1주차 스터디

딥러닝 CNN 완벽 가이드 – 세션1, 2, 3

세션1 – 딥러닝 개요와 경사하강법

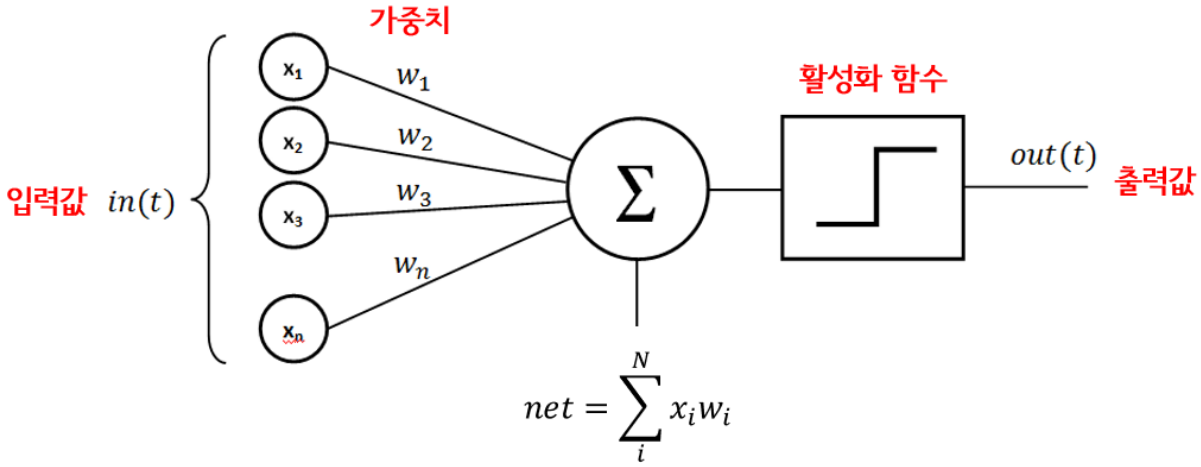
세션2 – 오차 역전파, 활성화 함수, 손실 함수, 옵티마이저

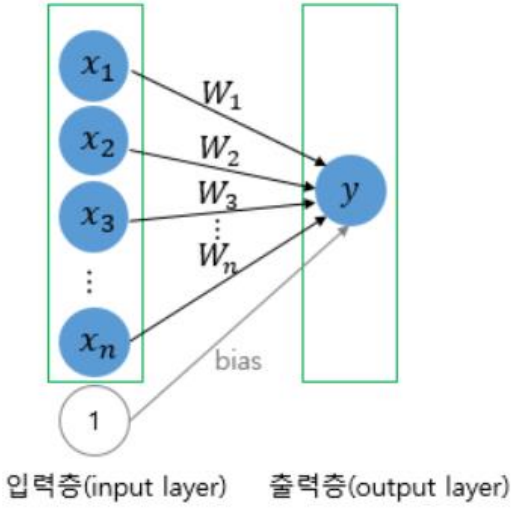
세션3 – Keras Framework

발표자 : 오홍석, 임수진

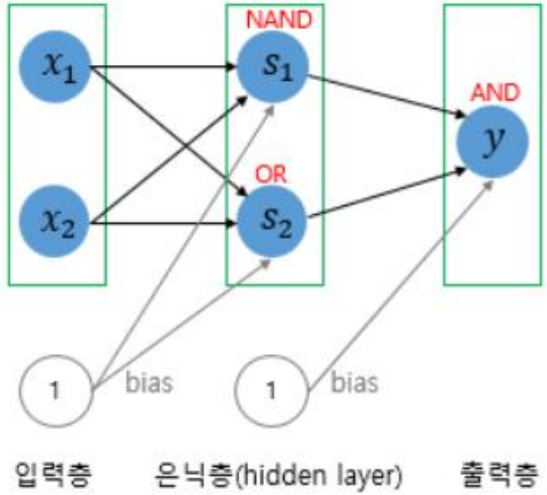
1. 퍼셉트론

단층 퍼셉트론



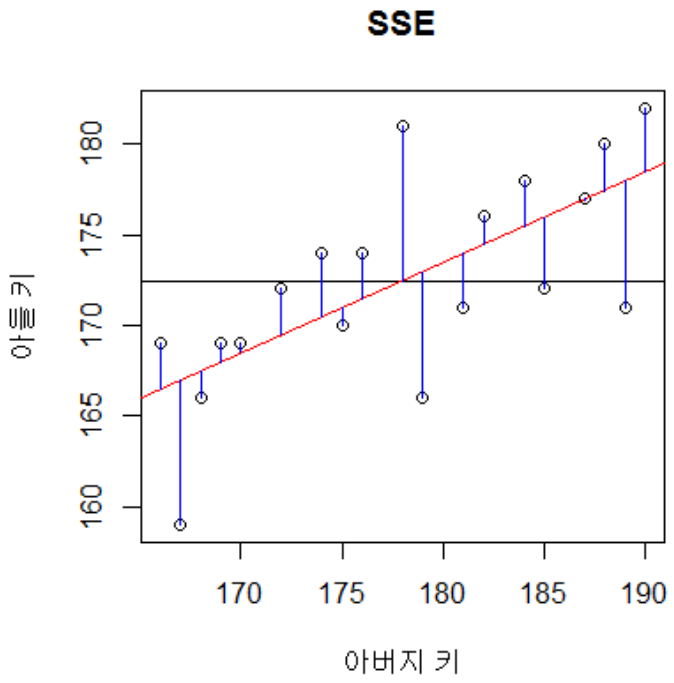


다층 퍼셉트론



** 은닉층들의 층 수가 늘어난다고 해서 반드시 학습 효과가 향상되는 것은 아니며, 1~2개 층을 배치한 것만으로도 최대 성능을 나타내는 경우가 흔하다.

* RSS, MSE의 이해



$$RSS(w_0, w_1) = \sum_{i=1}^{N} (y_i - (w_0 + w_1 * x_i))^2$$

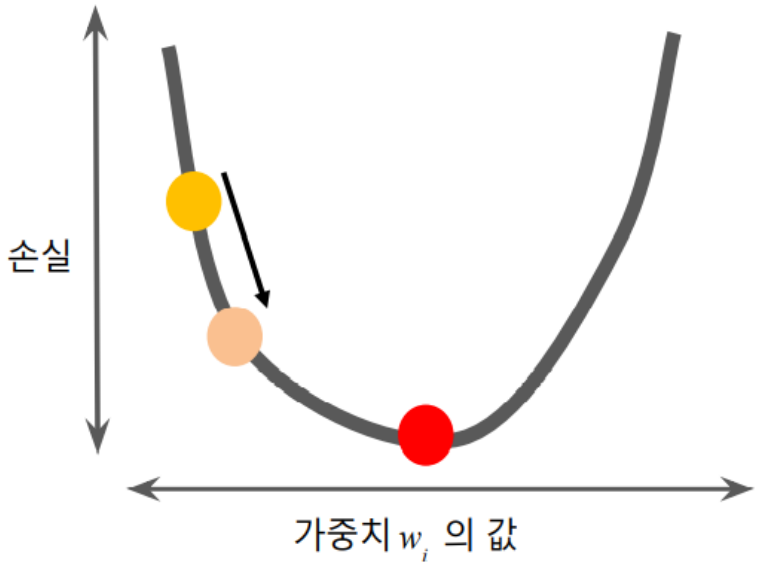
(i는 1부터 학습 데이터의 총 건수 N까지)

$$MSE(w_0, w_1) = \frac{1}{N} \sum_{i=1}^{N} (y_i - (w_0 + w_1 * x_i))^2$$

(i는 1부터 학습 데이터의 총 건수 N까지)

* 단순선형회귀의 비용함수

2. 경사 하강법



- 학습률이 너무 작으면?
최소점에 수렴하는데 너무 오래 걸림

- 학습률이 너무 크면?
최소점을 찾지 못하거나 오히려 발산할 가능성 존재

* 손실함수의 편미분

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

y_i = *actual value*
$\hat{y}_i$ = *predicted value*
n = *# of observations*

