TensorFlow (2)

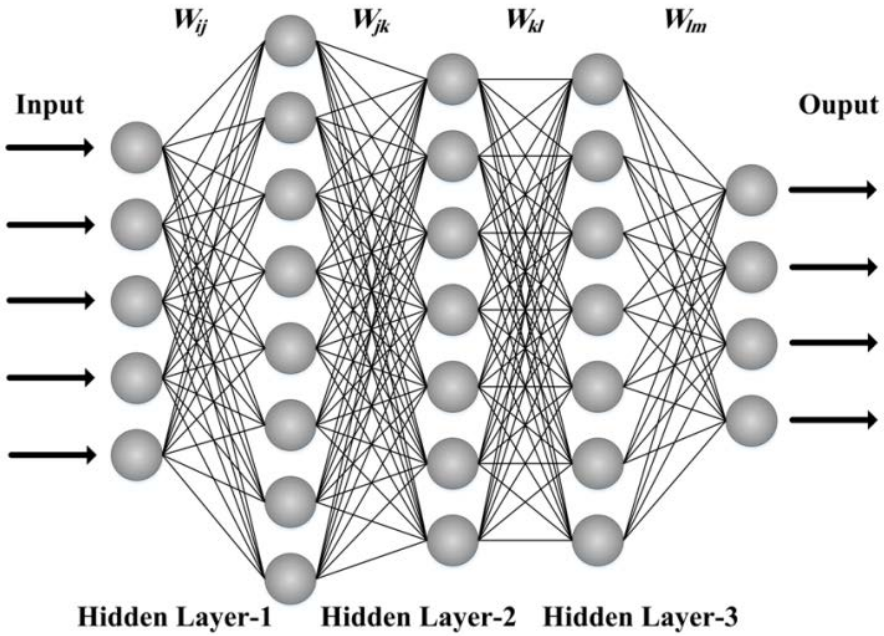
고려대학교 INI Lab

Contents

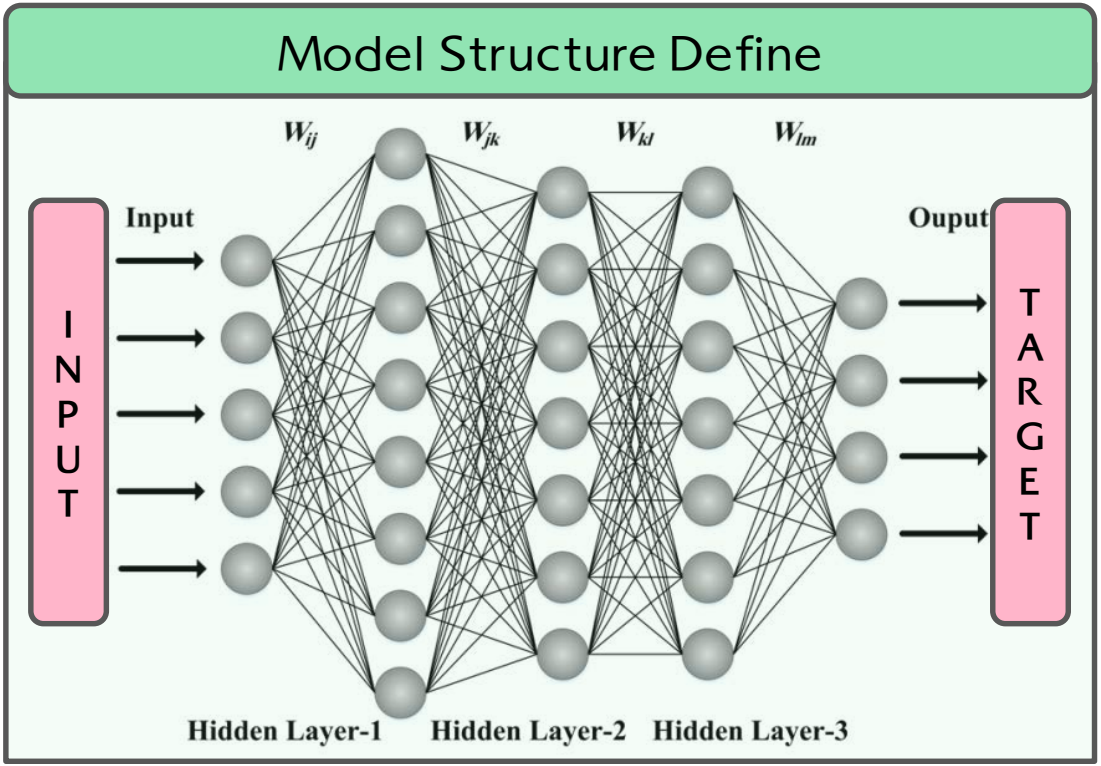
Feed Forward Neural Network

Feed Forward Neural Network

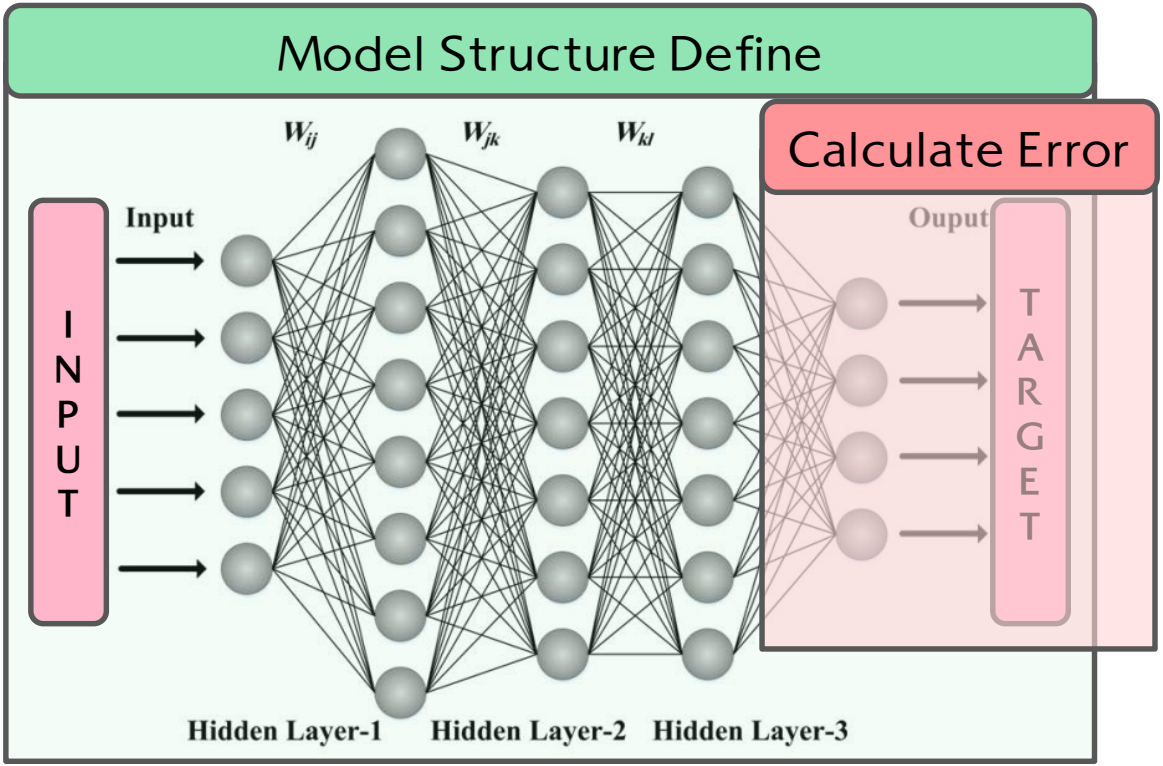
여러 hidden layer를 dense하게 쌓아서 학습시키는 모델
Multi-Layer Perceptron이라고도 부름



