방학 2주차

PERCEPTRON/MLP

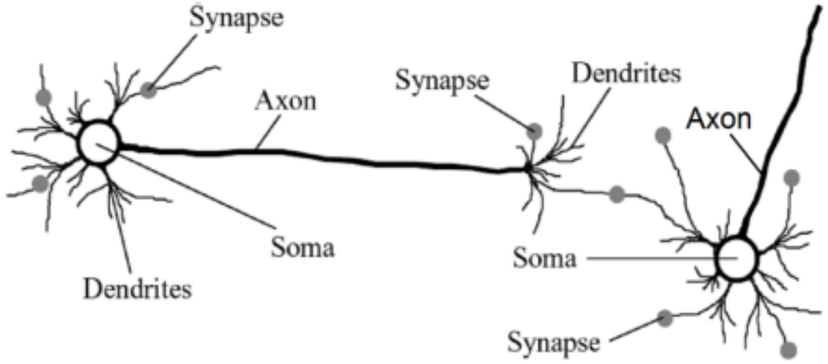
NEKA

Perceptron

(1) Perceptron이란

우리의 뇌는 neuron이라는 신경세포가 촘촘하게 연결되어 있음.

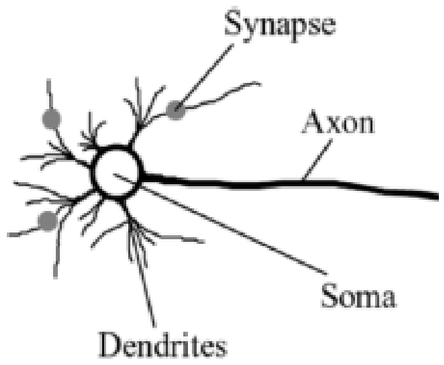
- 뉴런 : 전기 신호를 보내고 받는 신경 세포
- 신경계의 가장 기본이 되는 단위



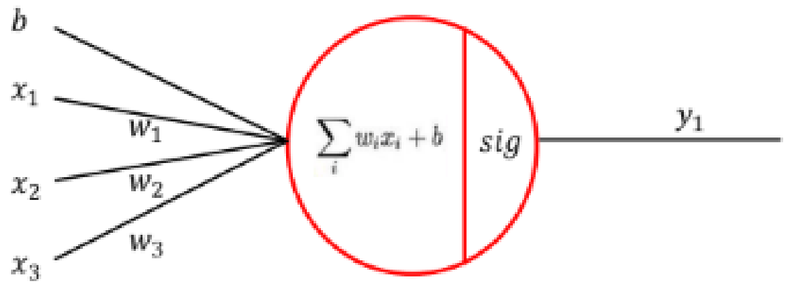
Perceptron

(1) Perceptron이란

Neuron



Perceptron

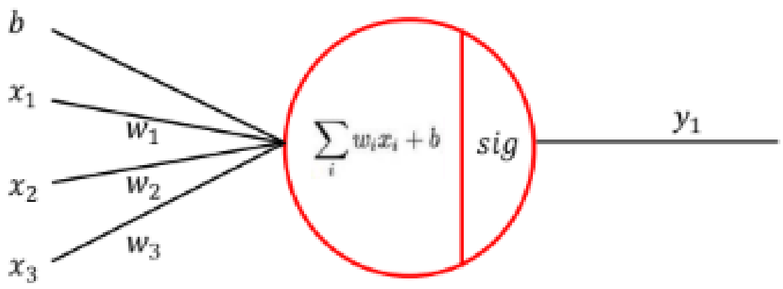


Perceptron

(1) Perceptron이란

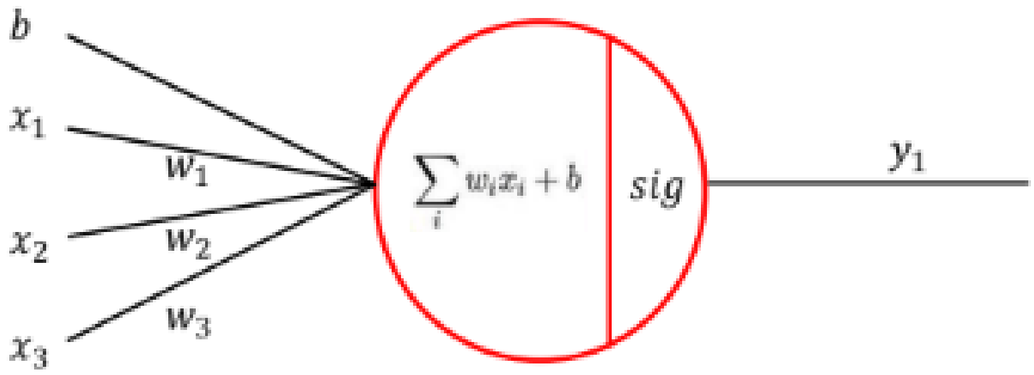
Perceptron은 이항분류를 하기 위한 알고리즘

- 뉴런의 돌기들은 각각 input과 output이 됨.
- 뉴런에서 각 synapse의 강도는 가중치가 됨.



Perceptron

(1) Perceptron이란



(A) Y=WX+B

(B) sigmoid

Perceptron

(1) Perceptron이란

(A) Y=WX+B

y=ax+b ⟶ y=a1x1+a2x2+a3x3+b ⟶ Y=WX+B

X	y
20	30
30	40
19	??

x1	x2	x3	y
20	30	23	56
30	40	23	50
19	25	54	??