$$\frac{dLoss(w)}{dw_1} = \frac{2}{N}\sum_{i=1}^{N} -x_i * (y_i - (w_0 + w_1 x_i) = -\frac{2}{N}\sum_{i=1}^{N} x_i * (\text{실제값}_i - \text{예측값}_i)$$

$$\frac{dLoss(w)}{dw_0} = \frac{2}{N}\sum_{i=1}^{N} - (y_i - (w_0 + w_1 x_i) = -\frac{2}{N}\sum_{i=1}^{N} (\text{실제값}_i - \text{예측값}_i)$$

* 가중치, 편향 업데이트

$$W = W - \alpha\frac{dL}{dw}$$

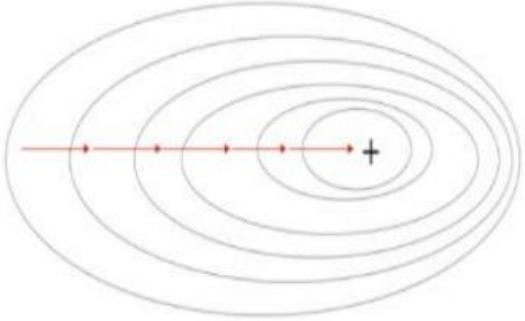
$$w_{1,new} = w_{1,old} - \eta\frac{dLoss(w)}{dw_1} = w_{1,old} + \eta(\frac{2}{N}\sum_{i=1}^{N} x_i * (\text{실제값}_i - \text{예측값}_i))$$

$$w_{0,new} = w_{0,old} - \eta\frac{dLoss(w)}{dw_0} = w_{0,old} + \eta(\frac{2}{N}\sum_{i=1}^{N} \text{실제값}_i - \text{예측값}_i)$$

2. 경사 하강법

* 경사하강법의 유형

1) 배치 경사 하강법

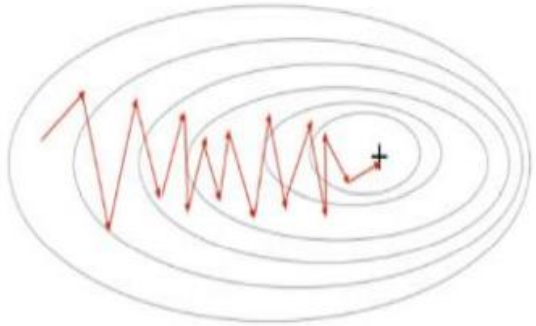


경사를 1회 계산하기 위해 전체 학습데이터 사용

-> 1개의 배치에 전체 학습데이터가 모두 들어감

- 일관된 방향을 향해 지속적으로 근접함.
- 데이터가 많으면 수행시간이 오래 걸림.

2) 확률적 경사 하강법

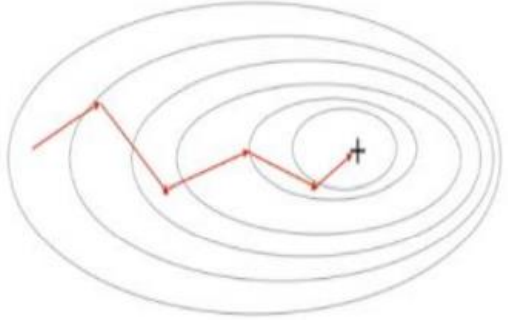


경사를 1회 계산하기 위해 1개의 학습데이터 사용

-> 1개의 배치에 임의의 학습데이터 1개만 들어감

- 데이터가 많아도 수행시간이 빠름.
- 무작위 선택에 의해서 노이즈가 많을 수 있고 데이터의 편중이 있을 수 있음.

