1 <처음보는 용어집>

1. Few shot learning : a form of prompt engineering in generative AI 로, 매우 적은 수의 라벨이 지정된 예제를 학습하여 정확한 예측을 하는 법을 배우는 머신러닝 프레임워크이다. 그래서 일반적으로 적절한 훈련 데이터가 부족할 때 분류 작업을 위한 모델을 훈련시키는 데 사용된다. 소수의 사례에서 배울 수 있는 인간의 능력을 모방하는 것을 목표로 한다. 즉 지도학습과 대조된다. (출처: 위키피디아, IBM)

2. Prompt engineering : 생성 AI 모델로 해석되고 이해될 수 있는 명령을 구조화하는 과정으로, 프롬프트에서 일시적으로 배울 2 수 있는 모델의 능력으로 정의된 문맥 학습에 의해 활성화된다. (출처: 위키피디아)

3. Retrieval Augmented Generation : 외부 소스에서 가져온 사실로 생성 AI 모델의 정확성과 신뢰성을 향상하는 기술이다. 즉 LLM 이 작동하는 방식에 대한 격차를 메운다. (출처: 위키피디아)

4. Automatic differentiation(자동 미분) : 컴퓨터 프로그램에 의해 지정된 함수의 부분 미분을 평가하는 기술로, 체인 규칙을 반복적으로 적용한다. (출처: 위키피디아)

5. Multi-Layer Perceptron(MLP) : 적어도 3 개의 층으로 구성된 비선형 활성화 기능을 가진 완전히 연결된 뉴런으로 구성된 현대 인공 신경망으로, 선형으로 분리할 수 없는 데이터를 구별할 수 있는 것으로 유명하다. 또한 역전파 방법을 사용하여 훈련된다. (출처: 위키피디아)

6. Bias-variance tradeoff : 모델의 복잡성, 예측의 정확성, 그리고 모델을 훈련시키는 데 사용되지 않은 이전에 보이지 않는 데이터에 대해 얼마나 잘 예측할 수 있는지 사이의 관계를 설명한다. 일반적으로, 모델에서 조정 가능한 매개 변수의 수를 늘리면 더 유연해지고 훈련 데이터 세트에 더 잘 맞을 수 있다. (출처: 위키피디아)

7. LLM : LLM 은 Large Language Model 의 약자로, 자연어 처리(NLP)에서 사용되는 인공지능 기술의 한 종류. 이 모델들은 대규모의 텍스트 데이터를 학습하여 언어의 구조와 의미를 이해하고, 그 학습을 바탕으로 텍스트 생성, 번역, 요약, 질문에 대한 답변 등 다양한 언어 관련 작업

8. RAG : RAG(Retrieval-Augmented Generation)는 대규모 언어 모델(LLM)의 한계를 극복하기 위해 제안된 새로운 자연어 처리 기술. LLM 은 방대한 양의 텍스트 데이터를 사전 학습하여 강력한 언어 이해 및 생성 능력을 갖추고 있지만, 학습 데이터에 없는 최신 정보나 특정 도메인 지식은 제공하기 어렵다는 단점을 가지고 있다. RAG 는 이러한 LLM 의 한계를 극복하기 위해 '지식 검색'과 '언어 생성'을 결합한 프레임워크. RAG 의 기본 아이디어는 질문에 답하기 위해 필요한 지식을 외부 데이터베이스에서 검색하여 활용하는 것.

9. 경사하강법 : 경사하강법(Gradient Descent)이란, 단계적으로 오차함수를 조금씩 줄여가며 반복적으로 가중치를 개선해 가며 최적의 가중치를 찾아가는 방법을 말한다.(지식백과 : AI 용어사전)

10. 신경망 : 기계 학습에서의 신경망은 인공 뉴런이나 노드로 구성된 인공 신경망을 의미한다.(지식백과 : AI 용어사전)

11. 활성함수 : 활성화 함수란 신경망의 뉴런에서 출력 신호를 결정하듯이 입력 신호의 가중치 합을 변환하여 출력하는 함수이다.(지식백과 : AI 용어사전)

12. 노드 : 노드(node)란, 신경망을 구성하고 있는 요소로 각각의 노드는 데이터를 포함하고 있으며 다른 노드와도 연결될 수 있다.(지식백과 : AI 용어사전)

13. 역전파 : 역전파(backpropagation)란, 다층 퍼셉트론 신경망 학습에서 주로 사용되는 알고리즘으로 목적 함수(objective function, loss function)를 최소로 하기 위해 수행된다.(지식백과 : AI 용어사전)

14. MLP (다층 퍼셉트론) : 퍼셉트론을 여러 층 쌓은 인공 신경망이다. 입력 층, 은닉 층, 출력 층으로 구성된다. 비선형 활성화 함 수를 사용해 입력 데이터로부터 출력 값을 만들고 복잡한 데이터를 학습한다.

15. Generalization (일반화) : 모델이 학습 데이터 뿐 아니라 일반적인 데이터에서도 성능을 발휘하는 것을 말한다. 일반적으로 train 과 test 에 대해 gap 이 생길 수 있는데, 이러한 gap 을 줄여 학습할 때 사용하지 않은 데이터에서도 좋 은 성능을 발휘하도록 한다.