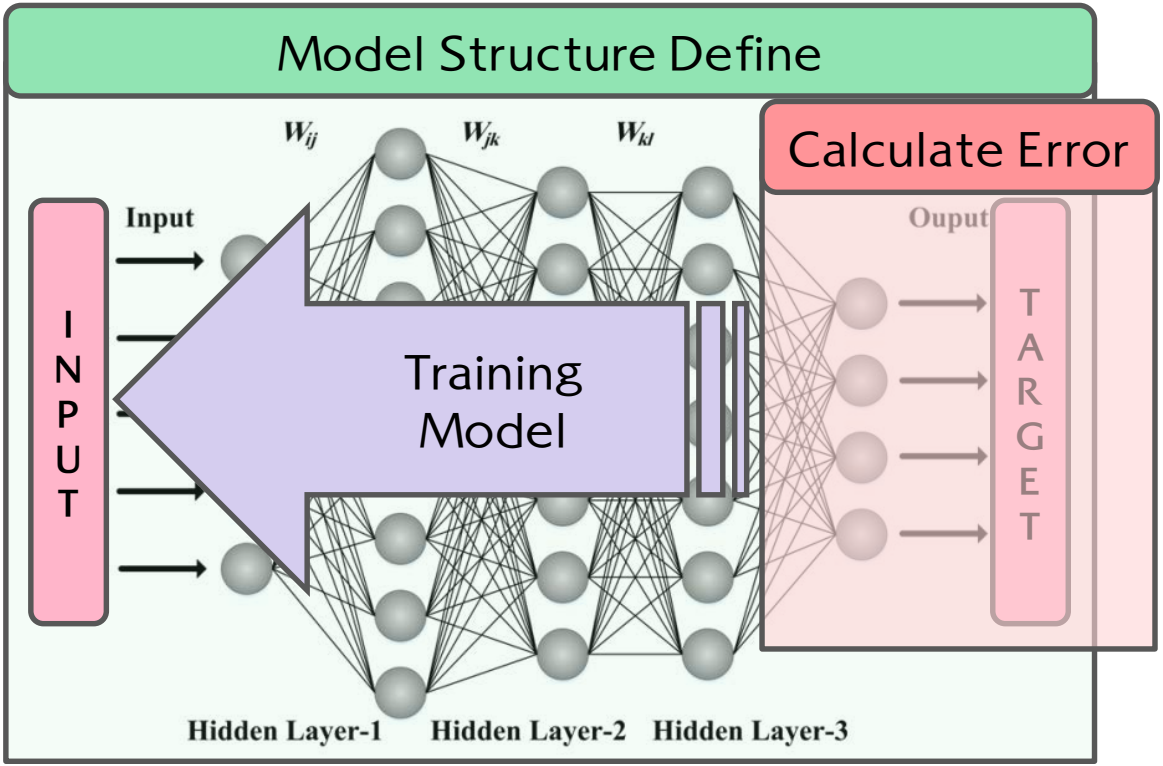
Build FFNN Model



Build FFNN Model

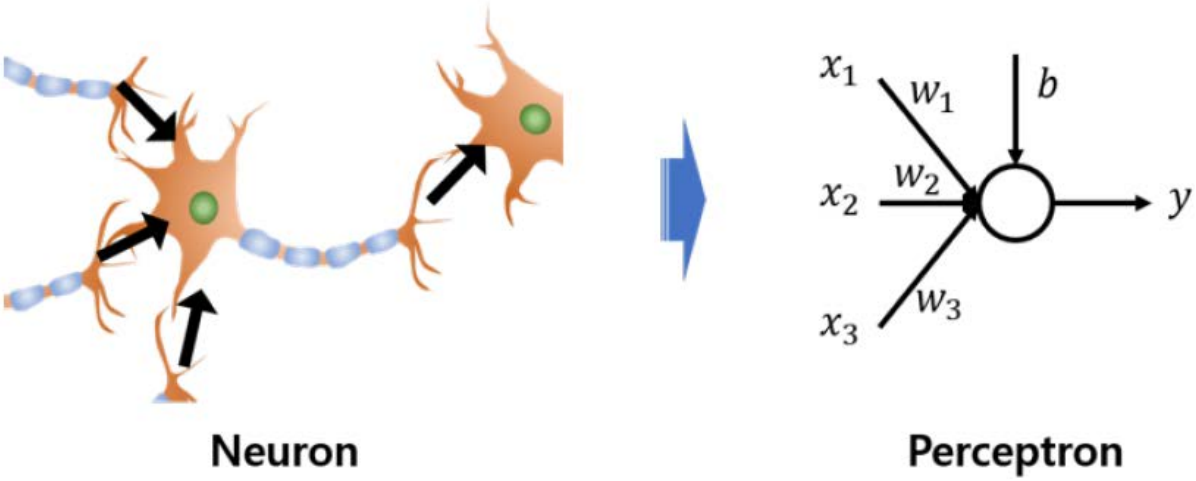


Build FFNN Model



Perceptron

입력 값을 받고, 이를 계산하여 값을 출력하는 neural network의 기본 단위

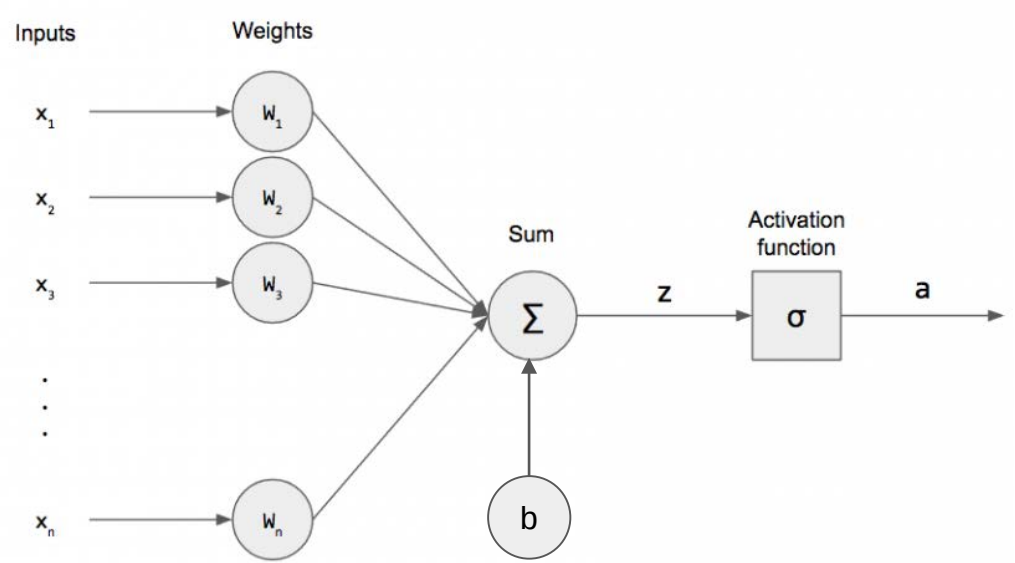


Perceptron

$$\sigma(X, W, b) = x_1 w_1 + x_2 w_2 + x_3 w_3 + \cdots + x_n w_n + b$$

$$= \sum_{1}^{n} x_i w_i + b$$

