주피터 노트북

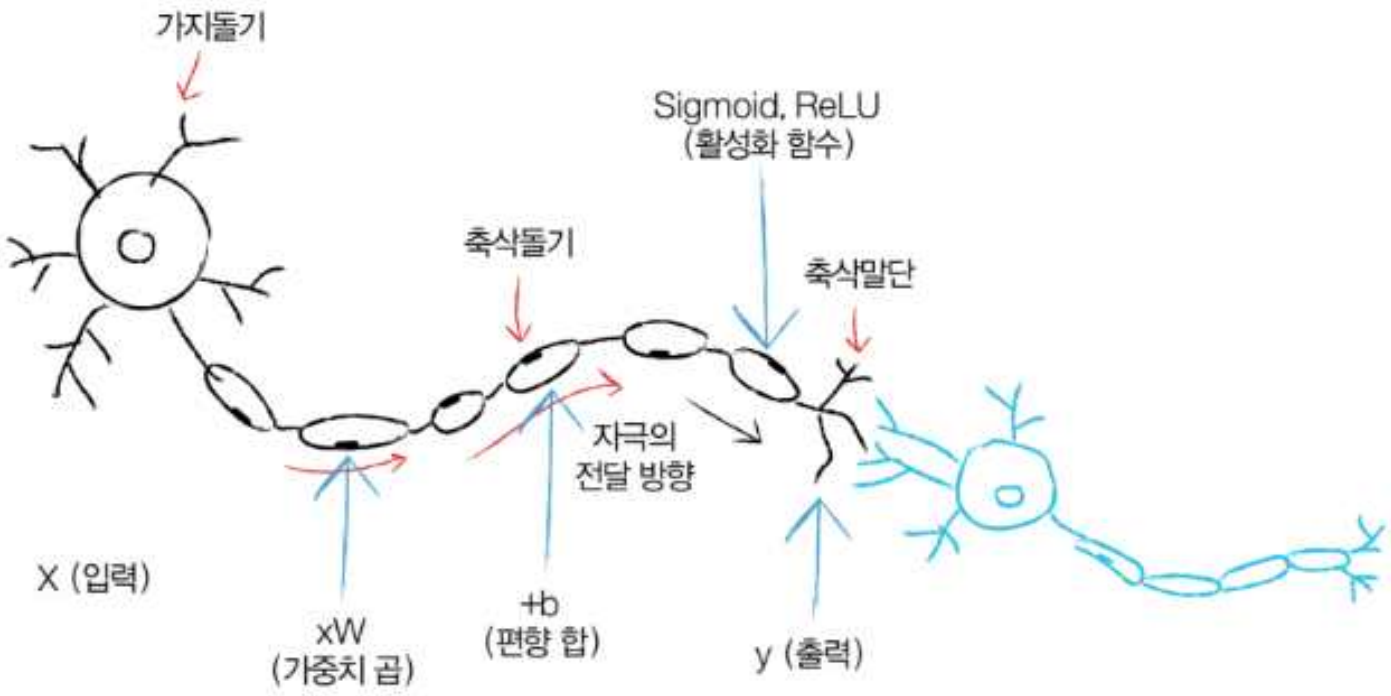
사용 설명

김 대 환

2022.

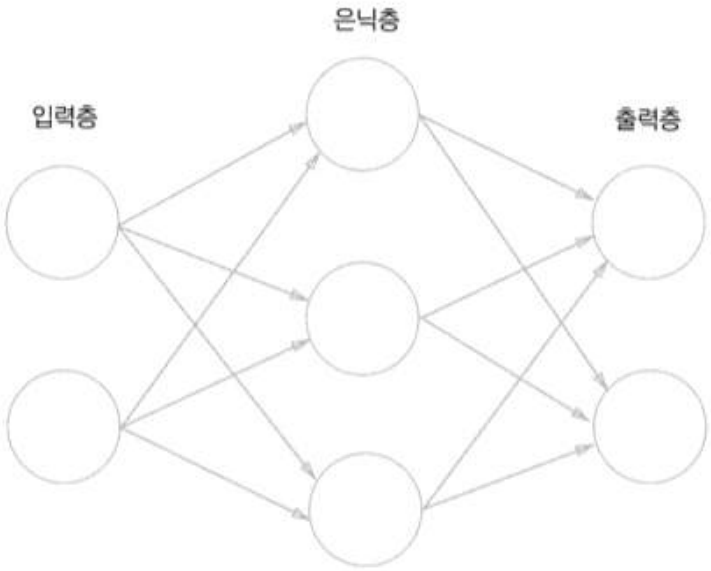
인공 신경망(Artificial Neural Network)