3) 미니배치 경사 하강법



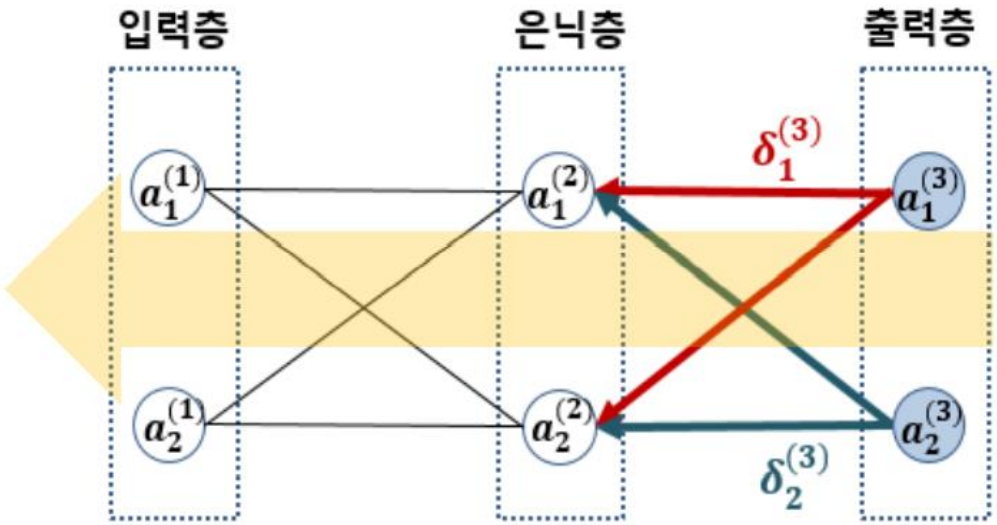
경사를 1회 계산하기 위해 일부 학습데이터 사용

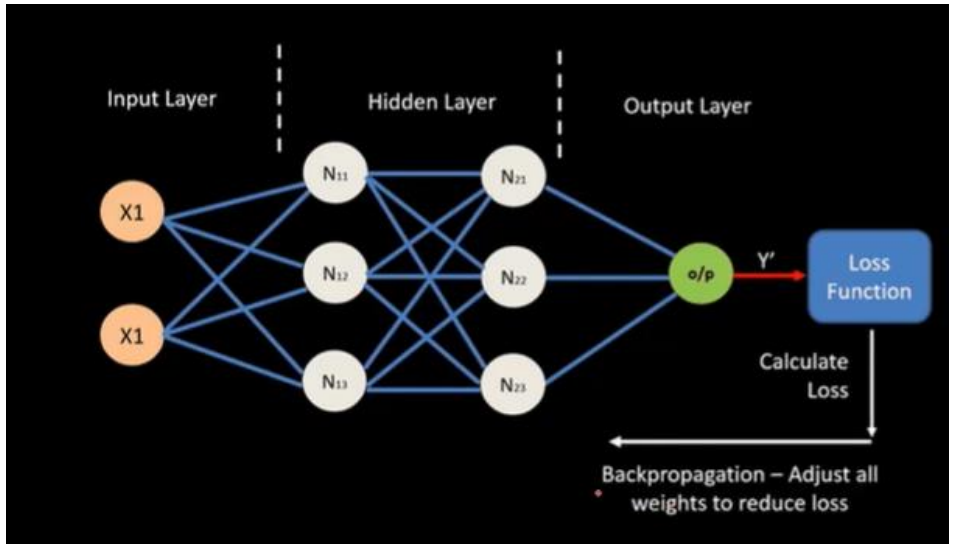
-> 1개의 배치에 임의의 학습데이터 여러개가 들어감

- 앞의 두가지 방법의 절충안.
- 배치 경사 하강법보다는 효율적이고, 확률적 경사 하강법보다는 노이즈가 적음.

