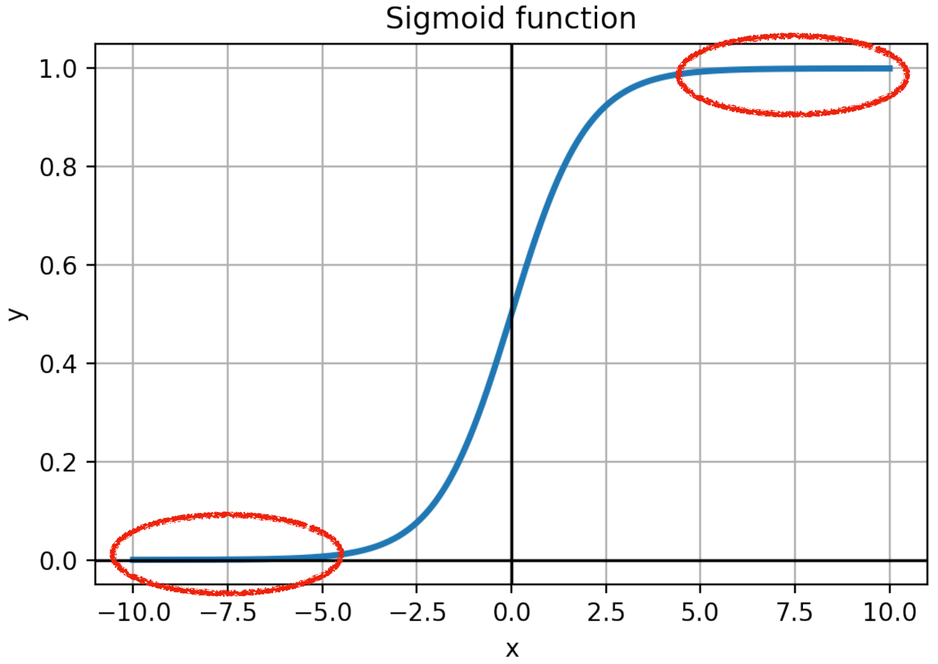


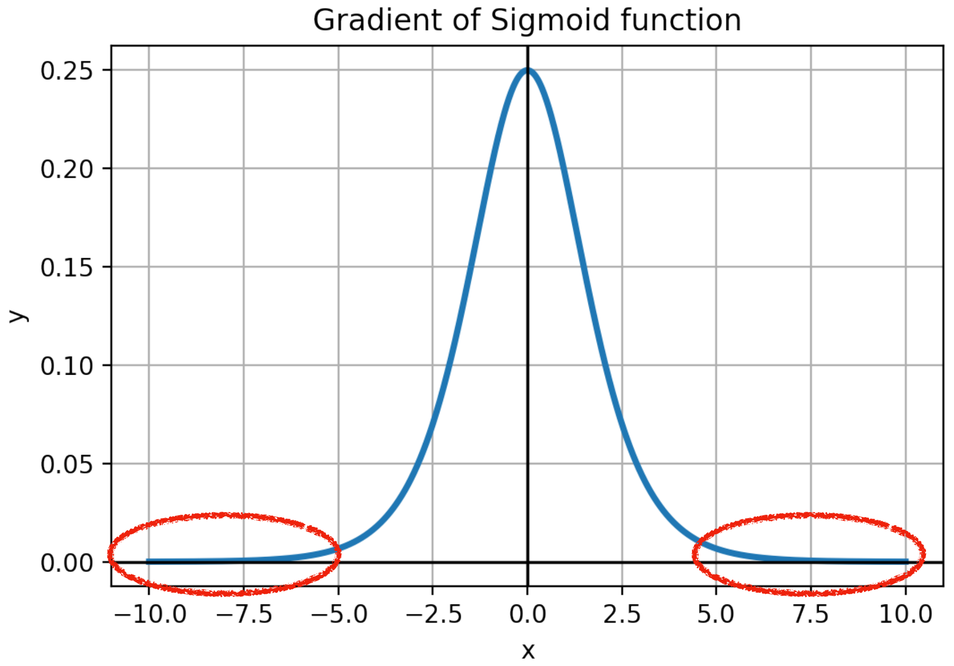
Activation Function

Sigmoid

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

- **Sigmoid 함수의 단점 1**:
  - 양 끝 부분의 지점에서 **Gradient가 거의 0에 가깝다. (saturation)**
  - 즉, **weight가 너무 크거나 작으면** 해당 **weight에 대한 gradient은 0에 가깝고 학습 속도가 매우 느릴 수 있다.**