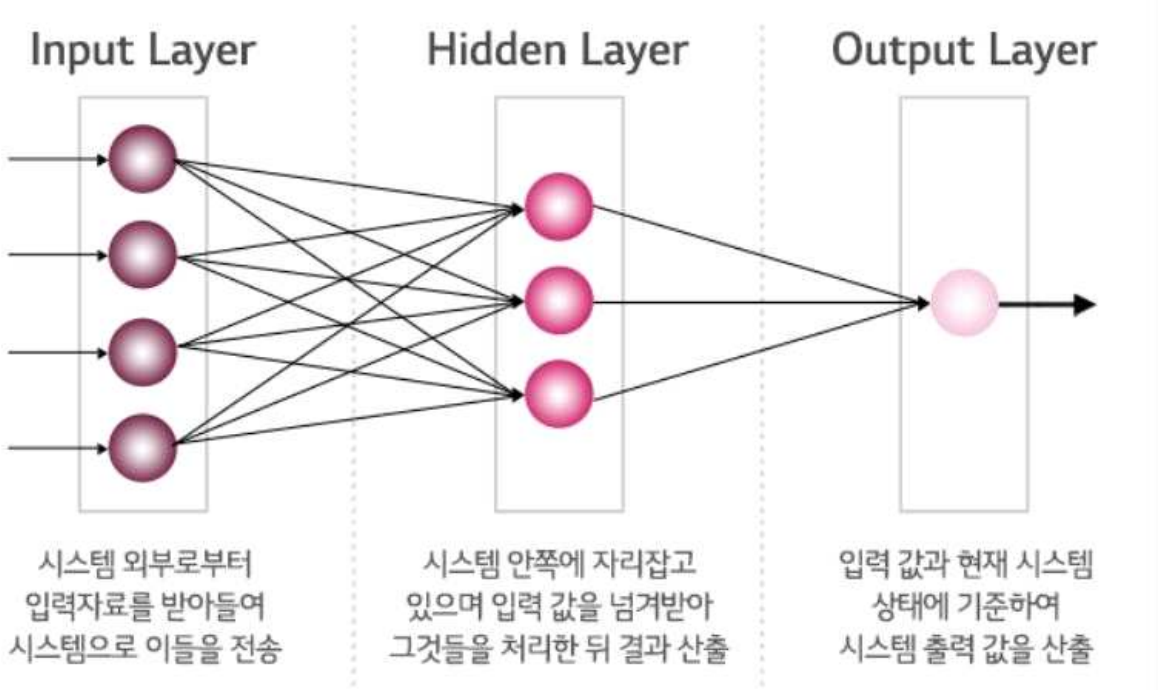
16. Bootstrapping : 주어진 데이터에서 중복을 허용하여 sample 을 만드는 방법이다. 원래의 데이터로부터 랜덤 샘플링을 통해 학습 데이터를 늘릴 수 있다.

17. 편미분 : 다변수 함수의 특정 변수를 제외한 나머지 변수를 상수로 간주하여 미분하는 것을 편미분 이라고 한다. 여러 변수들 중 하나에 대해서 미분하고, 나머지는 상수로 취급한다.

2 <팀 미션 - 문제1>

문제 1. 해당 강의에서는 "시그모이드(sigmoid) 함수나 tanh 함수는 전통적으로 많이 쓰이던 활성함수지만 딥러닝에선 ReLU 함수를 많이 쓰고 있다." 라고 말하고 있다. 활성화 함수의 개념에 대해 다시 한번 찾아보고, 각 함수마다의 특징들을 찾아보고 어느 모델, 기능들에 사용이 되는지, 또 왜 작금의 딥러닝에서는 ReLU 함수를 많이 사용하고 있는지에 대해 토의해보자 . [주제 : 신경망(활성화 함수)]

활성화 함수는 인공 신경망에서 입력값을 출력값으로 변환하는 함수이다. 인공 신경망은 인간의 두뇌를 모방하기 위해 뉴런의 구조를 참고했는데, 다음과 같이 입력값을 제공받는 input layer, 출력값을 내기 위한 연산을 수행하는 hidden layer, 그리고 사용자에게 출력값을 돌려주는 output layer 로 구성되어 있다.



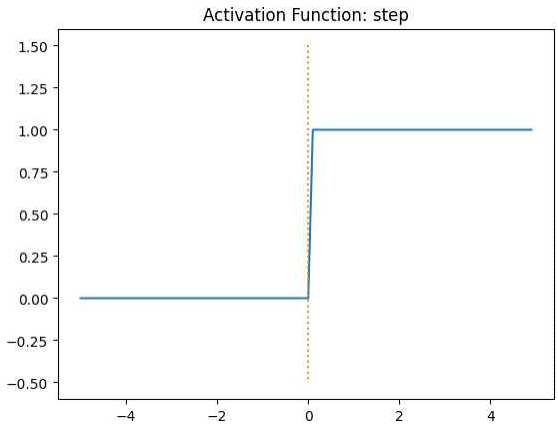
활성화 함수의 종류는 여러 가지가 있다. step function, sigmoid function, hyperbolic Tangent Function, ReLu function, leaky ReLu function 등등이 있는데, 이외에도 다양한 활성화 함수가 있으며 공통점은 비선형 함수라는 데에 있다.

선형 함수여서는 안되는 이유는 간단하다. 여러 개의 선형 함수가 겹쳐진다면 하나의 함수로 표현 가능하기 때문이다. 예를 들어 활성 함수로서 h(x) = 2x 라는 함수를 사용한다고 했을 때, 해당 활성 함수를 3 번 거친다면 h(h(h(x))) = 8x^3 이다. 하지만 이는 결국 h'(x) = 8x^3 으로 한번에 표현 가능하기 때문에 활성함수의 경우에는 비선형 함수만을 사용해야 한다.

Step Function

먼저, step function 은 최초의 인공 신경망에 사용되던 함수이다. 생명체의 뉴런(neuron)은 일정량 이상의 신호가 주어졌을 때에만 다음 뉴런으로 전달하는 특성을 가지고 있다. 즉 입력값이 0 이하라면 0 을, 그 이상이라면 1 을 돌려주는 함수라고 볼 수 있다. 이를 파이썬으로 구현한다면 다음과 같다.

```python
import numpy as np
from matplotlib import pyplot as plt
def step(x):
  return (x>0)*1 + (x<=0)*0


x = np.arange(-5.0, 5.0, 0.1)
y = step(x)

plt.plot(x,y)
plt.plot([0,0], [1.5, -0.5], ':')
plt.title("Activation Function: step")
plt.show()
```



하지만 step function 은 미분 불가능하다는 단점을 가지고 있다. 단적인 예시로는 다음의 그림이 있다. 인간은 다음 중 블루베리가 박힌 머핀과 치와와를 간단하게 구별 가능하지만, 인공지능에게 있어 다음 이미지에서 머핀과 치와와를 구분하는 것은 쉽지 않은 과제로 알려져 있다.