{20 30 23} {56}

Perceptron

(1) Perceptron이란

(B) sigmoid

Y=WX+B
(선형 회귀 모델)

——— activation function ⟶

Y = sigmoid(WX+B)
(분류 모델)

Perceptron

(1) Perceptron이란

(B) Activation Function

앞선 입력을 합쳐서 출력할 수 있도록 변환함

종류

1. Sigmoid

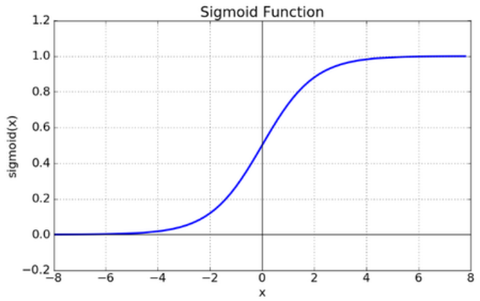
$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

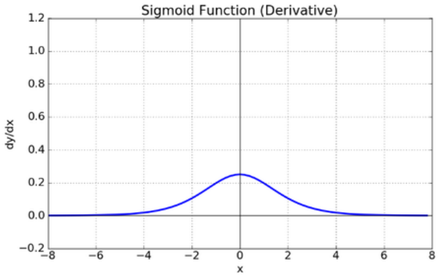
Sigmoid의 특성

- Logistic 이라고도 함.
- S자형 구조임.
- x값에 따라 0~1의 값을 가짐.

Sigmoid의 한계점

- Vanishing Gradient 문제가 있음





Perceptron

(1) Perceptron이란

(B) Activation Function

앞선 입력을 합쳐서 출력할 수 있도록 변환함

tanh의 특성

- '쌍곡선'과 관련 있는 개념임.
- S자형 구조임.
- 함수의 중심점이 0임.

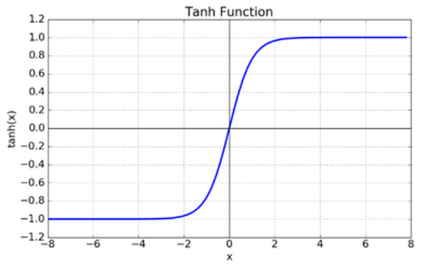
종류

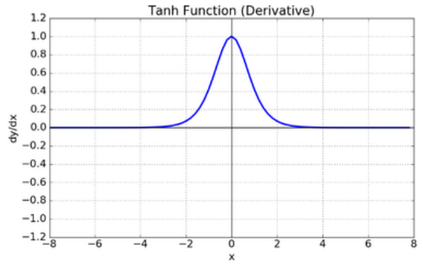
2. Tanh

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

tanh의 한계점

- Vanishing Gradient 문제가 여전히 있음





Perceptron

(1) Perceptron이란

(B) Activation Function

앞선 입력을 합쳐서 출력할 수 있도록 변환함

tanh의 특성

- '쌍곡선'과 관련 있는 개념임.
- S자형 구조임.
- 함수의 중심점이 0임.

