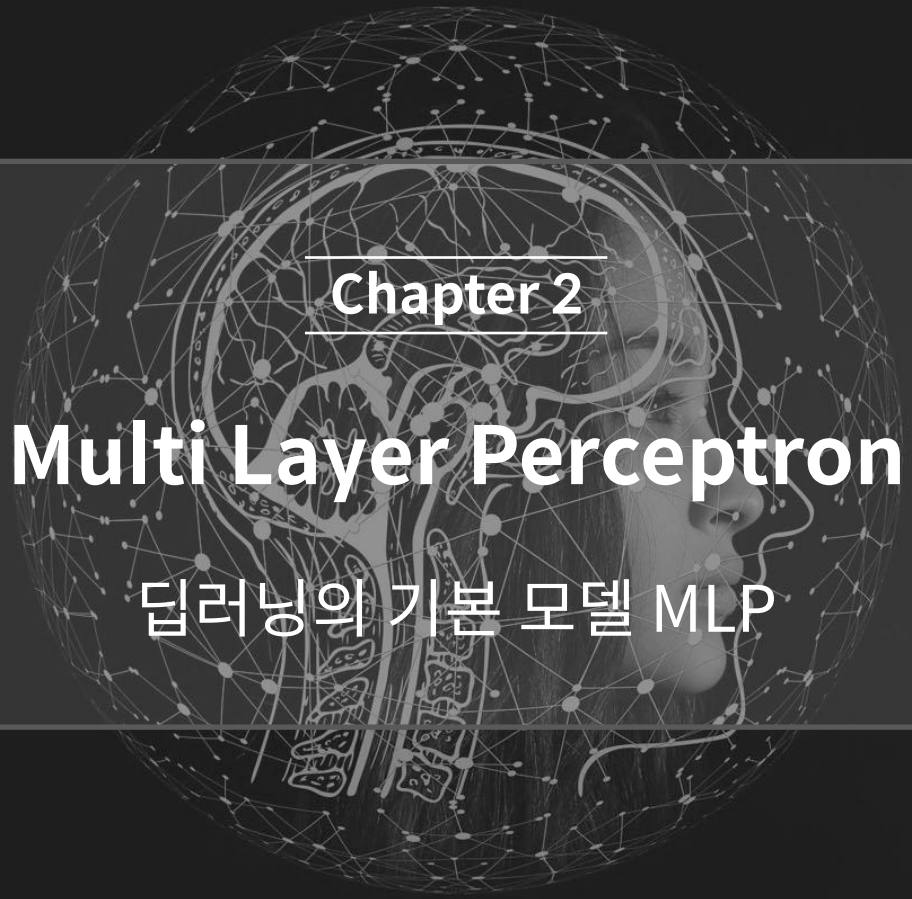


Chapter 2

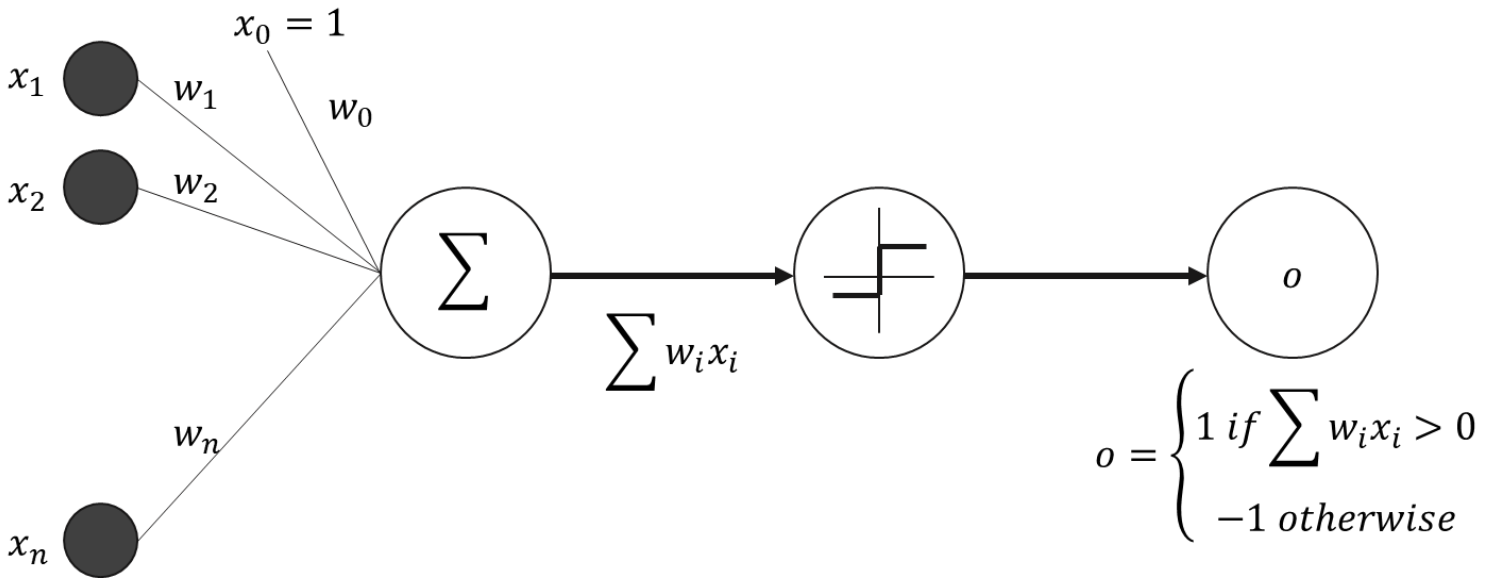
Multi Layer Perceptron

딥러닝의 기본 모델 MLP

최초의 인공지능 Perceptron

Perceptron

- Rosenblatt에 의해 1957년 개발된 인공 신경망 모형의 일종으로, feed-forward network 모형의 가장 간단한 형태인 선형 분류(linear classifier) 모형
- 아래 그림과 같이, 실수 형의 입력 벡터를 받아서 이들의 선형 조합 $\sum_{i=0}^{n} w_i x_i$를 계산하고 계산된 값의 크기가 기준(threshold)보다 크면 1을, 작으면 -1을 결과로 내보내는 함수 $o(x_1, \ldots, x_n)$ 로 구성됨



최초의 인공지능 Perceptron

Perceptron 학습 규칙

- 퍼셉트론은 처음에 Weight를 랜덤하게 설정하고 모델의 오류를 점차 개선해가며 Weight를 개선 해 나감

- 아래와 같은 학습규칙에 의해 Weight가 Update 됨. 모든 데이터를 올바르게 분류 할 때까지 이 과정을 거침

$$w_i \leftarrow w_i + \Delta w_i$$

$Where$

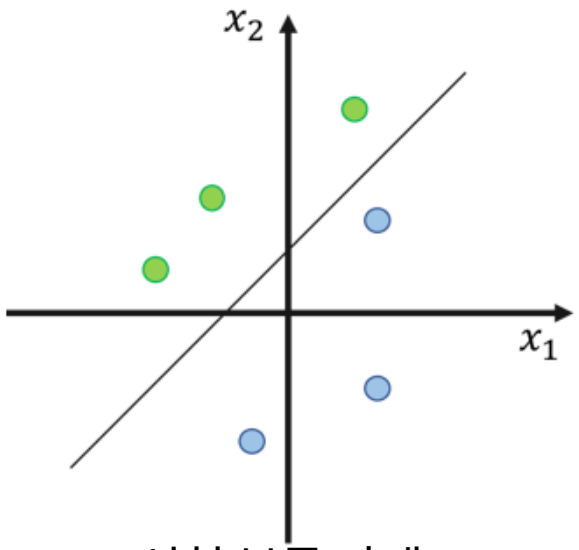
- $\Delta w_i = \eta(t - o)x_i$
  - t = 실제 값
  - o = 예측 값