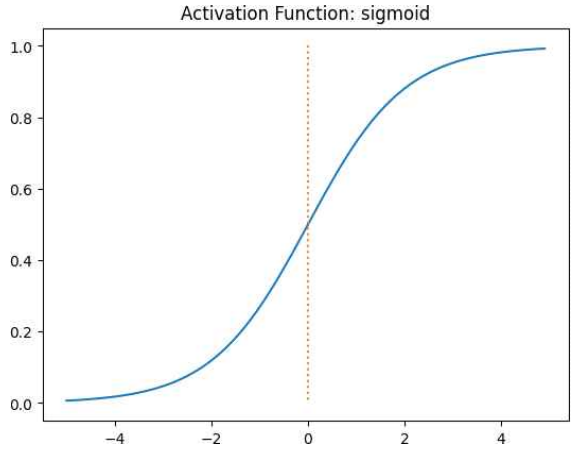
인간의 사고는 step function 과 같이 0 과 1 로만 이루어져 있지 않다. 분명한 기준점을 가지고 '그렇다'와 '아니다'로 답할 수 있는 문제들도 분명히 있지만, 그렇지 못한 문제들이 훨씬 더 많다. 위의 머핀과 치와와를 구분하는 것이 단순한 하나의 기준을 통해 이루어지지 않는 것처럼 말이다. 즉 인간의 연속적인 사고 과정을 따라가기에는 step function 과 같이 불연속적인 함수를 이용하기에는 부적합하다는 것을 알게 되👌고, 이후 활성화 함수로는 연속적인 함수를 사용하게 되👌다.

sigmoid function

무한대의 실수값을 0 과 1 사이의 확률값으로 변환하는 함수이다. 가장 보편적으로 활용되👌던 함수이지만, 입력값의 절대값이 커질수록 기울기가 0 으로 수렴하는 단점이 있다. 이를 기울기 소멸 문제, 혹은 Vanishing Gradient problem 이라고 한다. 활성함수의 기울기 값이 계속해서 곱해지다 보면 가중치에 따른 결과 값의 기울기도 0 에 가까워지므로, 기울기가 너무 작아져 종국에는 가중치가 거의 변하지 않는다는 특징을 가지고 있다. Sigmoid function 을 함수와 파이썬으로 나타낸다면 아래와 같다.

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

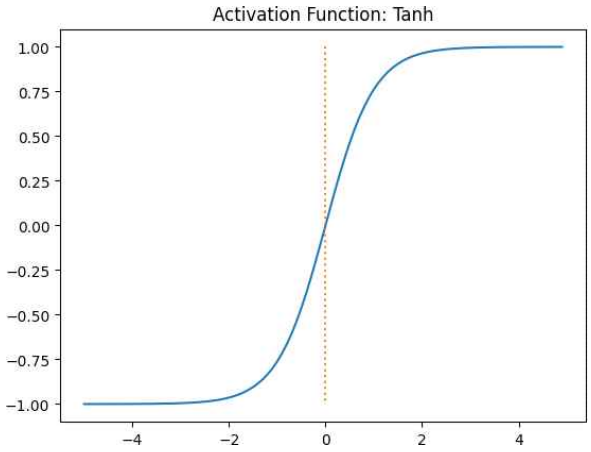
```
def sigmoid(x):
  return 1/(1+np.exp(-x))

x = np.arange(-5.0, 5.0, 0.1)
y = sigmoid(x)

plt.plot(x,y)
plt.plot([0,0], [1.0, 0.0], ':')
plt.title("Activation Function: sigmoid")
plt.show()
```



Tanh function(hyperbolic tangent function)

tanx 함수를 쌍곡선 형태로 구현한 함수이다. 기존의 sigmoid function 의 vanishing gradient 문제를 해결하기 위해 등장하였다. Sigmoid 대비 기울기가 작아지지 않는 구간이 더 넓기 때문에 양수와 음수 모두에서 학습 효율성이 더 높다. 그렇기 때문에 sigmoid function 이 필요할 때 대안적으로 사용되지만, 값이 클 때 여전히 gradient vanishing problem 을 겪는다.

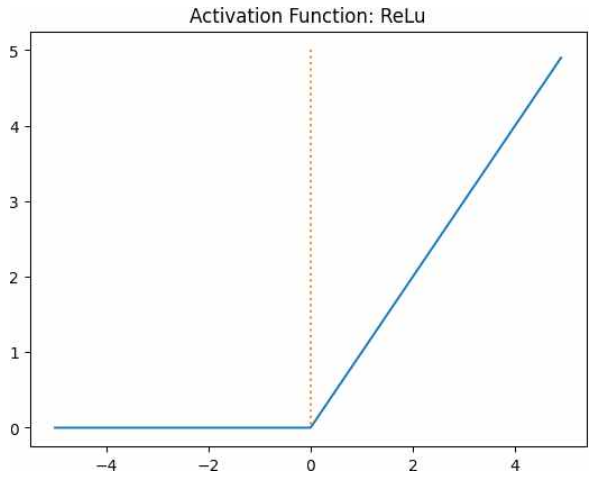
```
x = np.arange(-5.0, 5.0, 0.1)
y = np.tanh(x)

plt.plot(x,y)
plt.plot([0,0], [1.0, -1.0], ':')
plt.title("Activation Function: Tanh")
plt.show()
```



ReLu function

현재 딥러닝 분야에서 가장 기초적인 함수. 입력값이 음수이면 0 을 출력하고 양수값이라면 그대로 그 값을 반환하는 함수이다. sigmoid, tanh 함수의 고질적인 문제였던 Vanishing Gradient 를 해결하기도 했다. 참고로 ReLu 함수는 입력값이 너무 커지게 되면 모델이 입력값에 편향될 수 있으므로 max 값을 6 이하로 제한하기도 한다. 파이썬 코드와 그 구현 결과는 다음과 같다.