3. 오차 역전파

* 역전파 알고리즘





출력층으로부터 역순으로 gradient를 전달해서

전체 층의 가중치를 업데이트 하는 방식

4. Chain Rule

* 미분의 연쇄법칙 – 합성함수의 미분

$$z = f(g(x))$$

$$\frac{dz}{dx} = f'(g(x)) * g'(x)$$

* 미분의 연쇄법칙 – 의존 변수들의 순차적인 변화율

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dx}$$

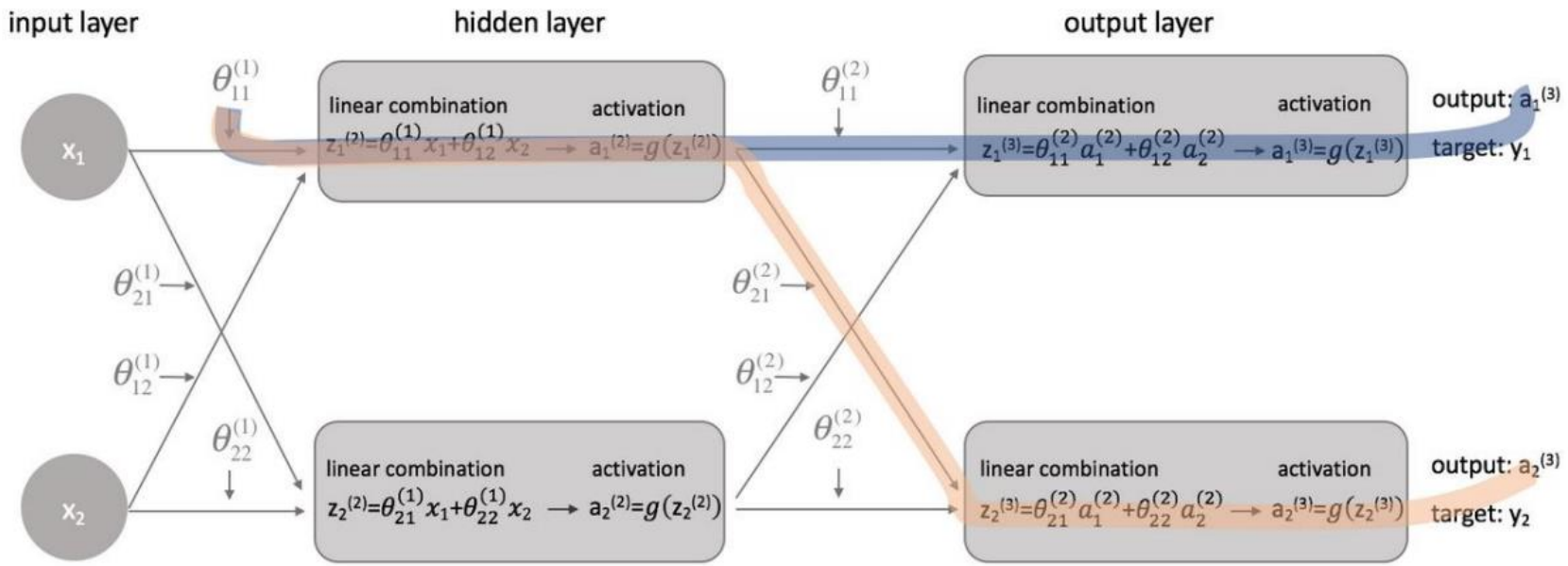
$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dz} \cdot \frac{dz}{dx}$$

- 변수가 여러 개일 때 어떤 변수에 대한 다른 변수의 변화율을 알아내기 위해 사용됨
- 변수 y가 변수 u에 의존하고, 다시 변수 u가 변수 x에 의존한다고 하면 x에 대한 y의 변화율은 u에 대 한 y의 변화율과 x에 대한 u을 변화율을 곱하여 계산할 수 있음

**** 심층 신경망은 합성 함수의 연쇄 결합**

4. Chain Rule

* 신경망의 역전파 계산



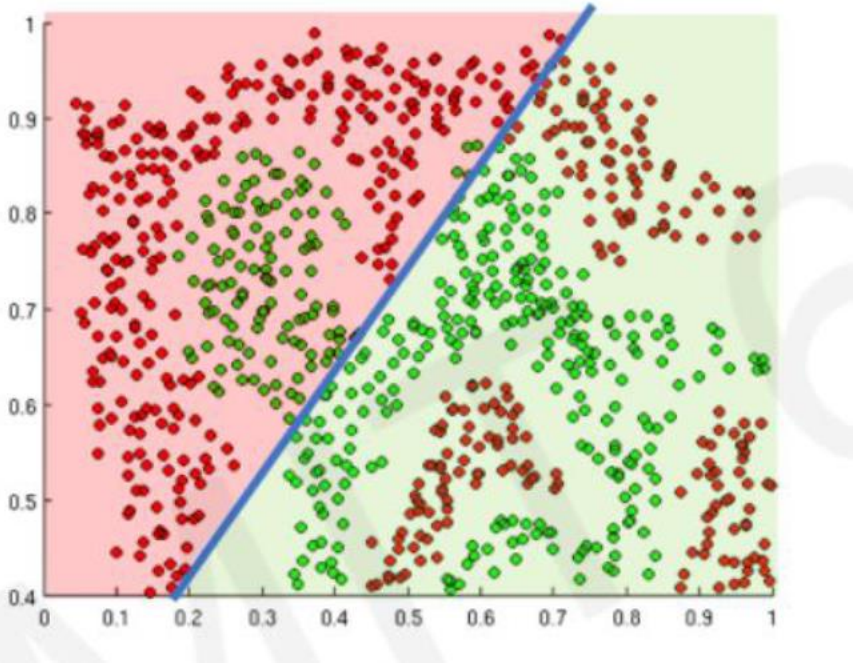
Local Gradient

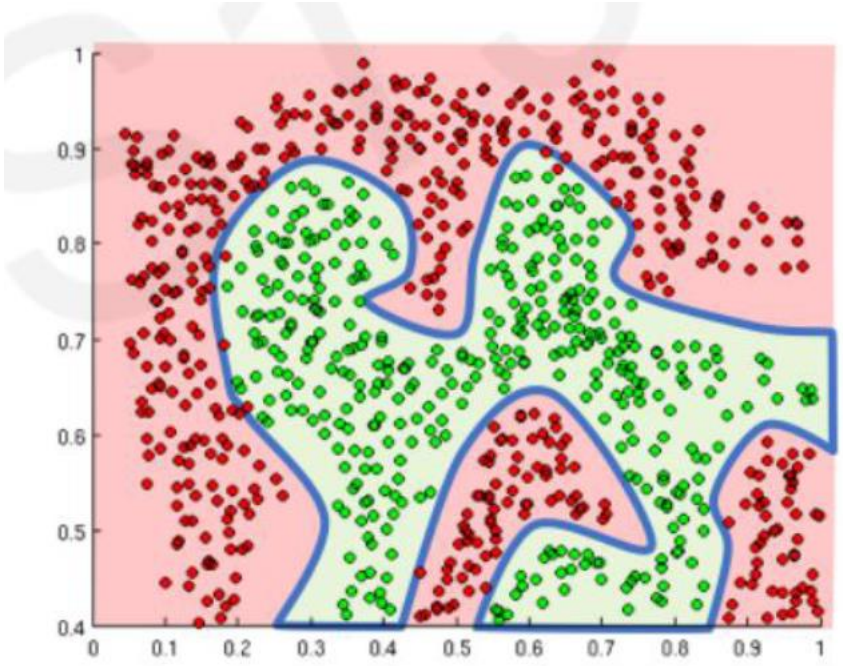
$$\frac{\partial J(\theta)}{\partial \theta_{11}^{(1)}} = \left(\frac{\partial J(\theta)}{\partial a_1^{(3)}}\right)\left(\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}}\right)\left(\frac{\partial z_1^{(3)}}{\partial a_1^{(2)}}\right)\left(\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}}\right) + \left(\frac{\partial J(\theta)}{\partial a_2^{(3)}}\right)\left(\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}}\right)\left(\frac{\partial z_2^{(3)}}{\partial a_1^{(2)}}\right)\left(\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}}\right)\left(\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}}\right)$$

