

Chapter6. 학습 관련 기술들

Optimization & Normalization

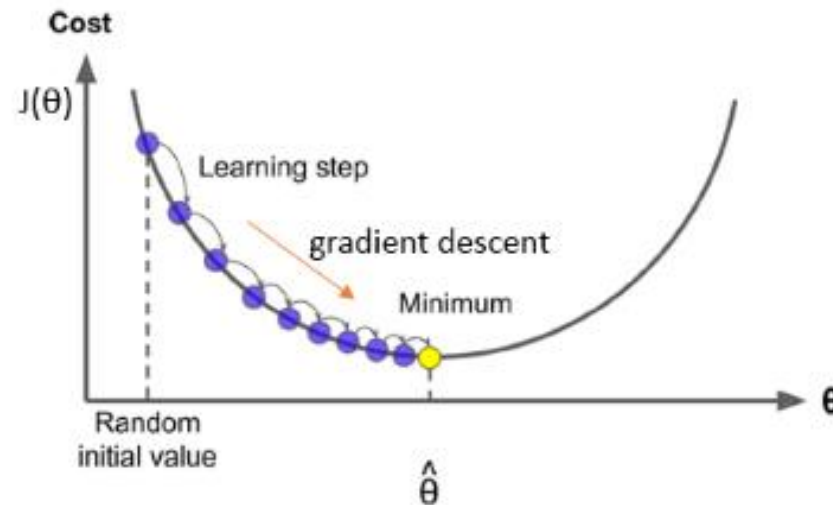
박종혁

Gradient Descent

경사하강법

경사하강법이란?

해당 함수의 최소값 위치를 찾기 위해 비용 함수(Cost Function)의 Gradient 반대 방향으로 정의한 step size를 가지고 조금씩 움직여 가면서 최적의 파라미터를 찾으려는 방법.



밑바닥부터 시작하는 딥러닝

BGD

Optimization

◆ BGD(Batch Gradient Descent)

- For one update, gradients are calculated for the whole dataset
- Batch gradient descent is guaranteed to converge to a local minimum
- Redundant computations for large redundant dataset

```
Repeat
   $\alpha = 0$ 
  for n = 1 to N (for all training data)
     $\alpha += \frac{\partial E_n}{\partial w}$ 
  end
   $w = w - \eta \alpha$ 
Until end condition satisfied
```

SGD

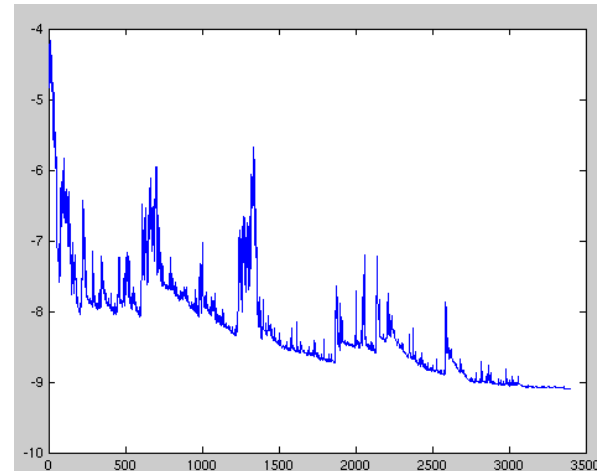
Optimization

◆ SGD(Stochastic Gradient Descent)

- For one update, gradients are calculated for one sample
- Usually faster and can be used to learn online
- Fluctuations: Maybe good or maybe bad
- With a small learn rate, show similar performance

```
Repeat
  for n = 1 to N (for all training data)
    
$$w = w - \eta \frac{\partial E_n}{\partial w}$$

  end
Until end condition satisfied
```



밑바닥부터 시작하는 딥러닝

BGD VS SGD

Optimization

[Batch Gradient Descent]

(장점)

1. 모든 데이터를 계산하여 최적의 한스텝을 찾는다.
2. Local minimum을 보장한다.

(단점)

1. 데이터가 크면 계산량이 많아져서 느리다.