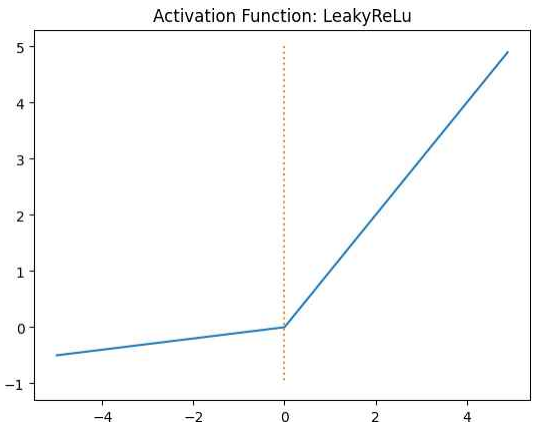
```
def ReLu(x):
  return np.maximum(0,x)

x = np.arange(-5.0, 5.0, 0.1)
y = ReLu(x)

plt.plot(x,y)
plt.plot([0,0], [5.0, 0.0], ':')
plt.title("Activation Function: ReLu")
plt.show()
```



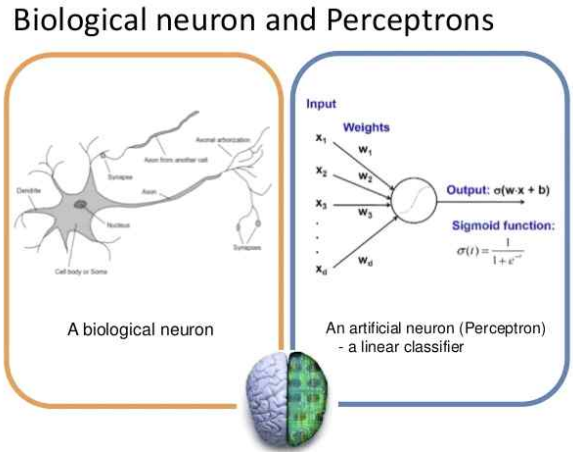
Leaky ReLu funtion

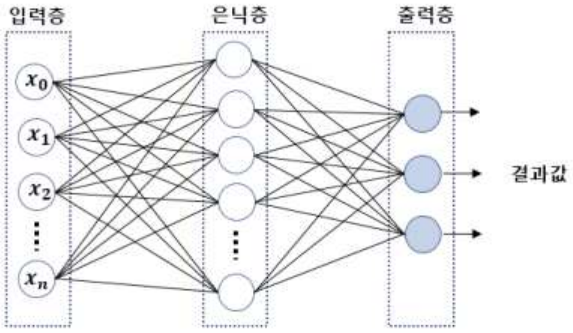
ReLu function 함수의 단점을 보완하기 위해 나온 함수. ReLu 는 음수값이 입력되면 0 을 반환하기 때문에 음수 영역에서는 학습이 어렵다.(Dying ReLu / Dead Neuron) 때문에 음수값에서는 아주 작은 값을 곱해 반환하는 Leaky ReLu function 이 등장했다. 음수값에서도 여전히 그 값을 어느 정도 반환한다는 점을 제외하고는 차이점이 거의 없다.

```
def LeakyReLu(x):
  return np.maximum(0.1*x,x)

x = np.arange(-5.0, 5.0, 0.1)
y = LeakyReLu(x)

plt.plot(x,y)
plt.plot([0,0], [5.0, -1.0], ':')
plt.title("Activation Function: LeakyReLu")
plt.show()
```



퍼셉트론의 기본적인 개념

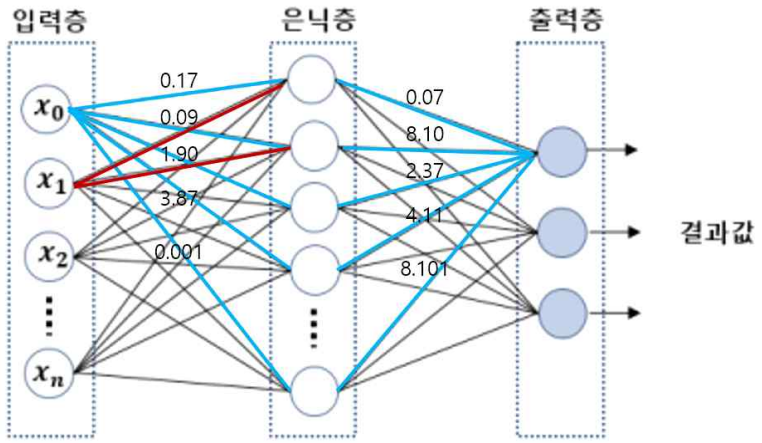




퍼셉트론은 다수의 입력신호(input)를 받아서 하나의 신호(output)를 출력한다 라는 의미를 갖고 있다.

인간의 뉴런의 형태를 본따서 만든 것으로 원하는 결과값 도출을 위해 입력층과 은닉층 사이, 은닉층과 출력층 사이에 딥러닝 학습을 통해 최적의 가중치를 결정한다. 학습 완료 후 최적의 가중치와 은닉층이 선택되는데 한 번 학습한 결과는 다음에도 똑같이 사용할 수 있다.

퍼셉트론 학습 방법 및 절차



처음에는 각각의 입력층에 대한 여러 은닉층과 여러 은닉층과 각각의 출력층에 대한 가중치가 랜덤으로 설정된다. 그리고 결과를 산출한다. 산출한 결과에 대해 오차값을 계산한다. 1000번, 10000번처럼 여러번 학습을 진행하여 점점 오차값을 줄여나감으로서 학습을 진행한다.

