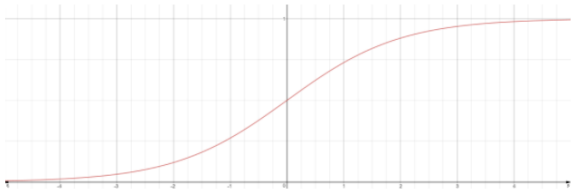
Activation function(비선형 활성화 함수)

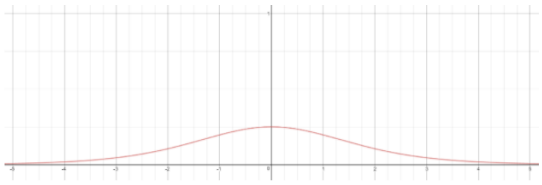
Sigmoid (logistic function)

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

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

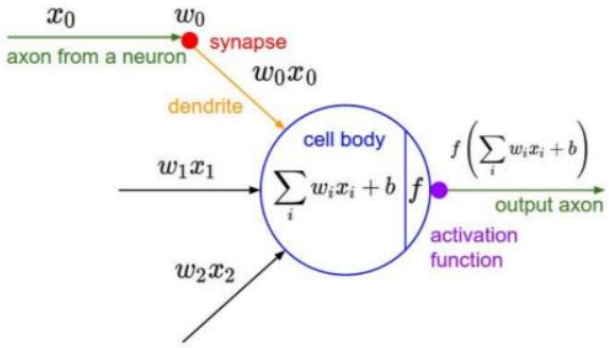


[Sigmoid]



[Sigmoid 1차 미분]

- S자 커브 형태로 자연스럽게 활성화 가능함.
- 입력 값에 따라 gradient 값(sigmoid 1차 미분값)이 지나치게 작아질 수 있다(학습이 잘 안됨)
- Exponential 연산 자체가 다소 무겁다. (학습이 느려짐)



$x0, x1, x2$는 모두 0 이상의 값을 갖는데, 직전 층에서 sigmoid로 인해서 양수([0,1])로 활성화된 출력이 입력으로 주어지기 때문이다.

$$\frac{\partial L}{\partial w_i} = x_i \times \delta$$

: Loss(L)를 weight(w)에 따라 미분한 것.

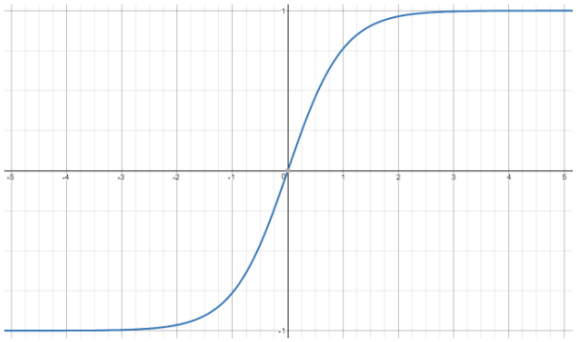
δ가 양수라면 Loss에 대한 $w0, w1, w2$ 각각의 gradient가 모두 양수 혹은 모두 음수가 된다. 따라서 데이터 x와 파라미터 w를 2차원 벡터로 가정한다면 1사분면과 3사분면에만 속하고, 2사분면과 4사분면에는 속할 수 없다. 결과적으로 허용되는 방향에 제약이 가해져(1사분면과 3사분면에만 가능) 학습 속도가 늦거나 수렴이 어렵게 된다. 이 문제는 하이퍼볼릭탄젠트(tanh)를 이용하여 해결할 수 있다.

tanh (하이퍼볼릭탄젠트)

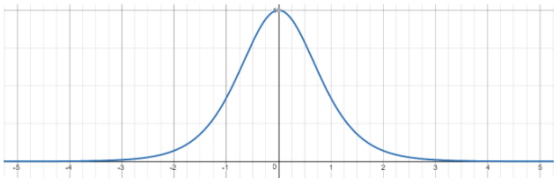
하이퍼볼릭탄젠트는 **sigmoid 함수의 크기와 위치를 조절(rescale and shift)한 함수이다.**

sigmoid 함수와의 관계식과 미분식이다.