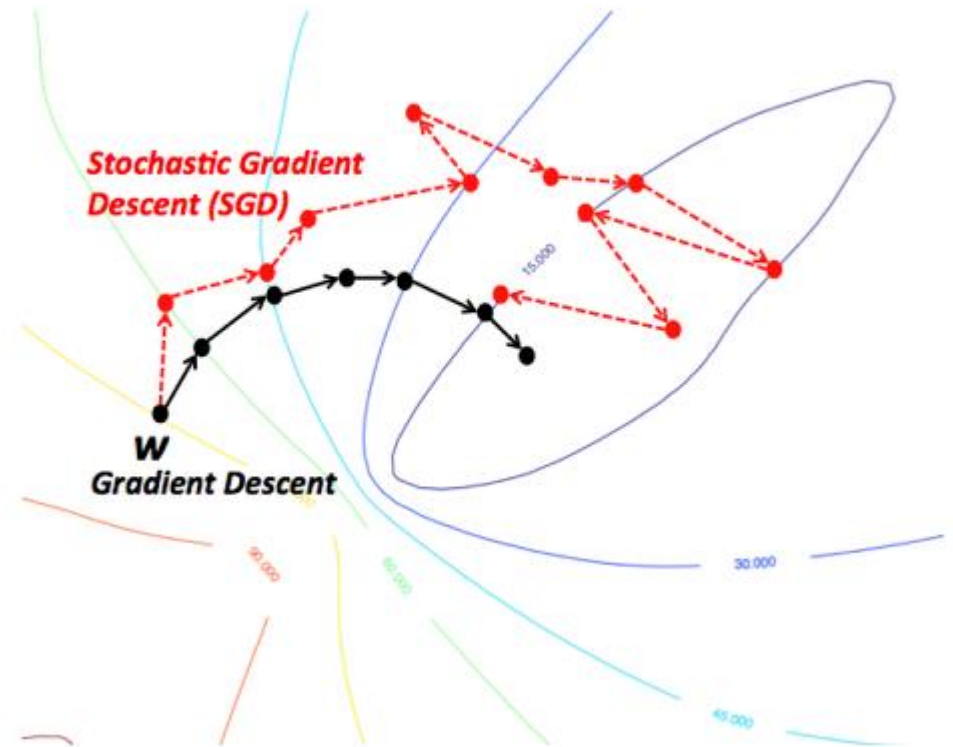
[Stochastic Gradient Descent]

(장점)

1. 실제 Gradient를 잘 찾을 수 있다.
2. 계산량이 적어 빠르다.
3. Batch가 찾지 못하는 Global minimum을 찾을수도 있다.

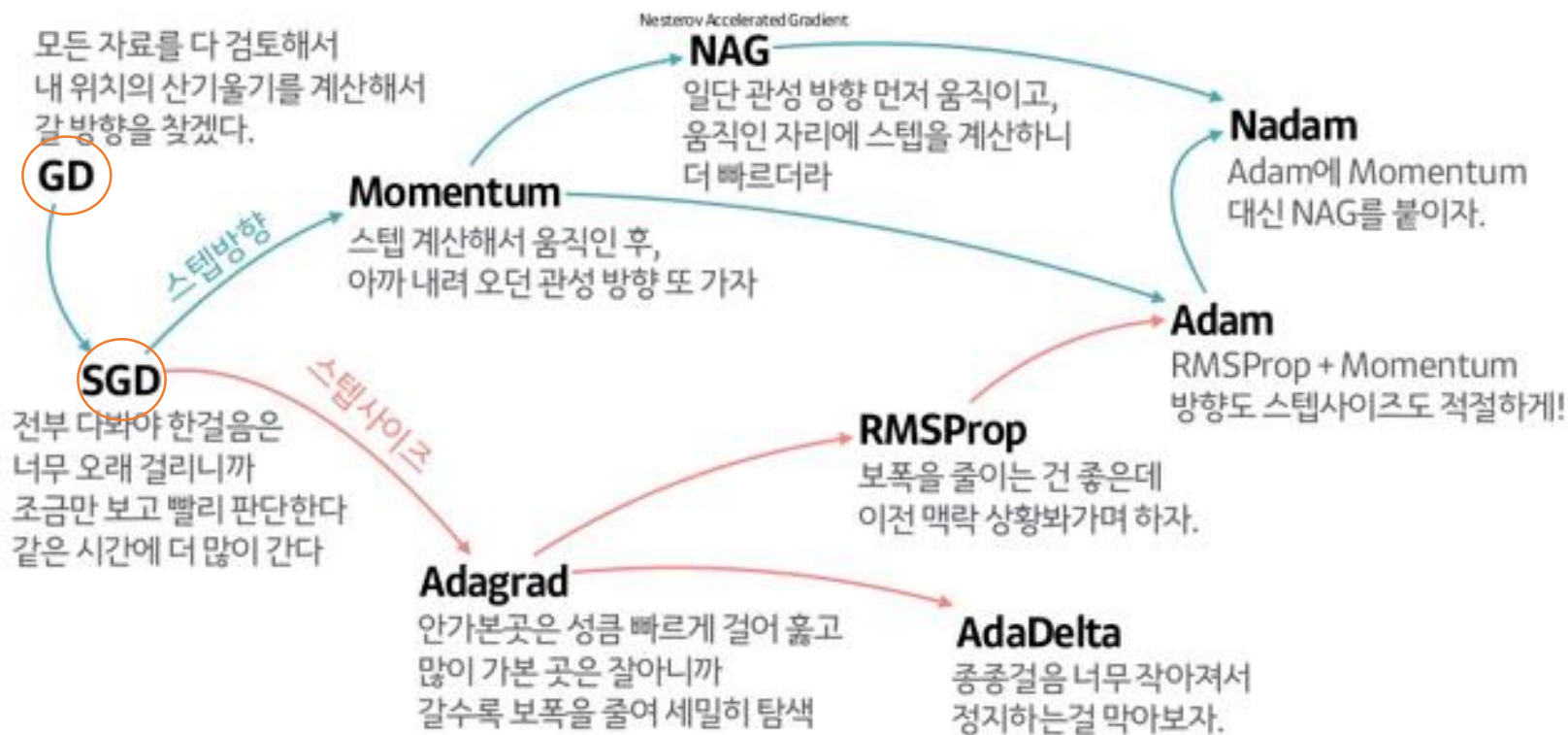
(단점)