최초의 인공지능 Perceptron

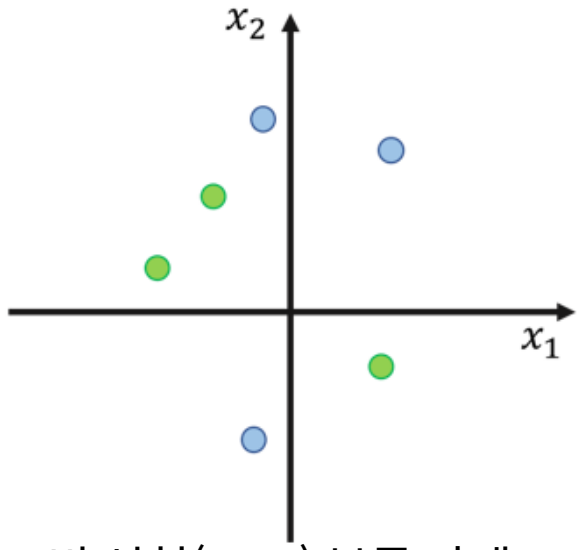
Perceptron의 한계

- 퍼셉트론은 선형 분류 모형의 형태를 가지고 있음. 그렇기 때문에 선형 문제밖에 풀지 못함.

- 비선형 분류 문제는 풀지 못한다라는 단점이 있음. 일정 오류를 허용하고 선을 그을 수 밖에 없음



선형 분류 사례



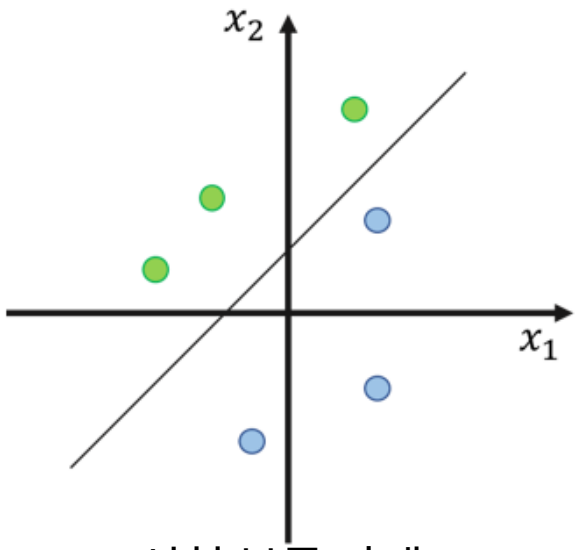
비 선형(XOR) 분류 사례

Multi Layer Perceptron

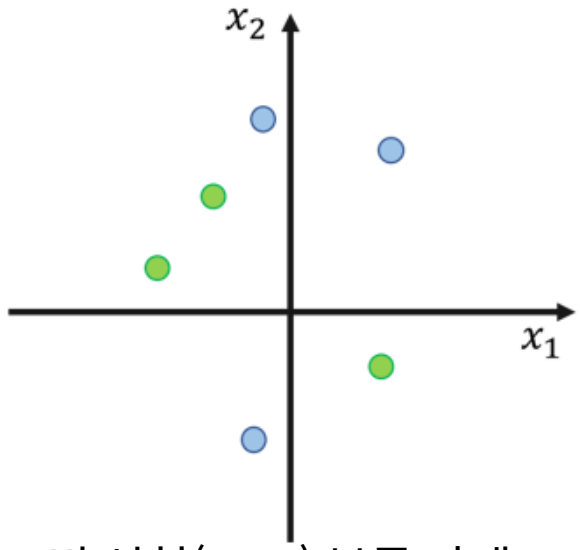
Perceptron의 한계

- 퍼셉트론은 선형 분류 모형의 형태를 가지고 있음. 그렇기 때문에 선형 문제밖에 풀지 못함.

- 비선형 분류 문제는 풀지 못한다라는 단점이 있음. 일정 오류를 허용하고 선을 그을 수 밖에 없음



선형 분류 사례

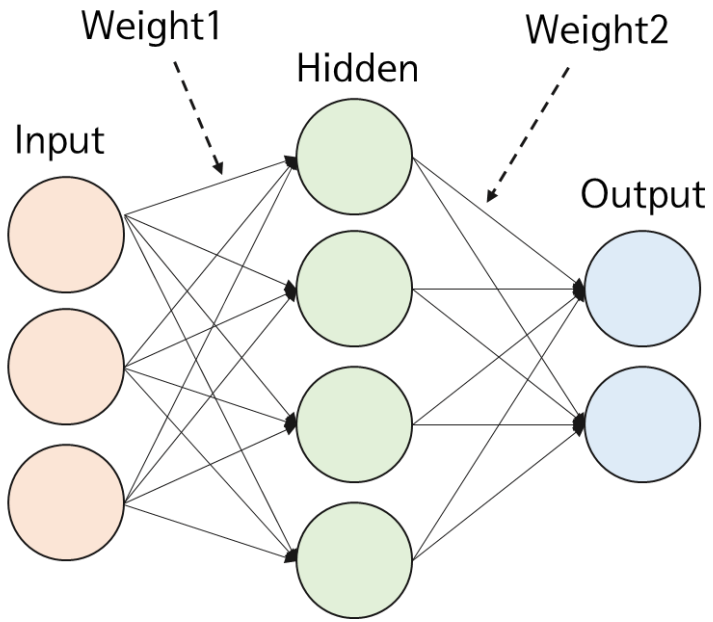


비 선형(XOR) 분류 사례

Multi Layer Perceptron

Multi Layer Perceptron

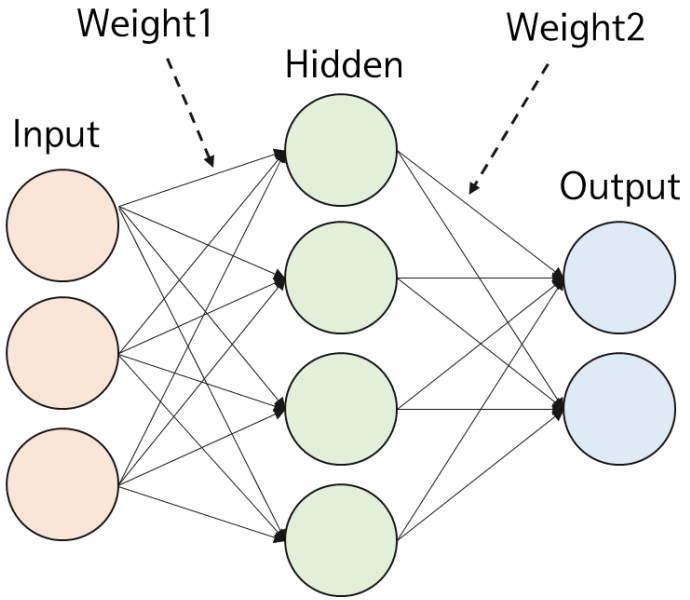
- 퍼셉트론이 가지는 한계점을 극복하기 위해 여러 Layer를 쌓아 올린 MLP가 등장하게 되었음.
- 여러 개의 퍼셉트론의 조합과 그것들의 재조합으로 복잡한 비선형적인 모형을 만들어내는 것.
- 딥러닝의 기본 구조가 되는 신경망이라고 하면 기본적으로 MLP를 의미. MLP의 Hidden Layer를 쌓으면 layer가 깊어지기 (Deep) 때문에 딥러닝의 기본적인 모델이 됨.