Upstream Gradient

5. 활성화 함수

* 딥러닝 네트워크에 비선형성을 적용하기 위함





5. 활성화 함수 (Sigmoid & softmax)

- Sigmoid :

마지막 classification(binary)

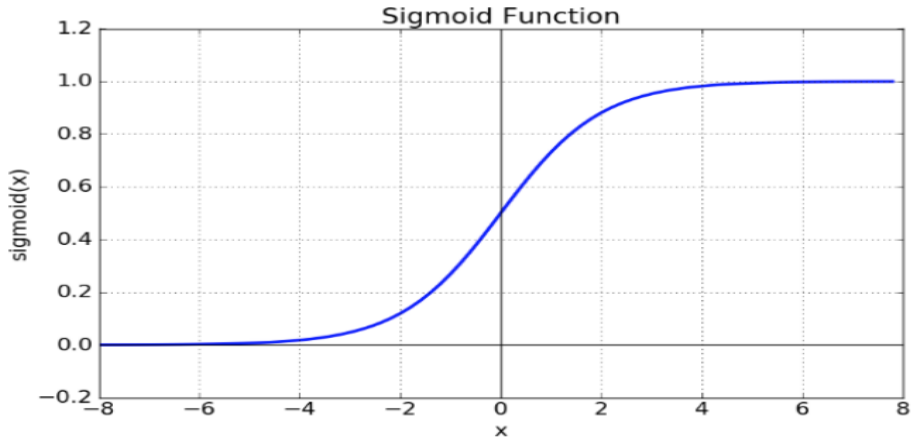
- Softmax :

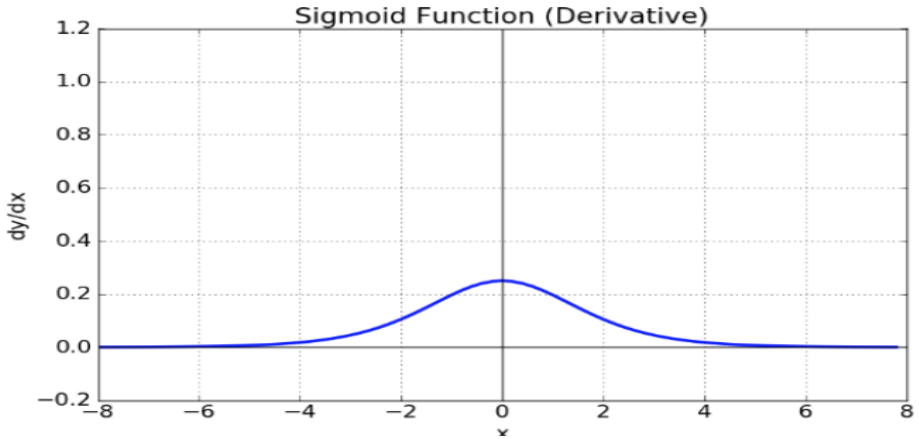
마지막 classification(multi)