- **뉴런(Neuron)**과 신경망의 원리
  - 입력 값 X에 가중치(W)를 곱하고 편향(b)을 더한 뒤, 활성화 함수(Sigmoid, ReLU 등)을 거쳐 결과 값 y를 만들어내는 것이 인공 뉴런의 동작 원리



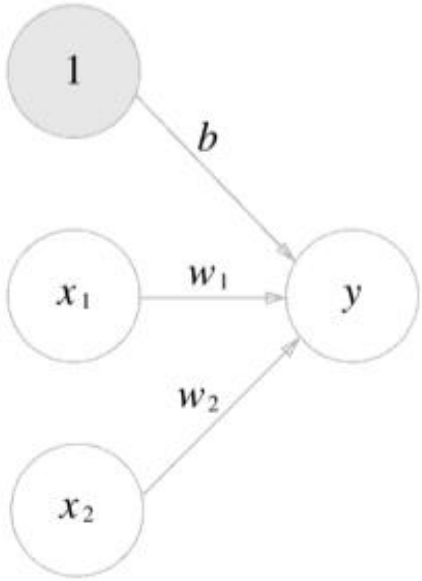
인공 신경망(Artificial Neural Network)

- 원리
  - 원하는 출력 y값을 위해 적절한 W와 b 값을 찾는 최적화 과정을 학습(learning) 또는 훈련(training)이라 한다
  - 퍼셉트론에서는 원하는 출력 값을 갖도록 수동으로 가중치 값을 선택하지만, 신경망에서는 데이터를 학습하여 가중치 값을 자동으로 조절한다
  - 신경망은 활성화 함수에서 퍼셉트론과 큰 차이를 보인다



인공 신경망(Artificial Neural Network)

- 입력 **x**에 대한 수식이 **0**을 넘으면 **1**을, 그렇지 않으면 **0**을 출력하는 예
  - 아래 그림은 편향 b (bias)가 1인 뉴런이 추가
  - 입력 신호의 총합이 h(x) 함수를 거쳐 변환되고, 그 변환된 값이 y의 출력



<편향을 명시한 퍼셉트론>

$$y = \begin{cases} 0\ (b + w_1x_1 + w_2x_2 \leq 0) \\ 1\ (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$

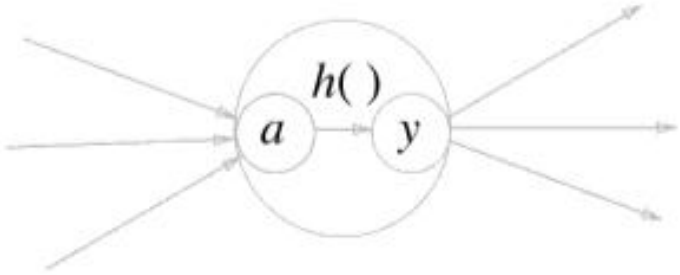
<퍼셉트론을 표현한 수식>

$$y = h(b + w_1x_1 + w_2x_2)$$

$$h(x) = \begin{cases} 0\ (x \leq 0) \\ 1\ (x > 0) \end{cases}$$

인공 신경망(Artificial Neural Network)