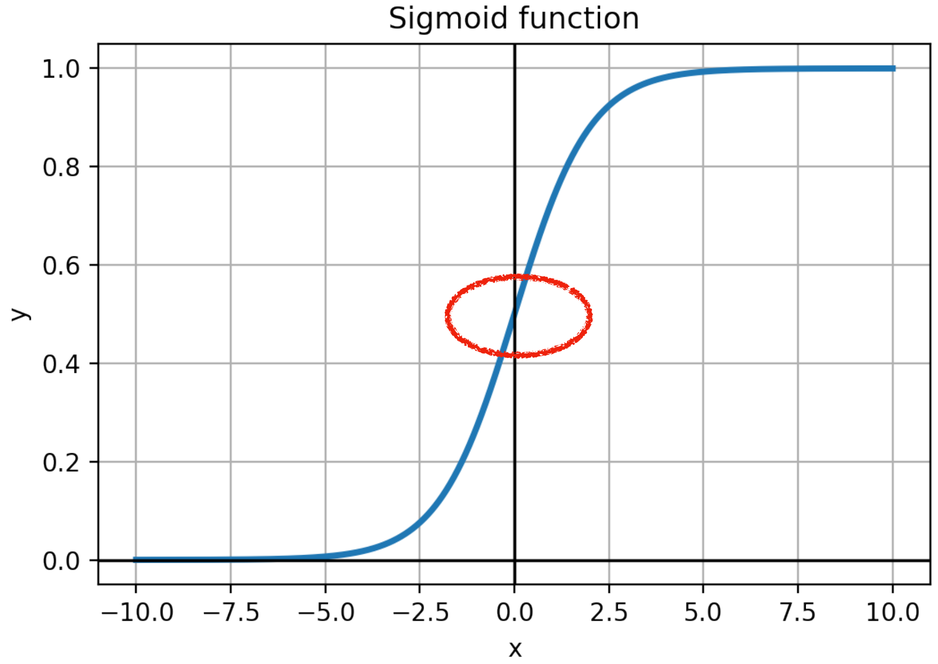
ACADENTIAL

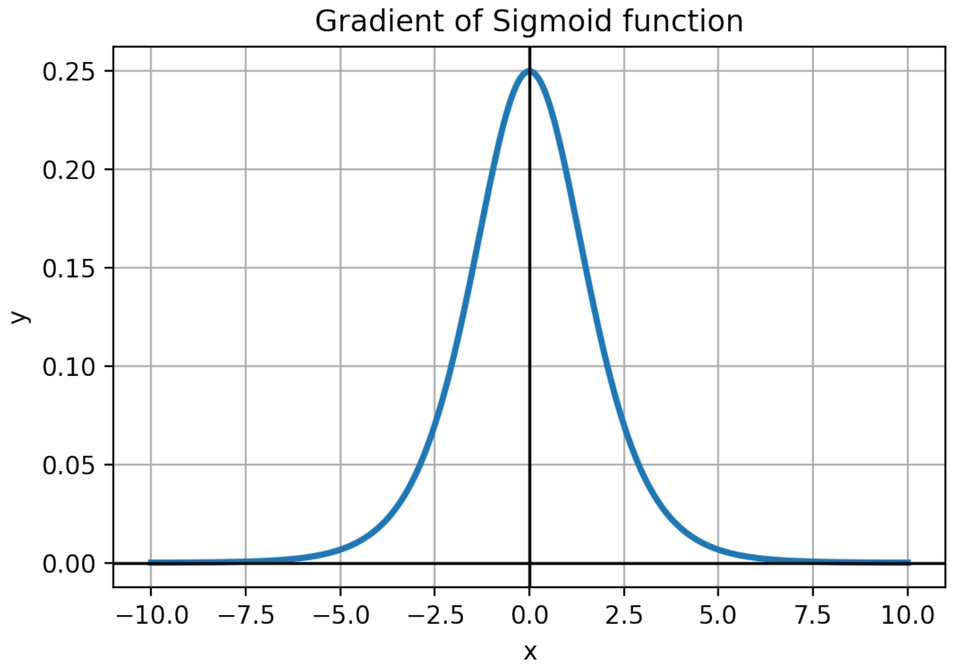
Activation Function

Sigmoid

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

- **Sigmoid 함수의 단점 2**:
  - Sigmoid 함수의 **중심이 되는 값은 0**이 아니라 0.5이고 Sigmoid 함수의 **출력값은 항상 양수**이다.
  - 따라서, sigmoid로 학습할시 지그재그로 움직이는 경향을 보일수 있다!



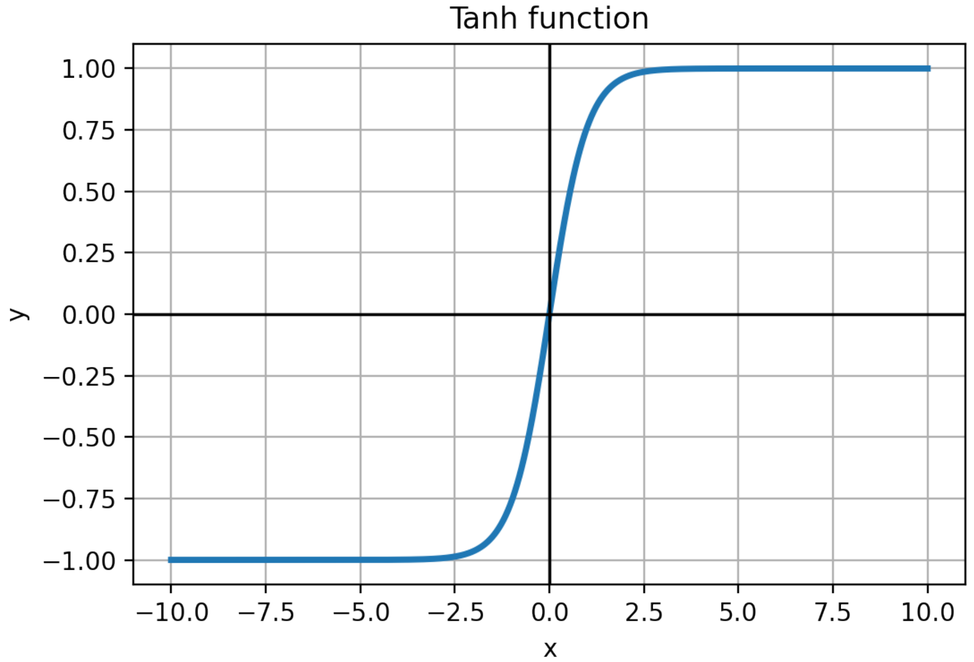


ACADENTIAL

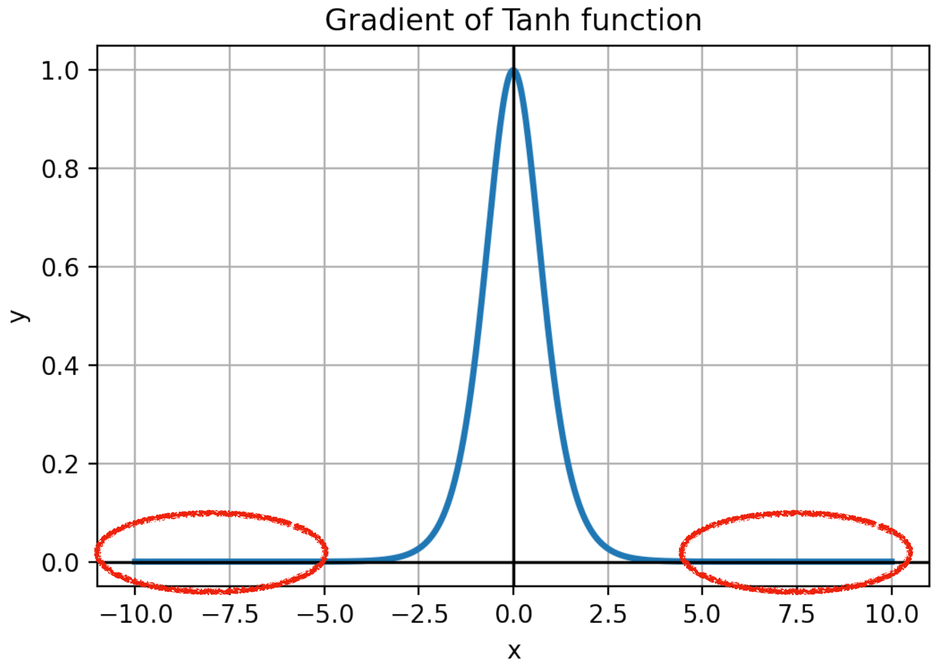
Activation Function

tanh

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



- (-∞, ∞) 범위의 인풋값 x을 -1~1 사이의 값으로 변환해준다.
- Sigmoid 함수와 마찬가지로 양 끝 부분에서 Gradient가 아주 작아지지만 0을 중심값으로 가진다.