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

- Vanishing Gradient



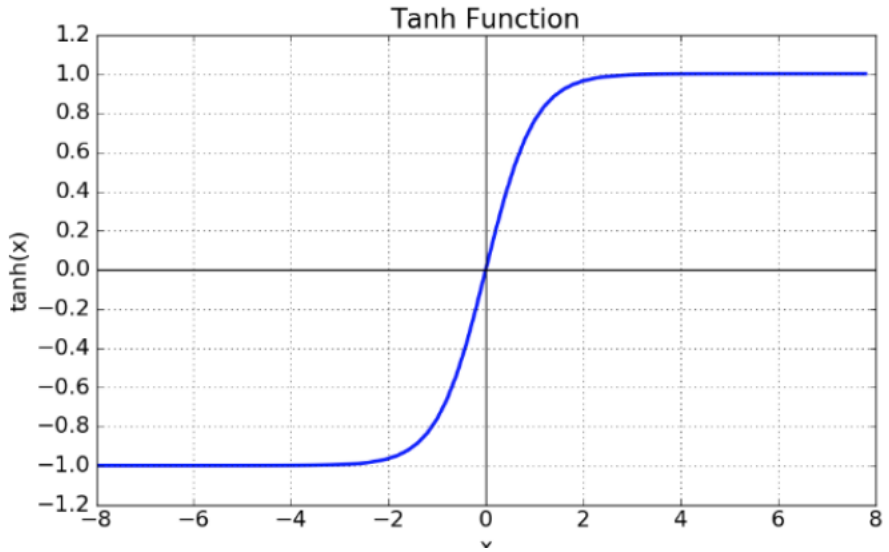


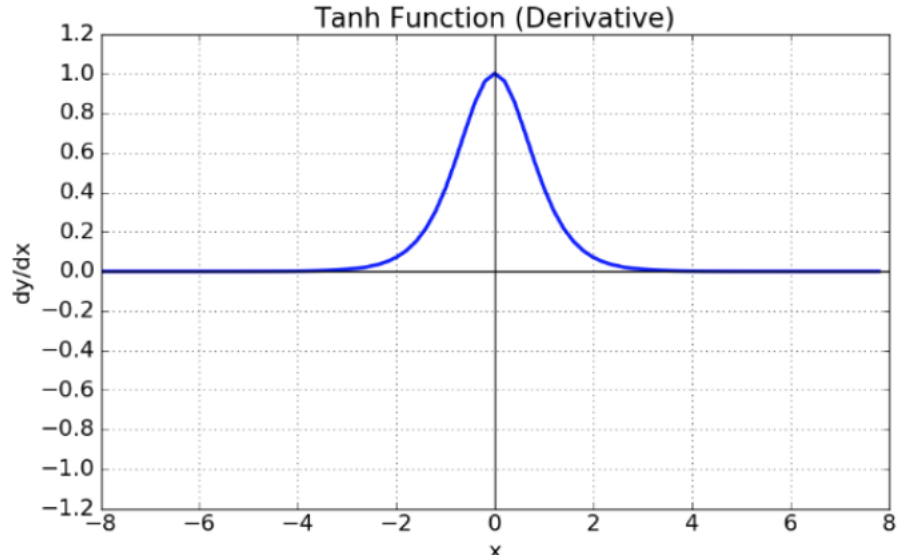
5. 활성화 함수 (Tanh)

* Vanishing Gradient 여전히 존재

$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$



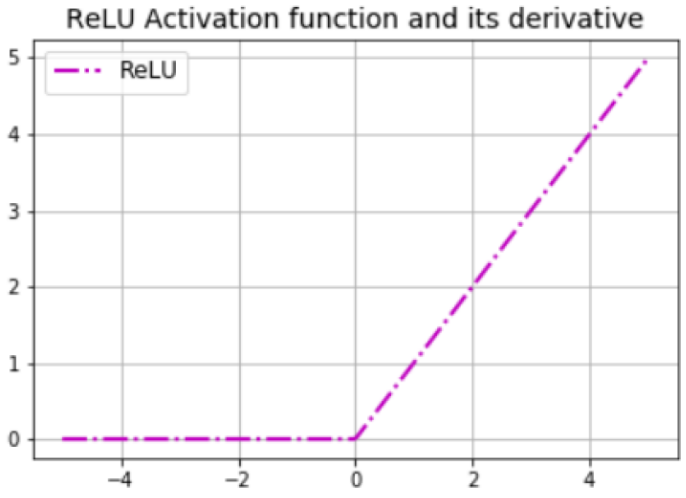


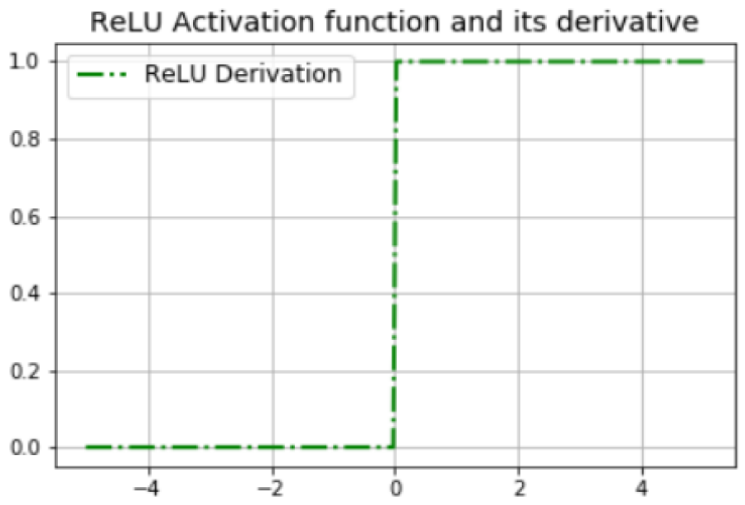
5. 활성화 함수 (Relu)

- 은닉층에 사용됨
- Zero centered

* Dying ReLU

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

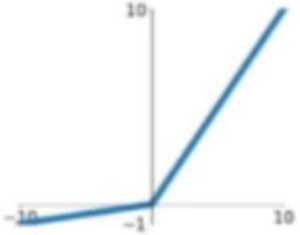




5. 활성화 함수 (Leaky ReLU / EELU / Maxout)

Leaky ReLU

$$\max(0.1x, x)$$

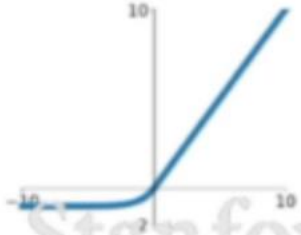


Maxout

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

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



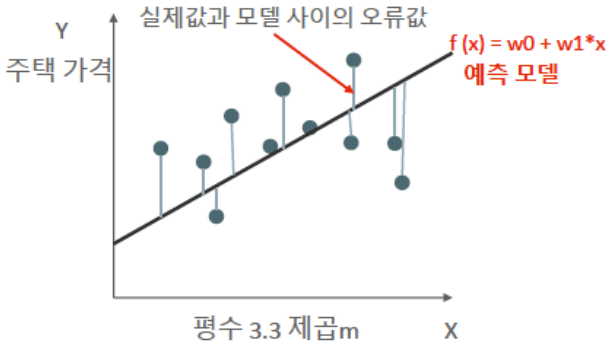
6. 손실함수

* 학습과정이 올바르게 이뤄질 수 있도록 적절한 가이드 제공

* 회귀 :

$$\mathbf{MSE}(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + w_1 * x_i))^2$$