- **활성화 함수 (Activation Function)**
  - 입력 신호의 총합을 출력 신호로 변환하는 h(x) 함수를 활성화 함수라 함
  - 아래 그림은 입력 신호와 편향의 총합을 계산하는 a를 활성화 함수 h()에 넣어서 y를 출력
    - ✓ 딥러닝 네트워크에서는 노드에 들어오는 값들을 곧바로 다음 레이어로 전달하지 않고, 주로 **비선형 함수**를 통과시켜 전달한다



  - 계단 함수(Step function)는 임계 값을 경계로 출력이 변하는 활성화 함수
    - ✓ 퍼셉트론은 활성화 함수로 계단 함수를 사용
  - 활성화 함수에는 다양한 종류가 있다

활성화 함수

- 계단 함수의 구현 예
  - 아래 함수는 실수**(부동 소수점)**만 입력으로 사용

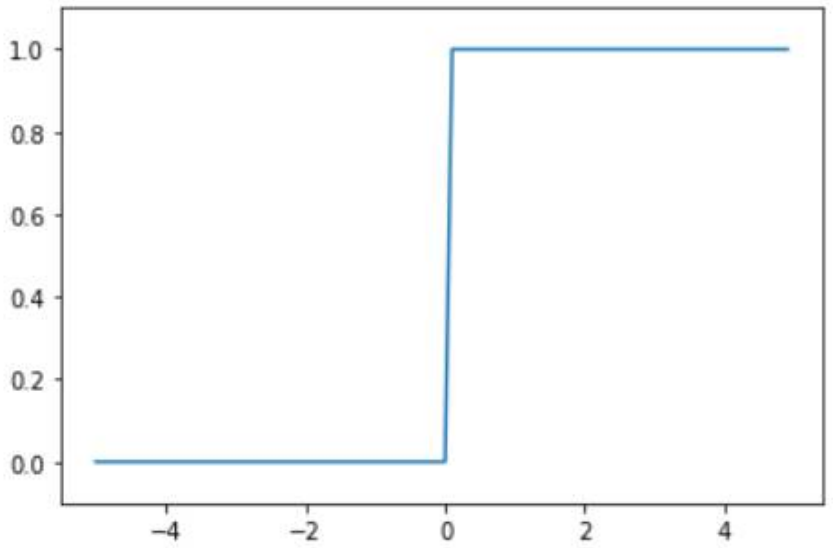
```
import numpy as np

x = np.array([-1.0, 1.0, 2.0])
print(x)

y = x > 0
print(y)

# 출력을 bool에서 int형으로 변경하고 싶을때
y = y.astype(np.int)
print(y)
```

```
# 출력결과:

[-1. 1. 2.]

[False True True]

[0 1 1]
```

활성화 함수

- 계단 함수의 그래프 표현
  - np.arange(-5.0, 5.0, 0.1)은 -5.0에서 5.0 전까지 0.1 간격의 넘파이 배열을 생성

```
import numpy as np
import matplotlib.pylab as plt

def step_function(x):
  return np.array(x > 0, dtype=np.int)

x = np.arange(-5.0, 5.0, 0.1)
y = step_function(x)
plt.plot(x, y)
plt.ylim(-0.1, 1.1) # y축 범위 지정
plt.show()
```



활성화 함수

- 시그모이드 **(Sigmoid)** 함수
  - 신경망에서 자주 사용되는 활성화 함수
  - 0과 1사이의 값을 가지는 비선형 함수
    - ✓ 보통 은닉층에서는 잘 사용하지 않고 이진 분류(0, 1) 출력을 가지는 출력층에서만 사용

```
def step_function(x):
  return (x > 0, dtype=np.int)
```