1. Local minimum을 보장하지 못한다.
2. 데이터의 분산이 크면 오히려 Batch보다 성능이 떨어질수도 있다.
3. 비등방성 함수에서는 비효율적이다.



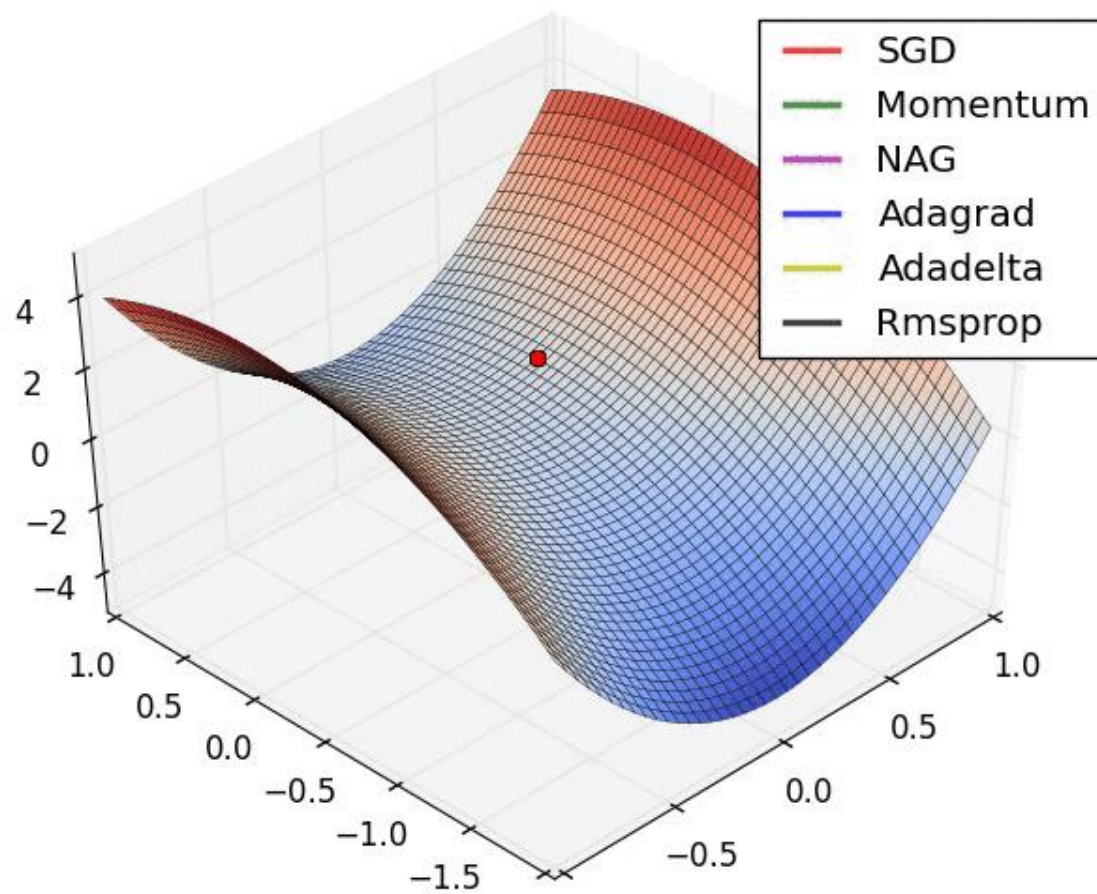
더 나은 방법을 찾아서..