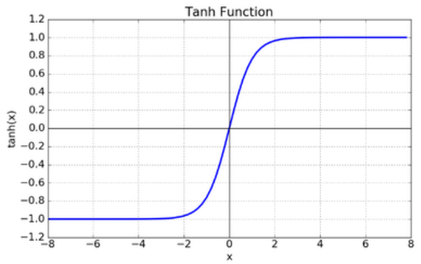
종류

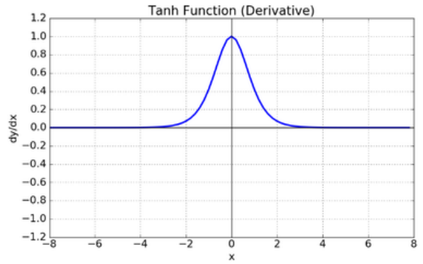
2. Tanh

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

tanh의 한계점

- Vanishing Gradient 문제가 여전히 있음





Perceptron

(1) Perceptron이란

(B) Activation Function

앞선 입력을 합쳐서 출력할 수 있도록 변환함

종류

3. ReLU (Rectified Linear Unit)

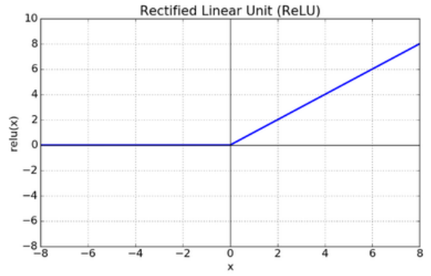
$$f(x) = max(0, x)$$

ReLU의 특성

- Vanishing Gradient를 해결함
- 0보다 작으면 값이 0임.
- 학습이 비교적 빠르고, 연산 비용이 적음

ReLU의 한계점

- Dying ReLU 문제가 있음



Perceptron

(1) Perceptron이란

(B) Activation Function

앞선 입력을 합쳐서 출력할 수 있도록 변환함

종류

4. Leaky ReLU

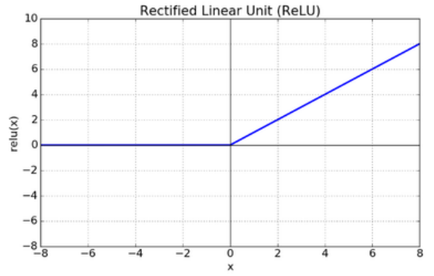
$$f(x) = max(0.01x, x)$$

Leaky ReLU의 특성

- Dying ReLU 문제를 해결함
- 0보다 작으면 값이 0임.
- 학습이 비교적 빠르고, 연산 비용이 적음

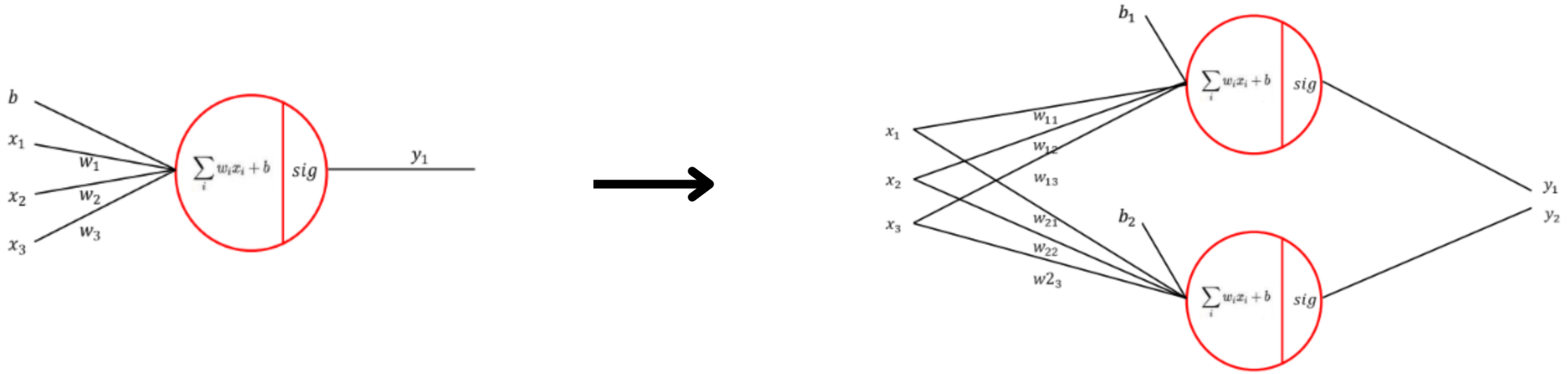
Leaky ReLU의 한계점

- 음수 파트가 중요할 때 쓰기 어려움



MLP

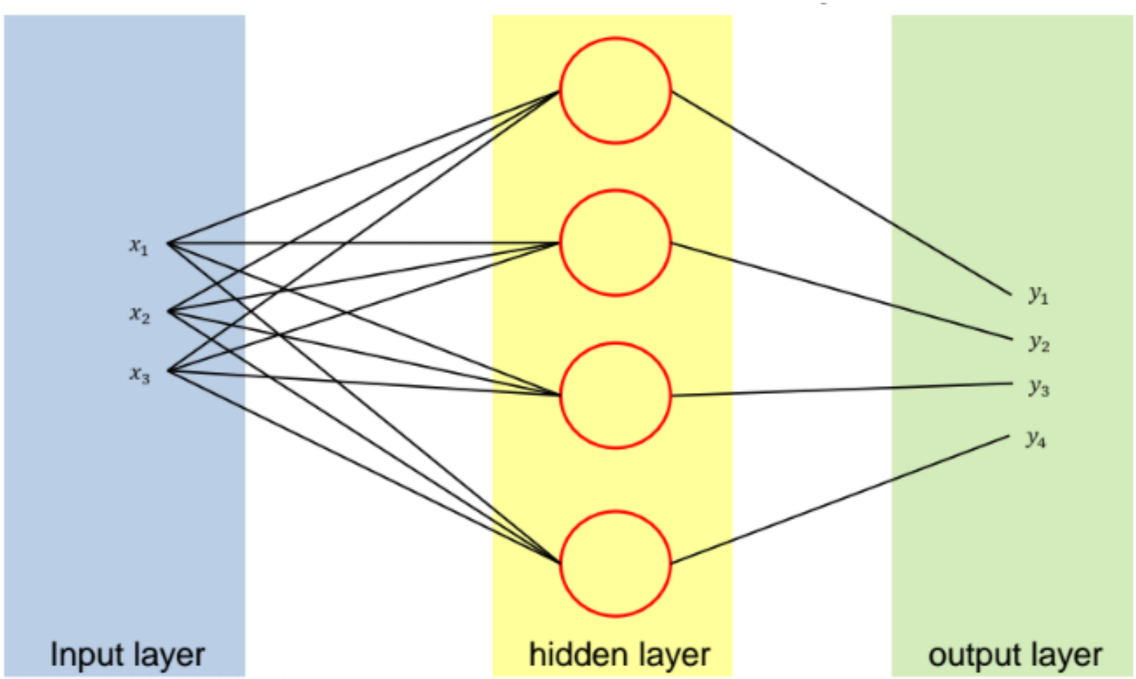
(1) MLP란?



단층 다층

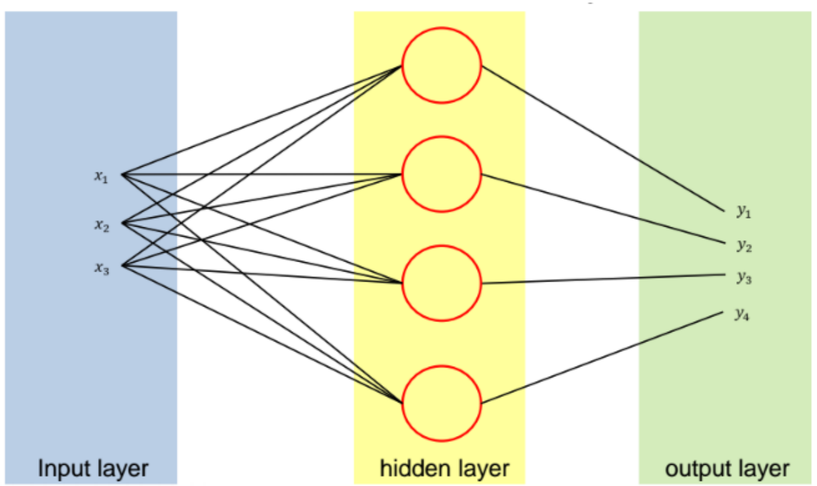
MLP

(1) MLP란?

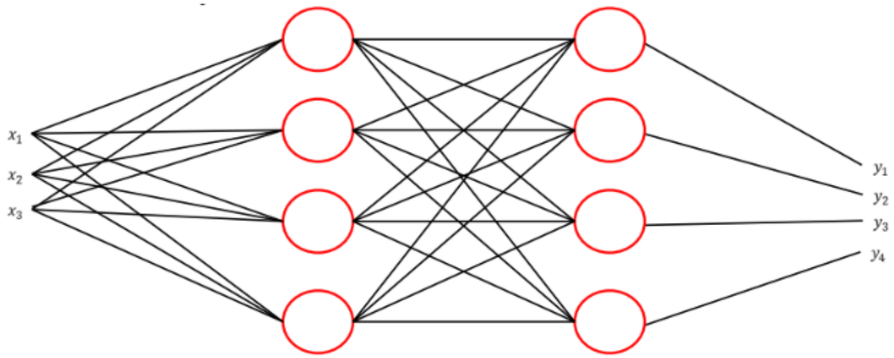


MLP

(1) MLP란?