$$= X \cdot W^T + b$$

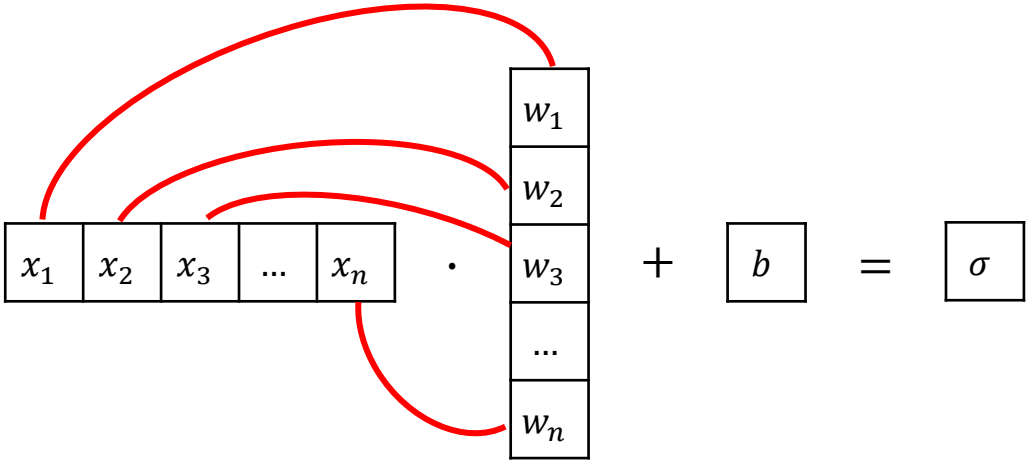


Perceptron

$$\sigma(X, W, b) = x_1 w_1 + x_2 w_2 + x_3 w_3 + \cdots + x_n w_n + b$$

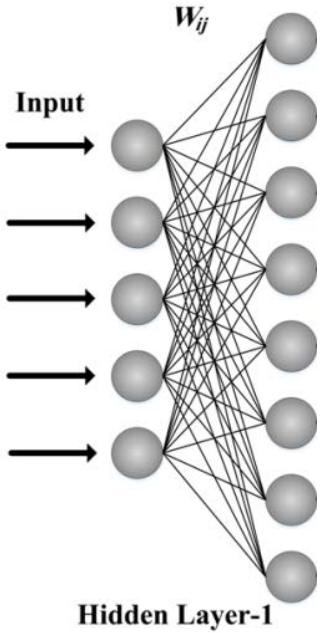
$$= \sum_{1}^{n} x_i w_i + b$$

$$= X \cdot W^T + b$$



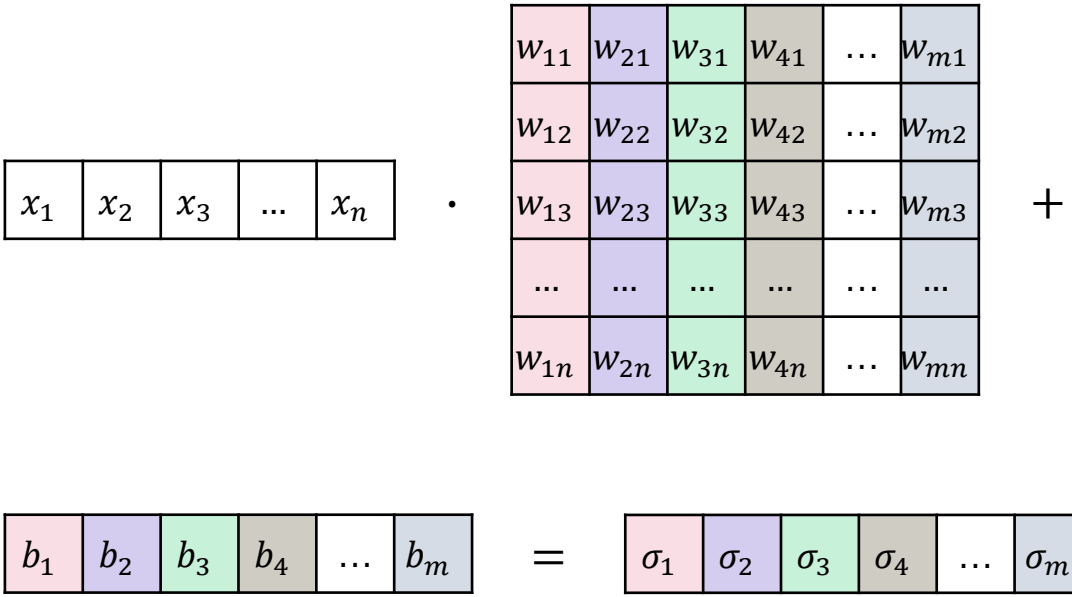
Neural Network Layer

각각의 layer는 여러 개의 perceptron의 묶음으로 이루어져 있음.