Optimization



Overview

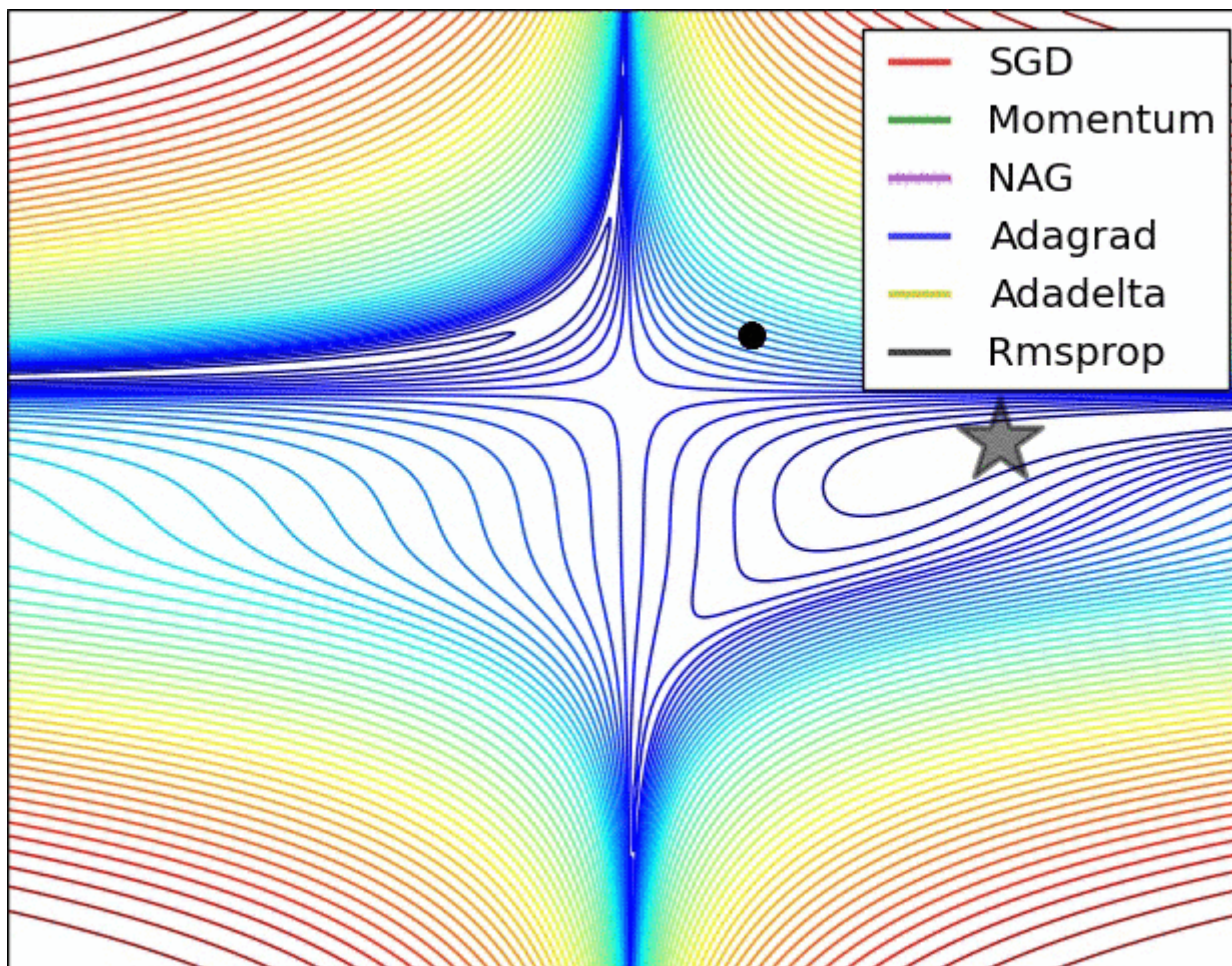
움짤로 살펴보자!



밑바닥부터 시작하는 딥러닝

Overview

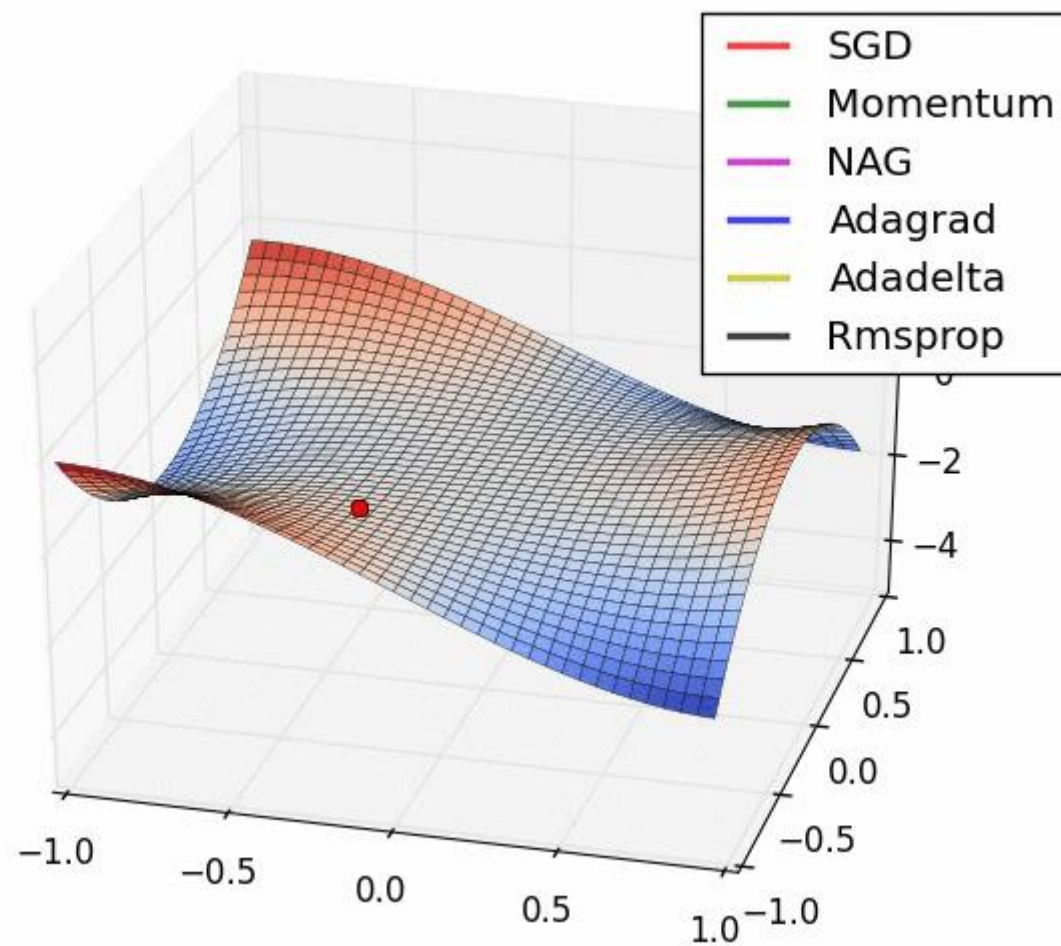
움짤로 살펴보자!