ANN

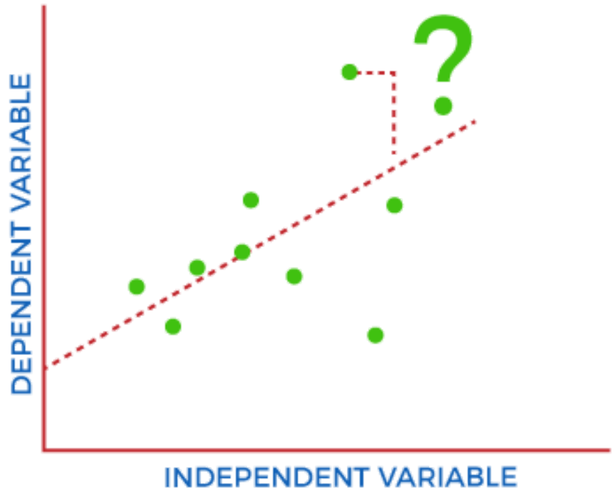


DNN

Cost Function

(1) Cost Function이란?

'실제 답'과 '학습의 결과'의 차이를 수치화한 것



Cost Function

(1) Cost Function이란?

종류

1. MSE : Mean Squared Error $$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

2. RMSE : Root Mean Squared Error $$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

3. RAE : Mean Absolute Error $$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

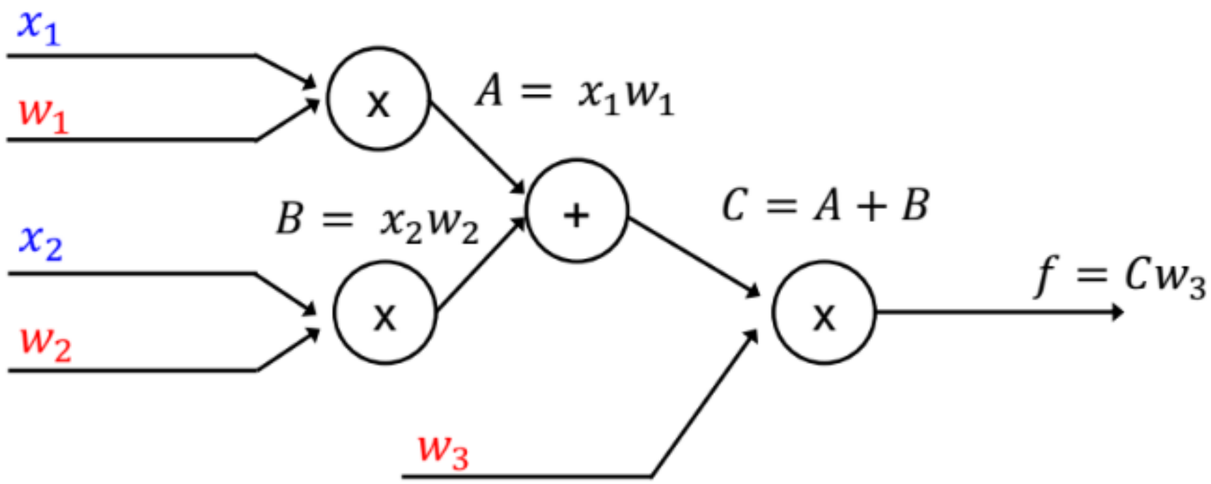
4. R^2 : $$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

