모두의 딥러닝 (Deep learning)

Nambeom Kim (nbumkim@gmail.com)

Neural network to Deep learning (Perceptron)

Deep learning

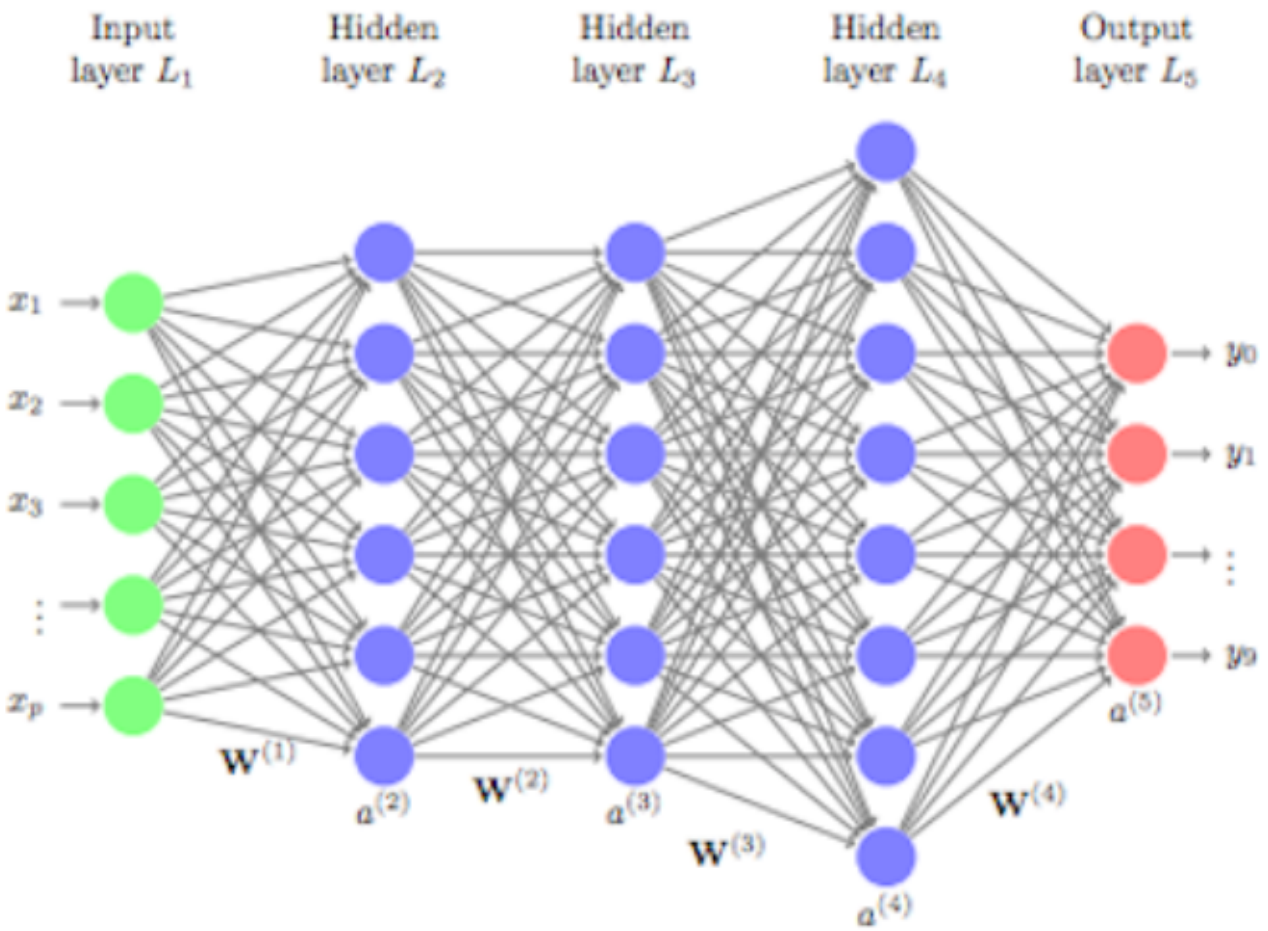
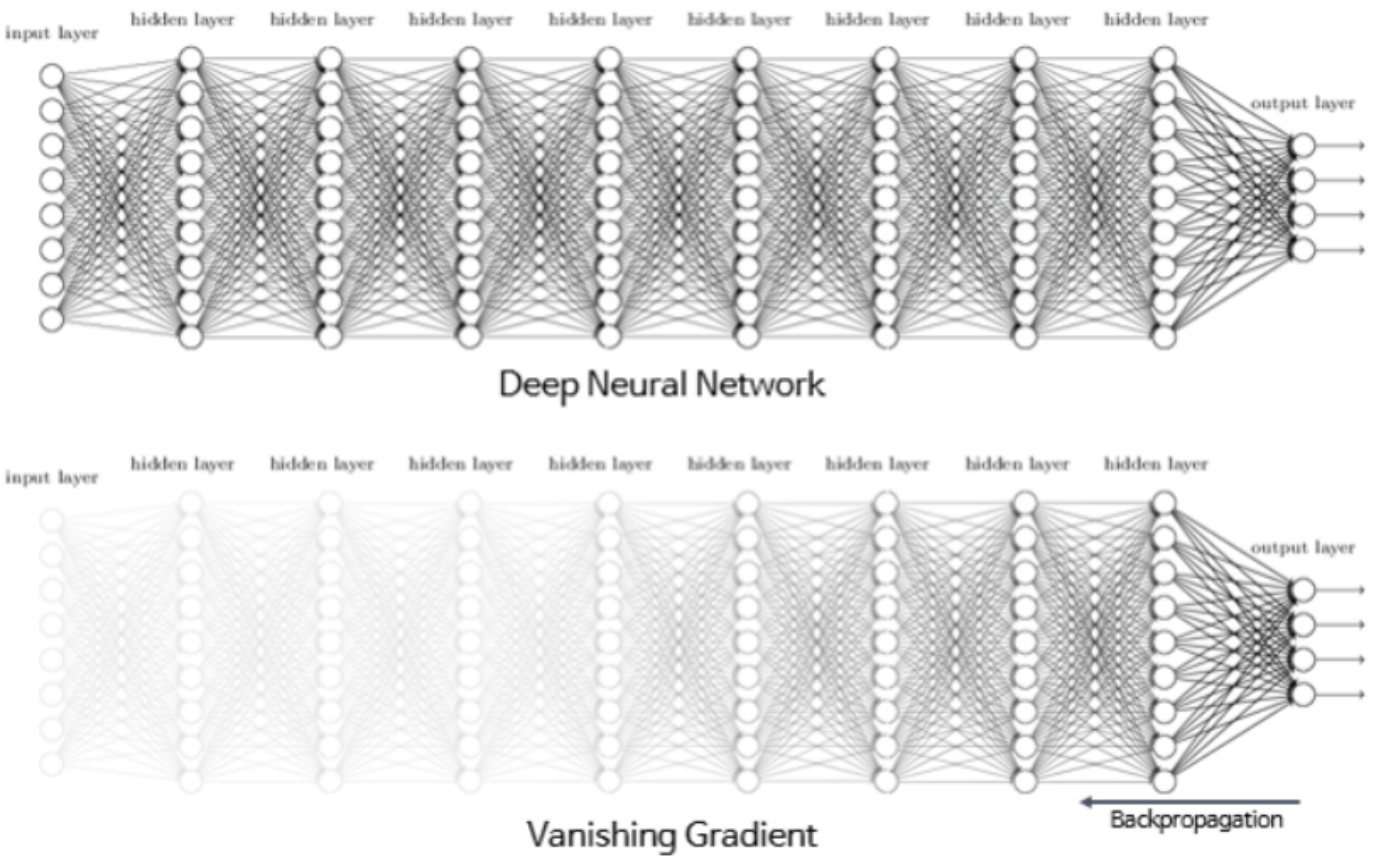