ACADENTIAL

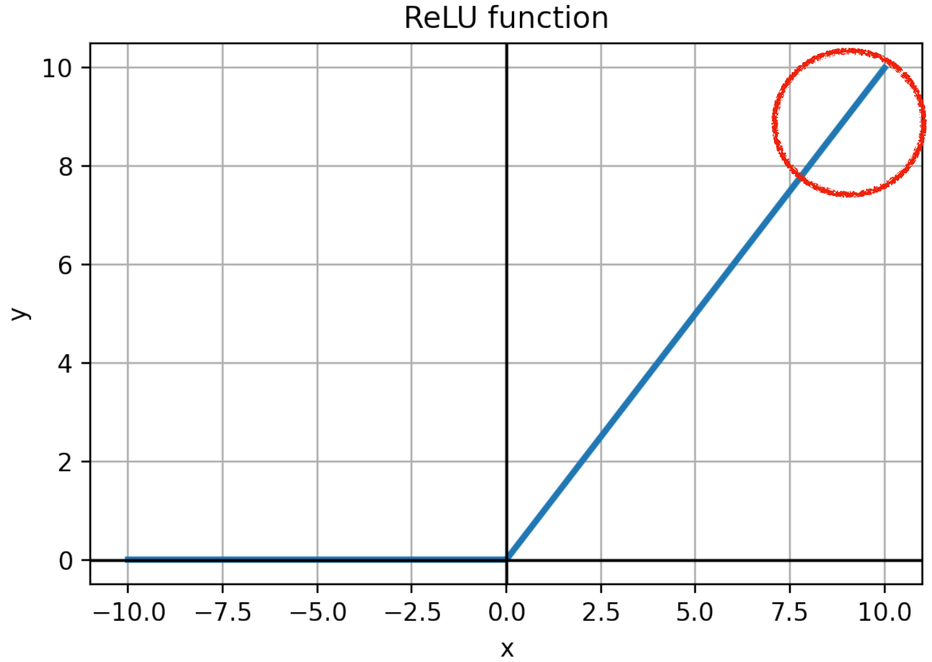
Activation Function

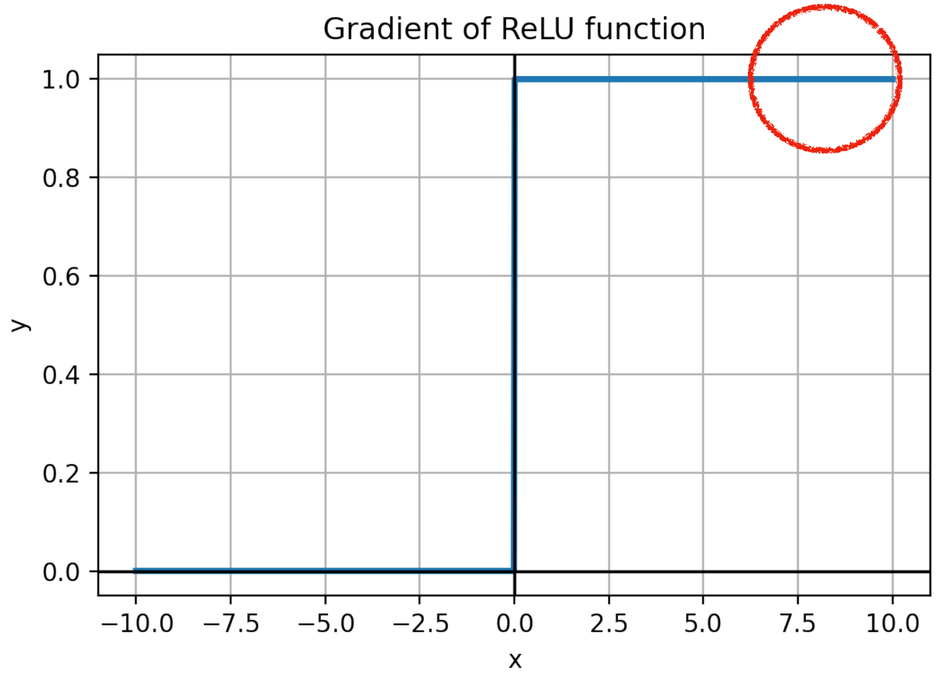
ReLU

$$\mathrm{ReLU}(x) = \max(0, x)$$

장점:

- Sigmoid, Tanh 함수들에 비해서 SGD의 학습 수렴 속도가 빠르다.
- Sigmoid, Tanh에 비해서 saturation의 문제로부터 비교적 자유롭다.





ACADENTIAL

ACADENTIAL

Activation function

ReLU

- ReLU 함수가 사용되는 Neural Network의 예시:
  - VGG (image classification)
  - DenseNet (image classification)
- 주로 Neural Network의 앞, 중간 단계에서 자주 사용된다.

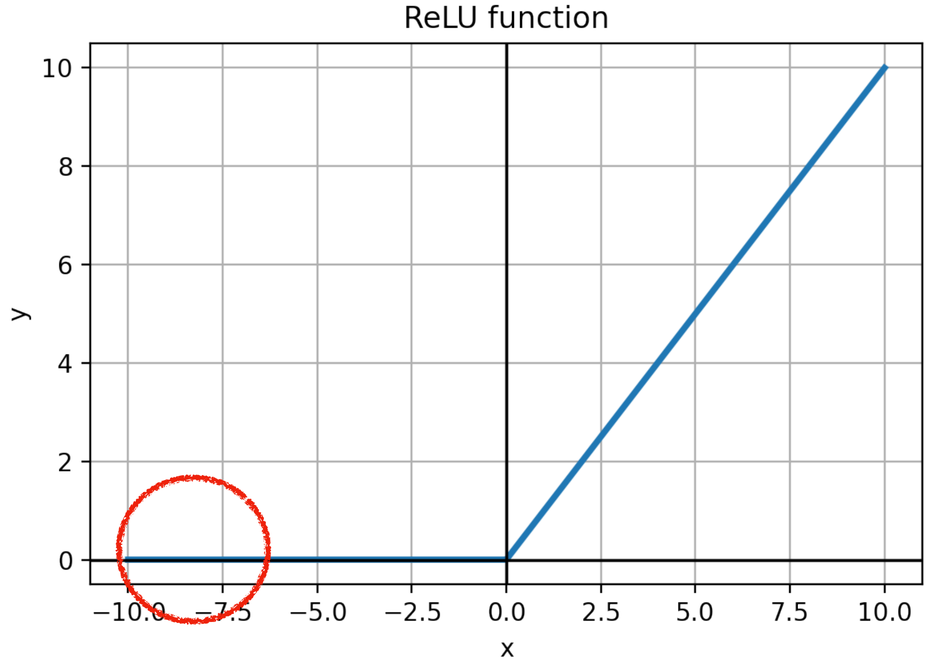
Activation Function

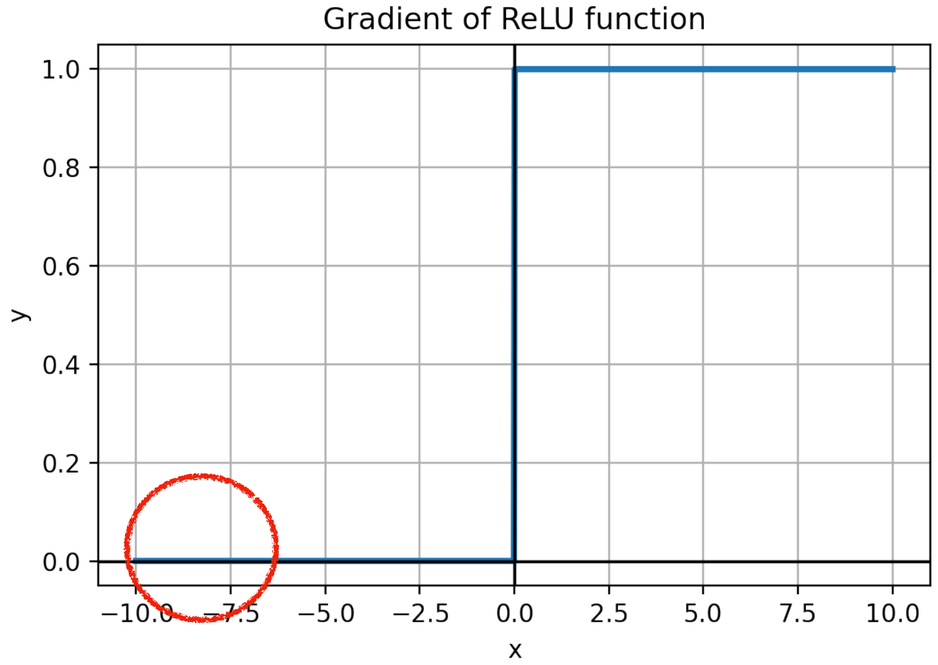
ReLU

$$\mathrm{ReLU}(x) = \max(0,x)$$

단점:

- $x < 0$에 대해서는 Gradient가 0이다.
- 이는 “dead” neuron의 문제점을 야기한다.