밑바닥부터 시작하는 딥러닝

Overview

움짤로 살펴보자!



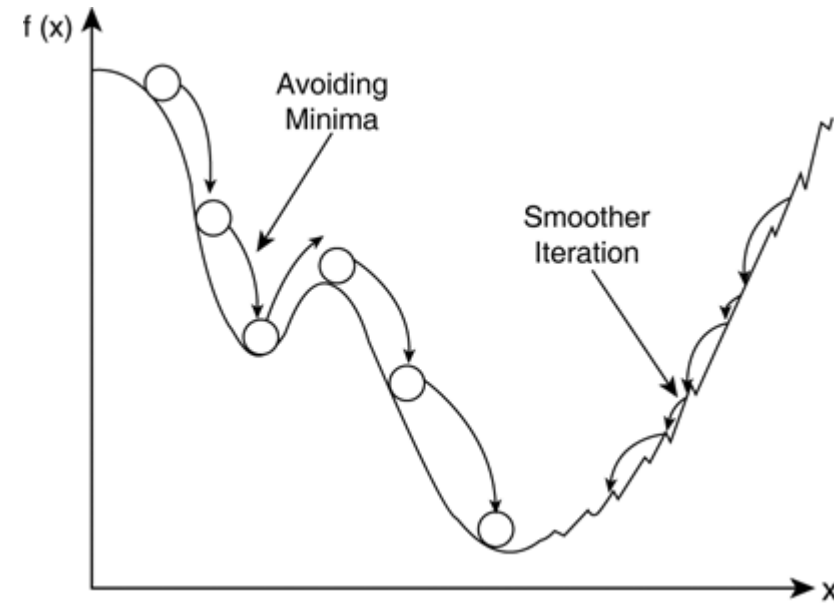
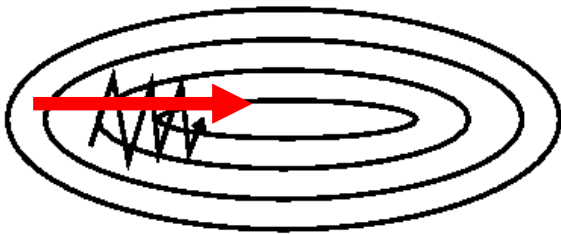
밑바닥부터 시작하는 딥러닝

Momentum

Step direction

Simple gradient method depends on the current position

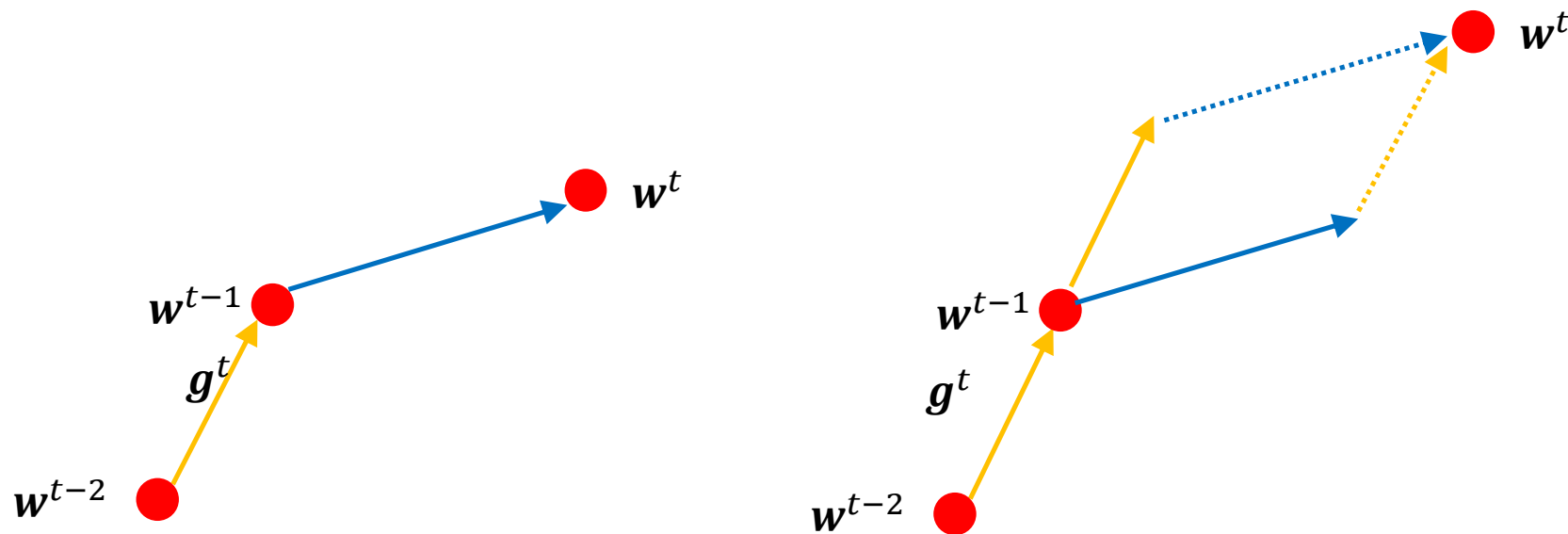
- Hard to avoid local minimum
- Gradient can vary much