Multi Layer Perceptron

Multi Layer Perceptron

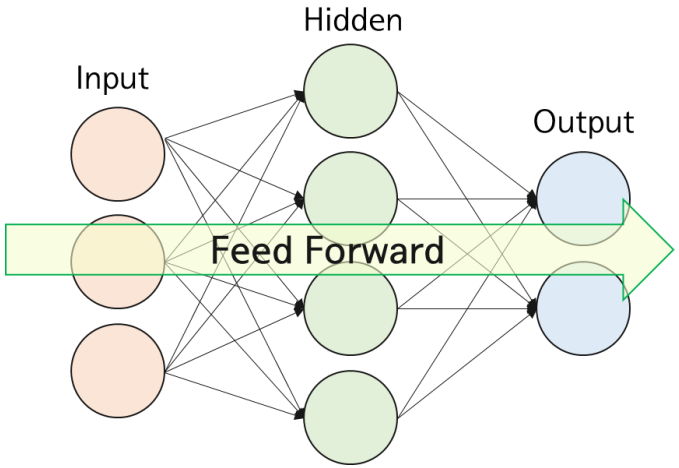
- 아래 그림은 Input하나 Hidden 하나 Output하나 총 세개의 Layer 로 연결 되어 있는 MLP이며, 각 원하나는 노드 (Node)라고 부름.
- Input Node의 수는 Input Data의 변수의 수가 되며 Hidden Layer 와 Hidden Node의 수는 사용자가 지정 해 주어야 할 Hyperparameter임.
- Output Layer는 최종적으로 모델의 결과값을 의미하기 때문에 Output Node의 수는 풀고자 하는 문제에 따라 달라짐. 회귀 분석을 하고자 하는 경우에는 Output Node의 수는 1이 되고, 0부터 9까지의 숫자 분류를 하고자 하는 경우에는 Output Node의 수는 10이 됨.

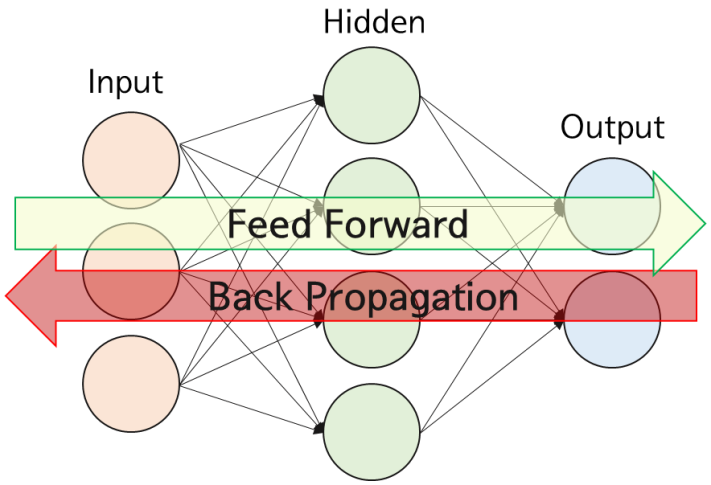


Multi Layer Perceptron

Multi Layer Perceptron 학습

- Feed Forward : 신경망의 Input에서 Weight와 Hidden을 거쳐 Output을 내보내는 과정
- Back Propagation : Feed Forward를 통해서 Input에서 Output까지 계산하며 Weight를 Update하는 과정





Feed Forward

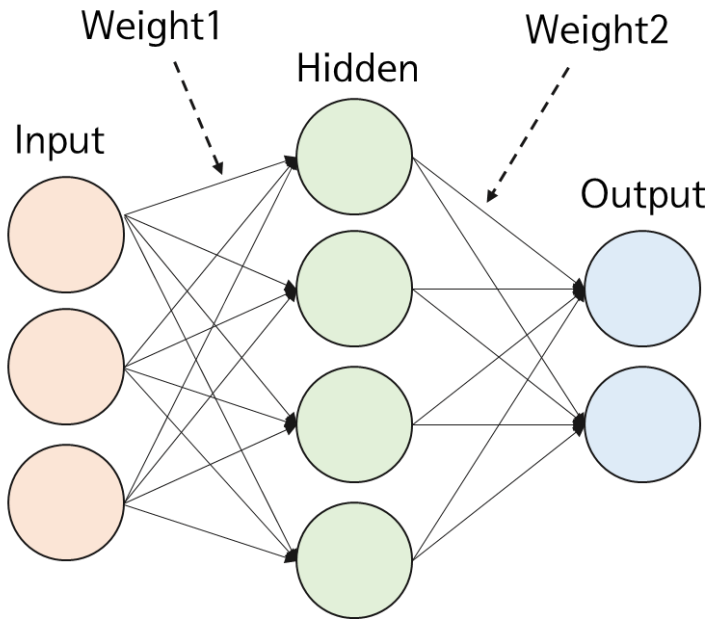
Input -> Hidden -> Output -> Error -> Hidden

Back Propagation

MLP Feed Forward

Multi Layer Perceptron

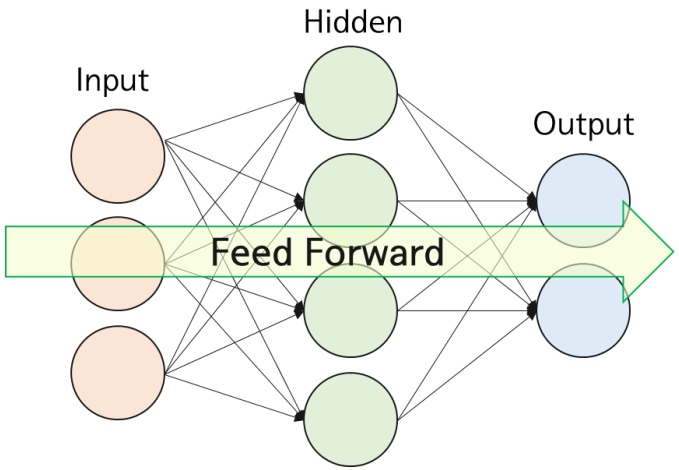
- 퍼셉트론이 가지는 한계점을 극복하기 위해 여러 Layer를 쌓아 올린 MLP가 등장하게 되었음.
- 여러 개의 퍼셉트론의 조합과 그것들의 재조합으로 복잡한 비선형적인 모형을 만들어내는 것.
- 딥러닝의 기본 구조가 되는 신경망이라고 하면 기본적으로 MLP를 의미. MLP의 Hidden Layer를 쌓으면 layer가 깊어지기 (Deep) 때문에 딥러닝의 기본적인 모델이 됨.

