**Activation Function**

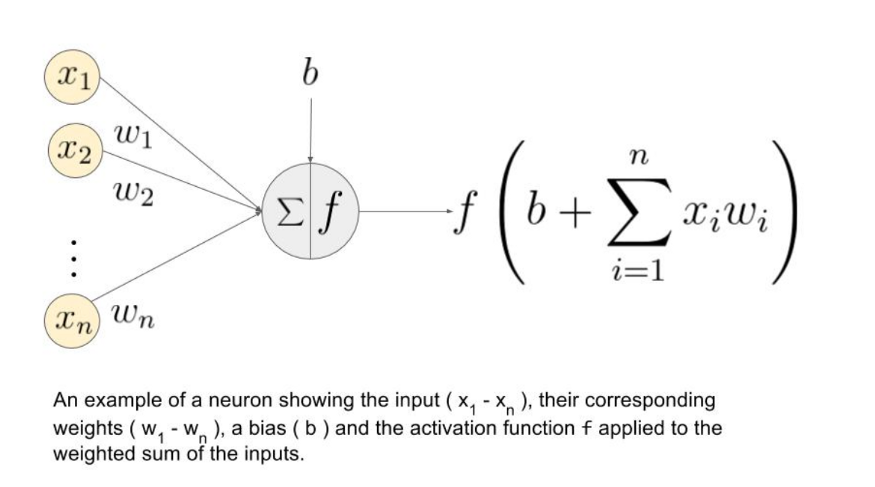
1. **Neural Network**

Neural network, 즉 인공 신경망이란 생물학의 신경망에서 영감을 얻은 학습 알고리즘이다. 이는 시냅스의 결합으로 네트워크를 형성한 인공 뉴런이 학습을 통해 시냅스의 결합 세기를 변화시켜 문제해결능력을 가지는 비선형 모델이다. 인공 신경망은 하나의 입력 계층, 하나 이상의 은닉 계층 및 하나의 출력 계층을 포함하는 노드 계층으로 구성되어 있다. 각 노드 또는 인공 뉴런은 다른 노드에 연결이 되며 연관된 가중치와 임계값을 갖는다. 개별 노드의 출력이 지정된 임계값을 초과하면 해당 노드가 활성화되어 네트워크의 다음 계층으로 데이터를 전송한다.

뉴런은 상호 연결되어 있는데, 이들은 전기 입력을 받아 또 다른 전기 신호를 발생시킨다. 그러나 입력을 받았을 때 즉시 반응하는 것이 아닌 입력 값이 어느 threshold에 도달하여야 출력을 발생시킨다. 이는 아래와 같은 수식으로 나타낼 수 있는데, 수식에서 sign과 같은 함수를 Activation function이라 한다.

1. **Activation Function**

Activation function이란 입력 신호의 총합을 출력 신호로 변환하는 함수이다. ‘activate’라는 이름에서 알 수 있듯이 입력 신호의 총합이 활성화를 일으키는 지 정하는 역할을 한다. <그림 1>은 Activation function의 Computational Graph이다. 이를 통해 input과 weight의 내적 값에 bias를 더한 값을 얼마나 출력시킬 지 정하기 위해 Activation function이 작동함을 알 수 있다.



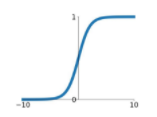
<그림 1> Activation function의 Computational Graph

신경망에서는 뉴런에 연산 값을 계속 전달하는 방식으로 가중치를 훈련하고 예측을 진행한다. 각 함수는 네트워크의 각 뉴런에 연결되어 있다. 각 뉴런의 입력이 모델의 예측과 관련 있는 지 여부에 따라 각 함수는 활성화된다. 이러한 활성화를 통해 신경망은 입력 값에서 필요한 정보를 학습한다.

Activation function은 훈련 과정에서 계산량이 많고 연적파를 사용해야 하므로 연산의 효율성이 중요하다. 따라서 연산의 효율성을 높이기 위한 Activation function이 많이 연구되고 있으며, 그 종류가 다양하다. 본 문서에서 설명할 Sigmoid와 Tanh, ReLU 이외에도 Leaky ReLU, Maxout, ELU 등의 Activation function이 존재한다.

1. **Sigmoid**
   1. **Sigmoid**

Sigmoid 함수는 대표적인 Logistic 함수이다. 해당 함수는 모든 실수 입력 값을 0보다 크고 1보다 작은 미분 가능한 수로 변환하는 특징을 가진다. 함수 식과 미분 값은 다음과 같고, <그림 2>는 함수의 그래프이다.