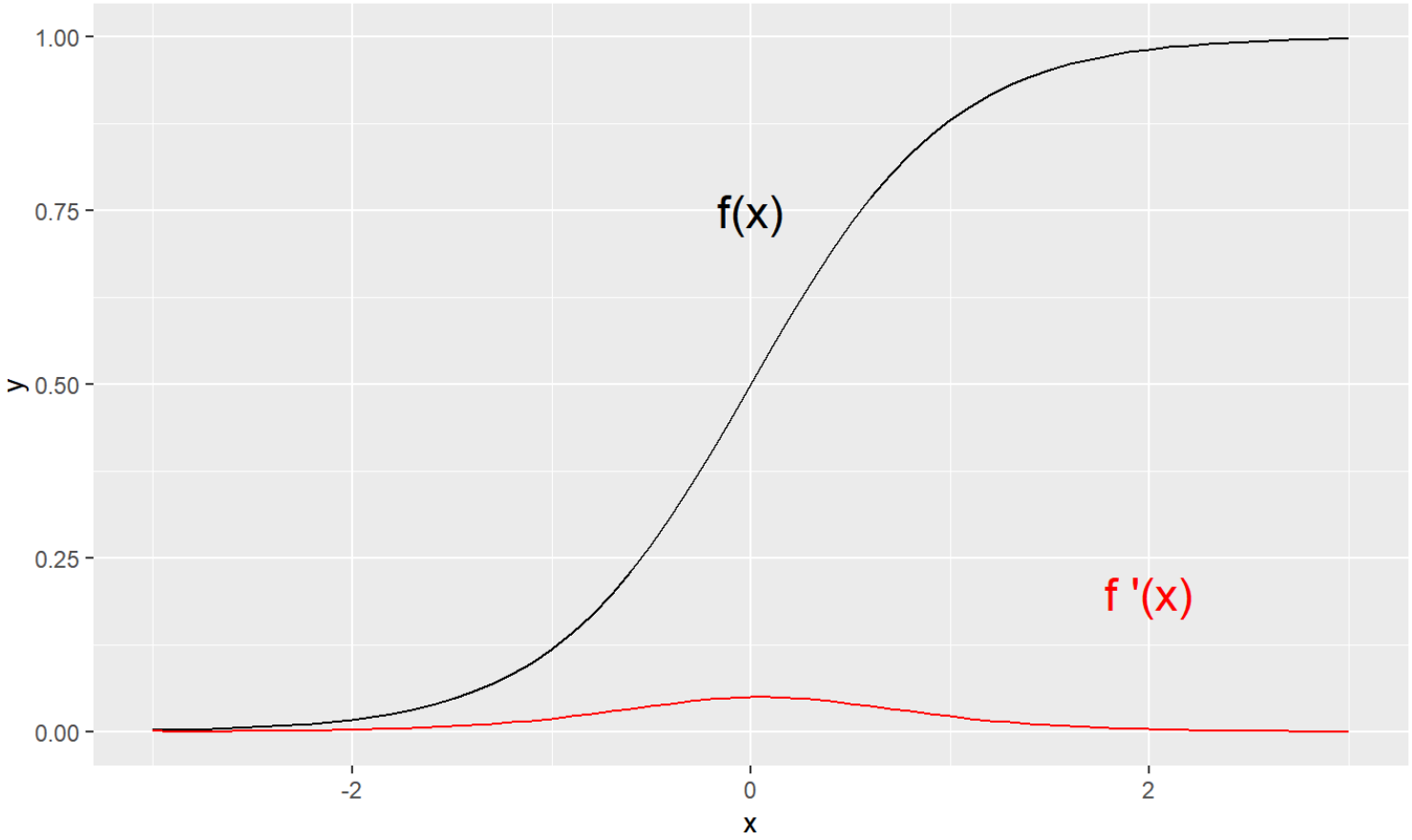
Figure 13.3: Representation of a deep feedforward neural network.