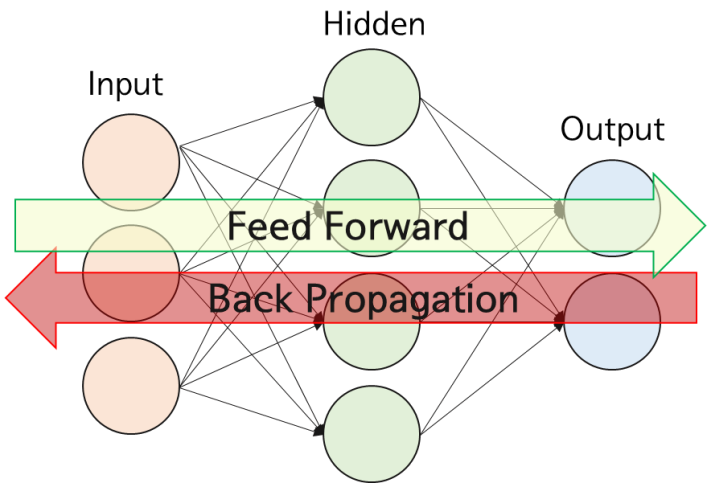


MLP Feed Forward

Multi Layer Perceptron 학습

- Feed Forward : 신경망의 Input에서 Weight와 Hidden을 거쳐 Output을 내보내는 과정
- Back Propagation : Feed Forward를 통해서 Input에서 Output까지 계산하며 Weight를 Update하는 과정





Feed Forward

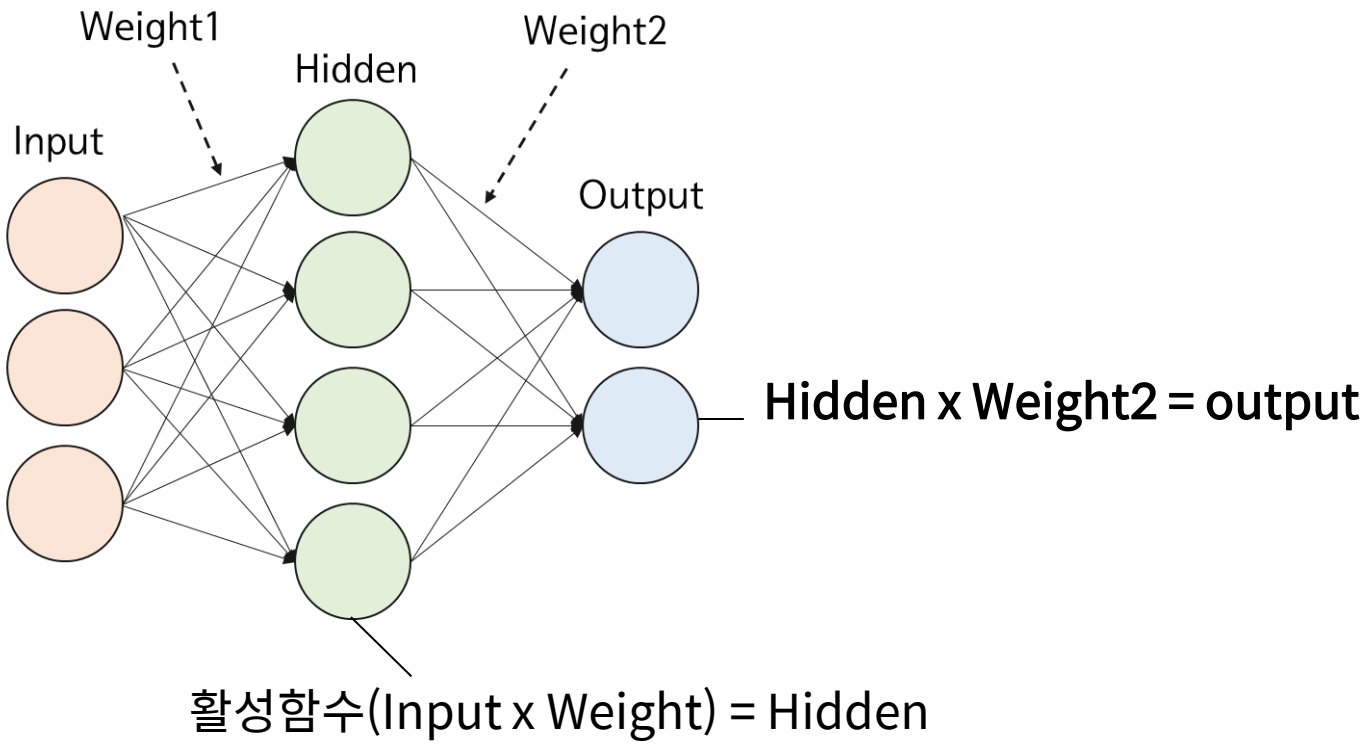
Input -> Hidden -> Output -> Error -> Hidden

Back Propagation

MLP Feed Forward

Feed Forward

- Feed Forward : 신경망은 Input에서 Weight와 Hidden을 거쳐 Output을 내보내는 과정
- Neural network는 feed forward와 back propagation 모두 행렬로 연산 됨

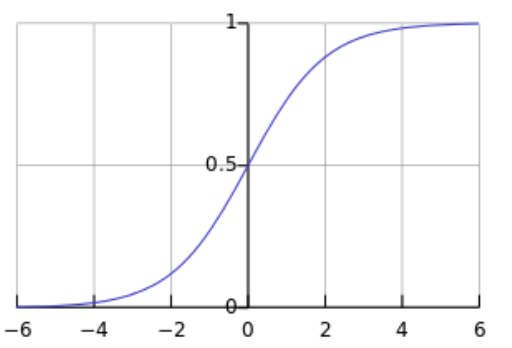


MLP Feed Forward

활성 함수 (Activation Function)

- 입력 값과 가중치 곱의 합을 입력으로 받아 출력하는 함수로, 일반적으로 미분 가능(differentiable)하고 단조 증가하는 시그모이드 함수(sigmoid function)를 기본적으로 사용
- 시그모이드 함수비선형 활성 함수 중에 가장 기본 적으로 사용하는 함수 (선형 함수만을 사용할 경우, 여러 층의 선형 함수들을 계층적으로 연결하더라도 선형적인 출력만이 가능)
- 입력 값이 0이하이면 0.5이하의 값을 출력하고, 입력 값이 0이상이면 0.5이상의 값을 출력
- 입력 값에 대하여 0부터 1사이로 Scaling해주는 개념
- 비선형 활성 함수를 사용 하는 이유는, 우리가 풀고자 하는 문제가 비선형적인 복잡한 문제이기 때문.
- Input과 Weight의 선형결합이 활성함수로 들어가게 됨. 선형결합을 비선형화 시킨 개념이라고 볼 수 있음. 기존에는 신경망 모형이 직선밖에 긋지 못했다면 비선형화 시켜 여러 곡선의 조합을 통해 복잡한 문제를 풀 수 있게 됨

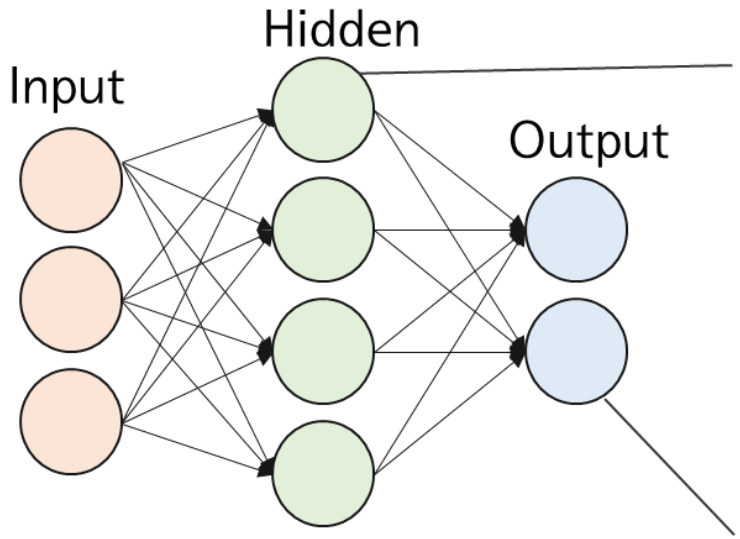
$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \text{and} \quad \frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$$