XOR을 통한 활성화 함수의 개념 이해

1) 계단함수 (Step Function) : 0또는 1의 값만 출력

2) 시그모이드 함수 (Sigmoid Function) : 0~1사이의 실수를 출력

3) ReLU(Rectified Linear Units) : 0이하면 0을 출력, 0이상이면 그 값을 출력

XOR : False False면 False를 출력, True False / False True / True True면 True를 출력한다. 근데 세상은 이렇게 단순하지 않다.(0과1로만 이루어져있지 않다.) 그래서 0과 1사이의 수를 반환하는 활성화 함수를 사용하여 복잡하고 비선형적인 문제를 해결할 수 있도록 도와준다.

따라서 활성화 함수는 신경망이 단순한 패턴뿐만 아니라, 복잡한 함수를 근사하고, XOR 같은 비선형 문제를 해결할 수 있도록 도와준다. (활성화 함수에 대한 개념은 밑에 설명)

- 활성화 함수 개념 숙지

활성화 함수(activation function) 개념 숙지

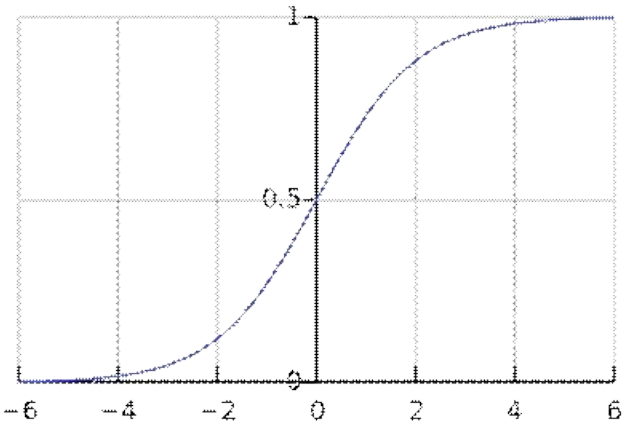
입력 노드에 대해 학습에 활용 가능한 출력 노드를 정의하는 수학적 함수이다.
입력받은 데이터의 활성화 여부를 설정하며, 활성화에 따라 가중치와 최종 결과에 영향을 미친다.
주로 여러 개의 층으로 구성된 다층 구조의 신경망에서 사용된다.

활성화 함수는 입력값에 대해 비선형 변환을 수행하여 모델이 복잡한 패턴을 학습할 수 있게 한다. 즉, 뉴런의 가중 합을 입력으로 받아 이를 출력값으로 변환하는데,
이는 네트워크의 학습과 성능에 큰 영향을 미친다. 활성화 함수가 없으면, 뉴런의 출력은 단순히 입력의 선형 조합에 불과하여 네트워크가 복잡한 패턴을 학습할 수 없게 된다.

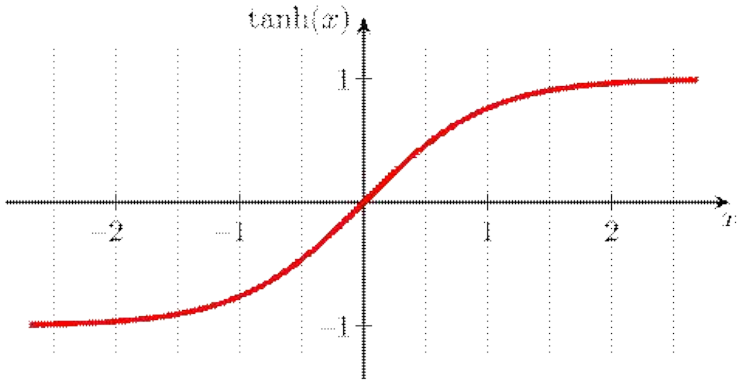
- 시그모이드, tanh, ReLU 개념 숙지

시그모이드 함수(Sigmoid Function)

입력된 데이터를 0과 1 사이의 값으로 출력하는 **비선형**함수로, 미분가능한 **연속성**을 가진 함수이다.
비선형성으로 인해 복잡한 관계를 모델링할 수 있다.



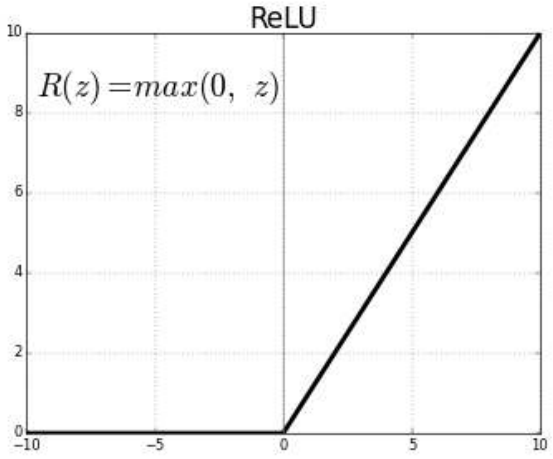
시그모이드 함수는 일반적으로 입력값인 x의 값이 증가할 때 함수값인 y의 값도 증가하는 양상을 보인다.
이에 따라, 시그모이드 함수는 그래프의 모양이 S자 모양의 곡선으로 나타나는 함수이다.
- 장점: 회귀분석에 있어서 선형 함수는 새로운 데이터의 추가가 기존의 분류 모델에 크게 영향을 미치는 반면, 시그모이드 함수에서는 해당 문제가 크지 않았다. 경사하강법을 시행할 때 기울기의 급격한 변화가 나타나지 않는다는 점이 그 이유이다.
- 단점: 정의역의 절댓값이 커질수록 변곡점의 기울기값이 0에 수렴하여 역전파 과정이 진행될수록 아래층에 전달되는 신호가 점차 사라지는 '기울기 소실(Gradient Vanishing)' 문제가 발생한다.



- tanh 함수(Tanh Function)

실수 값을 (-1, 1) 범위로 압축하는 **활성화 함수**이다. 원점을 중심으로 하여 값이 양의 무한대로 갈수록 1에 수렴하고, 값이 음의 무한대로 갈수록 -1에 수렴한다. 이 함수는 그 형태가 S자를 닮았기 때문에 시그모이드 **함수'**의 일종으로 간주되며, 이런 특징 덕분에 비선형성을 모델에 부여할 수 있다.