밑바닥부터 시작하는 딥러닝

Momentum

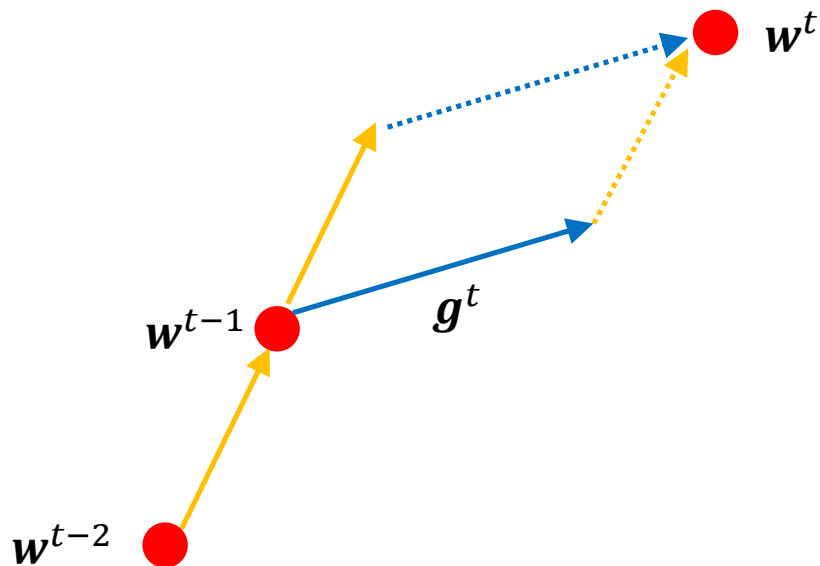
Step direction



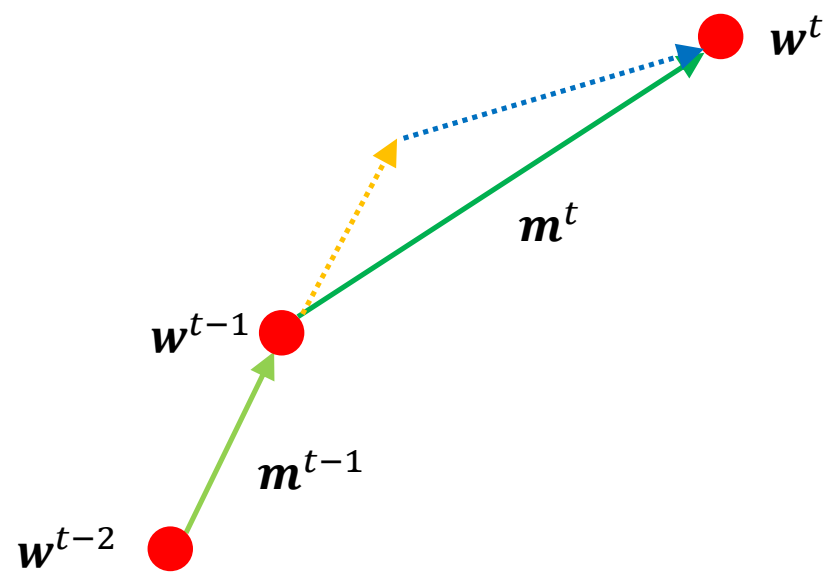
밑바닥부터 시작하는 딥러닝

Momentum

Step direction



$$m^t = m^{t-1} + g^t$$



밑바닥부터 시작하는 딥러닝

Momentum

Step direction

g^t : 시간 t에서의 기울기

$$w^t = w^{t-1} - \eta g^t$$

기존 방법

Momentum rate

Learning rate

$$m^t = \alpha m^{t-1} + \eta g^t$$

$$w^t = w^{t-1} - m^t$$

Momentum

- Update parameters considering both the momentum and the gradient of the current position