Neural Network Layer

Single Perceptron과 유사하게 layer-by-layer 연산 수행 가능



Error(cost) Calculation

모델이 학습한 결과인 Output Layer의 값과 실제 결과(Target)을 비교

▶ 두 값의 차이가 클 수록, 즉 error가 클 수록 잘못된 내용을 학습한 것

▶ mean-square error

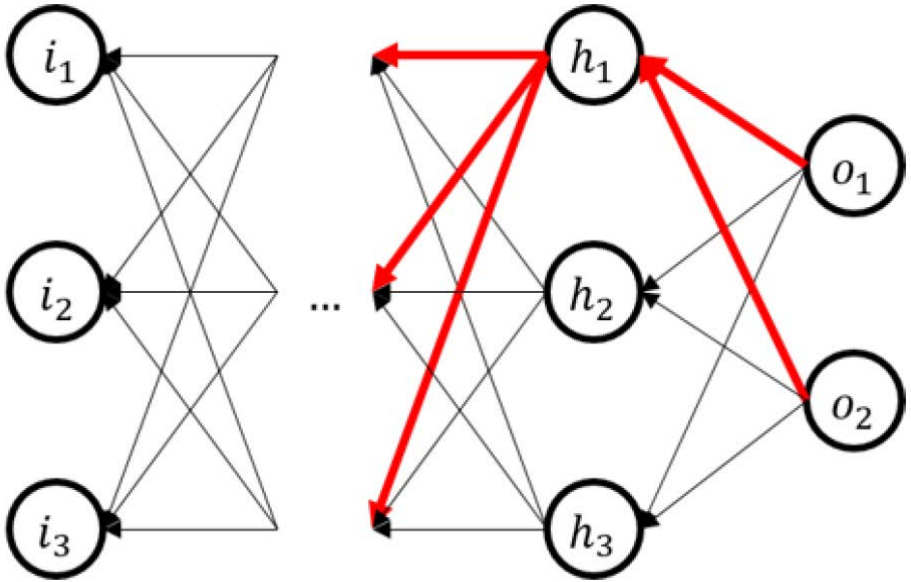
▶ cross-entropy

등의 기법을 통해 error 계산

Training Neural Network

Error Backpropagation(오류 역전파)

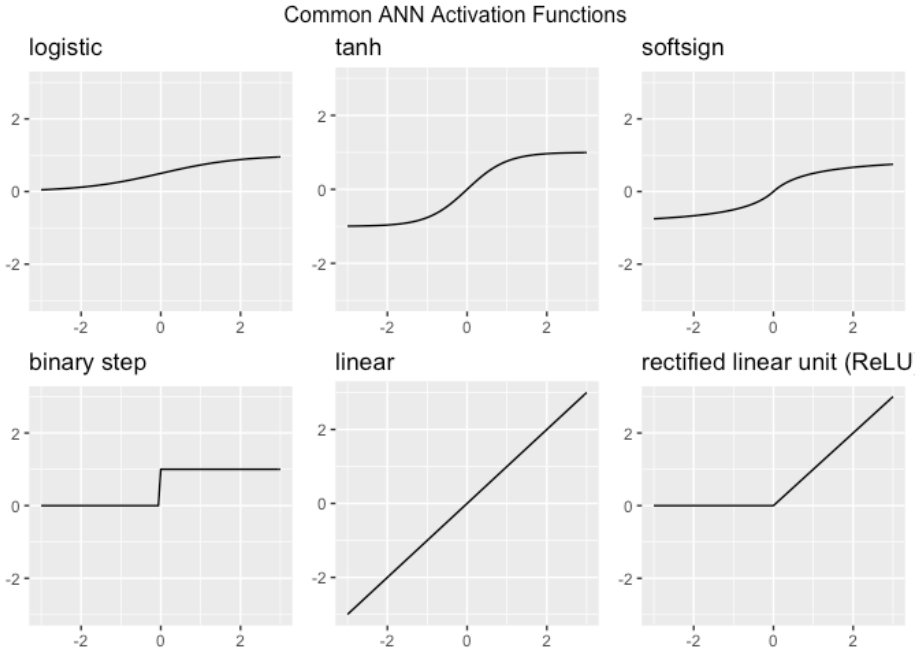
▶ 출력의 오차 값을 미분해서 다음 layer로 차례대로 전달



Improve Deep Learning Model

Activation Function

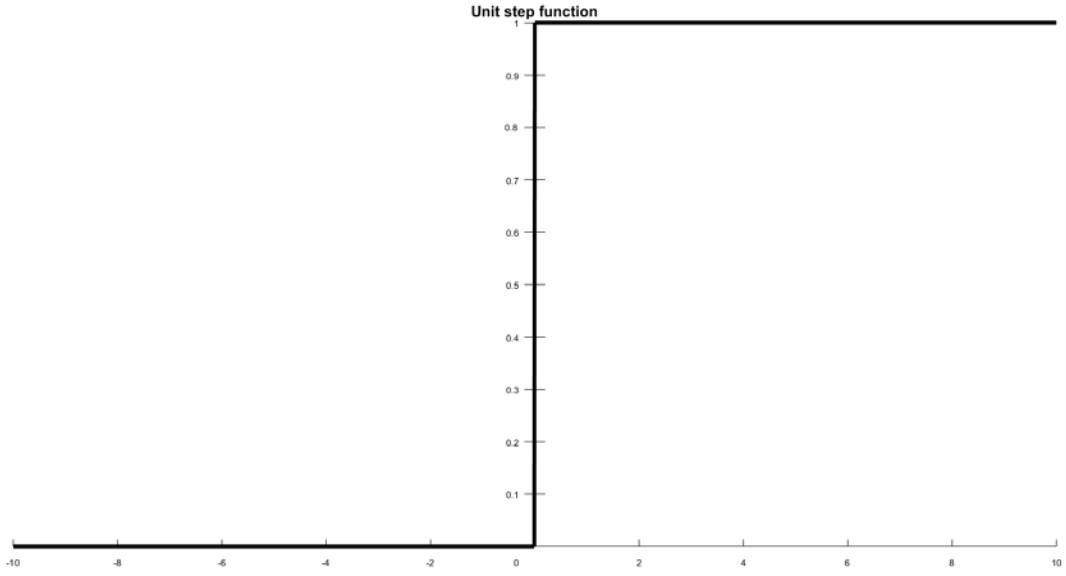
Layer에서 학습하는 결과를 Non-linear하게 나타내기 위해 사용



Binary(Step) Activation

Neuron의 원래 의미에 가장 가까운 Activation Function

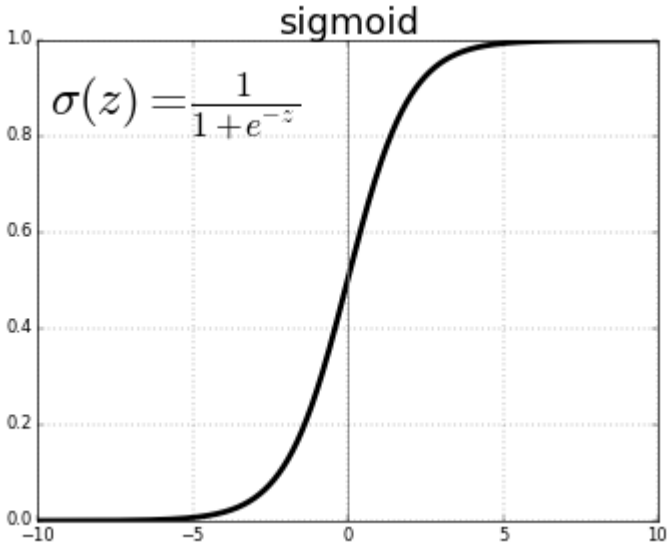
▶ 미분이 항상 0이기 때문에, 미분에 기반한 값 수정이 아닌 고정된 값을 이용한 update를 진행해야 함.



Sigmoid Activation

(0, 1) 사이의 실수 값으로 출력 값을 반환함.