해당 함수는 “시그모이드” 함수의 일종이지만, 시그모이드 함수가 갖는 단점을 일부 개선한 함수로 볼 수 있다. 시그모이드 함수는 미분이 간단해 모델 제작에 필요한 시간은 줄어드나, 미분 범위가 짧아 정보 손실을 갖는 큰 단점이 있다. 비록 tanh의 경우에도 지속되지만 0를 중심으로 하는 시그모이드보다 최적으로 볼 수 있다.



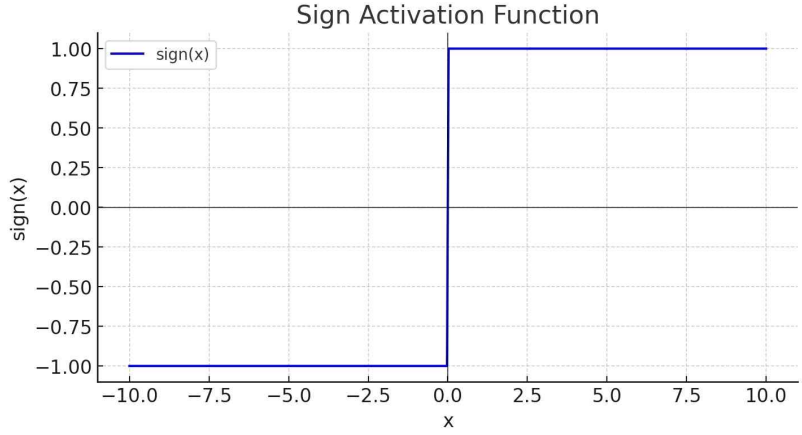
ReLU 함수(Rectified Linear Unit)

시그모이드 함수가 가진 문제였던 처리 속도의 한계와, 기울기 소실에 대한 부분을 개선한 해결한 모델이다. 기울기 소실 문제는 한 층에 사용된 활성화 함수의 출력값을 아래층의 입력값으로 사용하는 다층 구조에서 치명적으로 작용하여 반드시 개선이 필요한 문제였다.

ReLU는 비선형적이며 시그모이드 함수와 달리 역전파 오류가 없다는 장점이 있다. 또한 더 큰 신경망의 경우에도 ReLU를 기반으로 모델을 구축하는 속도는 시그모이드 함수를 사용하는 것에 비해 매우 빠르다.

출력 범위는 [0, ∞)로, 범위가 무한하여 패턴 표현이 대부분의 가중치에 상당한 영향을 미치기 때문에 일반적으로 훈련이 더 효율적이다.그러나 입력값이 0보다 작으면 출력이 0이 되어 뉴런이 죽는 문제(dying ReLU)가 발생할 수 있다.