Momentum

Step direction

Momentum is the exponential average of the past gradients

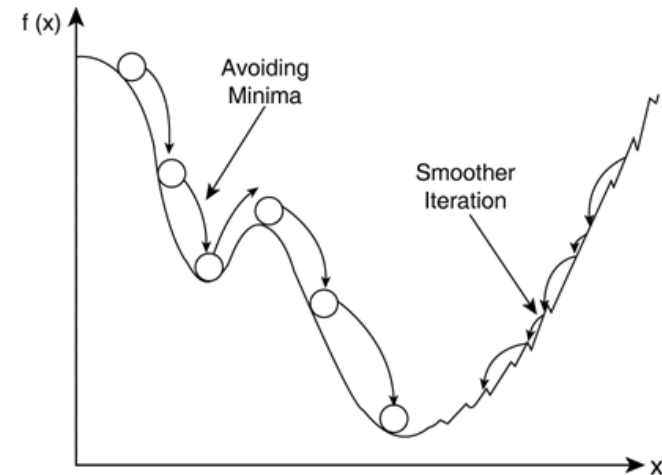
$$\mathbf{m}^t = \eta \mathbf{g}^t + \gamma \eta \mathbf{g}^{t-1} + \gamma^2 \eta \mathbf{g}^{t-2} + \dots$$

$$\mathbf{m}^t = \gamma \mathbf{m}^{t-1} + \eta \mathbf{g}^t$$

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \mathbf{m}^t$$



Oscillation



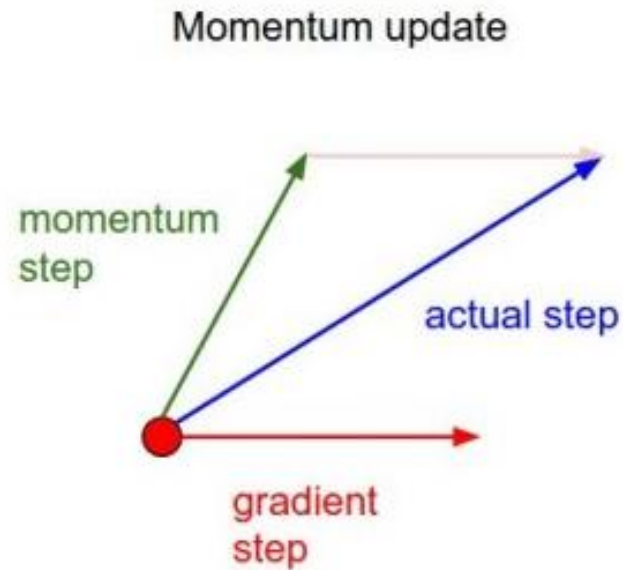
밑바닥부터 시작하는 딥러닝

Momentum

Step direction

◆ Disadvantage of Momentum

- Simple addition of momentum may cause excessive update
- We may miss the position where to stop



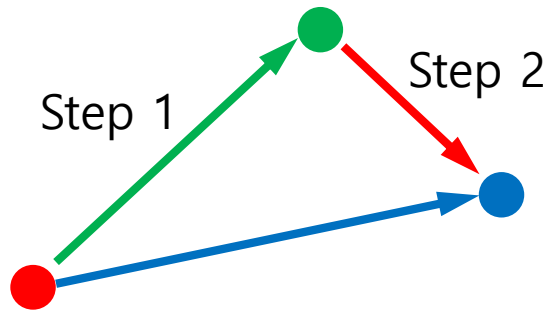
NAG

Step direction

NAG(Nesterov Accelerated Gradient)

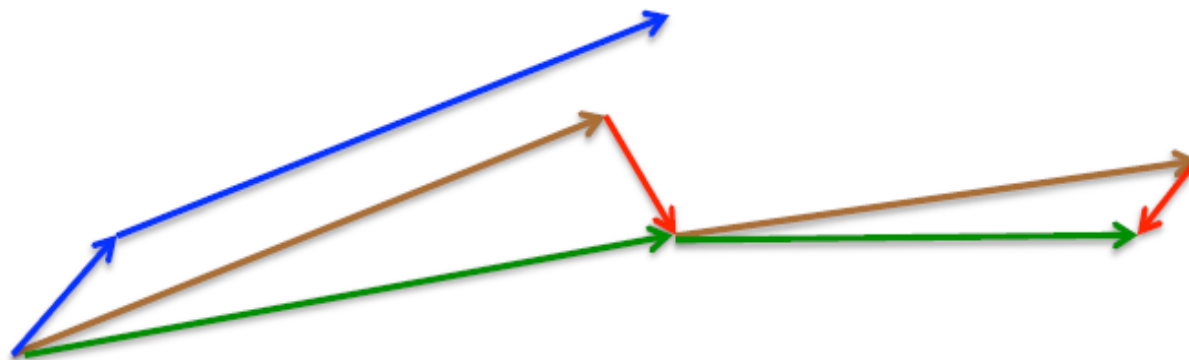
→ Momentum의 불안정한 Parameter update를 방지하고자 고안된 방법.