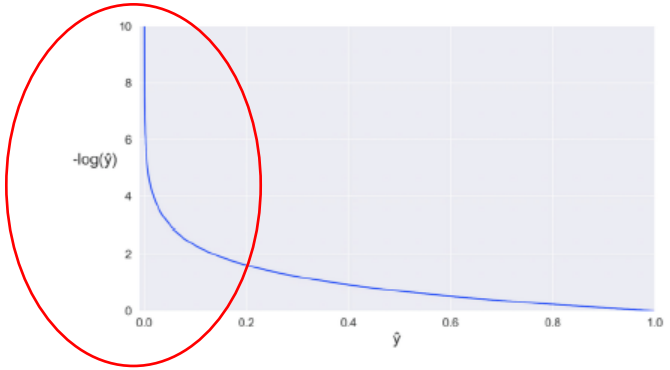
(i는 1부터 학습 데이터의 총 건수 N까지)



* 분류 :

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$



6. 손실함수

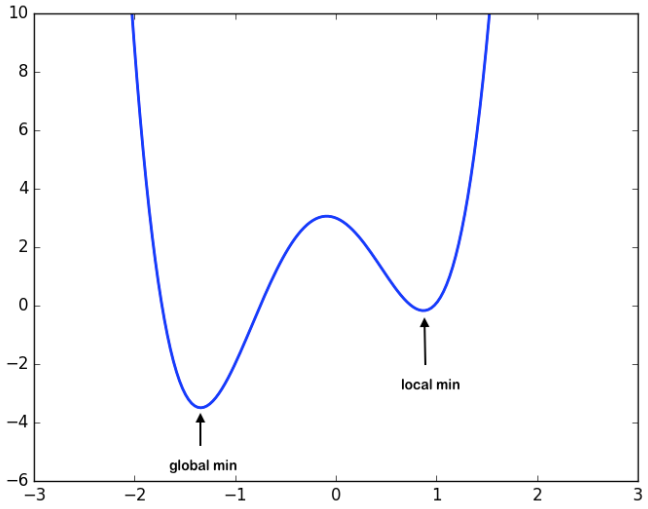
	예측 값	실제 값	CE	Squared Error
Case 1	0.1 0.9 0.0	0 1 0	$-\sum_{i=1}^{5} i$번째 실제값 $* log\,(i$ 번째예측값$)$ = - log(0.9) = **0.105**	0.1^2 = **0.01**
Case 2	0.1 0.7 0.2	0 1 0	= - log(0.7) = 0.356	$0.3^2 = 0.09$
Case 3	0.4 0.5 0.1	0 1 0	= - log(0.5) = 0.693	$0.5^2 = 0.25$
Case 4	0.990 0.01 ~~X~~ 0.01 0.0	0 1 0	= - log(0.01) = **6.90**	$0.99^2 =$ **0.980**

7. 옵티마이저

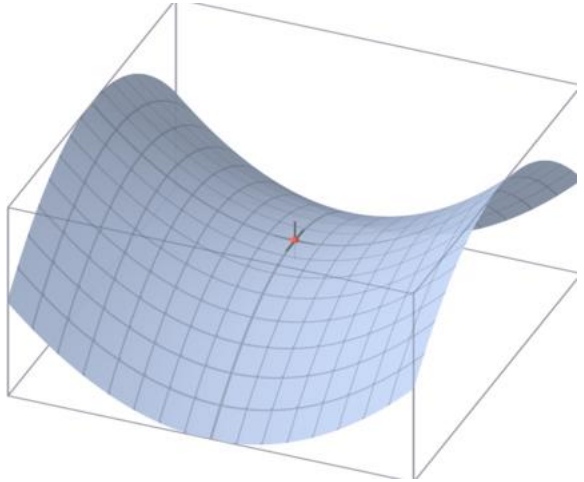
* 보다 최적으로 GD를 적용

* 최소 Loss로 보다 빠르고 안정적으로 수렴할 수 있는 기법 적용 필요

$$w_{t+1} = w_t - \eta \frac{dLoss}{dw_t}$$



Local minimum



Saddle point

7. 옵티마이저

* momentum

$$w_{t+1} = w_t - \eta \frac{dLoss}{dw_t}$$

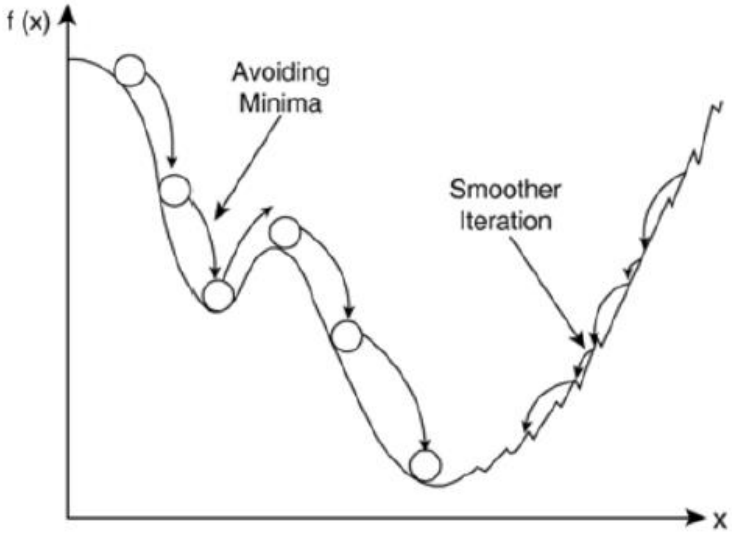
Momentum 적용 식

momentum 계수

momentum을 반영한 새로운 Gradient

$$v_t = \gamma v_{t-1} + \eta \frac{dLoss}{dw_t}$$

$$w_{t+1} = w_t - v_t$$



7. 옵티마이저

* Adagrad

$$w_{t+1} = w_t - \eta \frac{dLoss}{dw_t}$$

$$s_t = s_{t-1} + g_t^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{s_t + \varepsilon}} * g_t$$