4. CrossEntropy : $$H(p,q) = -\int_x p(x)\log q(x)dx$$

FP와 BP

(1) Forward Propagation

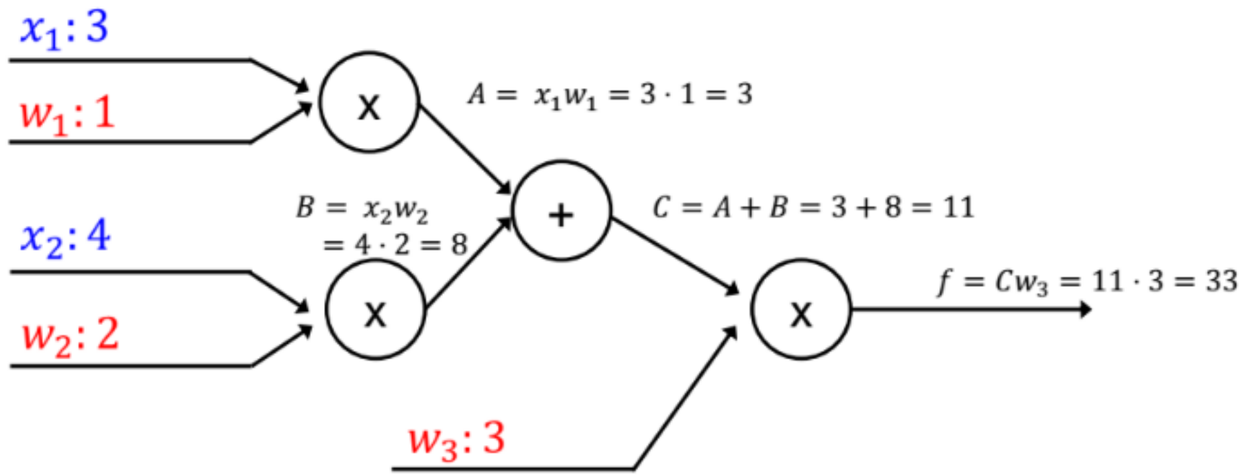
$$f = (x_1 w_1 + x_2 w_2) w_3$$



FP와 BP

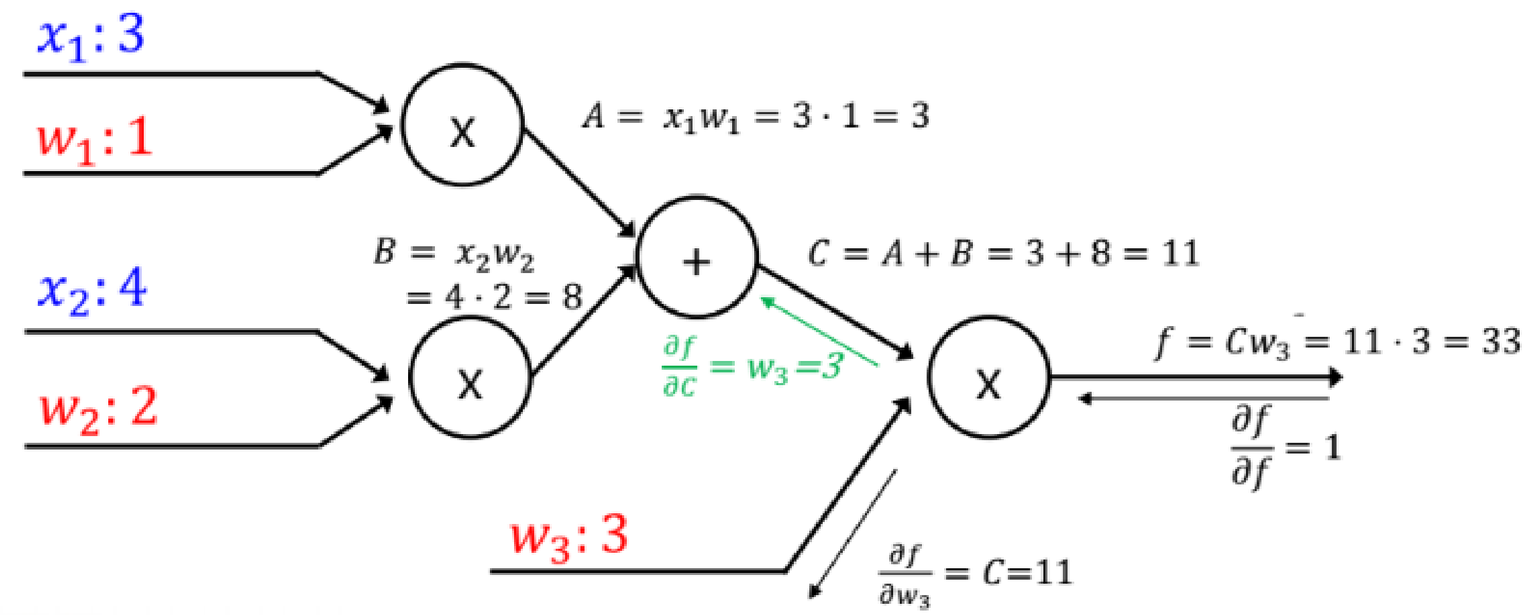
(1) Forward Propagation

$$f = (x_1 w_1 + x_2 w_2) w_3$$



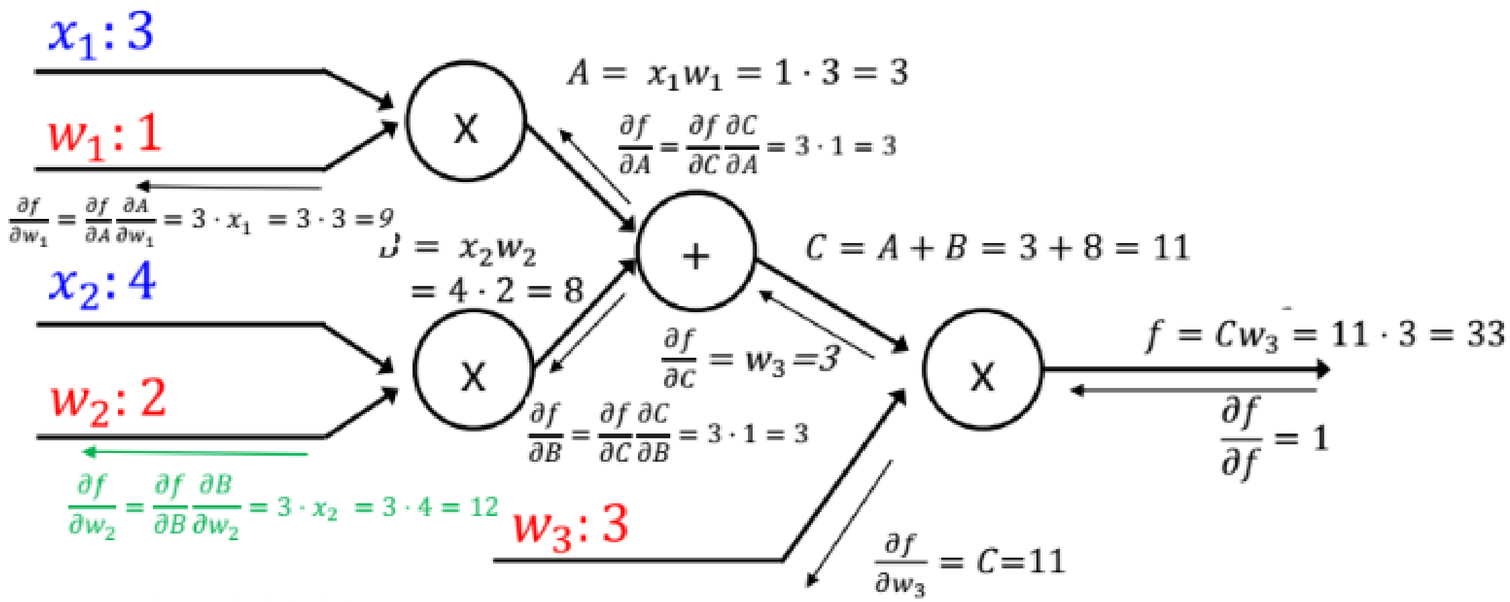
FP와 BP

(2) Back Propagation



FP와 BP

(2) Back Propagation



FP와 BP

(2) Back Propagation

$$\frac{\partial f}{\partial w_1} = \frac{\partial f}{\partial A}\frac{\partial A}{\partial w1} \quad \longrightarrow \quad \frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial f}\frac{\partial f}{\partial A}\frac{\partial A}{\partial w1}$$

Gradient Descent

(1) Gradient Descent란?