MLP Feed Forward

Feed Forward 계산

- Input data x
- 학습 해야 할 Weight w_{ih}, w_{hj}
- Output y
- True label t



$$Z_h{}^k = \sum w_{ih}{}^k x_i{}^k$$

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

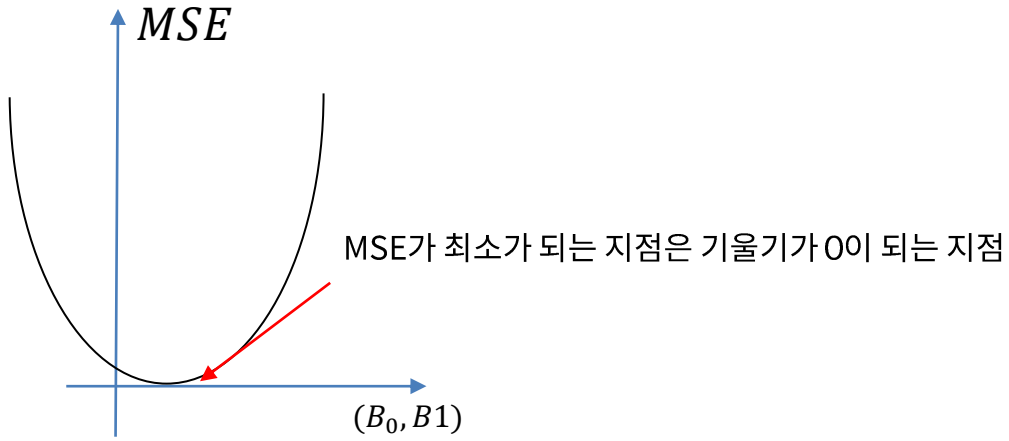
$$y_j{}^k = \sum w_{hj}{}^k \sigma(Z_h{}^k)$$

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

MLP Back Propagation

Gradient Descent Method

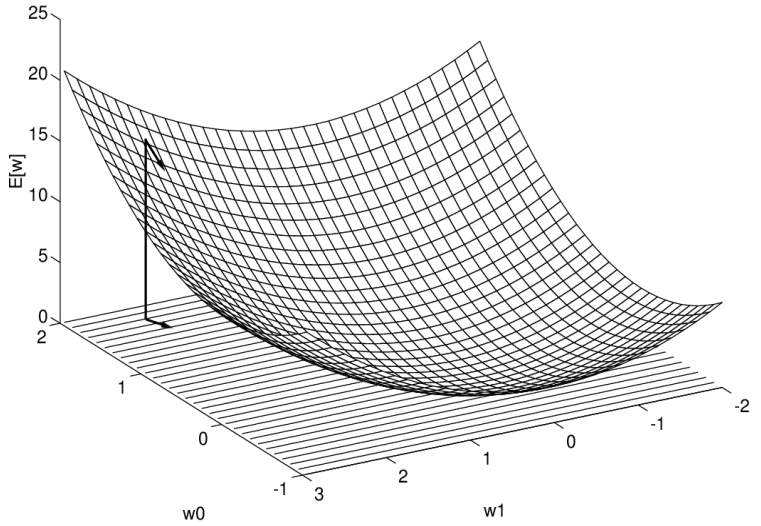
- 가장 간단한 선형 회귀 모형은 Loss Function을 MSE로 설정하여 MSE가 감소하도록 회귀계수 (β)를 추정
- MSE는 2차함수의 형태이므로 이 MSE가 최소가 되는 지점을 찾을 수 있음
- MSE는 회귀계수에 대한 함수이므로 회귀계수로 미분하여 기울기가 0이 되는 지점을 찾으면 그때의 회귀계수는 MSE가 최소가 되는 회귀 계수
- 직선으로 이루어진 비교적 간단한 모델이기 때문에, MSE를 회귀계수(β)로 미분하여 기울기가 0이 되는 지점을 한번에 찾을 수가 있음



MLP Back Propagation

Gradient Descent Method

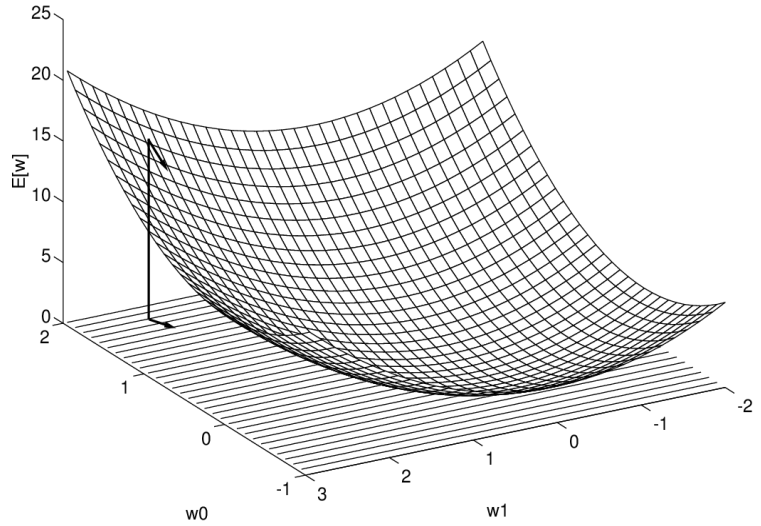
- 신경망 모형 같은 경우는 앞서 서술하였듯이 매우 복잡한 모형.
- Hidden Layer가 깊어지면 깊어질수록, Hidden Node가 많아지면 많아질수록 더욱 복잡 해짐. 그렇기 때문에, 신경망 모형에서 MSE를 신경망 모형의 Weight로 미분해서 기울기가 0이 지점을 찾을 수 없음
- MSE를 신경망 모형의 Weight로 미분하여 기울기를 감소시켜 최소가 되는 지점을 찾아갑니다. 다시 말해, 한번에 기울기가 최소가 되는 지점을 찾기 어렵기 때문에, 산 정상에서 차근차근 내려오면서 길을 찾듯이 기울기를 조금씩 구해 MSE가 낮아지는 지점을 찾아 감



MLP Back Propagation

Gradient Descent Method

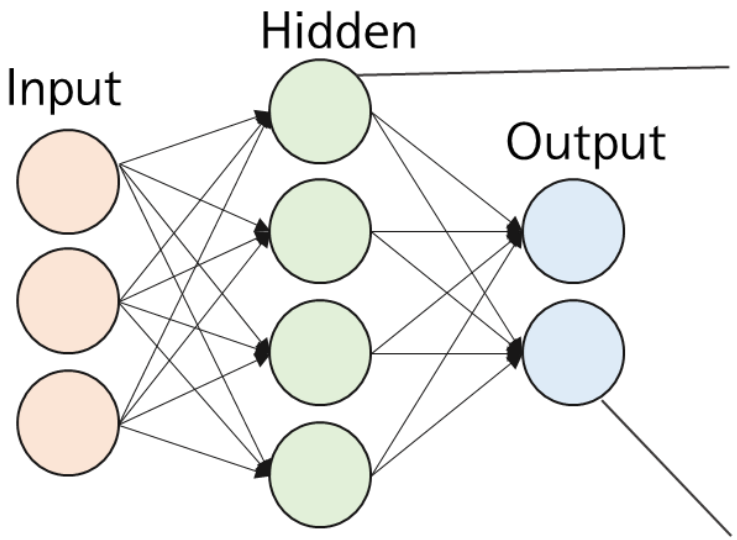
- Weight에 의해 MSE값이 달라지게 되고, 초기 지점에서 MSE가 최소가 되는 지점을 기울기 구해 찾아감.
- 초기 Weight를 어떻게 설정 하느냐에 따라서 MSE가 최소가 되는 지점을 찾아가는 시간이 달라 질 수 있음.
- 기본적으로는 정규분포 (Normal Distribution) 또는 균등분포 (Uniform Distribution)에서 난수를 추출해서 초기 Weight를 설정하지만, 더 좋은 초기 Weight를 설정하기 위한 방법 또한 존재. 이러한 기법을 Initialization 기법이라 칭함