ACADENTIAL

Activation Function

ReLU

“dead” neuron이란

- ReLU$(w \cdot x + b)$에서
- **만약에** 모델이 학습하는 과정에서 $b \ll 0$
- 즉, **매우 작은 음수의 값을 가지도록 bias term b을 학습 가정.**

Activation Function

ReLU

“dead” neuron이란

- ReLU$(w \cdot x + b)$에서
- 만약에 모델이 학습하는 과정에서 $b \ll 0$
- 즉, 매우 작은 음수의 값을 가지도록 bias term b을 학습을 하게되면
- $w \cdot x$**와는 상관없이** **ReLU**$(w \cdot x + b) < 0$**이 되어버리고**
- **기울기는** ∇_w**ReLU**$(w \cdot x + b) = 0$

ACADENTIAL

Activation Function

ReLU

“dead” neuron이란

- ReLU$(w \cdot x + b)$에서
- 만약에 모델이 학습하는 과정에서 $b \ll 0$
- 즉, 매우 작은 음수의 값을 가지도록 bias term b을 학습을 하게되면
- $w \cdot x$와는 상관없이 ReLU$(w \cdot x + b) < 0$이 되어버리고
- 기울기는 $\nabla_w \text{ReLU}(w \cdot x + b) = 0$
- **따라서** $w \rightarrow w - \lambda \cdot \frac{df(w,x)}{dw}^{=0}$
- **해당 뉴론의 weight에 대한 기울기는 항상 0이 되어버려 w가 바뀌지 않고 그대로 “죽어버리는” 문제점을 의미한다!**

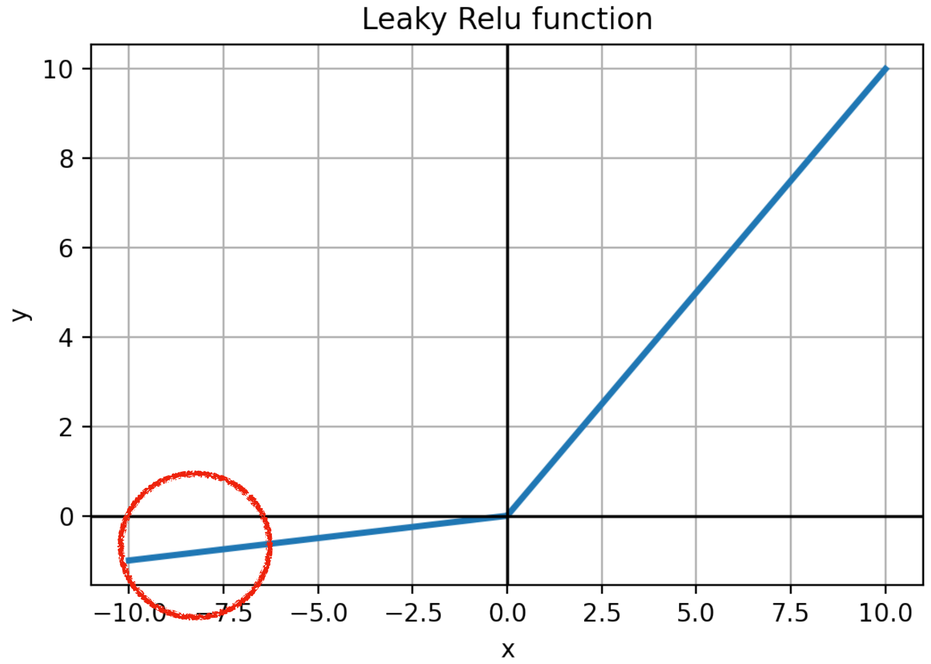
ACADENTIAL

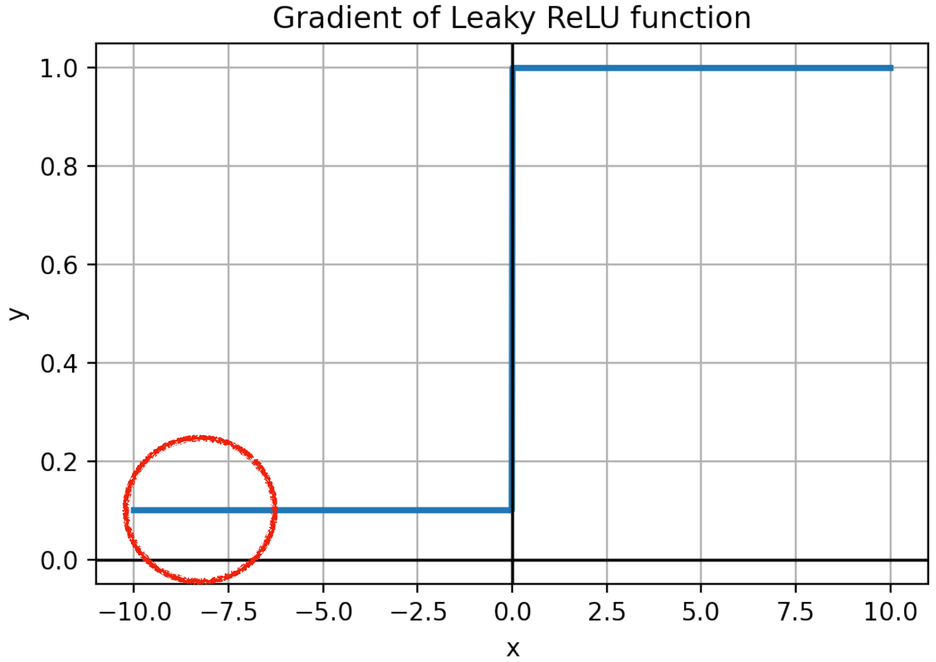
Activation Function

Leaky ReLU

$$\text{LeakyReLU}(x) = \max(0.1x, x)$$

- ReLU의 “dead” neuron 문제점을 해결하기 위해서 제안됨.
- $x \ll 0$에 대해서도 non-zero Gradient값을 가진다!