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



[tanh]

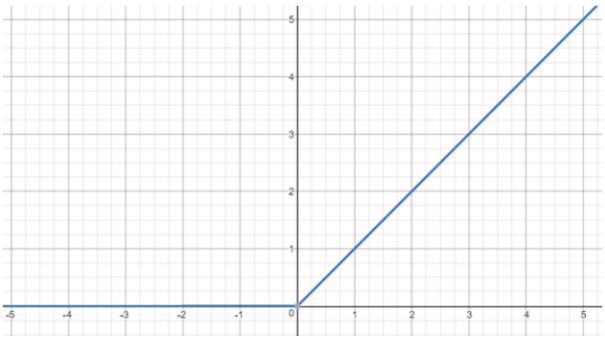


[tanh 1차 미분]

- sigmoid 함수[0, 1]와 달리 tanh는 -1과 1사이의 값을 가진다([-1, 1]
- 그래프의 모양이 sigmoid 함수와 달리 0을 기준으로 대칭인 점을 확인할 수 있다. -> sigmoid를 활성화 함수로 사용할 때보다 학습 수렴 속도가 빠르다.
- 입력 값에 따라 gradient 값(tanh1차 미분값)이 지나치게 작아질 수 있다(학습이 잘 안됨)

ReLU(Rectified Linear Unit)

$$
f(x) = max(0, x)
$$

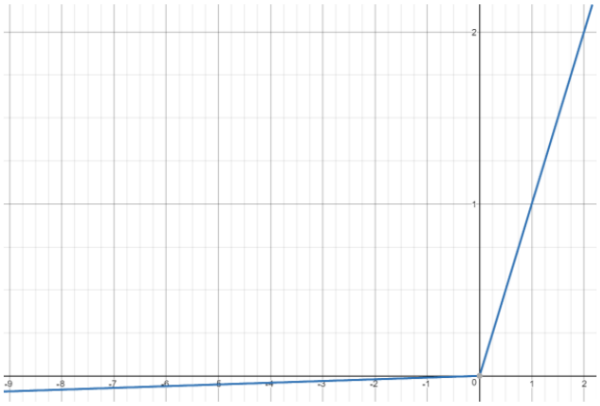


- 음수일 때 무조건 gradient 가 0이 되는 단점이 있다. -> 이를 해결하기 위해서 Leaky ReLu를 사용한다.

- x = 0에서 미분이 불가능하다. 이는 보통 x = 0에서 미분 값을 0으로 만들어주는 방식을 통해 따로 처리한다. -> x <= 0에서 미분 값이 0이 된다.
- 앞서 소개된 활성화 함수에 비해 학습 속도가 훨씬 빠르다

Leaky ReLU

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



- **x 가 음수일 때 gradient가 0.01**이라는 점을 제외하고는 ReLU와 같다.

PReLU

$$f(x) = max(\alpha x, x)$$

- Leaky ReLU와 거의 유사하지만 **새로운 파라미터 α를 추가하여 x < 0에서 기울기를 학습할 수 있게 하였다.**

ELU(Exponential Linear Unit)

$$f(x) = x \quad if \quad x > 0$$
$$f(x) = \alpha(e^x - 1) \quad if \quad x \leq 0$$

- ReLU의 특성을 공유하고, ***Dying ReLU 문제가 발생하지 않는다.**
- 출력값이 거의 zero-centered 에 가깝다.
- 일반적인 ReLU와 달리 exp 함수를 계산하는 비용이 발생한다.

Maxout 함수

$$f(x) = max(w_1^T x + b_1, w_2^T x + b_2)$$

- ReLU의 특성을 공유하고, **Dying ReLU 문제가 발생하지 않는다.**

- 계산량이 복잡하다

---

*Dying ReLu : Forward propagation 과정에서 어떤 뉴런에서든 음수 값이 들어오게 되면 BackPropagation 시에 0이라는 real value 가 가중치에 곱해지면서 해당 노드가 통째로 죽어버리는 현상

무조건 좋지 않은 현상은 아닌 것으로 보인다. 오히려 이 점이 ReLU가 가지는 큰 강점으로 작용한다는 것.(Spacrse Repersentation을 부여함으로써) 거시적 관점에서 일종의 일반화 방식으로 dropout과 유사하게 overfitting 을 줄이는 역할을 한다. **단, 신경망이 깊고, 넓으며, 충분한 Epoch가 주어진 상황에서만 일반화에 도움이 된다.**

Dying ReLu 참고 :

1) https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks
2) https://brunch.co.kr/@kdh7575070/27

전체 내용 참고 :

- https://ratsgo.github.io/deep%20learning/2017/04/22/NNtricks/