Sign 활성화 함수



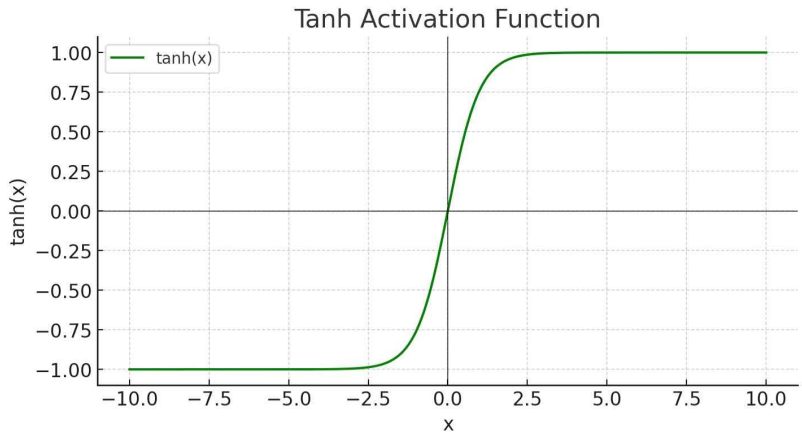
-입력값이 양수면 1을 출력

-입력값이 음수면 -1을 출력

-이진 분류 문제에 사용됨

-다른 활성함수보다 성능이 낮아 많이 사용되지 않음

-Decision Boundary간 거리 정보를 고려하지 않는다는 단점이 있음



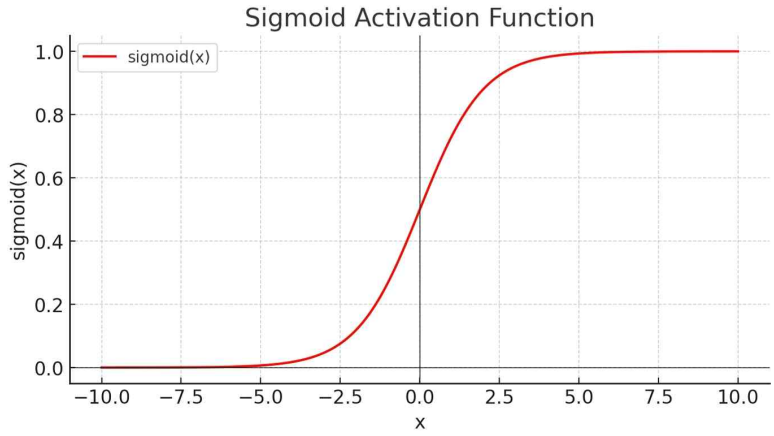
Tanh 활성화 함수

-넘파이에서 제공하는 함수로 별도의 함수 작성이 필요치 않음

-시그모이드 함수와 비슷하지만 출력값이 -1에서 1까지임

-순환 신경망에서 주로사용

Sigmoid 활성화 함수



-1980년대부터 사용되어온 전통적인 활성화 함수

-매끄러운 S자 곡선 형태

-미분 가능

-경사하강 최적화 기법 적용 가능

-0부터 1까지 연속적인 실수 출력을 제공함

Softmax 활성화 함수

$$softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

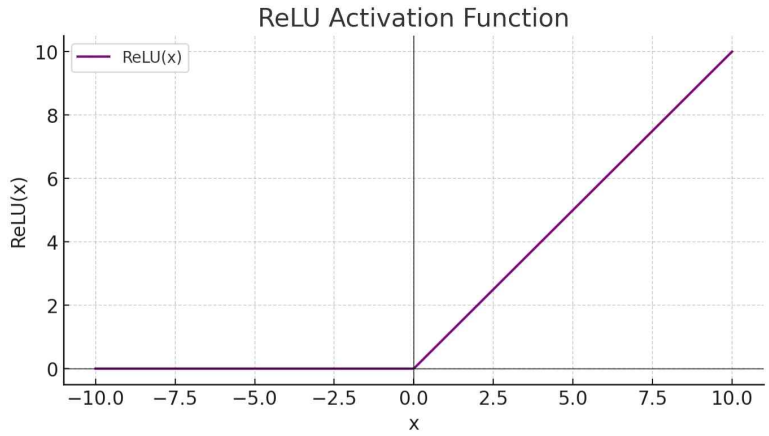
-n가지 출력값을 갖는 함수

-입력값을 N가지 클래스 중 하나로 분류하는 multi-class Classification에 주로 사용됨

-정규화로 인해 출력값이 0~1 사이값으로 나옴

-출력값의 총합은 1

ReLU(Rectified Linear Unit) 활성화 함수



-딥러닝에서 많이 쓰이는 활성화 함수

-입력값이 양수면 output이 input과 동일하게 출력됨

-입력값이 0, 혹은 음수면 출력값은 0으로 고정

-계산 비용이 적고 간단한 미분 계산이 가능

-ReLU 활성함수를 쓰면 심층 신경망에서 나타나는 Gradient Vanishing 현상이 일어나지 않음

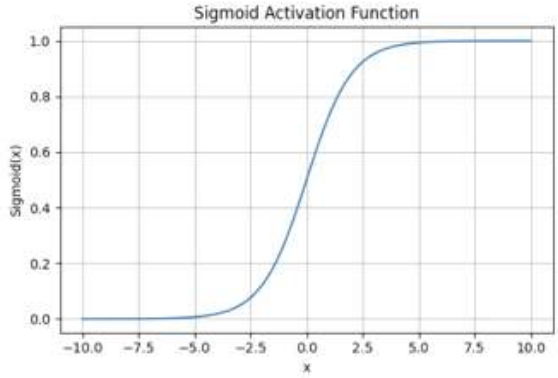
-미분치가 감소하여 역전파에서 일어나는 동작문제를 해결할 수 있음

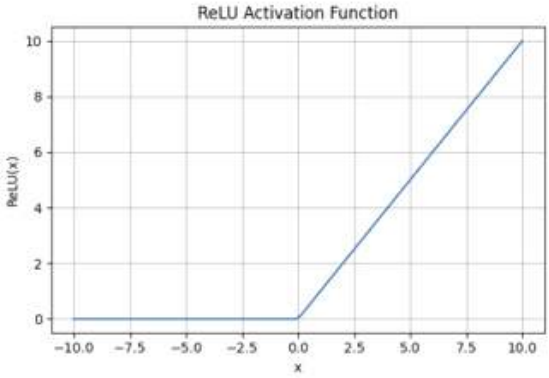
(2)의 활성함수 python으로 구현해보기

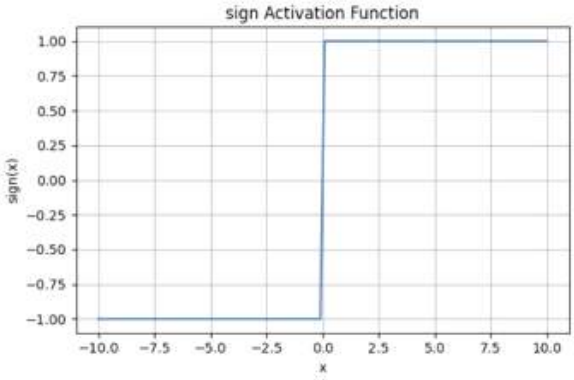
Activation Function

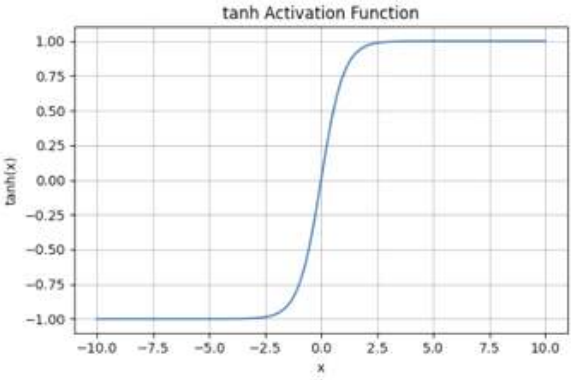
```
import numpy as np
import matplotlib.pyplot as plt

def sigmoid(x):
  return 1.0/(1.0 + np.exp(-x))

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

def sign(x):
  return np.sign(x)

def tanh(x):
  return np.tanh(x)

def softmax(x):
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum(axis=0)

x = np.linspace(-10, 10, 100)
plt.figure(figsize=(12, 12))

# Sigmoid Activation Function
ax_sigmoid = plt.subplot(3, 2, 1)
ax_sigmoid.plot(x, sigmoid(x))
ax_sigmoid.set_xlabel('x')
ax_sigmoid.set_ylabel('Sigmoid(x)')
ax_sigmoid.set_title('Sigmoid Activation Function')
ax_sigmoid.grid(True)

# ReLU Activation Function
ax_ReLU = plt.subplot(3, 2, 2)
ax_ReLU.plot(x, relu(x))
ax_ReLU.set_xlabel('x')
ax_ReLU.set_ylabel('ReLU(x)')
ax_ReLU.set_title('ReLU Activation Function')
ax_ReLU.grid(True)
```