▶ 미분을 간단하게 표현할 수 있음

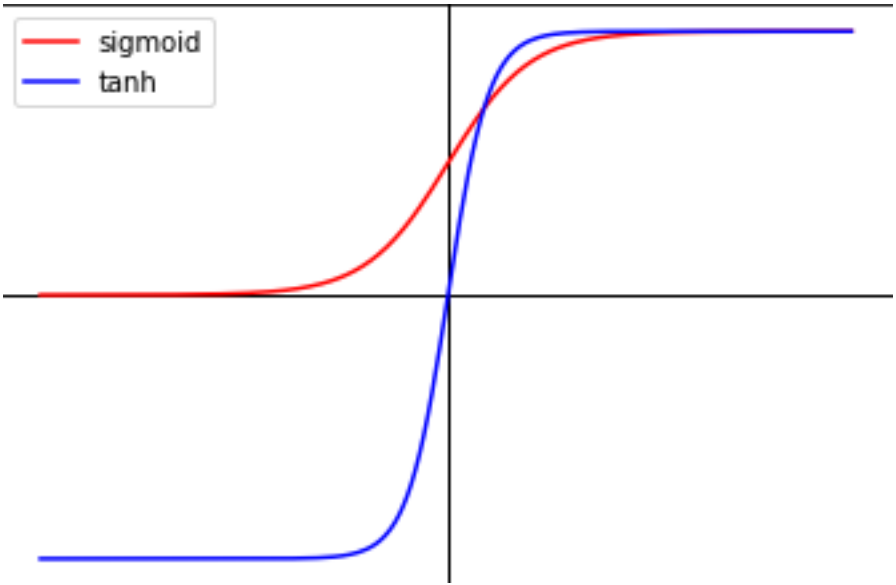


tanh Activation

(−1, 1) 사이의 실수 값으로 출력 값을 반환함.

▶ 0에 가까운 경우, sigmoid보다 더 큰 미분 값을 가지기 때문에 weight값을 더 빠르게 수정

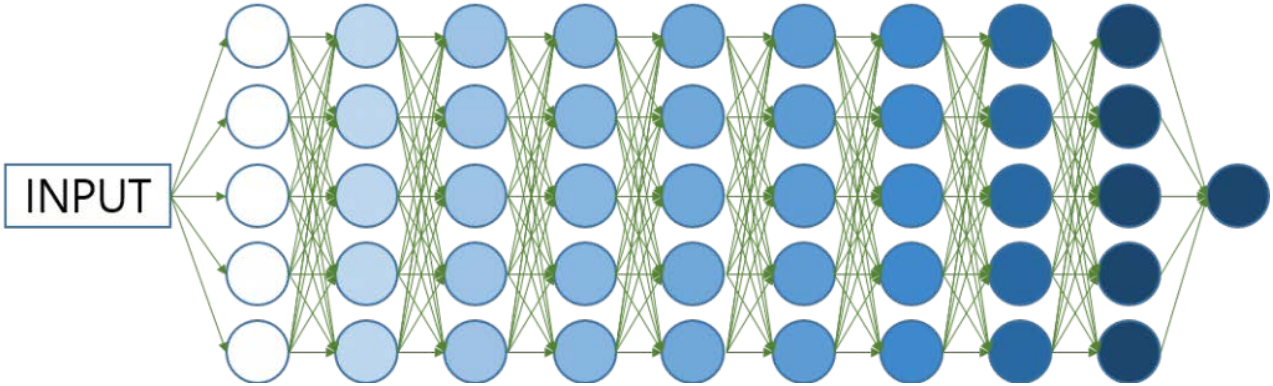
▶ 미분 역시 간단함



Vanishing Gradient Problem

Layer가 깊어질 수록 gradient 값이 점점 작아짐

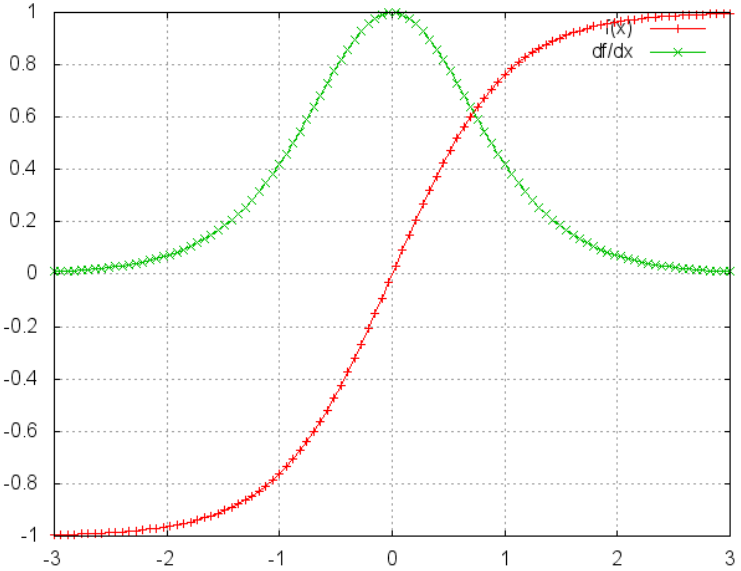
▶ Output Layer에서 멀어질 수록 weight값이 거의 업데이트 되지 않음



Vanishing Gradient Problem

Layer가 깊어질 수록 gradient 값이 점점 작아짐

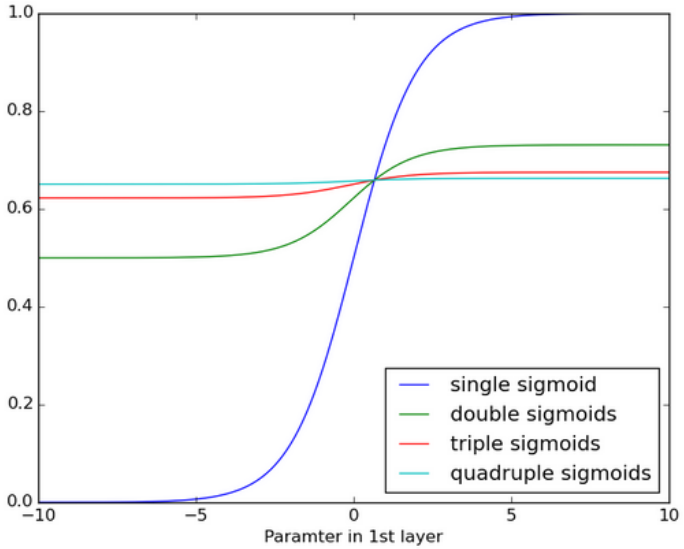
▶ Output Layer에서 멀어질 수록 weight값이 거의 업데이트 되지 않음



Vanishing Gradient Problem

Layer가 깊어질 수록 gradient 값이 점점 작아짐

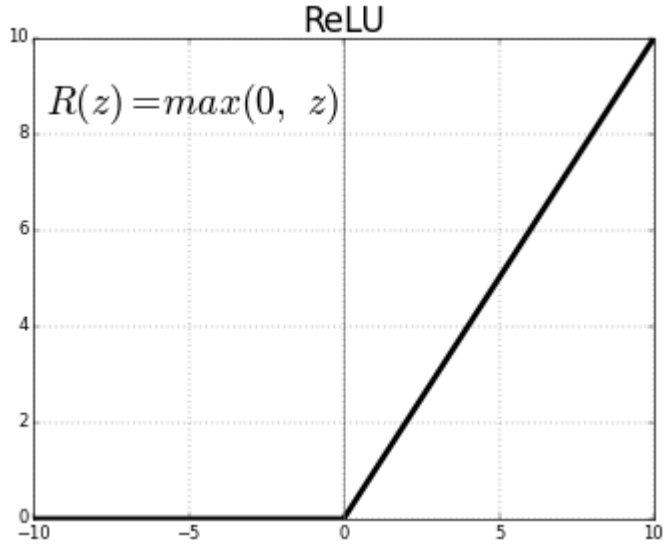
▶ Output Layer에서 멀어질 수록 weight값이 거의 업데이트 되지 않음



ReLU Activation

(−1,∞) 사이의 실수 값으로 출력 값을 반환함.