Vanishing gradient



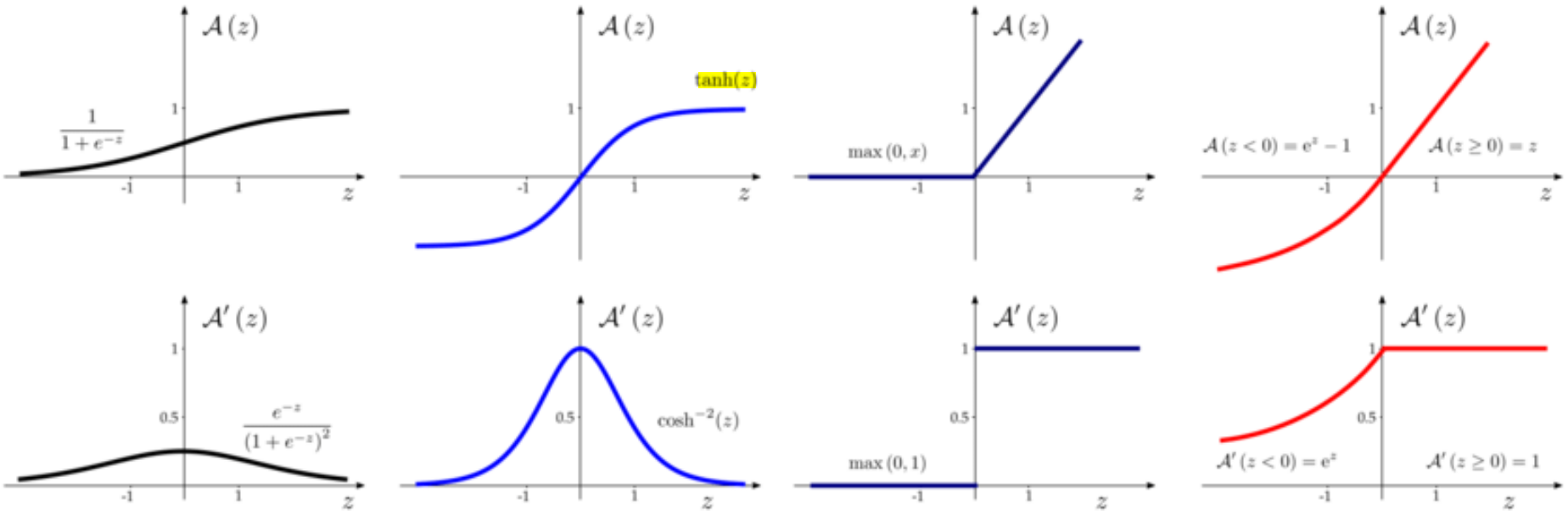
https://trendy00develope.tistory.com/37

Differentiation of sigmoid



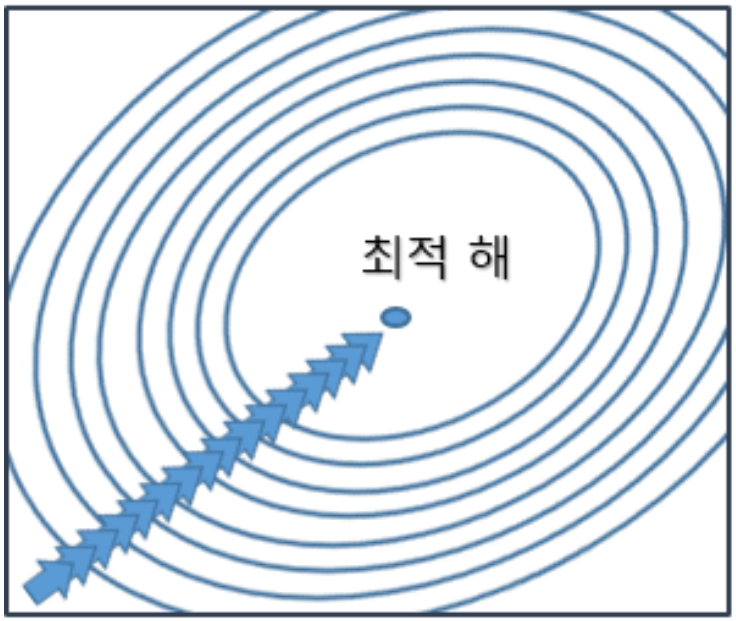
안드레이 카패시 미디엄 블로그

Differentiation of activation functions

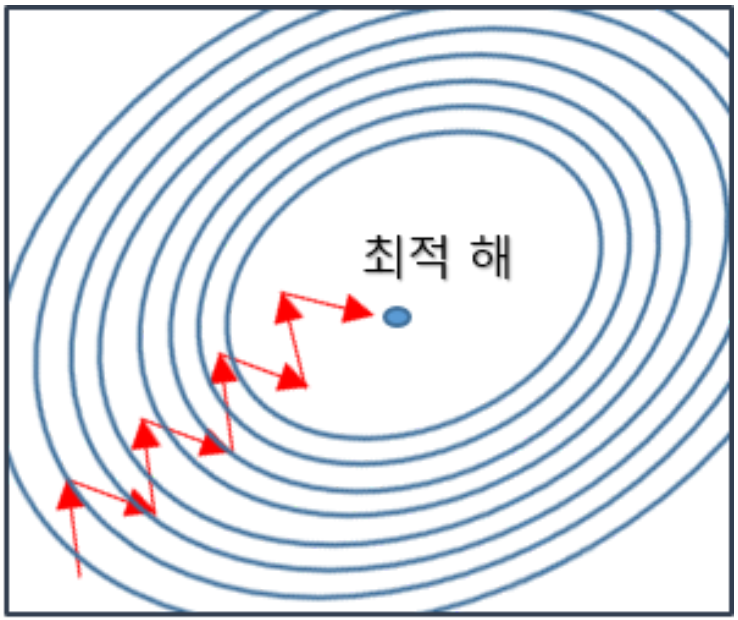


경사하강 **vs** 확률적 경사하강

$$\theta_j^{(n+1)} = \theta_j^{(n)} - \gamma \frac{\partial}{\partial \theta_j} J(\theta^{(n)})$$