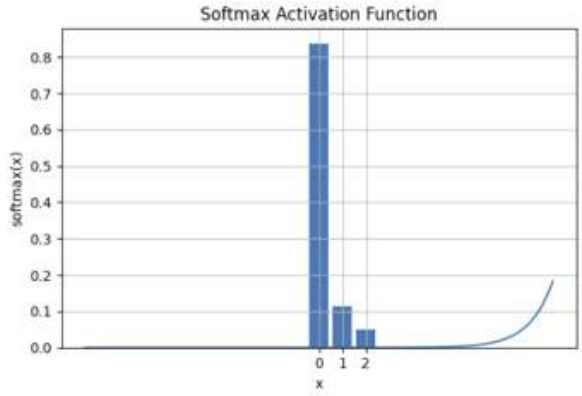
```
# sign Activation Function
ax_sign = plt.subplot(3, 2, 3)
ax_sign.plot(x, sign(x))
ax_sign.set_xlabel('x')
ax_sign.set_ylabel('sign(x)')
ax_sign.set_title('sign Activation Function')
ax_sign.grid(True)

# tanh Activation Function
ax_tanh = plt.subplot(3, 2, 4)
ax_tanh.plot(x, tanh(x))
ax_tanh.set_xlabel('x')
ax_tanh.set_ylabel('tanh(x)')
ax_tanh.set_title('tanh Activation Function')
ax_tanh.grid(True)

# softmax Activation Function
scores = np.array([3.0, 1.0, 0.2])

ax_softmax = plt.subplot(3, 2, 5)
ax_softmax.plot(x, softmax(x))
ax_softmax.set_xlabel('x')
ax_softmax.set_ylabel('softmax(x)')
ax_softmax.set_title('softmax Activation Function')
ax_softmax.grid(True)


plt.bar(range(len(scores)), softmax(scores))
plt.title('Softmax Activation Function')
plt.xticks(range(len(scores)))

plt.tight_layout()
plt.show()
```


Sigmoid Activation Function
Sigmoid(x)
x


ReLU Activation Function
ReLU(x)
x


sign Activation Function
sign(x)
x


tanh Activation Function
tanh(x)
x


Softmax Activation Function
softmax(x)
x

문제 2. 인공지능의 발전함에 있어서 생길 수 있는 문제점들에 대해 생각해보고 토론해보도록 하자. [주제 : AI 윤리 (권한과 책임) - 토의 유도]

주제1. 의료와 인공지능

환자의치료방법을의료AI시스템이결정하는경우,이에대한권한과책임은의료전문가와인공지능사이에어떻게분배되어야할까?법원에서어떠한사건에대한처분을AI가결정할때,이에대한권한과책임은판사와인공지능사이에어떻게분배되어야할까?

환자의 60% 이상이 의료분야 AI 신뢰 부족하다고 답했다. 데이터 프라이버시, 편견, AI 프로세스 투명성 부족에 기인하는 것으로 파악된다. 의료 분야에서의 AI는 기술적인 부분만이 아니라 도덕적, 사회적 의무를 기반으로 배포되어야 한다. 의료 행위의 본질을 주도하는 것이 인간이라는 전제를 기반으로 보조 도구로서의 기능을 AI가 수행하고 이에 따라서, 책임 소재도 인간 중심으로 가져가는 이 타당하다고 본다.
(참고 기사: Ethical AI in Healthcare: A Focus on Responsibility, Trust, and Safety)

주제2. 예술과 인공지능

미국 저작권청(USCO)은 인공지능(AI) 생성작품도 저작권을 인정할 것이라고 밝혔다. 다만 최종 완성품에 사람의 창의적 노력이 포함됐다는 것을 증명할 수 있어야 한다는 기준이 있다고 한다. 더레지스터 등 외신에 따르면 미 저작권청은 AI에 의해 생성된 자료를 포함하는 저작물에 대한 가이드라인을 최근 발표했다고 하는데, 미국과 달리 국내에 챗GPT, 달리(DALL-E), 스테이블 디퓨전 같은 AI도구를 이용한 창작물에 대한 저작권은 어떻게 인정되고 있는가?

우리나라 정부는 작년 말 저작권 등록 관련 규정을 개정하면서 인공지능(AI)이 만든 그림, 시, 소설 등 창작물은 저작권을 등록할 수 없다는 내용을 명시했다. 인간과 AI가 함께 작업한 창작물도 인간 행위에 의한 결과임이 명백한 부분에 대해서만 제한적으로 저작권을 인정하겠다는 입장이다.
문화체육관광부와 한국저작권위원회는 작년 12월 생성형 AI 저작권 가이드라인을 발행했다. 해당 안내서에는 AI 사업자 및 이용자, 저작권자에 대한 안내사항과 AI 산출물 및 저작권 등록에 대한 내용이 담겨 있다.
주요 내용을 살펴보면, AI 산출물은 저작권법으로 보호될 수 없다고 한다. 저작권법에서는 인간이 만들어 창작성을 인정받을 수 있는 저작물을 보호대상으로 하고 있기 때문에, 법에 의거하여 AI 산출물은 저작권을 가질 수 없다.

또한 AI 산출물 제작을 위해 입력하는 개별적인 프롬프트에 대한 저작권은 창작성 유무에 따라 달라질 수 있으며, 프롬프트 입력 행위만으로는 그 표현에 있어서의 기여도를 인정하기 어려울 것이라고 한다.

AI 산출물은 직접 생성했다고 해도 저작권 침해 및 약관 위반 등의 문제가 발생할 수 있으므로, 관련 내용에 유의하여 이용해야 한다고 한다. AI 산출물이라고 하더라도 기존의 저작물과 동일하거나 유사하다고 판단될 경우에는 저작권 침해의 문제가 발생할 수 있다고 한다.

[참고자료] [단독] AI가 만든 그림 · 소설 저작권 불인정... 저작권 등록규정에 명시 – 매일경제