- ▶ 0보다 작을 경우 update X, 0보다 클 경우 항상 고정된 값으로 미분
- ▶ 0에서 미분이 불가능하다는 단점
- ▶ 초기 weight 값 지정에 영향을 많이 받음

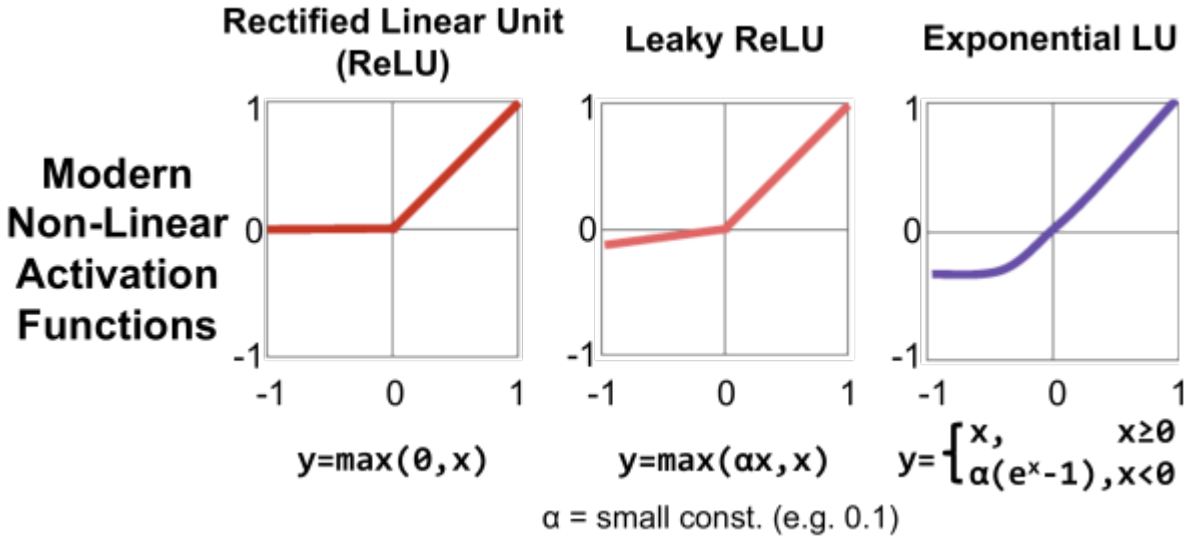


Various ReLU Activation

0보다 작을 경우 update X

0에서 미분이 불가능하다는 단점

▶ 두 가지 단점을 해결하기 위한 다양한 접근 방법들

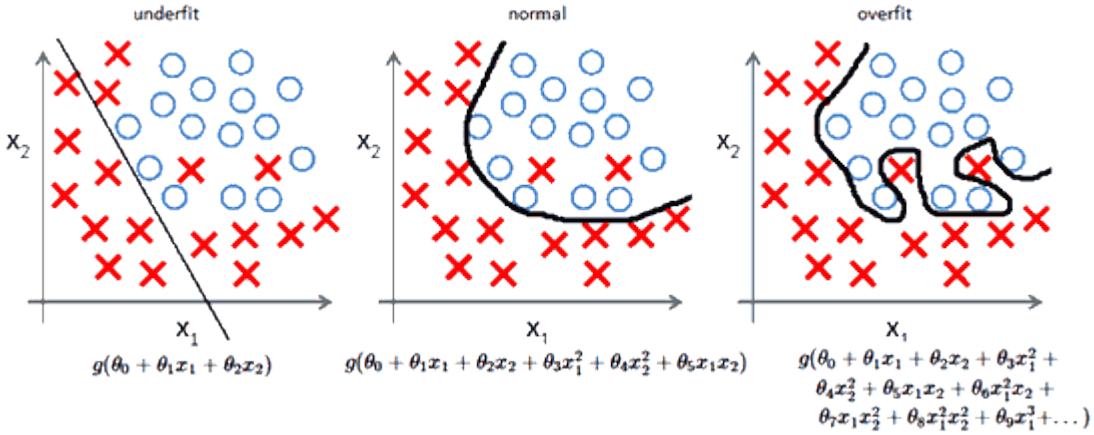


Overfit and Underfit

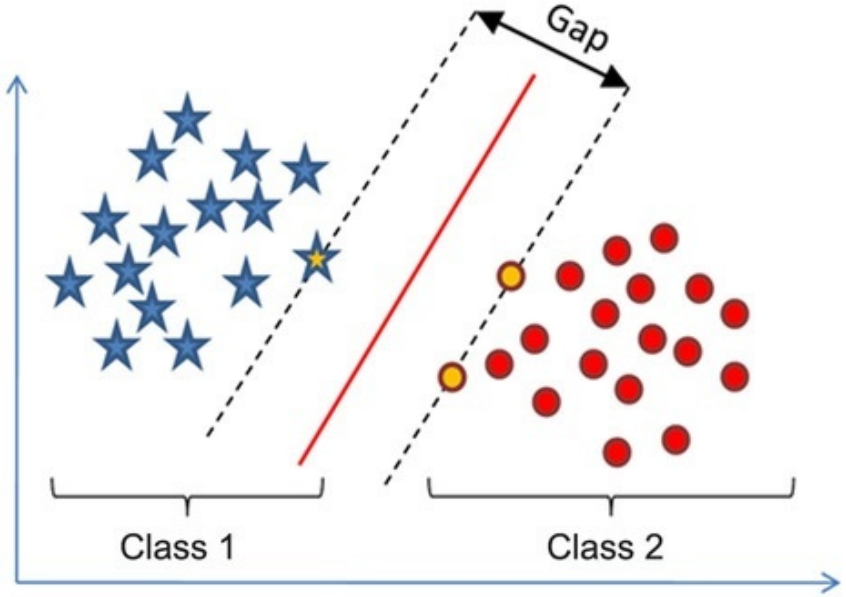
Underfit : 아직 모델이 제대로 학습되지 않았을 경우 생기는 문제