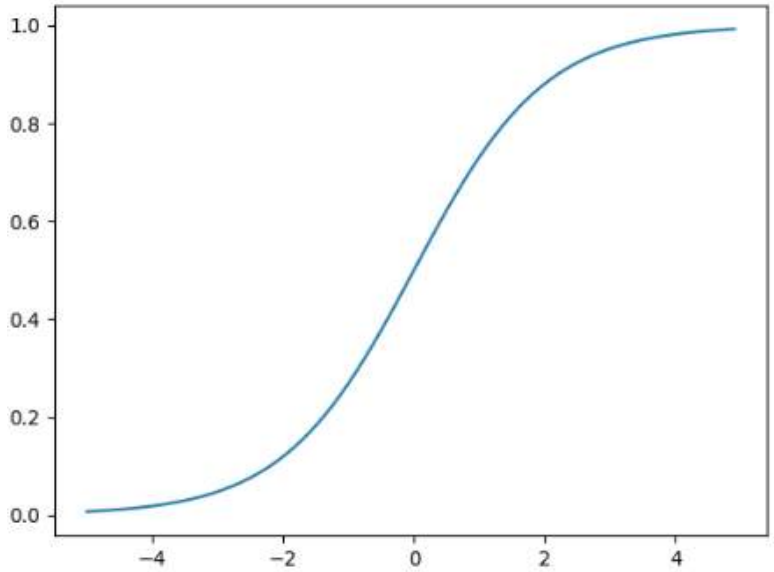
$$h(x) = \frac{1}{1 + exp^{(-x)}}$$

(exp는 자연상수)

활성화 함수

- 시그모이드 함수의 그래프 표현
  - 입력에 따라 출력이 연속적으로 변하며 부드러운 곡선으로 표현된다
  - 계단 함수는 0과 1 중 하나의 값만 반환하지만 시그모이드 함수는 실수 값을 반환

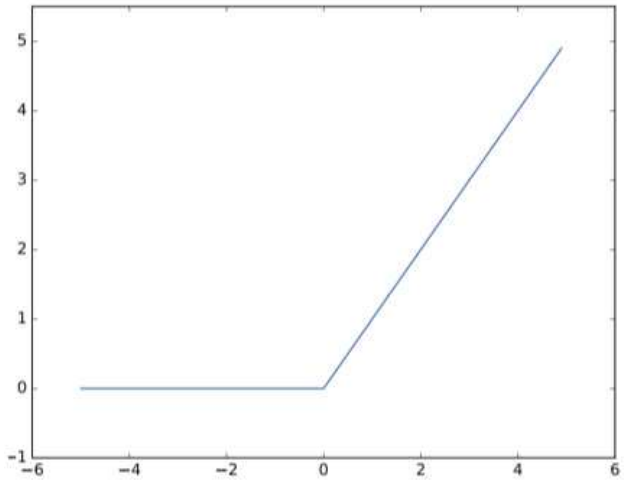
```
import numpy as np
import matplotlib.pylab as plt

def sigmoid(x):
    return 1/(1 + np.exp(-x))

x = np.arange(-5.0, 5.0, 0.1)
y = sigmoid(x)
plt.plot(x, y)
plt.ylim(-0.1, 1.1)
plt.show()
```



활성화 함수

- **ReLU(Rectified Linear Unit) 함수**
  - 최근 시그모이드 함수 대신 많이 사용
  - 입력이 0을 넘으면 입력 그대로 출력하고, 0 이하면 0을 출력
  - Gradient Vanishing 문제(layer가 늘어날 때 값이 사라지는 현상) 해결로 가장 기본적인 활성화 함수로 사용
  - 파이썬 코드로 아래와 같이 구현할 수 있음

```
def relu(x):
  return np.maximum(0, x)
```



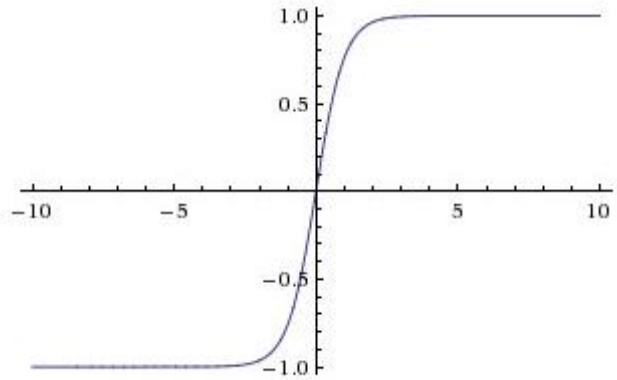
$$h(x) = \begin{cases} 0\,(x \leq 0) \\ x\,(x > 0) \end{cases}$$

