Optimization

Overview

우리가 배워야 할 것

저번주와 이번주에 Activate Function과 Weight Initialization에 대해 배웠다
오늘 Optimization만 배운다면, 딥러닝의 모델을 개선시킬 수 있는 세 가지 핵심 기술을 모두 습득한 것

```
learning_rate = 0.000001

w1 = np.random.uniform(low=-0.058, high=+0.058, size=(784, 1000))
w2 = np.random.uniform(low=-0.077, high=+0.077, size=(1000, 10))

num_epoch = 100

for epoch in range(num_epoch):
    # Forward propagation
    z1 = X_train.dot(w1)
    a1 = sigmoid(z1)
    z2 = a1.dot(w2)
    a2 = sigmoid(z2)

    # Backpropagation
    d2 = a2 - y_train_hot
    d1 = d2.dot(w2.T) * a1 * (1 - a1)

    w2 = w2 - learning_rate * a1.T.dot(d2)
    w1 = w1 - learning_rate * X_train.T.dot(d1)
```

앞으로 우리가 배워야 할 것

저번주와 이번주에 Activate Function과 Weight Initialization에 대해 배웠다
오늘 Optimization만 배운다면, 딥러닝의 모델을 개선시킬 수 있는 세 가지 핵심 기술을 모두 습득한 것

```
learning_rate = 0.000001

w1 = np.random.uniform(low=-0.058, high=+0.058, size=(784, 1000))
w2 = np.random.uniform(low=-0.077, high=+0.077, size=(1000, 10))

num_epoch = 100

for epoch in range(num_epoch):
    # Forward propagation
    z1 = X_train.dot(w1)
    a1 = sigmoid(z1)
    z2 = a1.dot(w2)
    a2 = sigmoid(z2)

    # Backpropagation
    d2 = a2 - y_train_hot
    d1 = d2.dot(w2.T) * a1 * (1 - a1)

    w2 = w2 - learning_rate * a1.T.dot(d2)
    w1 = w1 - learning_rate * X_train.T.dot(d1)
```

Weight Initialization

Activation Function

Optimizer

앞으로 우리가 배워야 할 것

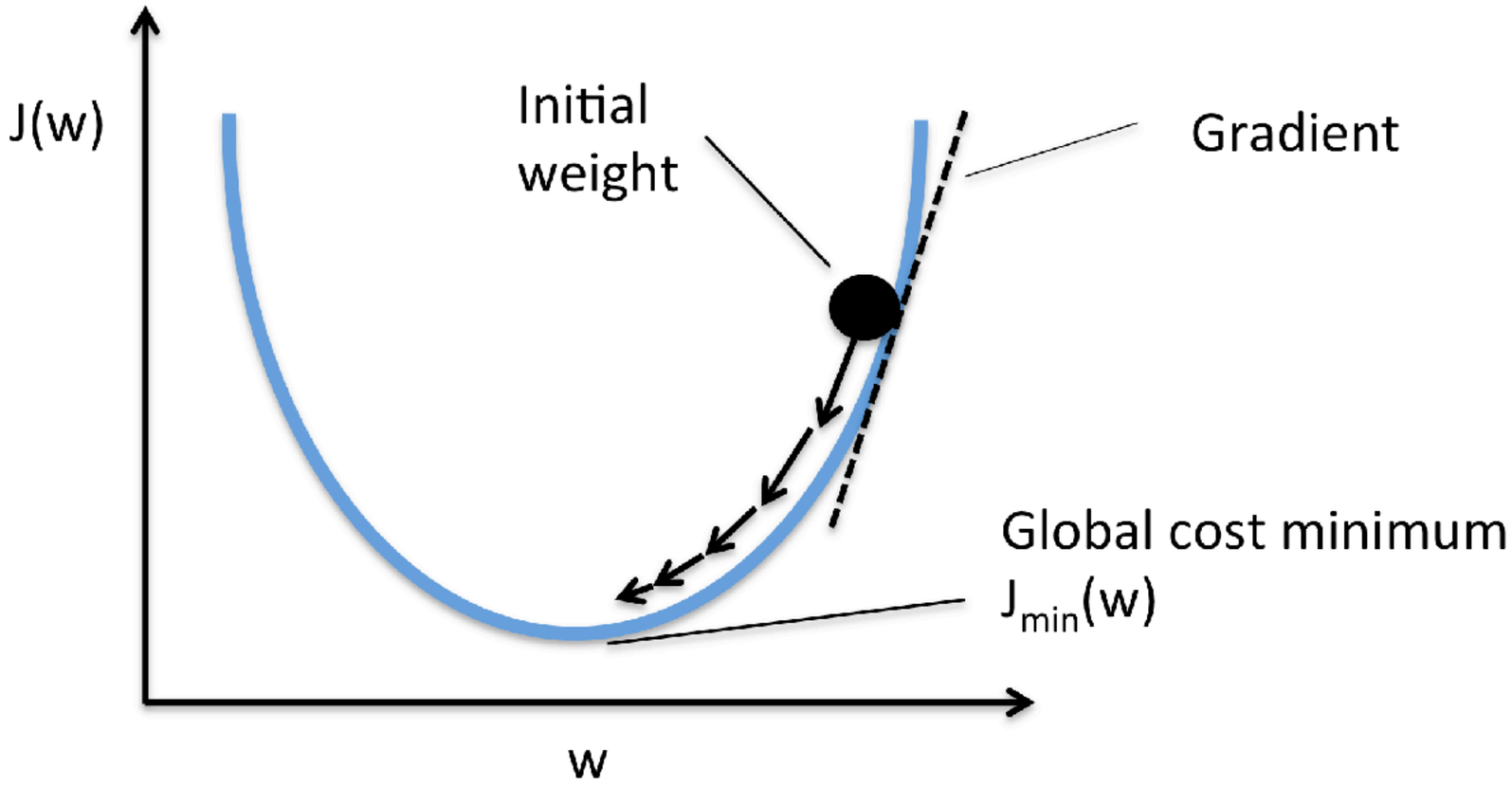
저번주와 이번주에 Activate Function과 Weight Initialization에 대해 배웠다
오늘 Optimization만 배운다면, 딥러닝의 모델을 개선시킬 수 있는 세 가지 핵심 기술을 모두 습득한 것

```
learning_rate = 0.000001

w1 = np.random.uniform(low=-0.058, high=+0.058, size=(784, 1000))
w2 = np.random.uniform(low=-0.077, high=+0.077, size=(1000, 10))

num_epoch = 100

for epoch in range(num_epoch):
    # Forward propagation
    z1 = X_train.dot(w1)
    a1 = sigmoid(z1)
    z2 = a1.dot(w2)
    a2 = sigmoid(z2)

    # Backpropagation
    d2 = a2 - y_train_hot
    d1 = d2.dot(w2.T) * a1 * (1 - a1)

    w2 = w2 - learning_rate * a1.T.dot(d2)
    w1 = w1 - learning_rate * X_train.T.dot(d1)
```