<그림 2> Sigmoid 함수

* 1. **Sigmoid의 장점**

① 미분식이 단순하고, 유연한 미분 값을 가진다.

② 출력 값의 범위가 (0,1)로 제한되어, 정규화 관점에서 exploding gradient 문제를 방지한다.

* 1. **Sigmoid의 단점**

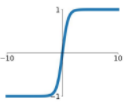
① Vanishing Gradient 문제가 발생한다. 입력 값이 아무리 커도 미분 값이 범위가 제한적이기 때문에 층이 쌓일수록 gradient 값이 0에 수렴한다. 따라서 학습 효율성이 매우 떨어지게 되고, 극한 값으로 갈수록 값이 포화된다.

② 출력 중심이 0이 아니다. X가 모두 양수로 들어올 경우 gradient 값이 모두 양수 또는 음수의 형태를 지녀 지그재그 꼴로 학습하게 된다. 이는 비용이나 효율 측면에서 좋지 않은 결과이다.

③ exp 연산은 비용이 크다.

1. **Tanh**
   1. **Tanh**

Hyperbolic Tangent(Tanh) 함수는 Sigmoid의 대체제로 사용될 수 있는 Activation function이다. Tanh는 Sigmoid가 확장된 함수이다. 둘의 차이점은 Sigmoid의 출력 범위가 0에서 1 사이인 반면 tanh와 출력 범위는 -1에서 1사이라는 점이다. Sigmoid와 비교하여 tanh와는 출력 범위가 더 넓고 경사면이 큰 범위가 더 크기 때문에 더 빠르게 수렴하여 학습하는 특성이 있다. 다음 식은 Tanh 함수의 식과 미분 값이며, <그림 3>은 한수의 그래프이다.



<그림 3> Tanh 함수

* 1. **Tanh의 장점**

① 중앙값이 0이기 때문에 Gradient descent 사용 시 Sigmoid 함수에서 발생하는 편향 이동이 발생하지 않는다.

② 기울기가 양수와 음수 모두 나올 수 있기 때문에 Sigmoid 함수보다 학습 효율성이 뛰어나다.

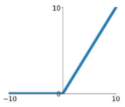
③ Sigmoid 함수보다 범위가 넓기에 출력 값의 변화폭이 더 커서 Gradient Vanishing이 완화된다.

* 1. **Tanh의 단점**

① Sigmoid 함수보다 범위는 넓지만 그 구간이 크지 않은 편이므로 여전히 Gradient Vanishing 문제가 존재한다.

1. **ReLU**
   1. **ReLU**

Rectified Linear Unit(ReLU) 함수는 개선 선형 함수로, 실제 뇌와 같이 모든 정보에 반응하는 것이 아닌 일부 정보에 대해 무시와 수용을 통해 보다 효율적인 결과를 낸다. ReLU 함수의 식과 미분 값은 다음과 같고, <그림 4>는 ReLU 함수의 그래프이다.



<그림 4> ReLU 함수

* 1. **ReLU의 장점**

① 연산이 매우 빠르다. 함수의 원형을 통해서 알 수 있듯이 비교 연산 한 번을 통해 함수 값을 구할 수 있다. 수렴 속도 자체는 Sigmoid 함수나 Tanh 함수보다 6배가량 빠르다. 특히 ReLU는 편미분 시 1로 일정하므로 가중치 업데이트 속도가 매우 빠른 것으로 알려져 있다.

② 비선형 함수이다. 모양은 선형 같지만, 도함수를 가지며 backpropagtion을 허용한다.

③ Gradient Vanishing 문제가 발생하지 않는다. ReLU 함수는 양수는 그대로 반환하며 음수는 0으로 반환한다. 따라서 특정 양수 값에 수렴하지 않는다. 즉, 출력 값의 범위가 넓고 양수 경우 자신을 그대로 반환하기 때문에 Gradient Vanishing 문제가 없다.

* 1. **ReLU의 단점**

① 입력 값이 0이나 음수인 경우 gradient 값은 모두 0이 된다. 따라서 기울기가 0으로 나와 학습을 할 수 없게 된다. 이를 Dying ReLU 현상이라고 한다.

② Gradient Vanishing 문제 방지를 위해 사용하는 함수이므로 은닉층에서만 사용할 것을 추천한다. ReLU의 출력 값과 기울기는 모두 양수이다. 이로 인해 Sigmoid 함수처럼 가중치를 업데이트할 시 지그재그 현상이 발생한다.

③ 0에서 미분이 불가능하다.