MLP Back Propagation

Feed Forward 계산

- Input data x / 학습 해야 할 Weight w_{ih}, w_{hj} / Output y / True label t



$$Z_h{}^k = \sum w_{ih}{}^k x_i{}^k$$

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

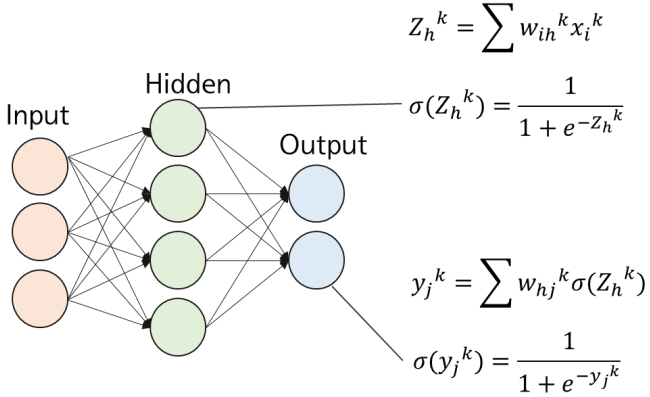
$$y_j{}^k = \sum w_{hj}{}^k \sigma(Z_h{}^k)$$

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

$$\varepsilon_k = \frac{1}{2}\sum \left(t_j^k - \sigma(y_j{}^k)\right)^2$$

MLP Back Propagation

Back Propagation 계산



- 신경망의 Loss를 아래와 같이 MSE로 설정하였을 때, 이 loss를 각각w_{ih}와 w_{hj}로 미분하여 기울기를 구함. 그리고 현재 Weight 에서 기울기만큼 빼줌

$$\varepsilon_k = \frac{1}{2}\sum\left(t_j^k - \sigma(y_j{}^k)\right)^2$$

- 정리하면, 각각의 Weight는 다음과 같은 식으로 Update 할 수 있음

Input data x / 학습 해야 할 Weight w_{ih}, w_{hj} / Output y / True label t

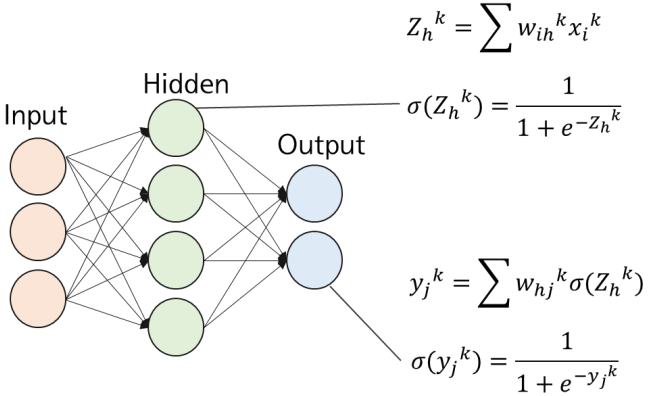
$$w_{ih}{}^{k+1} = w_{ih}{}^k + \Delta w_{ih}{}^k \qquad w_{hj}{}^{k+1} = w_{hj}{}^k + \Delta w_{hj}{}^k$$

$$= w_{ih}{}^k - \frac{\partial \varepsilon_k}{\partial w_{ih}{}^k} \qquad = w_{hj}{}^k - \frac{\partial \varepsilon_k}{\partial w_{hj}{}^k}$$

MLP Back Propagation

Back Propagation 계산

- 뒤의 Weight부터 Update해야 하기 때문에, w_{hj}로 미분
- ε_k를 한번 w_{hj}로 미분하는 것이 어려우므로 Chain Rule에 의해 다음과 같이 쓸 수 있고, 세개의 Term에 대해서 각각 아래와 같이 미분 할 수 있음.



$$\therefore \frac{\partial \varepsilon_k}{\partial w_{hj}^k} = \frac{\partial \varepsilon_k}{\partial \sigma(y_j^k)} \frac{\partial \sigma(y_j^k)}{\partial y_j^k} \frac{\partial y_j^k}{\partial w_{hj}^k}$$

$$-\left(t_j^k - \sigma(y_j^k)\right) = -e_j^k \qquad \sigma(y_j^k)\left(1-\sigma(y_j^k)\right) \qquad \sigma(Z_h^k)$$

$$\frac{\partial \varepsilon_k}{\partial w_{hj}^k} = -e_j^k \, \sigma(y_j^k)' \sigma(Z_h^k)$$

Input data x / 학습 해야 할 Weight w_{ih}, w_{hj} / Output y / True label t

$$w_{ih}^{k+1} = w_{ih}^k + \Delta w_{ih}^k \qquad w_{hj}^{k+1} = w_{hj}^k + \Delta w_{hj}^k$$

$$= w_{ih}^k - \frac{\partial \varepsilon_k}{\partial w_{ih}^k} \qquad = w_{hj}^k - \frac{\partial \varepsilon_k}{\partial w_{hj}^k}$$

MLP Back Propagation

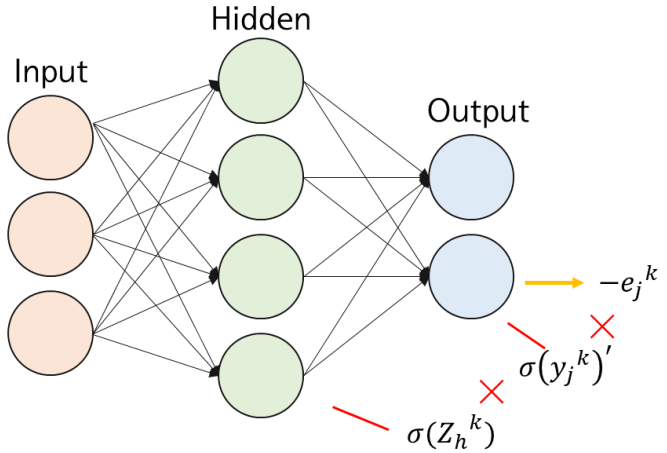
Back Propagation 계산

- **Weight의 기울기는 Error부터 전파 되어 옴**

$$\therefore \frac{\partial \varepsilon_k}{\partial {w_{hj}}^k} = \frac{\partial \varepsilon_k}{\partial \sigma({y_j}^k)} \frac{\partial \sigma({y_j}^k)}{\partial {y_j}^k} \frac{\partial {y_j}^k}{\partial {w_{hj}}^k}$$

$$-\left(t_j^k - \sigma\left({y_j}^k\right)\right) = -{e_j}^k \qquad \sigma({y_j}^k)\left(1 - \sigma\left({y_j}^k\right)\right) \qquad \sigma({Z_h}^k)$$

$$\frac{\partial \varepsilon_k}{\partial {w_{hj}}^k} = -{e_j}^k \, \sigma\left({y_j}^k\right)' \sigma({Z_h}^k)$$