ACADENTIAL

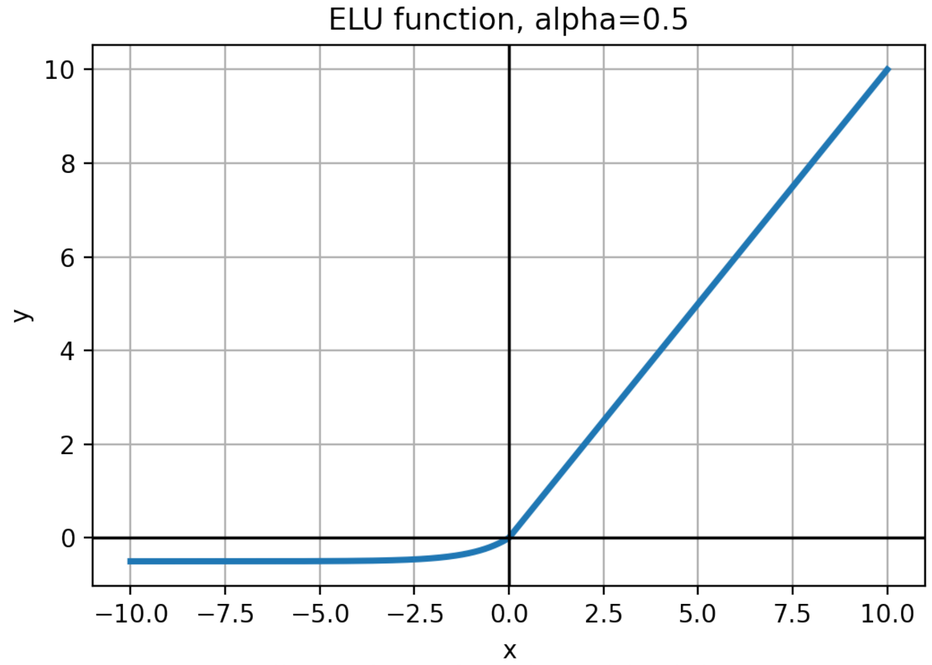
Activation Function

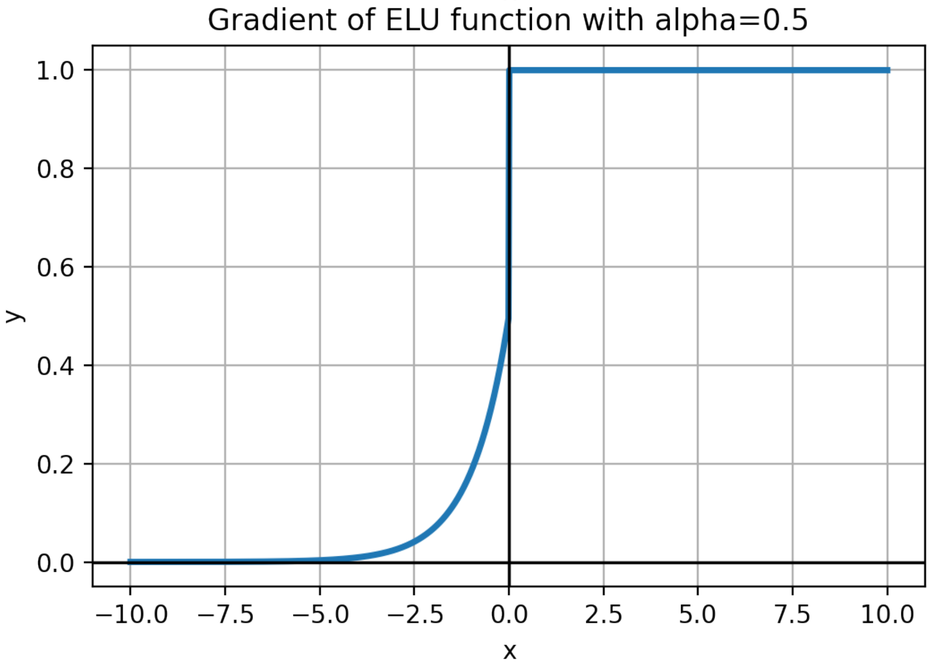
ELU

$$\mathrm{ELU}(x) = x \qquad \text{if} \quad x > 0$$

$$\alpha(e^x - 1) \qquad \text{else}$$

- ELU의 경우 $x < 0$에 대해서도 non-zero gradient을 가진다.
- 하지만 $x \ll 0$에 대해서는 saturate하여 gradient가 0에 수렴한다.





ACADENTIAL

Activation Function

Softmax

Softmax

$$\text{Softmax}(\mathbf{x}, T)_i = \frac{e^{x_i}/T}{\sum_{j=0} e^{x_j}/T}$$

- 사용 목적: Neural Network의 마지막 classification layer에서 **output한 logit**의 값을 **normalization** 해주는 역할.

Activation Function

Softmax

10개 class에 대한 classification task을 예로 들어보자. 마지막 layer에서 출력된 예측값은 다음과 같다.

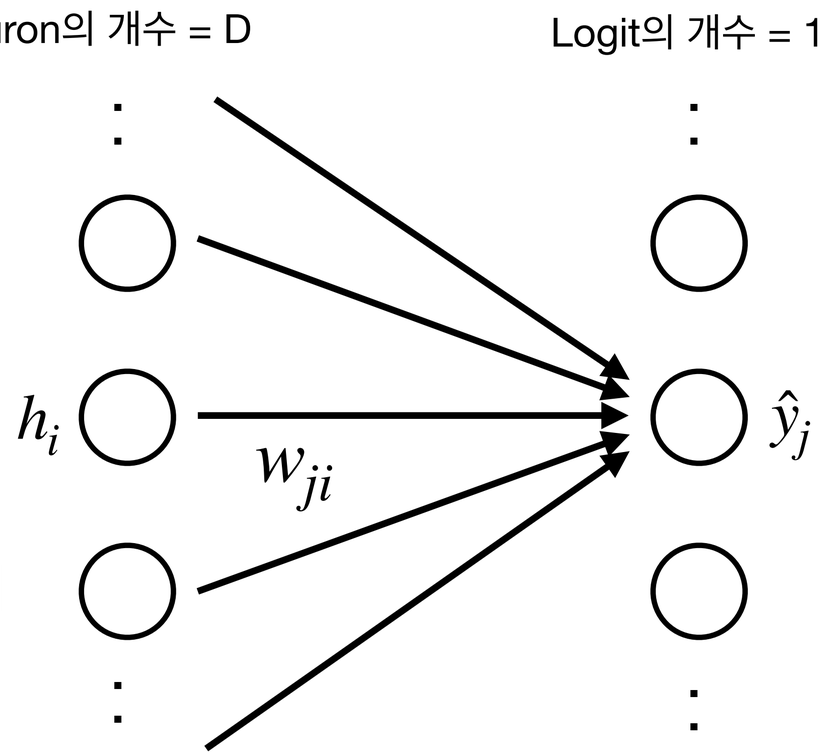
$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{h} + \mathbf{b}$$

$\hat{\mathbf{y}} \in \mathbb{R}^{10}, \mathbf{h} \in \mathbb{R}^{D}, \mathbf{b} \in \mathbb{R}^{10}, \mathbf{W} \in \mathbb{R}^{D \times 10}$

이때 CE Loss$(\hat{\mathbf{y}}, \mathbf{y})$을 계산하고 싶다고 했을때, $\hat{y}_j$은 [0, 1] 사이의 값이어야 한다.

참고:

- 주로 마지막 layer에서 (Softmax 취하기 전에) 출력된 값을 logit이라고 부른다.
- CE Loss = Cross Entropy Loss



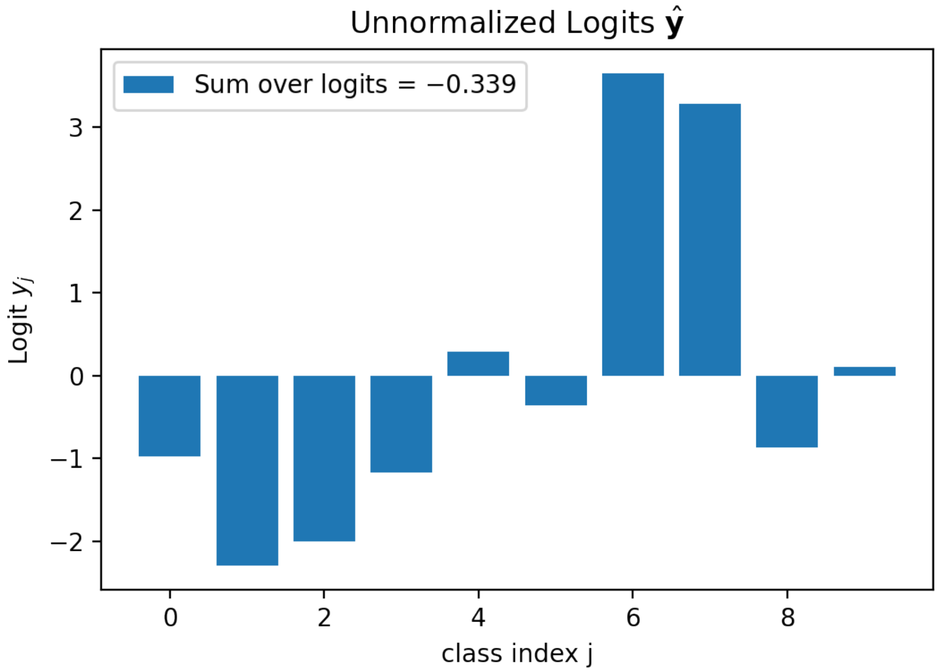
$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{h} + \mathbf{b} = \sum_i w_{ji} \cdot h_i + b_i$$