기타 활성화 함수

- **Tanh Function (하이퍼볼릭탄젠트 함수)**
  - -1 ~ 1 사이의 값을 가지며, 0 ~ 1 사이의 값을 갖는 Sigmoid Function을 평행이동한 것과 동일
  - 데이터 중심을 0으로 위치시키는 효과가 있기 때문에, 다음 층의 학습이 더 쉽게 이루어진다.

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

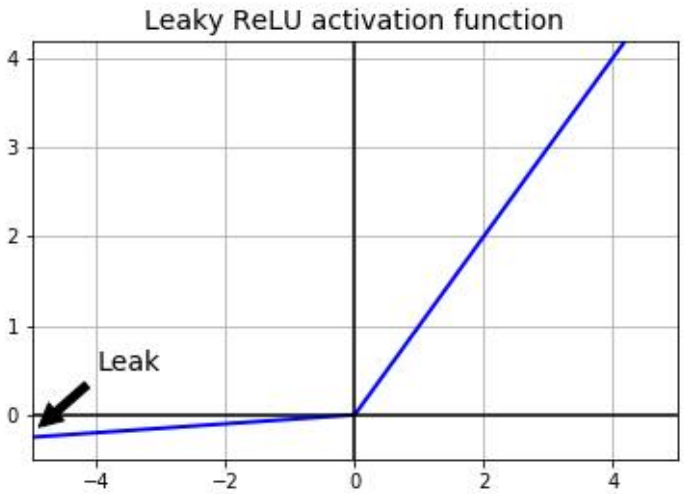
$$tanh'(x) = 1 - tanh^{2}(x)$$



기타 활성화 함수

- **leaky RELU Function (리키 렐루 함수)**
  - x가 음수일 때 미분값이 0이 되는 RELU 함수 대신, 약간의 기울기를 갖는 함수
  - 대개 ReLU 보다 잘 작동하지만 실제로는 잘 사용하지는 않는다

$$f(x) = max(0.01x, x)$$



참고: Gradient Vanishing 정의 및 원인

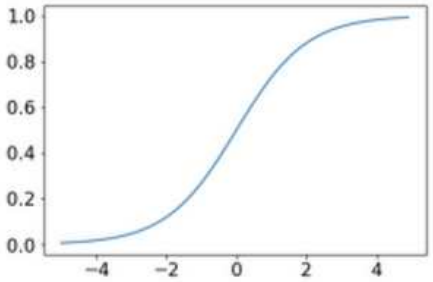
- **MLP** 학습 방법인 **Backpropagation** 중, **Gradient** 항이 사라지는 문제
  - Cost Function의 Gradient 항이 0이나 0에 가까워져 학습이 불가능해지는 현상

$$w^{+} = w - \eta * \frac{\partial E}{\partial w}$$

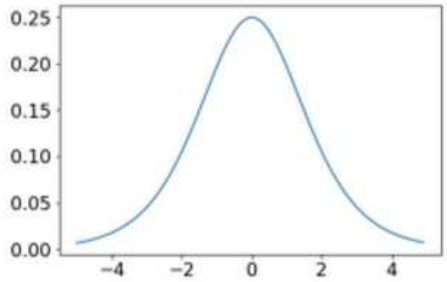
learning rate : 한번에 얼마나 학습할지

gradient : 어떤 방향으로 학습할지

  - Activation Function으로 Sigmoid Function (혹은, Tanh Function)을 사용했기 때문
    - ✓ 도함수를 살펴보면 최대값이 0.25로, 망이 깊어질수록 Gradient가 1/4씩 줄어든다는 의미