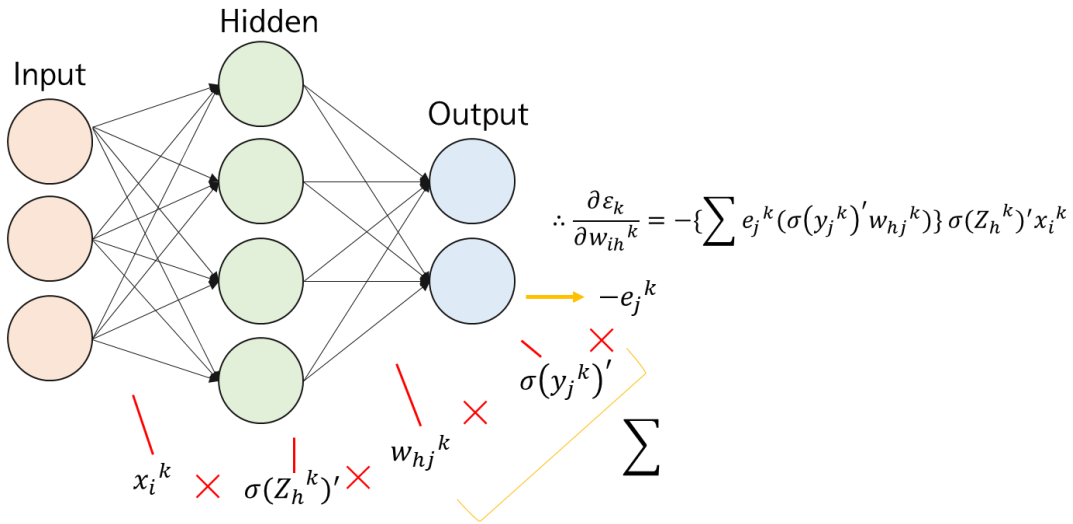
MLP Back Propagation

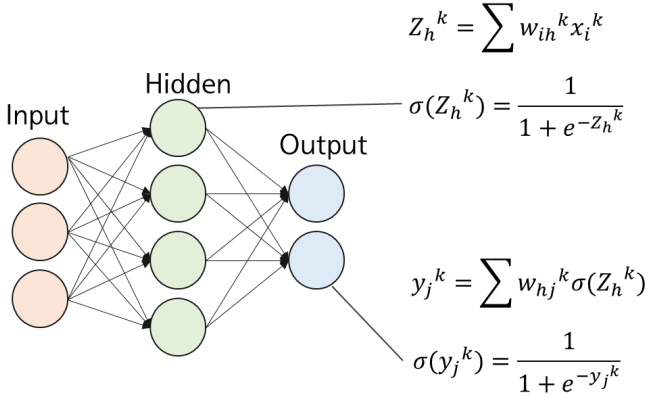
Back Propagation 계산

- ε_k에 대하여 첫번째 Weight w_{ih}로 미분

$$\frac{\partial \varepsilon_k}{\partial {w_{ih}}^k} = \frac{\partial \varepsilon_k}{\partial \sigma({Z_h}^k)} \frac{\partial \sigma({Z_h}^k)}{\partial {Z_h}^k} \frac{\partial {Z_h}^k}{\partial {w_{ih}}^k}$$

$$\sum \frac{\partial \varepsilon_k}{\partial {y_j}^k} \frac{\partial {y_j}^k}{\partial \sigma({Z_h}^k)} = \sum -{e_j}^k (\sigma({y_j}^k)' {w_{hj}}^k) \quad \sigma({Z_h}^k)' \quad {x_i}^k$$





Input data x / 학습 해야 할 Weight w_{ih}, w_{hj} / Output y / True label t

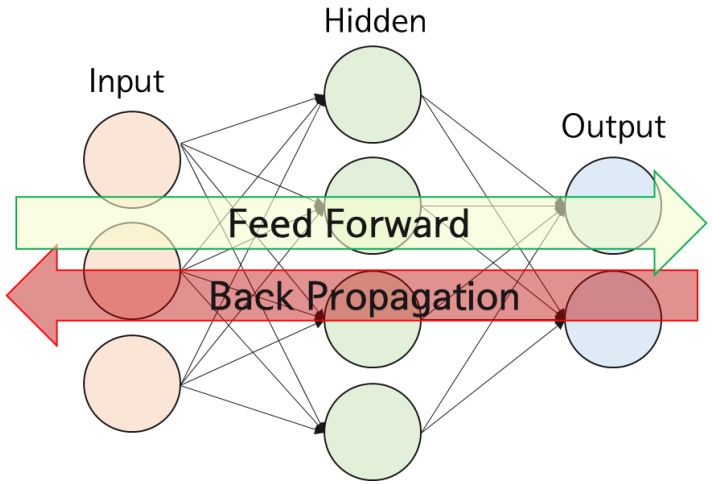
$${w_{ih}}^{k+1} = {w_{ih}}^k + \Delta {w_{ih}}^k \qquad {w_{hj}}^{k+1} = {w_{hj}}^k + \Delta {w_{hj}}^k$$

$$= {w_{ih}}^k - \frac{\partial \varepsilon_k}{\partial {w_{ih}}^k} \qquad = {w_{hj}}^k - \frac{\partial \varepsilon_k}{\partial {w_{hj}}^k}$$

MLP Back Propagation

Multi Layer Perceptron 학습

- MLP는 Feed Forward와 Back Propagation을 번갈아가며 학습을 진행함
- 학습함에 따라 점차 학습 데이터에 대한 MSE가 줄어들게 됩니다



Feed Forward

Input -> Hidden -> Output -> Error -> Hidden

Back Propagation

MLP Back Propagation

Multi Layer Perceptron 학습

- 우리가 가지고 있는 모든 데이터를 한번에 Feed Forward하지는 않고, 너무 많은 연산을 필요로 하여 컴퓨팅 문제가 발생하기 때문에 굉장히 비효율적인 학습 과정을 거치게 됨.
- 우리가 가지고 있는 데이터를 쪼개서 Feed Forward함. 우리가 가지고 있는 전체 데이터가 1,000개라고 하면 100개씩 쪼개서 10번 Feed Forward와 Back Propagation을 반복함. 이 한 과정을 Epoch(세대)라고 하고 여기서 100개의 데이터를 Mini-Batch라고 부르며 100의 크기에 대해서는 Batch Size라 부름.
- 데이터를 쪼개서 Gradient Descent Method하는 방법을 Stochastic Gradient Descent (SGD)라고 부르며 이렇게 Gradient Descent해주는 것 들을 통틀어서 Optimizer라고 부름.

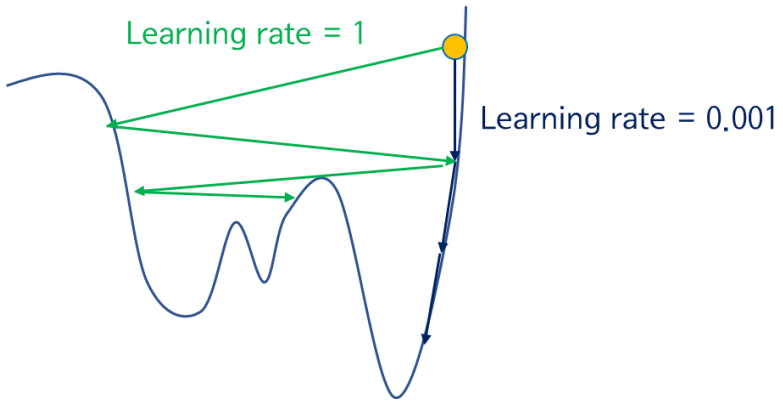
MLP Back Propagation

Multi Layer Perceptron 학습

- Weight의 Gradient는 위 잎서 언급한 식 과 같이 Update 되지만, 일반적으로 구해지는 Gradient의 크기는 매우 큼.
- 그래서 이 Gradient의 크기를 조절해줄 필요가 있는데, 이 조절해주는 상수를 **Learning Rate** (η)라고 부르고 Learning Rate까지 넣었 을 때 Weight가 Update되는 식은 다음과 같이 쓸 수 있음
- Learning Rate를 지정해주지 않으면 아래와 같이 Gradient를 구했을 때 Loss가 제대로 감소되는 방향을 구하지 못할 확률이 높음

$$w_{ih}{}^{k+1} = w_{ih}{}^{k} + \Delta w_{ih}{}^{k} \qquad w_{hj}{}^{k+1} = w_{hj}{}^{k} + \Delta w_{hj}{}^{k}$$

$$= w_{ih}{}^{k} - \eta \frac{\partial \varepsilon_k}{\partial w_{ih}{}^{k}} \qquad = w_{hj}{}^{k} - \eta \frac{\partial \varepsilon_k}{\partial w_{hj}{}^{k}}$$