7. 옵티마이저

* RMSProp

$$s_t = s_{t-1} + g_t^2$$

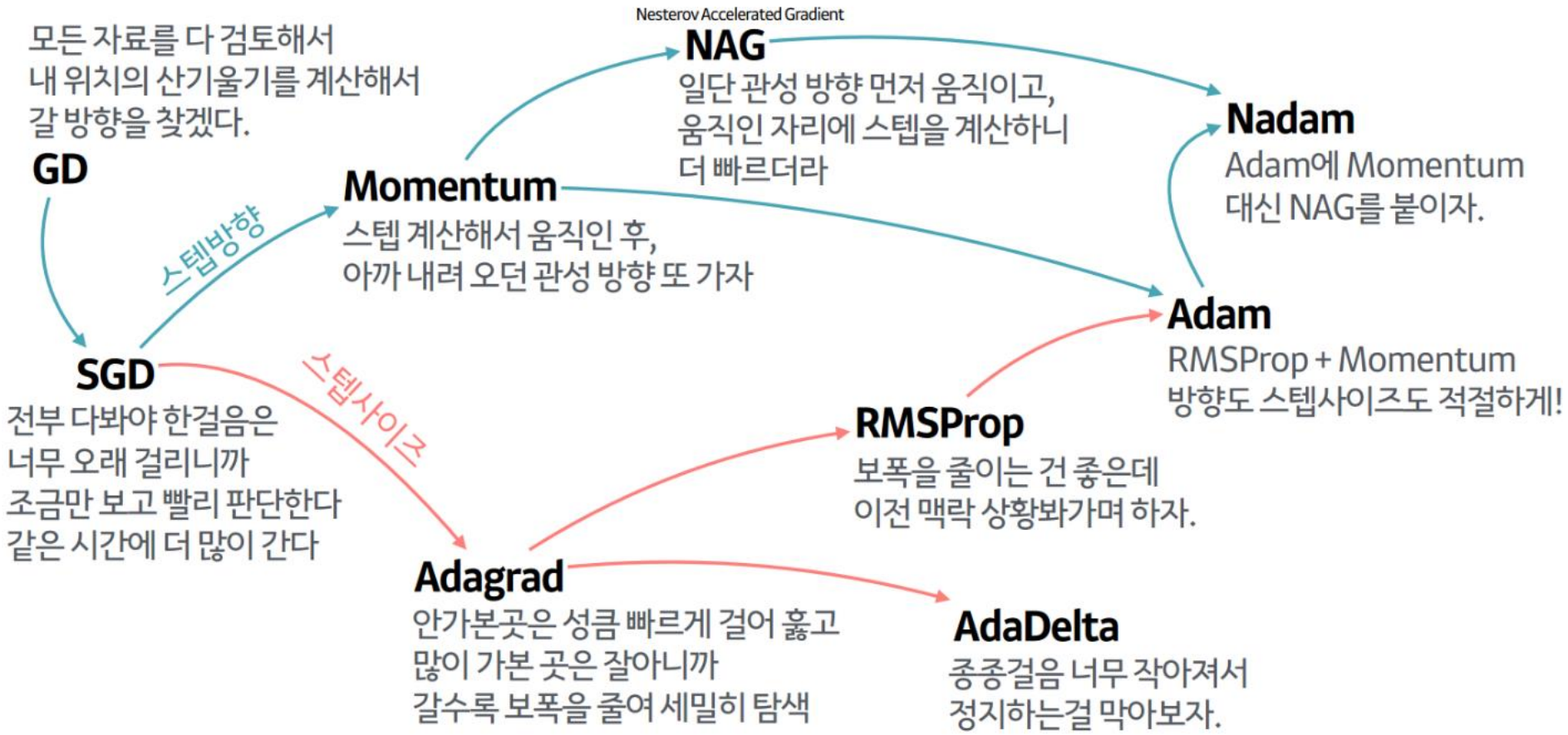
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{s_t + \varepsilon}} * g_t$$

지수 가중 평균 계수

$$s_t = \gamma s_{t-1} + (1-\gamma)\, g_t^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{s_t + \varepsilon}} * g_t$$

7. 옵티마이저


모든 자료를 다 검토해서
내 위치의 산기울기를 계산해서
갈 방향을 찾겠다.
GD
스텝방향
SGD
전부 다봐야 한걸음은
너무 오래 걸리니까
조금만 보고 빨리 판단한다
같은 시간에 더 많이 간다
Momentum
스텝 계산해서 움직인 후,
아까 내려 오던 관성 방향 또 가자
Nesterov Accelerated Gradient
NAG
일단 관성 방향 먼저 움직이고,
움직인 자리에 스텝을 계산하니
더 빠르더라
Nadam
Adam에 Momentum
대신 NAG를 붙이자.
Adam
RMSProp + Momentum
방향도 스텝사이즈도 적절하게!
스텝사이즈
RMSProp
보폭을 줄이는 건 좋은데
이전 맥락 상황봐가며 하자.
Adagrad
안가본곳은 성큼 빠르게 걸어 훑고
많이 가본 곳은 잘아니까
갈수록 보폭을 줄여 세밀히 탐색
AdaDelta
종종걸음 너무 작아져서
정지하는걸 막아보자.

감사합니다