경사 하강법

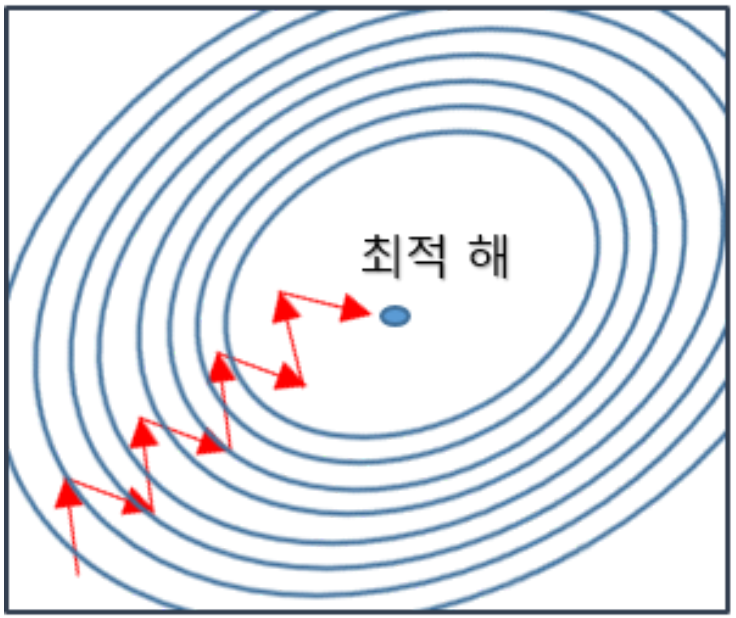


확률적 경사 하강법

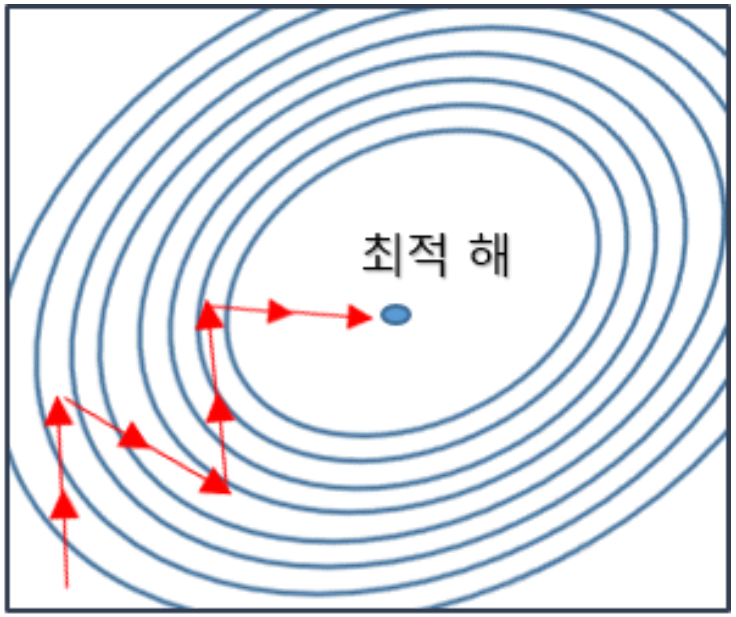
https://twinw.tistory.com/247

모멘텀 (Momentum)

$$\theta_j^{(n+1)} = \theta_j^{(n)} + v_{n-1} - \gamma \frac{\partial}{\partial \theta_j} J(\theta^{(n)})$$
$$v_n = \sum_{i=0}^{n} \rho^{n-i} [-\gamma \frac{\partial}{\partial \theta_j} J(\theta^{(i)})]$$