1. Update the current position considering the momentum
2. Evaluate the gradient at the new position
3. Update the position consider the gradient



NAG

Step direction



brown vector = jump, red vector = correction, green vector = accumulated gradient

blue vectors = standard momentum

Adaptive Learning Rates

Step size

- ◆ Why do we use the SAME learning rate for all the weights?
 - Can we use different learning rate?
 - Hmm.. Interesting, but why do we need different learning rate?
- ◆ Some weight are frequently updated and some are not
 - What if some inputs may be 0 but some may have none-zero values in most of training data
 - OK.. Let's make a large update for the less updated parameters

Adagrad

Step size

- Adagrad
 - Update much less-updated parameters
 - Update less much-updated parameter

g_i^t : parameter i 의 시간 t 에서의 기울기

G_i^t : parameter i 가 지금까지 update된 총량

$$w_i^t = w_i^{t-1} - \eta g_i^t$$

표준 방법

$$G_i^t = G_i^{t-1} + (g_i^t)^2$$
$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$

Adagrad

Very small value

밑바닥부터 시작하는 딥러닝

Adagrad

Step size

- Disadvantage of Adagrad
 - Eventually, G_i^t becomes large as time goes on
 - Parameters are rarely updated at some time

g_i^t : parameter i 의 시간 t 에서의 기울기

G_i^t : parameter i 가 지금까지 update된 총량

$$w_i^t = w_i^{t-1} - \eta g_i^t$$

표준 방법

$$G_i^t = G_i^{t-1} + (g_i^t)^2$$
$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$

Adagrad

Very small value

밑바닥부터 시작하는 딥러닝

RMSProp

Step size

- RMSProp
 - Instead of considering the total amount of updates, let's consider the amount of recent updates

g_i^t : parameter i 의 시간 t 에서의 기울기

G_i^t : parameter i 가 update된 양

$$G_i^t = G_i^{t-1} + (g_i^t)^2$$
$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$

Adagrad

$$G_i^t = \gamma G_i^{t-1} + (1 - \gamma)(g_i^t)^2$$
$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$

RMSProp

일반적으로 γ 는 0.9 정도로 잡는다.

Very small value

밑바닥부터 시작하는 딥러닝

Adam

Step direction + Step size

- Adam
 - RMSProp + Momentum

g_i^t : parameter i 의 시간 t 에서의 기울기

G_i^t : parameter i 가 update된 양

Momentum

$$m_i^t = \gamma m_i^{t-1} + \eta g_i^t$$

$$w_i^t = w_i^{t-1} - m_i^t$$

$$\rightarrow m_i^t = \beta_1 m_i^{t-1} + (1 - \beta_1) g_i^t$$

RMSProp

$$G_i^t = \gamma G_i^{t-1} + (1 - \gamma) (g_i^t)^2$$

$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$

$$\rightarrow G_i^t = \beta_2 G_i^{t-1} + (1 - \beta_2) (g_i^t)^2$$

Adam

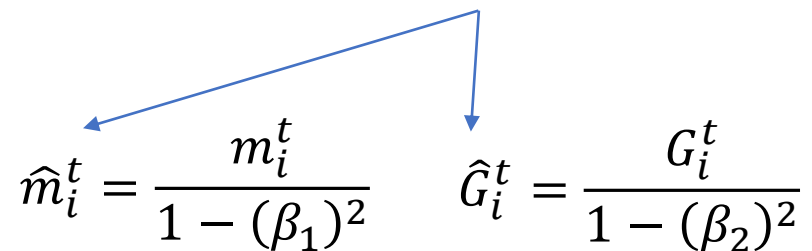
Step direction + Step size

- Adam
 - RMSProp + Momentum

$$m_i^t = \beta_1 m_i^{t-1} + (1 - \beta_1) g_i^t$$

$$G_i^t = \beta_2 G_i^{t-1} + (1 - \beta_2) (g_i^t)^2$$

Unbiased expectation


$$\hat{m}_i^t = \frac{m_i^t}{1 - (\beta_1)^t} \quad \hat{G}_i^t = \frac{G_i^t}{1 - (\beta_2)^t}$$

$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{\hat{G}_i^t + \epsilon}} \hat{m}_i^t$$

Bias Correction

What is it?

밑바닥부터 시작하는 딥러닝

Adam

Step direction + Step size

- Adam
 - RMSProp + Momentum

$$\begin{aligned}m_i^t &= \gamma m_i^{t-1} + \eta g_i^t \\w_i^t &= w_i^{t-1} - m_i^t\end{aligned}$$

Momentum

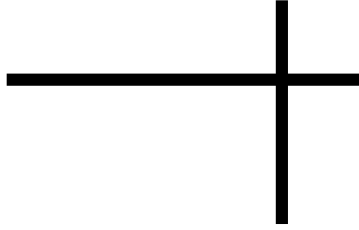
$$\begin{aligned}G_i^t &= \gamma G_i^{t-1} + (1 - \gamma)(g_i^t)^2 \\w_i^t &= w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t\end{aligned}$$

RMSProp

$$\begin{aligned}m_i^t &= \beta_1 m_i^{t-1} + (1 - \beta_1) g_i^t \\G_i^t &= \beta_2 G_i^{t-1} + (1 - \beta_2)(g_i^t)^2 \\ \hat{m}_i^t &= \frac{m_i^t}{1 - (\beta_1)^2} \quad \hat{G}_i^t = \frac{G_i^t}{1 - (\beta_2)^2}\end{aligned}$$

$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{\hat{G}_i^t + \epsilon}} \hat{m}_i^t$$

Adam



Thank You!

Any Question?