비용함수를 최소화하는 값을 찾기

$$w := w - \alpha \frac{\partial c}{\partial w}$$

Optimization

(1) Optimization이란?

비용함수를 최소화하는 값을 찾기

Gradient Descent

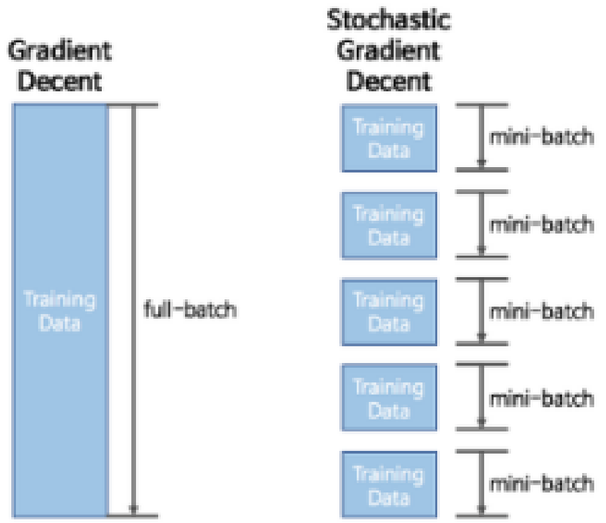
$$w := w - \alpha \frac{\partial C}{\partial w}$$

+ a?

Optimization

(2) Optimization의 종류

(1) SGD



GD vs SGD

- GD는 전부다 학습이 진행된 다음에 업데이트를 함

  vs

- SGD는 학습 중간마다 업데이트를 함

Algorithm 2 Stochastic Gradient Descent at Iteration k

Require: Learning rate ϵ_k

Require: Initial Parameter θ

while stopping criteria not met **do**

  Sample example $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ from training set

  Compute gradient estimate:

  $\hat{\mathbf{g}} \leftarrow +\nabla_\theta L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$

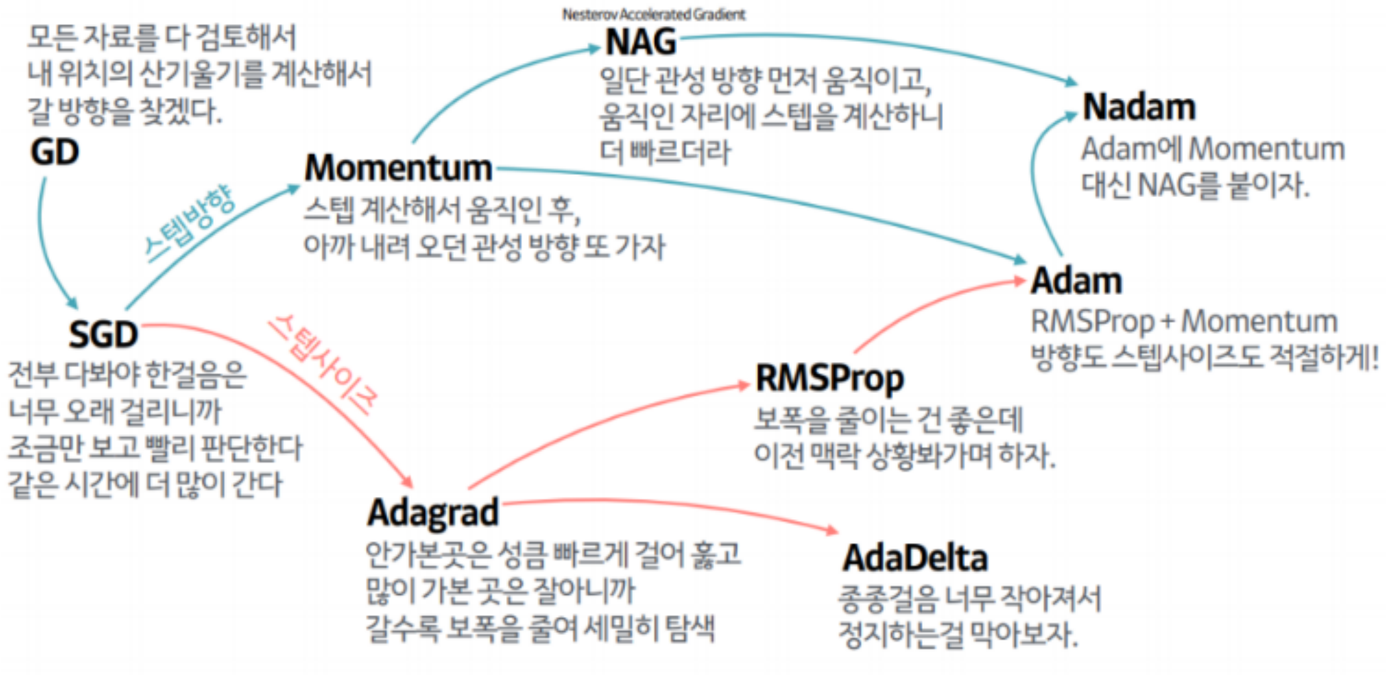
  Apply Update: $\theta \leftarrow \theta - \epsilon\hat{\mathbf{g}}$

