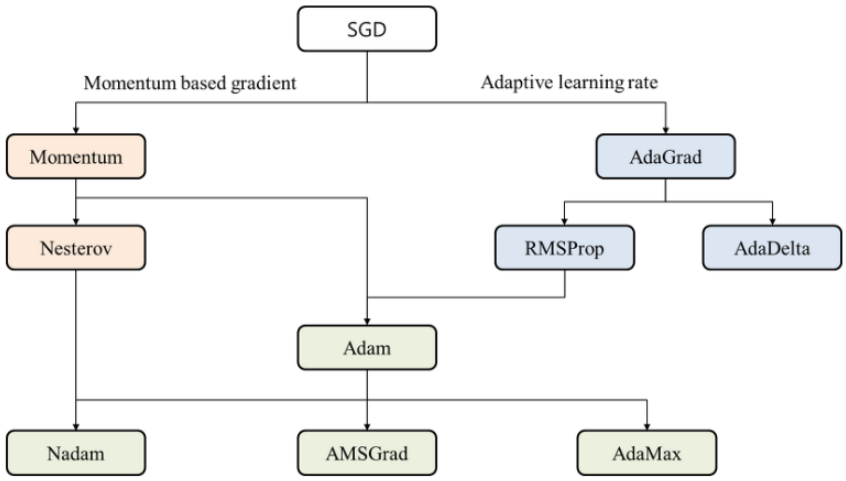
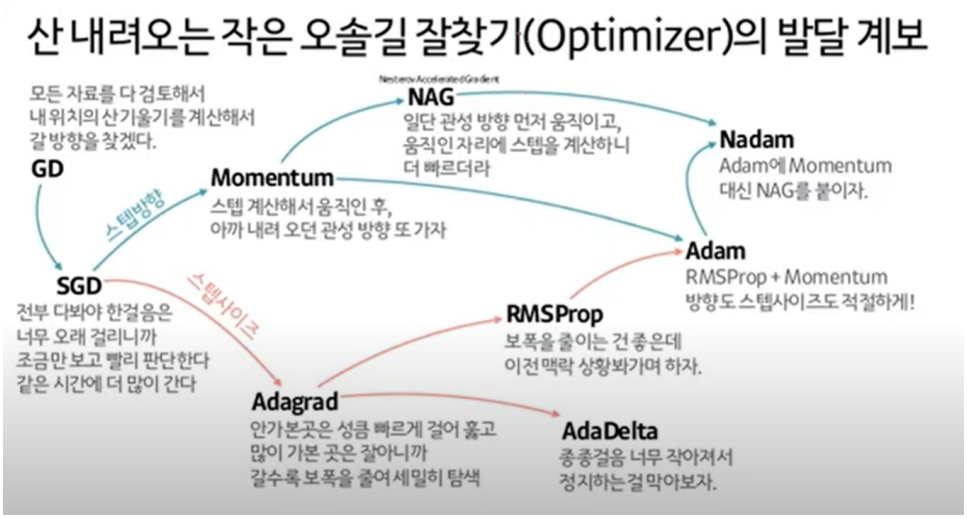
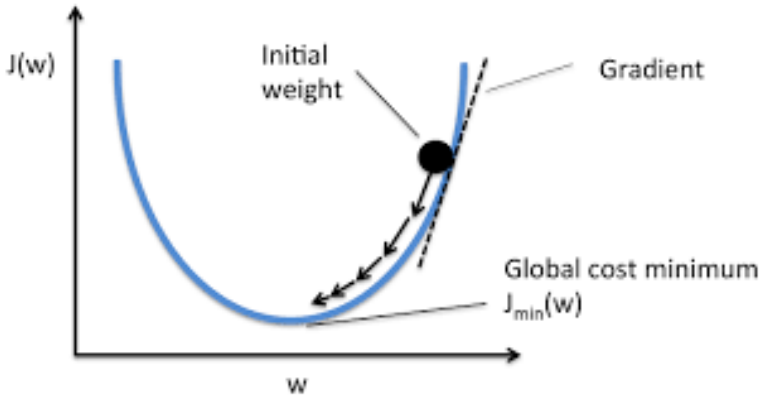
Optimization methods





1. gradient descent

$$\theta_j := \theta_j - \alpha \times \nabla J(\theta)$$



2. momentum based gradient

→ 아까 내려오던 관성 **방향**을 고려하자!!