ACADENTIAL

Activation Function

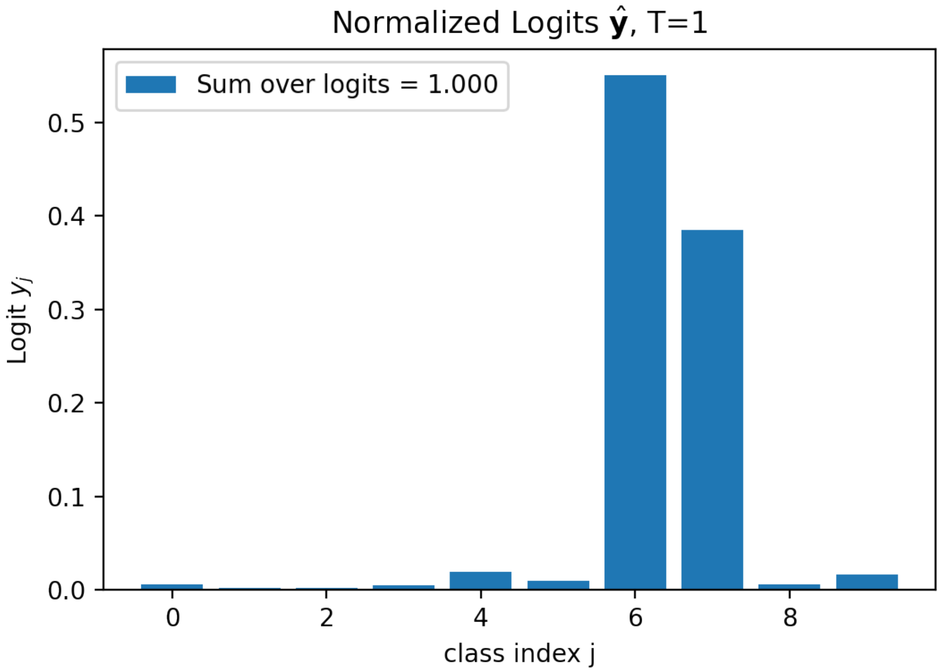
Softmax

$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{h} + \mathbf{b}$$



$$\text{Softmax}(\hat{\mathbf{y}}, T = 1)_i = \frac{e^{\hat{y}_i}}{\sum_{j=0} e^{\hat{y}_j}}$$

$$\text{Softmax}(\hat{\mathbf{y}}, T = 1)$$



ACADENTIAL

Activation Function

Softmax - What does temperature T does?

Softmax

$$\text{Softmax}(\mathbf{x}, T)_i = \frac{e^{x_i}/T}{\sum_{j=0} e^{x_j}/T}$$

Temperature T의 효과:

- T가 높을수록 softmax의 출력값들이 더 “넓게”, “고르게” 퍼진다.
- T가 낮을수록 softamx의 출력값들이 더 “좁게”, “쏠린다”.

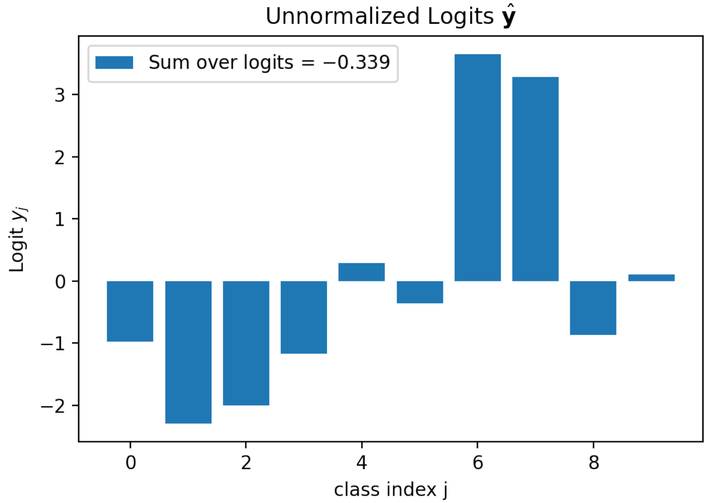
Activation Function

Softmax

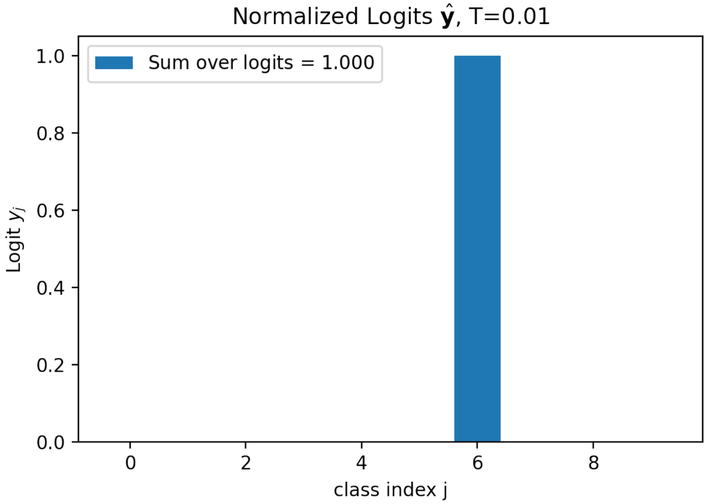
참고 사항:

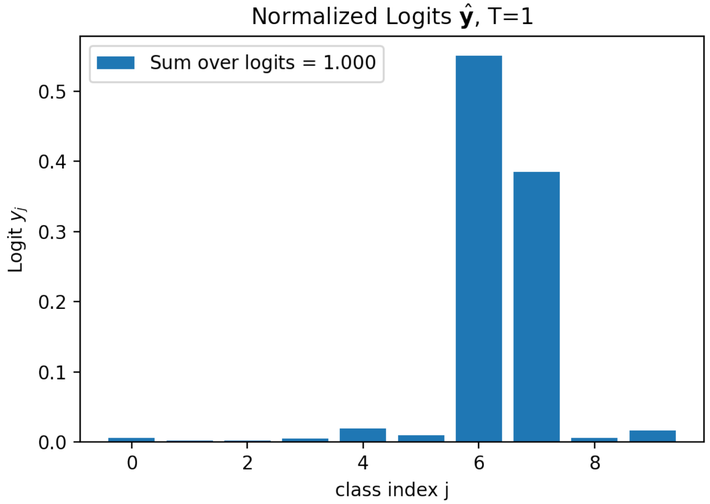
- T 값이 달라도 Argmax을 취했을시 결과는 같다.
- T 값이 달라도 총합은 1로 모두 동일하다.

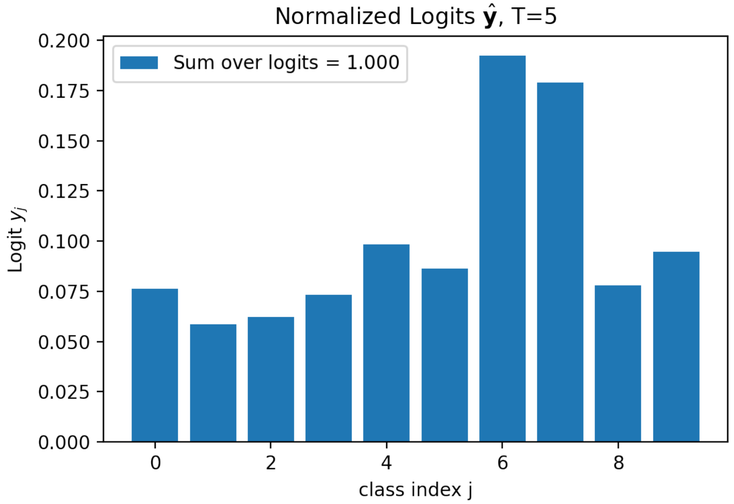
Before Softmax



T = 0.01 < T = 1 < T = 5







Entropy = 1.04 e-14 < Entropy = 0.989 < Entropy = 2.21

ACADENTIAL

참고 사항: What is Entropy?

$$\text{Entropy}(\mathbf{p}) = -\sum_i p_i \log(p_i)$$

- Entropy = 불확실성에 대한 정도 (measure of uncertainty)
- Entropy가 높을수록 불확실성이 높고 (probability distribution이 넓게 퍼져있다)
- Entropy가 낮을수록 불확실성이 낮은 것 (probability distribution이 쏠려있다)
- Temperature가 낮을수록 → Softmax logit의 분포가 쏠린다 → Entropy가 낮다
- Temperature가 높을수록 → Softmax logit의 분포가 넓게 퍼진다 → Entropy가 높다

ACADENTIAL

7-3. Pytorch로 구현해보는 Activation Function

ACADENTIAL

7-4. Section 7 요약

ACADENTIAL

Section Summary

Activation Function

- (Non-linear) Activation Function은 non-linear한 복잡한 decision boundary을 학습하기 위해 필요하다!