end while

ϵ_k is learning rate at step k

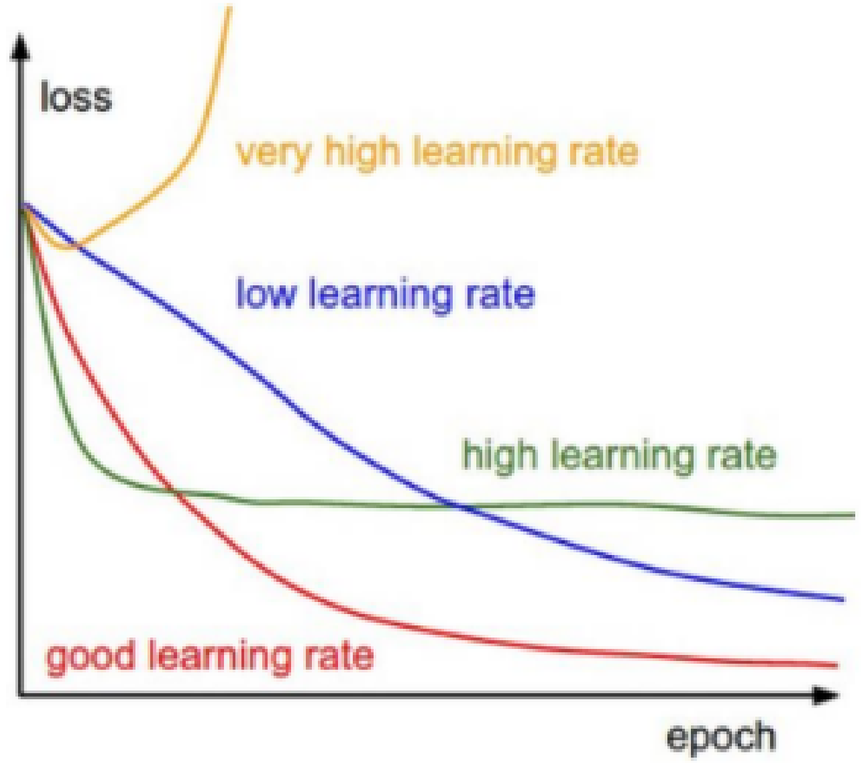
Optimization

(2) Optimization의 종류



Optimization

(2) Optimization의 종류

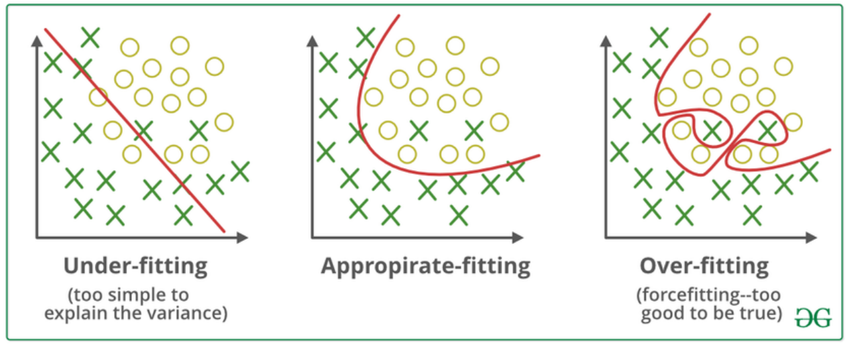


Overfitting

(1) Overfitting과 Underfitting

Overfitting : Training loss는 낮은데 Test loss가 높을 때

Underfitting : Training과 Test 모두 loss가 높을 때

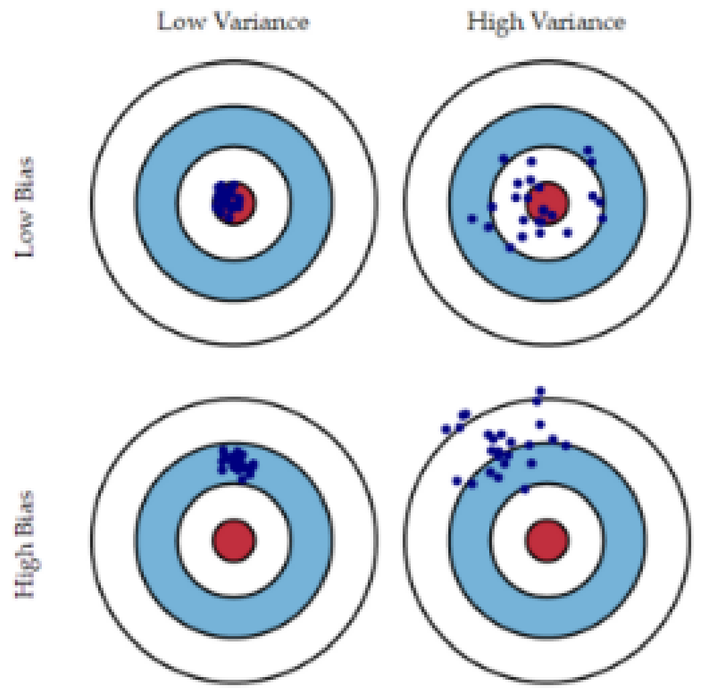


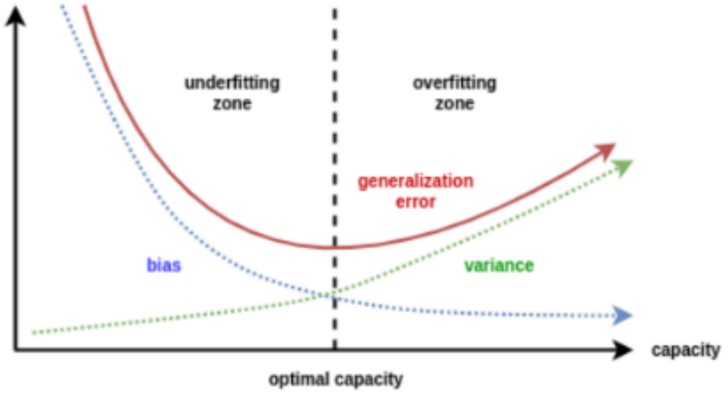
Overfitting

(2) bias와 variance

bias : 예측값과 정답의 차이에 대한 정도

variance : 예측값 사이의 차이에 대한 정도





Overfitting

(3) Overfitting을 해결하는 법

간단한 방법

1. 데이터셋의 숫자를 늘리기
2. 모델의 복잡도를 줄이기

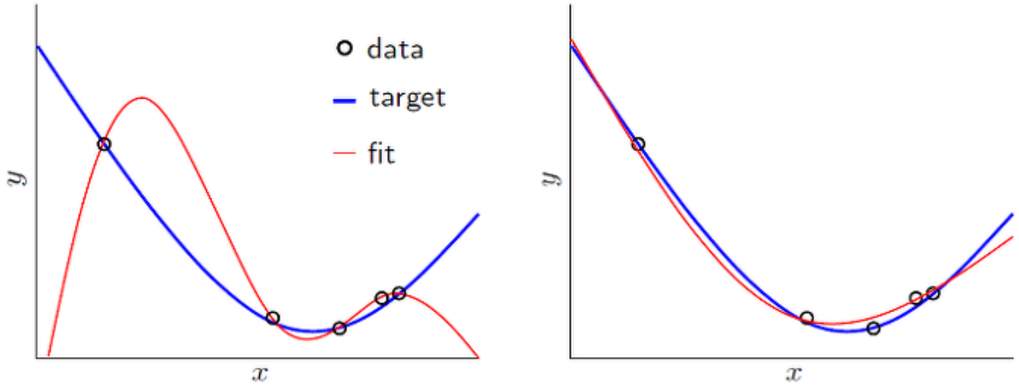
복잡한 방법

1. Regularization
2. Dropout
3. Back-Normalization