Universal Approximation Theorem

Universal Approximation Theorem

- 신경망 이론 중에 가장 유명하고 신경망 학습의 특성을 잘 나타내 주는 이론
- Hidden layer가 하나 이상인 신경망은 학습데이터 내에서 어떠한 함수 든 근사 (Approximation)시킬 수 있다
- 기본적인 MLP이상의 신경망은 학습 데이터 내에서 어떠한 모델이든 만들 수 있다라는 것
- 내가 가지고 있는 학습 데이터 내에서는 어떠한 함수 든 만들 수 있지만, 실제 데이터에는 잘 맞는다라는 보장은 하지 못하는 것
- 결국에 Overfitting의 개념과 이어짐. 학습 데이터내에서 완벽하게 맞춘다라는 것은 이전에 서술했던 것처럼 Overfitting이 굉장히 심해질 수 밖에 없다는 것을 뜻함

신경망 모형의 단점

Overfitting

- 신경망 모형 뿐만이 아니라, AI나 Data Science 전반적으로 굉장히 큰 이슈
- 실험 설계를 통해 적절히 데이터를 나누어 과적합 정도를 파악하고, 적절한 모델과 변수를 선택
- 신경망의 목적은 결국 학습 데이터내에서 loss를 최소화 시키는 것이기 때문에 더 좋은 Decision Boundary를 만들려고 노력하지 않음

신경망 모형의 단점

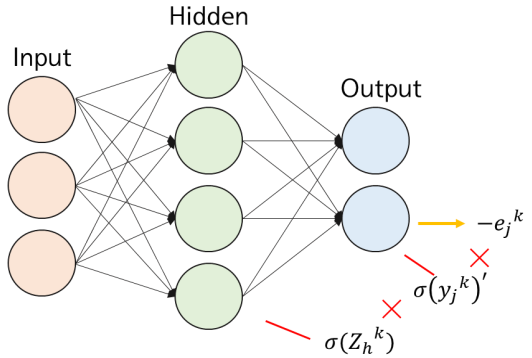
Gradient Vanishing Problem

- 기울기가 사라지는 현상을 의미

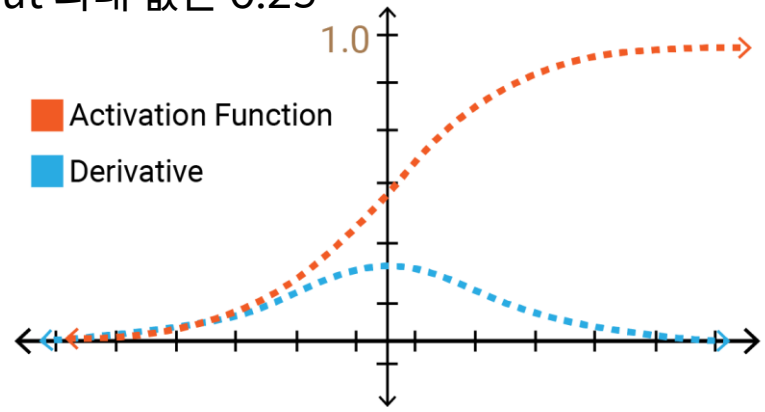
Back Propagation 계산 - Weight의 기울기는 Error부터 전파 되어 옴

$$\therefore \frac{\partial \varepsilon_k}{\partial {w_{hj}}^k} = \frac{\partial \varepsilon_k}{\partial \sigma({y_j}^k)} \frac{\partial \sigma({y_j}^k)}{\partial {y_j}^k} \frac{\partial {y_j}^k}{\partial {w_{hj}}^k}$$

$$-\left(t_j^k - \sigma\left({y_j}^k\right)\right) = -{e_j}^k \sigma({y_j}^k)\left(1 - \sigma\left({y_j}^k\right)\right) \qquad \sigma({Z_h}^k)$$



- Sigmoid의 미분한 값의 Output 최대 값은 0.25

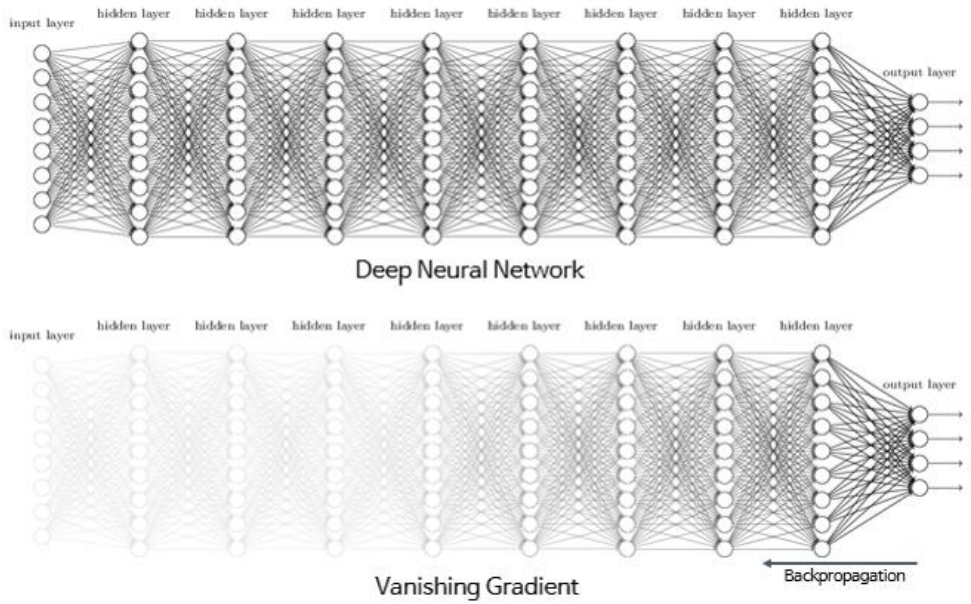


출처 : https://codeodysseys.com/posts/gradient-calculation/

신경망 모형의 단점

Gradient Vanishing Problem

- Hidden Layer가 깊어지면 깊어질 수록 앞의 Gradient를 구해주기 위해서는 이 Sigmoid의 미분한 값을 계속 곱해 주어야 함
- Layer가 깊어지면 깊어질수록 앞 부분의 Weight의 Gradient는 큰 변화가 없어지고, Weight의 변화가 거의 일어나지 않는 현상
- Layer가 깊으면 깊을수록 Gradient Vanishing이 일어나면서 신경망이 가지는 장점이 발휘 되지 못함



출처 : https://excelsior-cjh.tistory.com/177