확률적 경사 하강법



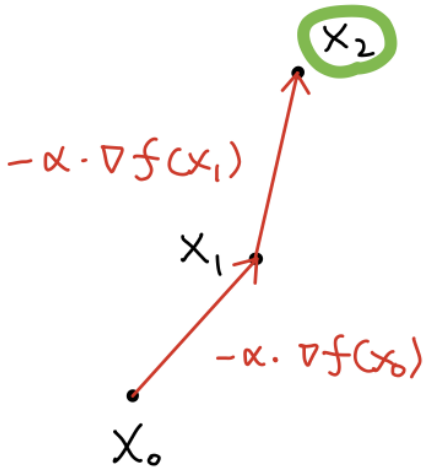
모멘텀

https://twinw.tistory.com/247

이전 방향을 고려하여 비슷한 방향으로 진행

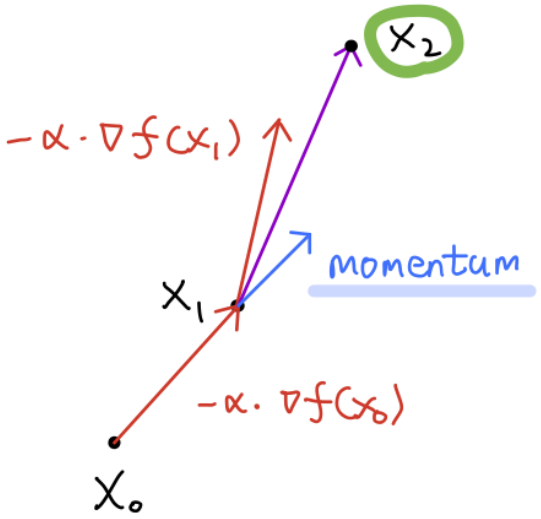
이전 방향과 완전히 다른 방향으로 움직이는 것 = 확률적 경사 하강법

Gradient Descent



$$X_1 = X_0 - \alpha \cdot f(x_0)$$
$$X_2 = X_1 - \alpha \cdot f(x_1)$$

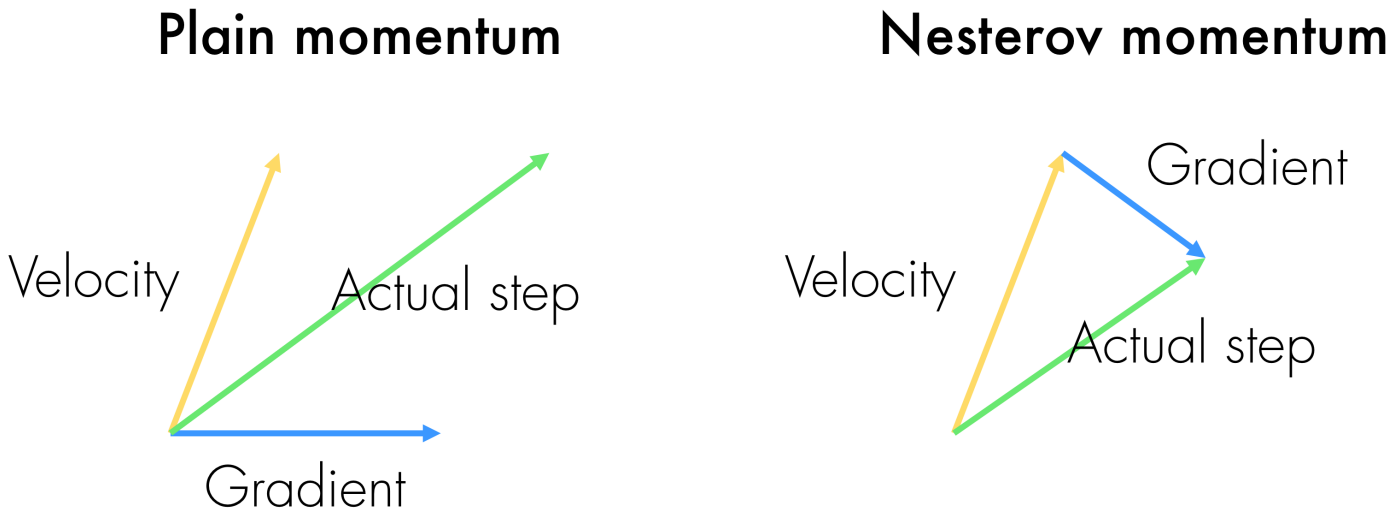
Momentum



$$X_1 = X_0 - \alpha \cdot f(x_0)$$
$$X_2 = X_1 - \alpha \cdot f(x_1) + \text{momentum}$$