> 💡 v : 현재 위치 이전까지 누적된 그래디언트 벡터

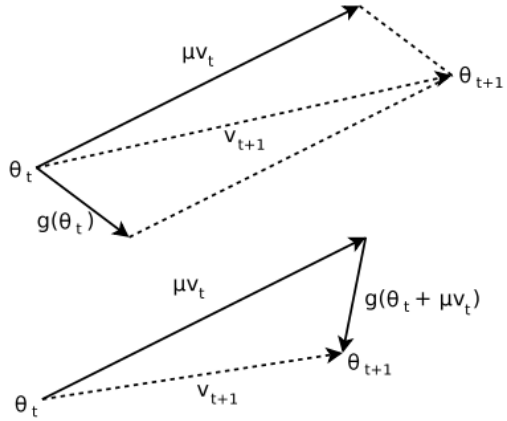


Figure 1. **(Top)** Classical Momentum **(Bottom)** Nesterov Accelerated Gradient

Momentum

$$v_{t+1} = \mu v_t - \alpha \times \nabla J(\theta)$$

$$\theta_{j,t+1} = \theta_{j,t} + v_{t+1}$$

Nestrov Accelerated Gradient(NAG)

$$v_{t+1} = \mu v_t - \alpha \times \nabla J(\theta_t + \mu v_t)$$

$$\theta_{j,t+1} = \theta_{j,t} + v_{t+1}$$

선험적/모험적으로 먼저 진행한 후 에러를 교정

3. adaptive learning rate

→ 지금까지 조금 움직인 곳은 앞으로 성큼성큼 움직이고, 지금까지 많이 움직인 곳은 앞으로 천천히 움직이자!!! # 변수 별 맞춤형 **학습률**

AdaGrad

$$h := h + \left(\frac{\partial E}{\partial w(t)}\right)^2$$

$$w := w - \alpha \frac{1}{\sqrt{(h)}} \times \left(\frac{\partial E}{\partial w(t)}\right)$$

→ h 값이 항상 증가하기 때문에, 나중에는 걸음이 너무 작아져 학습이 정지하게 되는 문제점 발생

→ **RMSProp** : h값을 exponential moving average 방법으로 조정하자

$$h := \gamma h + (1-\gamma)\left(\frac{\partial E}{\partial w(t)}\right)^2$$

$$w := w - \alpha \frac{1}{\sqrt{(h)}} \times \left(\frac{\partial E}{\partial w(t)}\right)$$

→ **Adadelta** : h값 조정 + h값 변화에 맞춰 learning rate도 조정하자 (수식 생략)

4. Adam, Nadam

- Adam = RMSProp + momentum
- Nadam = RMSProp + Nestrov
- AMSGrad: adam이 일부 데이터에 대해 optimal solution에 수렴이 어렵다는 문제를 해결하기 위해 exponential moving average 대신 second momentum 값들 중 가장 큰 값을 사용하도록 변형한 것.
- AdaMax: Adam에서 L2 norm을 기반으로 learning rate를 조절하는 부분을 Lp norm으로 확장시킨 알고리즘.

참고한 사이트 링크

https://brunch.co.kr/@jaeyuelpark/64

https://tensorflow.blog/2017/03/22/momentum-nesterov-momentum/

https://deep-learning-study.tistory.com/158

https://untitledtblog.tistory.com/149

https://tgd.kr/c/deeplearning/19860071

https://ardino.tistory.com/29?category=1088001

https://ardino.tistory.com/30?category=1088001