Overfit : 학습데이터를 과하게 학습할 경우 생기는 문제

▶ 두 경우에 빠지지 않았을 경우를 general한 경우라고 일컫음



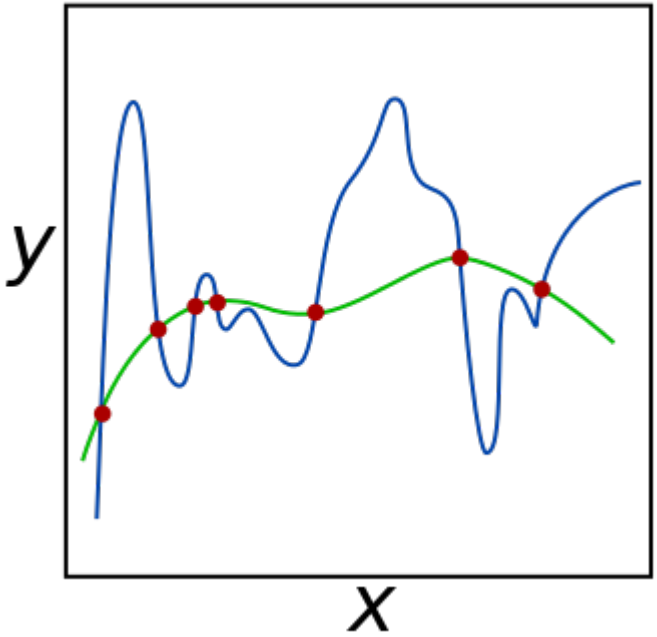
SVM and generalization



Regularization

과적합을 방지하기 위해 모델에 패널티를 부여

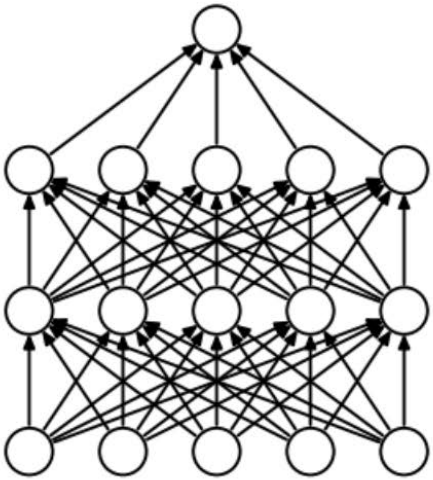
▶ 일반적으로 L1 또는 L2 regularization을 사용



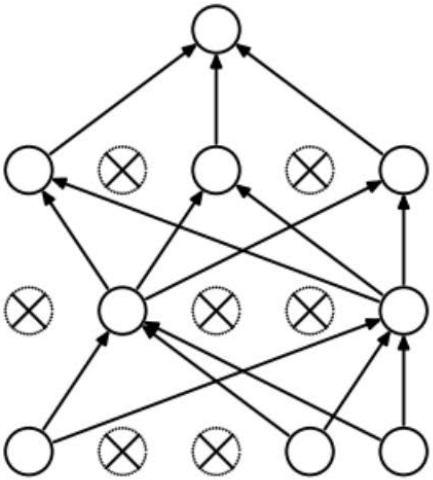
Dropout

매 epoch(mini-batch)마다 임의의 node를 제외하고 학습 진행

- ▶ Co-adaption 회피
- ▶ Ensemble 효과



(a) Standard Neural Net

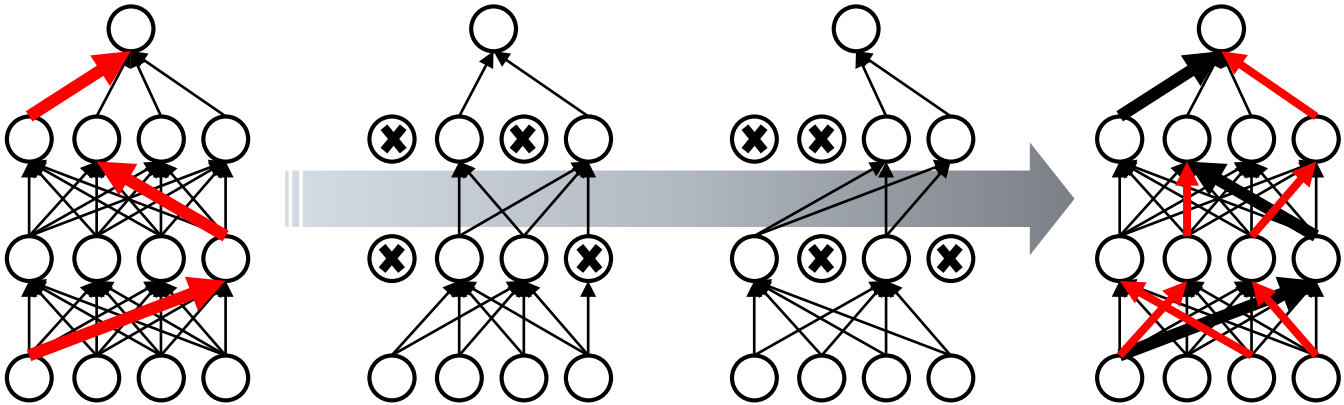


(b) After applying dropout.

Dropout

Co-adaption 회피

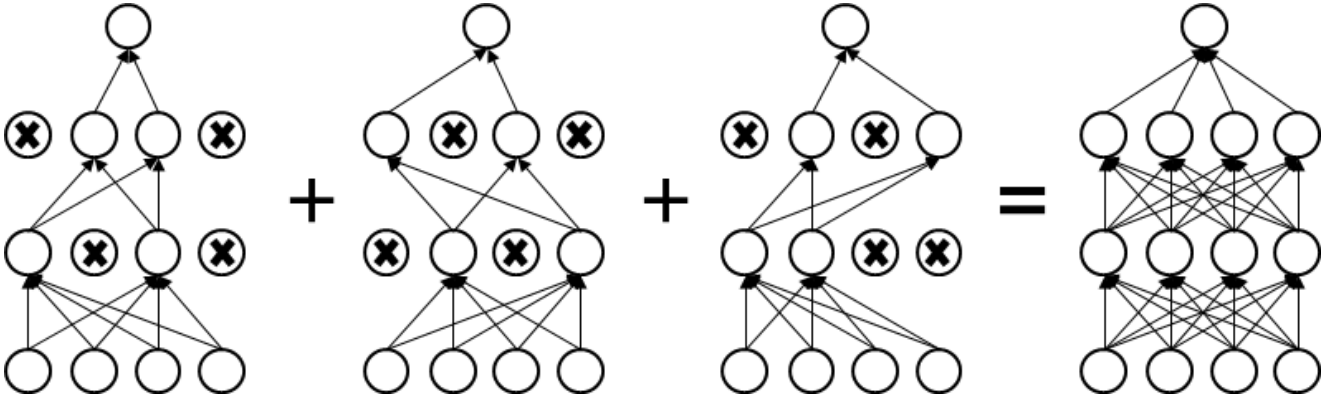
- ▶ 특정 노드의 영향력이 지나치게 강해지면 다른 노드는 학습이 진행 안 됨
- ▶ 영향력이 강한 노드를 제외시켜서 다른 노드를 학습시킴