https://icim.nims.re.kr/post/easyMath/428

Nesterov Momentum



https://hyunw.kim/blog/2017/11/01/Optimization.html

Adagrad and RMSProp

Update

$$\theta_j^{(n+1)} = \theta_j^{(n)} - \frac{\gamma}{\sqrt{h}} \frac{\partial}{\partial \theta_j} J(\theta^{(n)})$$

너무 빨리 움직이는 것은 제외, 잘 움직이지 않는 것을 움직이기 위해 사용하는 h

Adagrad

$$h = \sum_{i=0}^{n-1} (\frac{\partial}{\partial \theta_j} J(\theta^{(i)}))^2$$

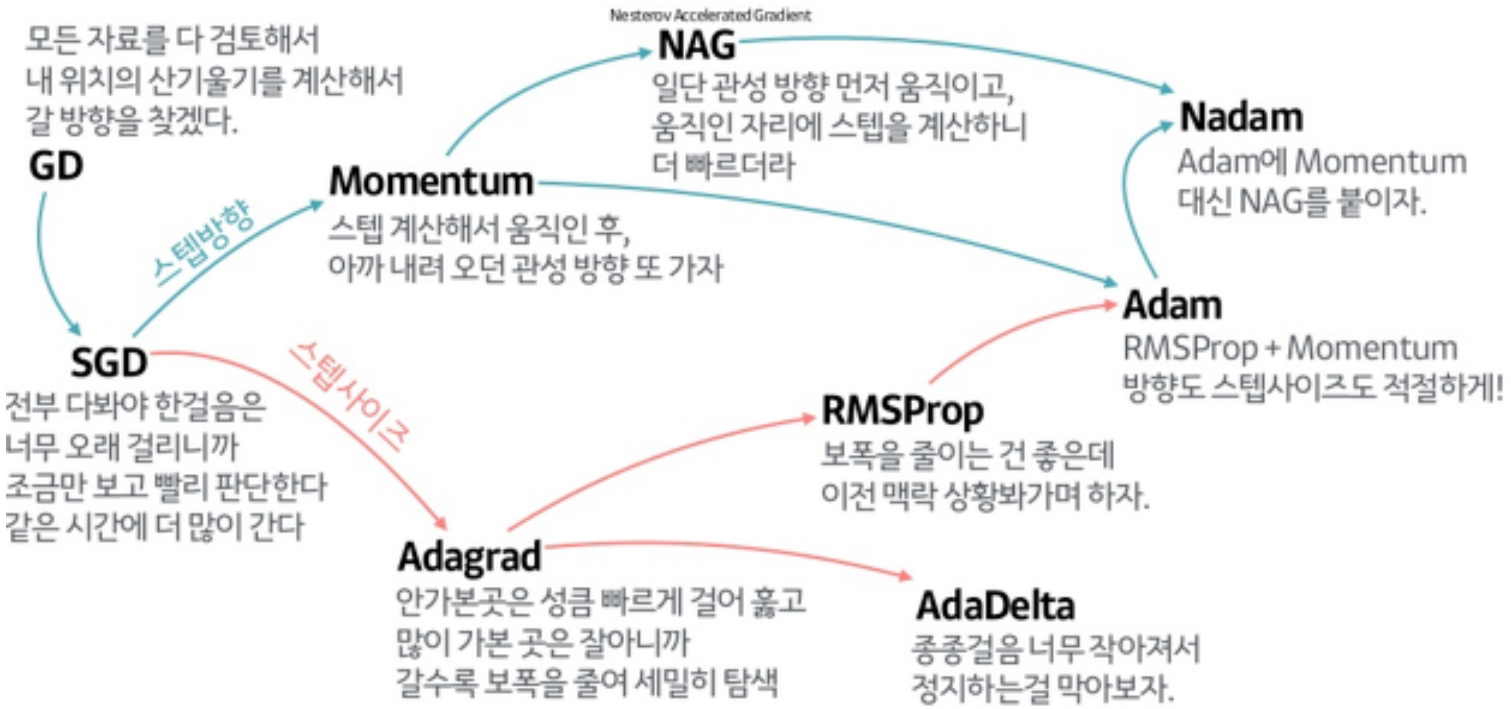
RMSProp

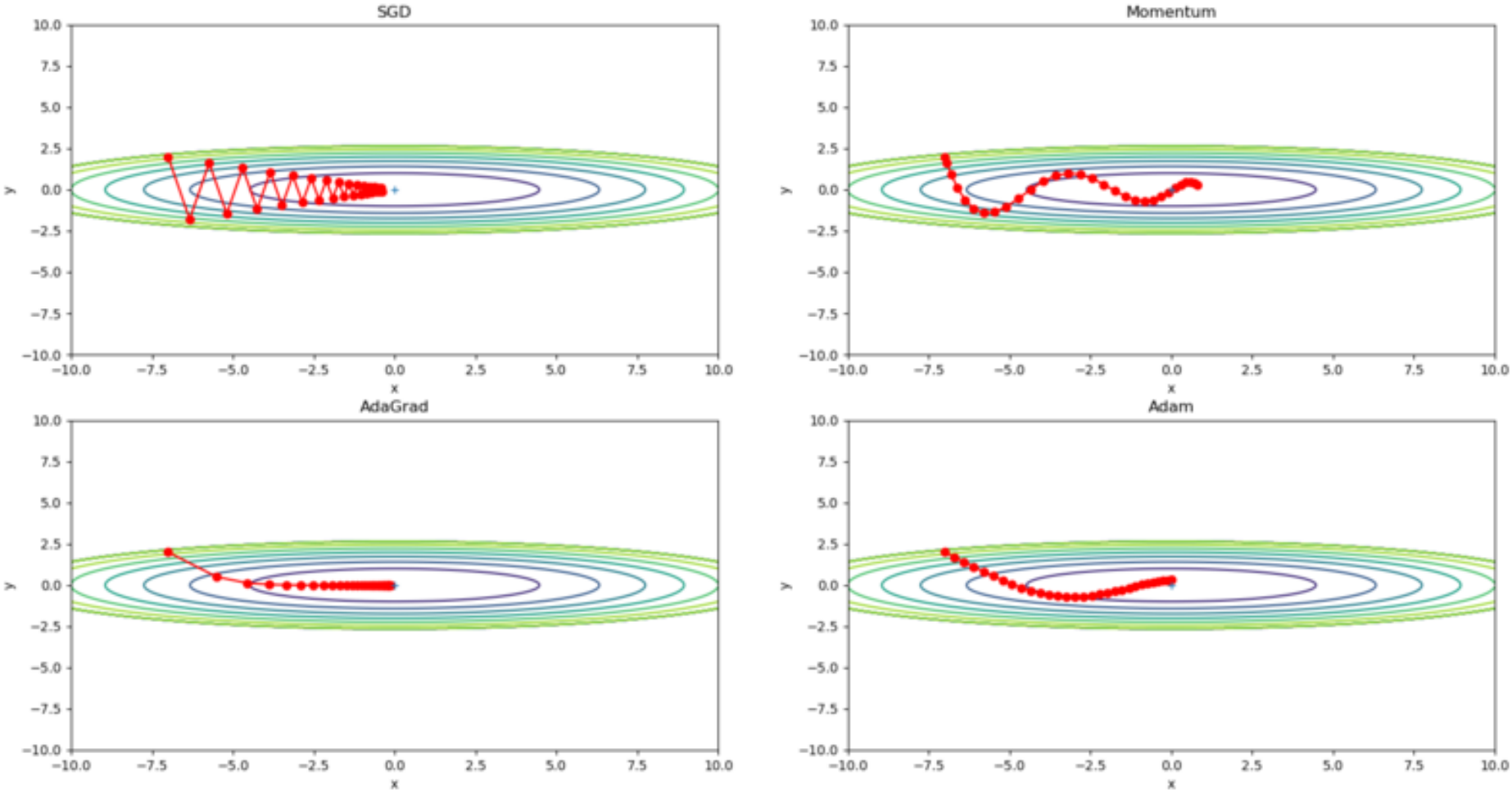
$$h = \rho h_{n-1} + (1-\rho)(\frac{\partial}{\partial \theta_j} J(\theta^{(i)}))^2$$

Adam

- Momentum + RMSProp

최적화 기법 (Optimization)





https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=tinz6461&logNo=221589073944