Overfitting

(4) Regularization

가중치가 너무 커지는 것을 막음



(a) without regularization (b) with regularization

L1 Regularization (Lasso 정규화)

$$C = C_0 + \frac{\lambda}{n} \sum_w |w|$$

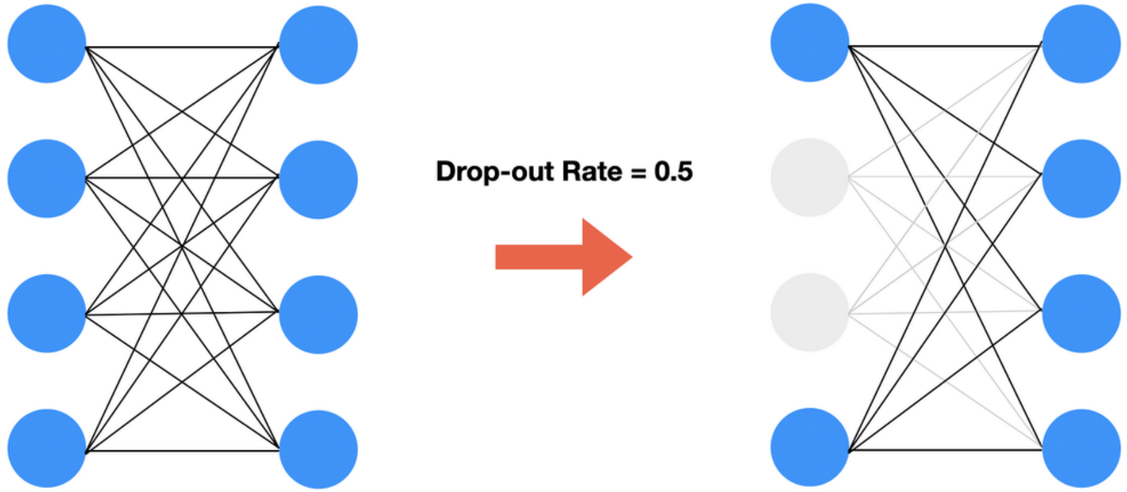
L2 Regularization (Ridge 정규화)

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2$$

Overfitting

(5) Dropout

레이어에서 특정 확률로 뉴런을 제거함

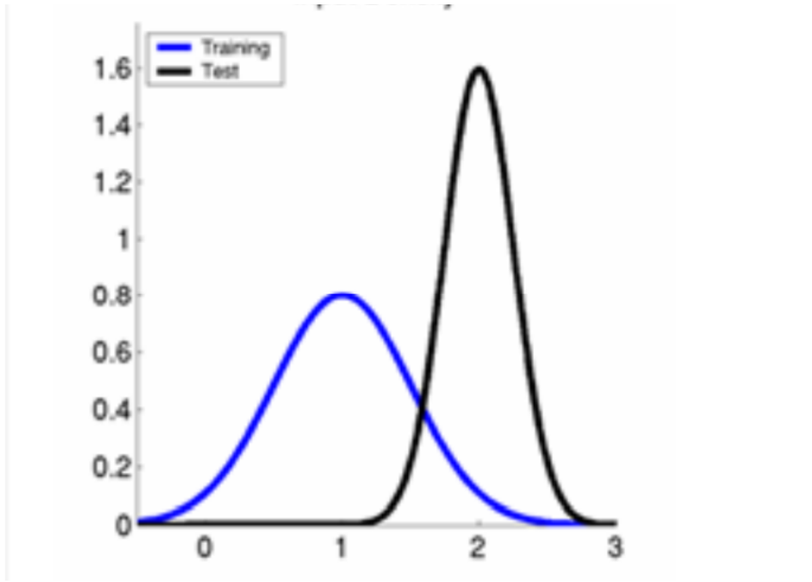


https://heytech.tistory.com/

Overfitting

(6) Batch Normalization

- Internal Covariance Shift : 레이어를 지나갈 때마다 입력값의 분포가 달라짐



Overfitting

(6) Batch Normalization

- 정규화 과정을 통해 평균과 분산을 일정하게 조정함

$$\hat{x}^k = \frac{x^k - E[x^k]}{\sqrt{Var[x^k]}}$$

$$y^k = \gamma^k \hat{x} + \beta^k$$

NEKA

THANK YOU