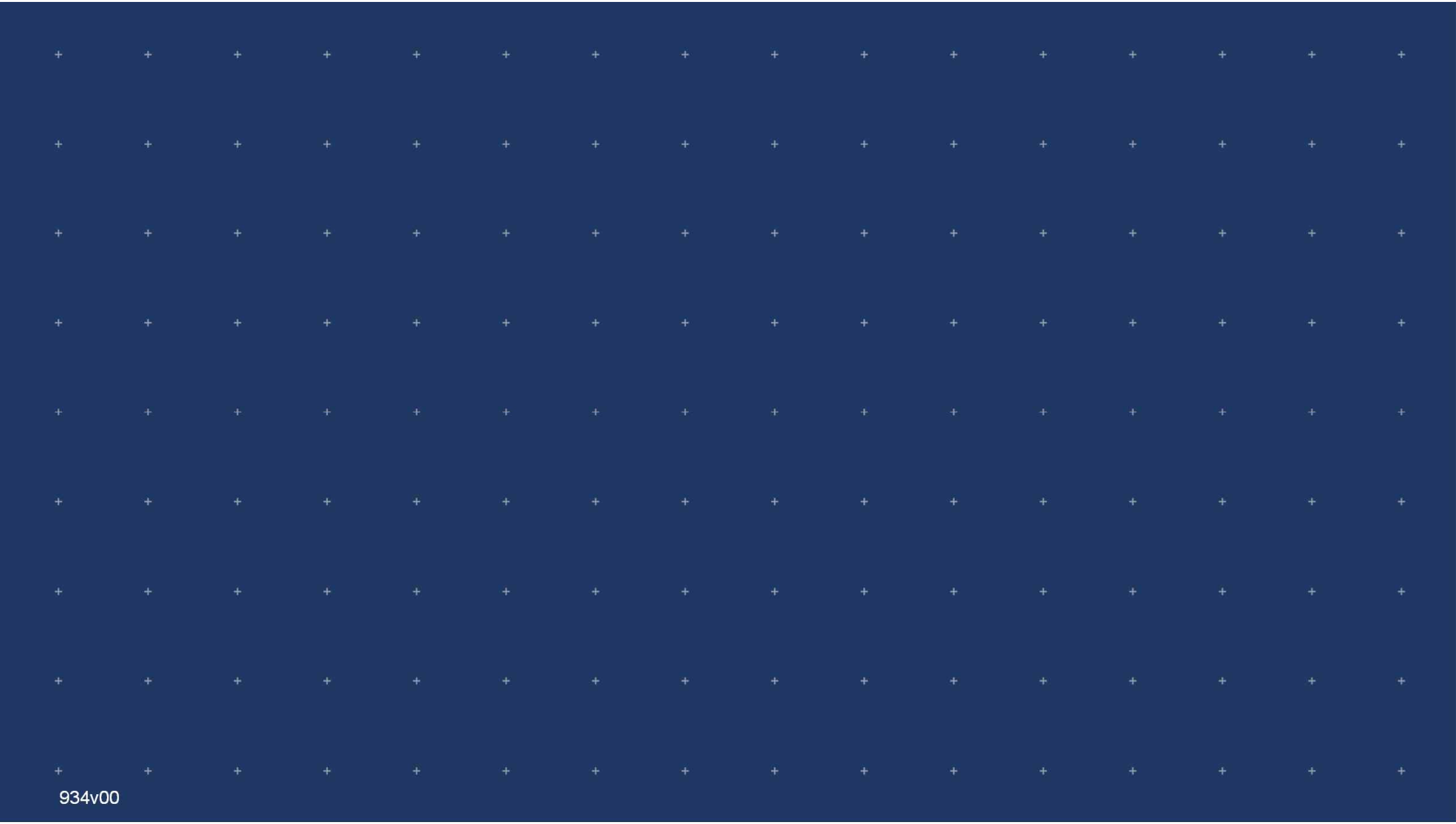


$$Sigmoid(x) = \frac{1}{1+e^{-x}}$$



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

(0 , ¼] 최대점 ¼) 에서 수평


934v00