Dropout

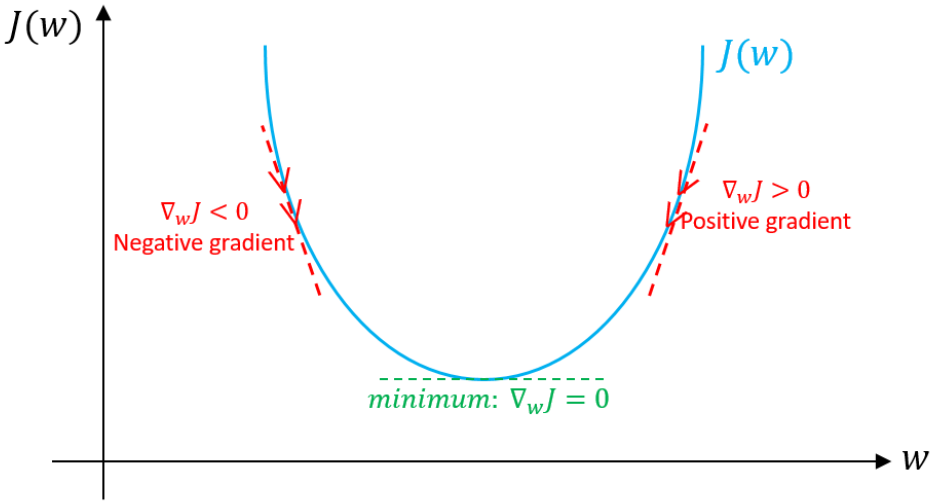
Ensemble 효과

▶ 각 batch에 따라 학습된 여러 neural network의 ensemble 관점으로 접근 가능



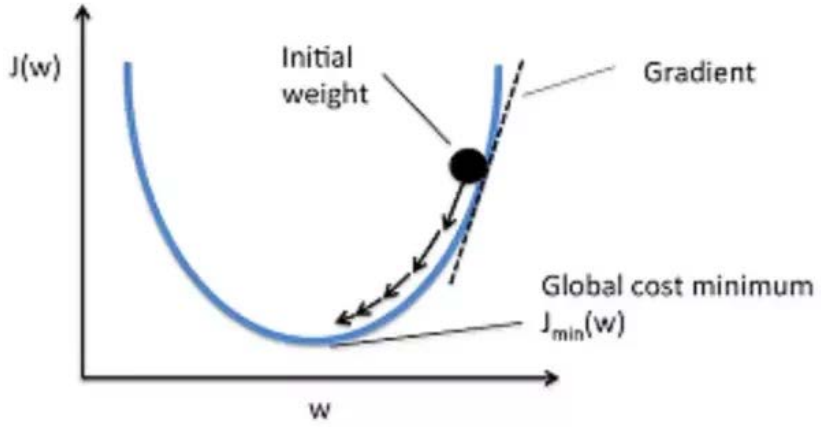
Optimizer

Error값을 최소화하는 방향으로 model을 학습시키는 기법



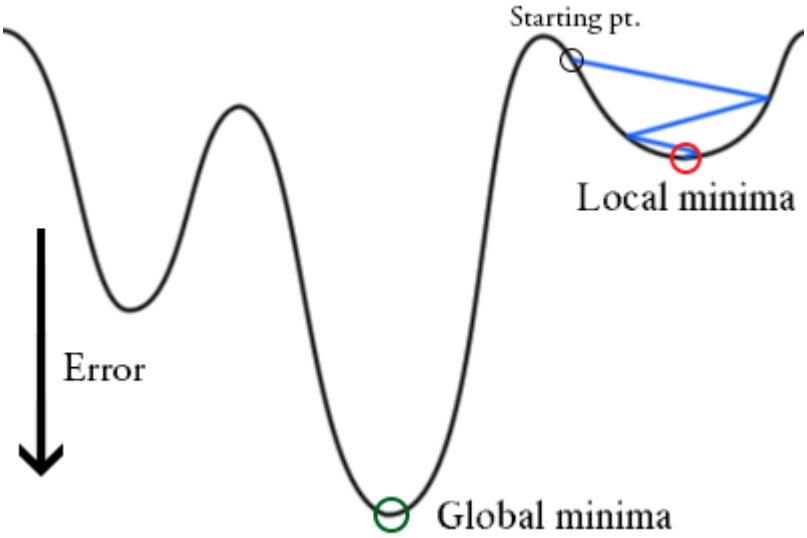
Gradient descent

매 epoch마다 error값을 계산하고, error의 미분 값에 따라 error값이 점점 작아지는(또는 커지는)형태로 모델을 수정하는 기법



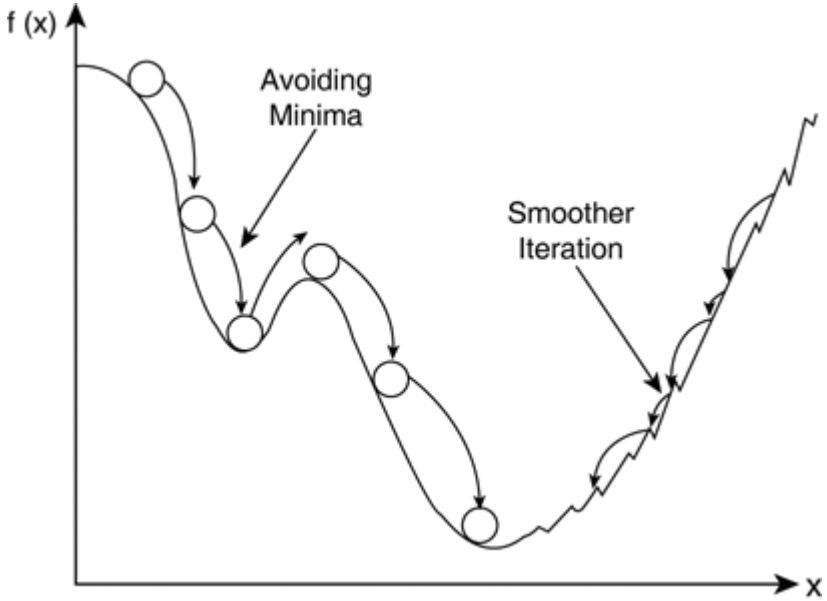
Global Minima vs Local Minima

Error가 작아지는 방향으로 내려가기 때문에, 지역 최소값에 빠지는 문제점이 발생



Momentum Optimizer

가속도 값을 이용해서 원래 이동하려는 방향으로 계속 나아가려는 성질을 통해, 얕은 local minima를 빠져나갈 수 있도록 학습



Various Optimizer

그 외에 다양한 optimizer 기법들이 존재

▶ (2017년도 기준) Adagrad기법 혹은 Rmsprop가 성능이 뛰어나다는 실험결과들이 있음