ACADENTIAL

Section Summary

Activation Function

- (Non-linear) Activation Function은 non-linear한 복잡한 decision boundary을 학습하기 위해 필요하다!

- Activation Function의 non-linearity 덕분에 feature space을 “휘고, 꺾거나, 왜곡시켜서” non-linear한 decision boundary을 그려낼 수 있다.

ACADENTIAL

Section Summary

Activation Function

- (Non-linear) Activation Function은 non-linear한 복잡한 decision boundary을 학습하기 위해 필요하다!
- Activation Function의 non-linearity 덕분에 feature space을 “휘고, 꺾거나, 왜곡시켜서” non-linear한 decision boundary을 그려낼 수 있다.
  - non-linearity 덕분에 feature space의 각 grid은 non-uniform하게 변형된다!

ACADENTIAL

Section Summary

Activation Function

- (Non-linear) Activation Function은 non-linear한 복잡한 decision boundary을 학습하기 위해 필요하다!
- Activation Function의 non-linearity 덕분에 feature space을 “휘고, 꺾거나, 왜곡시켜서” non-linear한 decision boundary을 그려낼 수 있다.
  - non-linearity 덕분에 feature space의 각 grid은 non-uniform하게 변형된다!
- linear한 activation function을 사용하면 아무리 많은 Layer들을 쌓아도 Single Layer Neural Network에 불과하다!

ACADENTIAL

Section Summary

Activation Function

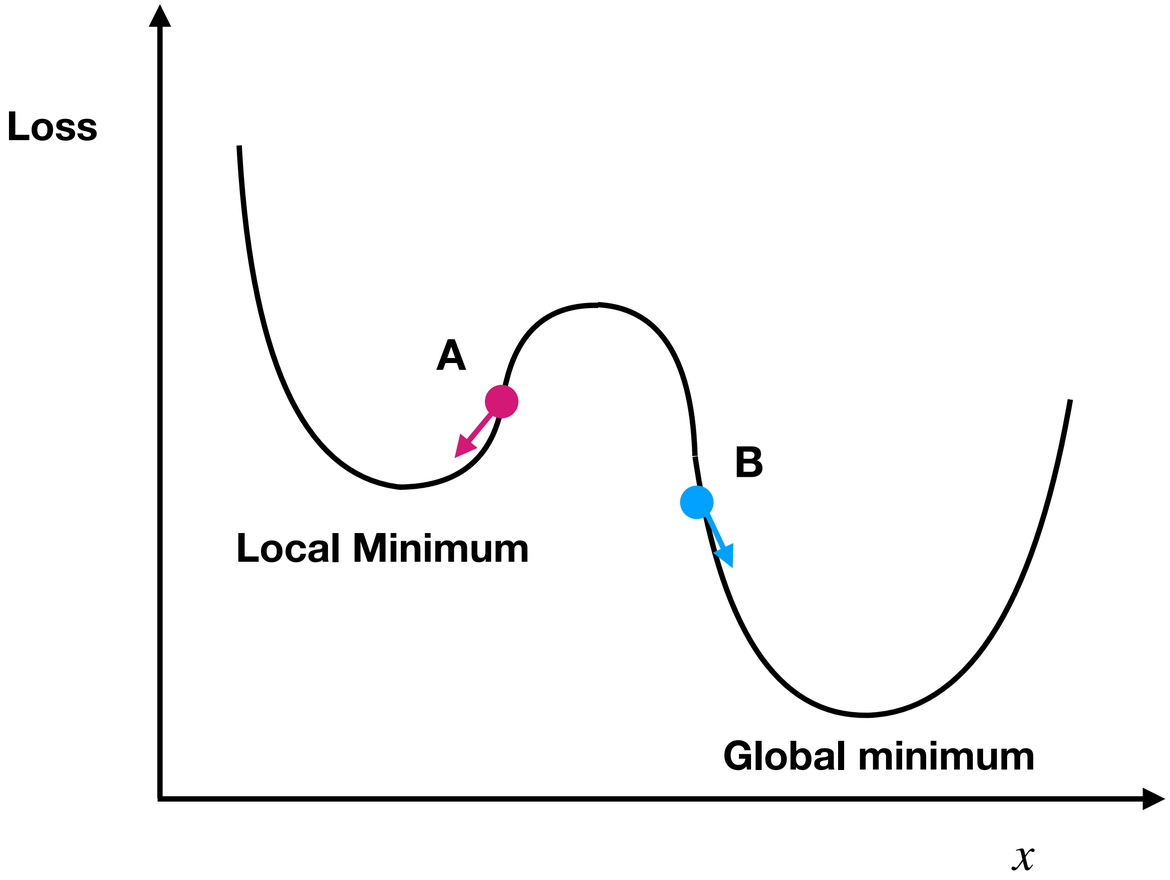
- 다음과 같은 Activation function의 종류들을 살펴보았다!
  - Sigmoid
  - Tanh
  - ReLU
  - Leaky ReLU
  - ELU
  - Softmax

Activation Function 종류	장점	단점	Comment
Sigmoid		단점 1: Gradient saturation 단점 2: Sigmoid 함수의 출력값은 항상 양수이다. SGD 학습시 지그재그로 움직이는 경향을 보일수 있다!	범위: $(-\infty, +\infty) \rightarrow (0,1)$
Tanh	장점 1: 중심값이 0이다. (Sigmoid의) 지그재그로 움직이는 문제 해소	단점 1: Gradient Saturation	범위: $(-\infty, +\infty) \rightarrow (-1,1)$
ReLU	장점 1: Sigmoid, Tanh 함수들에 비해서 SGD의 학습 수렴 속도가 빠르다. 장점 2: Sigmoid, Tanh에 비해서 saturation의 문제 일부 해소	단점 1: x < 0에 대해서는 Gradient가 0이다. “dead” neuron의 문제점을 야기한다.	CNN에서 많이 사용됨.
Leaky ReLU	(ReLU의 장점) 장점 3: Dead Neuron의 문제점 해소		CNN에서 많이 사용됨.
ELU	(ReLU의 장점) 장점 3: Dead Neuron의 문제점 일부 해소	단점 1: x << 0 에 대해서는 Gradient가 0이다. 이 때는 Dead Neuron의 문제점 발생함.	CNN에서 많이 사용됨.
SoftMax			Neural Network의 마지막 classification layer에서 사용됨 output한 logit의 값을 normalization 해주는 역할.

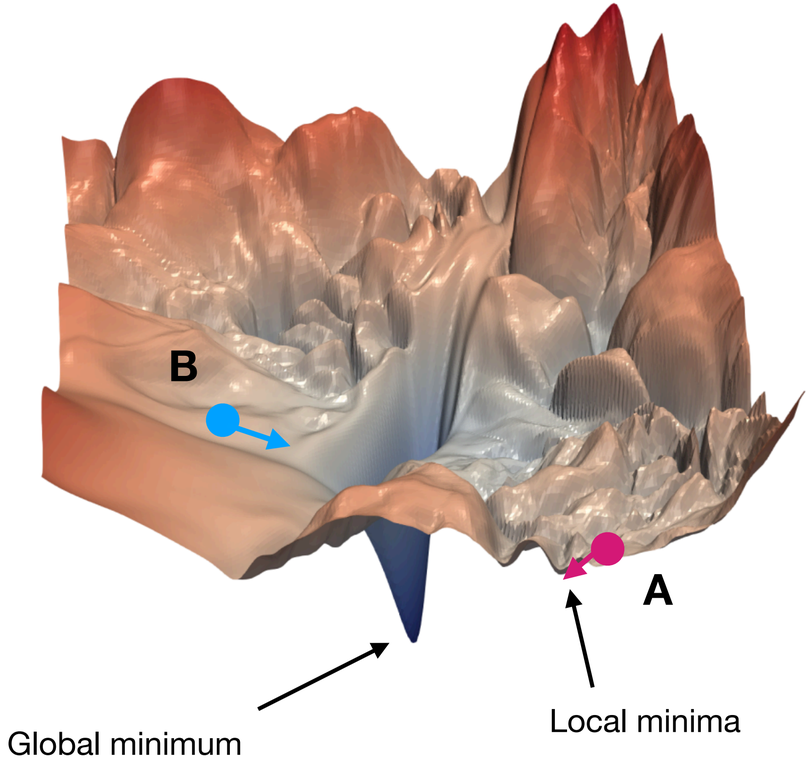
ACADENTIAL

Next Up!

Next Up!
Initialization



출처: Visualizing the Loss Landscape of Neural Nets (NeurIPS 2018)

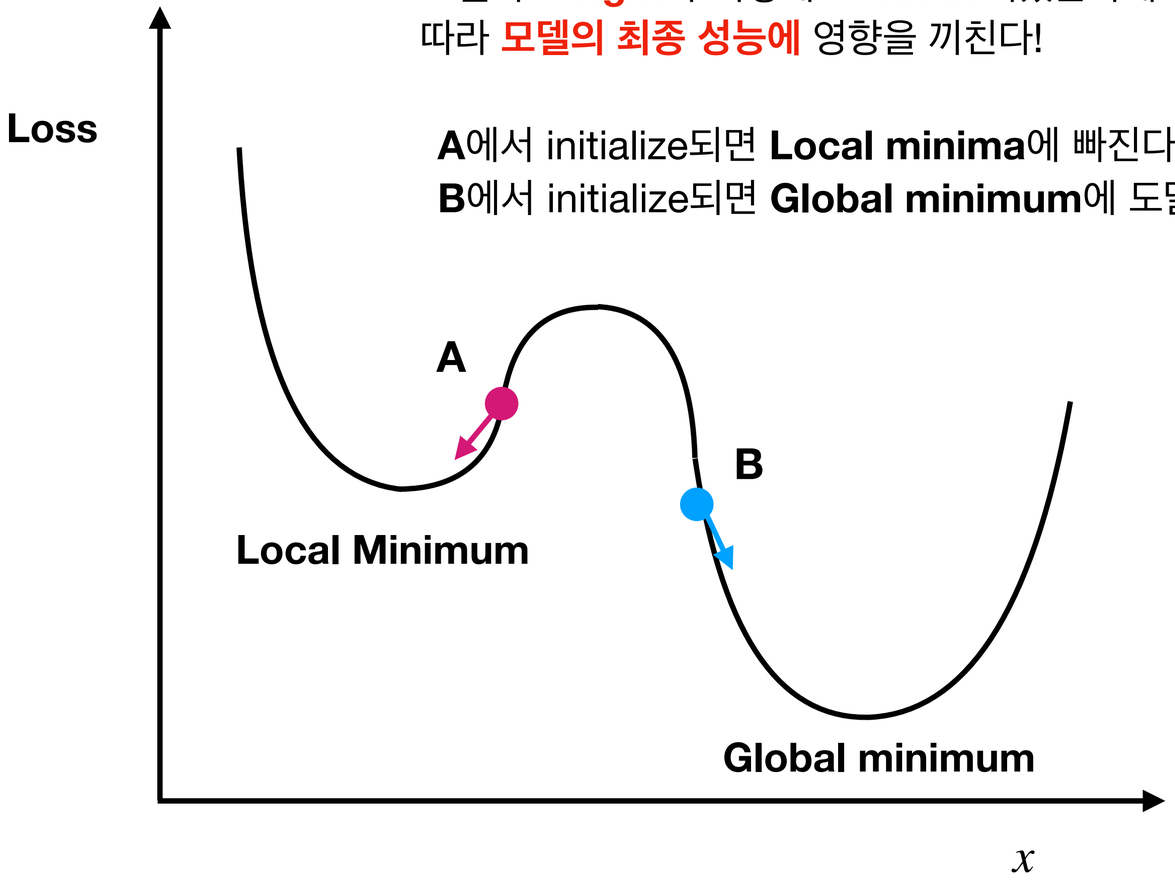


Next Up!

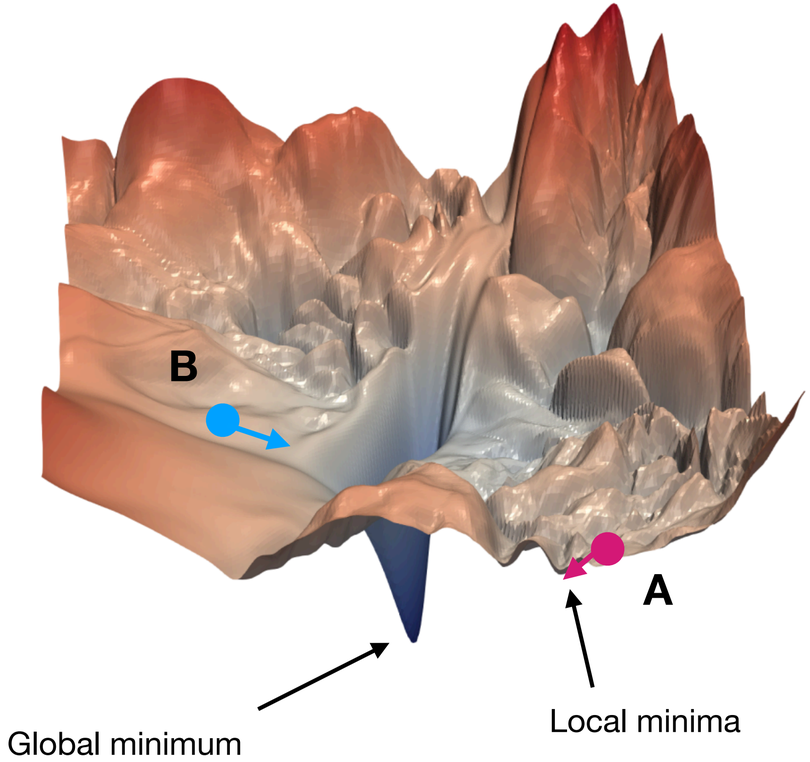
Initialization

모델의 **weight**가 어떻게 **initialize**되었는가에
따라 **모델의 최종 성능에** 영향을 끼친다!

A에서 initialize되면 **Local minima**에 빠진다.
B에서 initialize되면 **Global minimum**에 도달할 수 있다!



출처: Visualizing the Loss Landscape of Neural Nets (NeurIPS 2018)



Next Up!
Initialization

다음 Chapter에서는 **initialization**에 대해서 살펴보자!