Optimizer

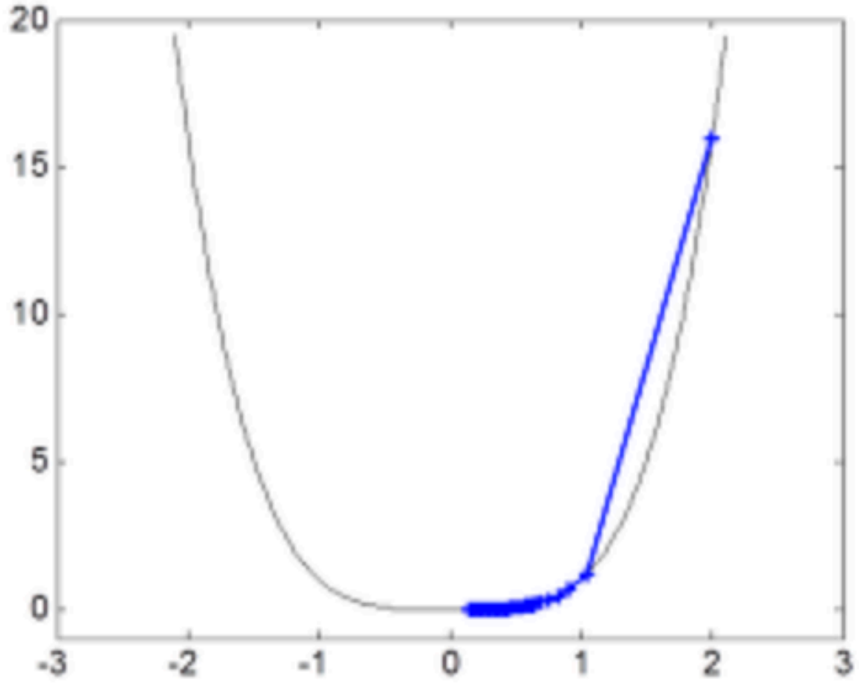
An Overview of Gradient Descent

예측값(predict)과 정답(actual)간의 차이를 알 수 있는 미분 가능한 함수(loss function)이 있다면,
이를 편미분한 뒤 기울기(gradient)를 타고 극소점으로 내려가면 된다.



Gradient Descent - Problem 1

초반에는 가파르게 떨어지다가, 어느 순간 수렴 속도가 굉장히 느려진다
이는 극소점 주변은 기울기가 굉장히 평평하기 때문

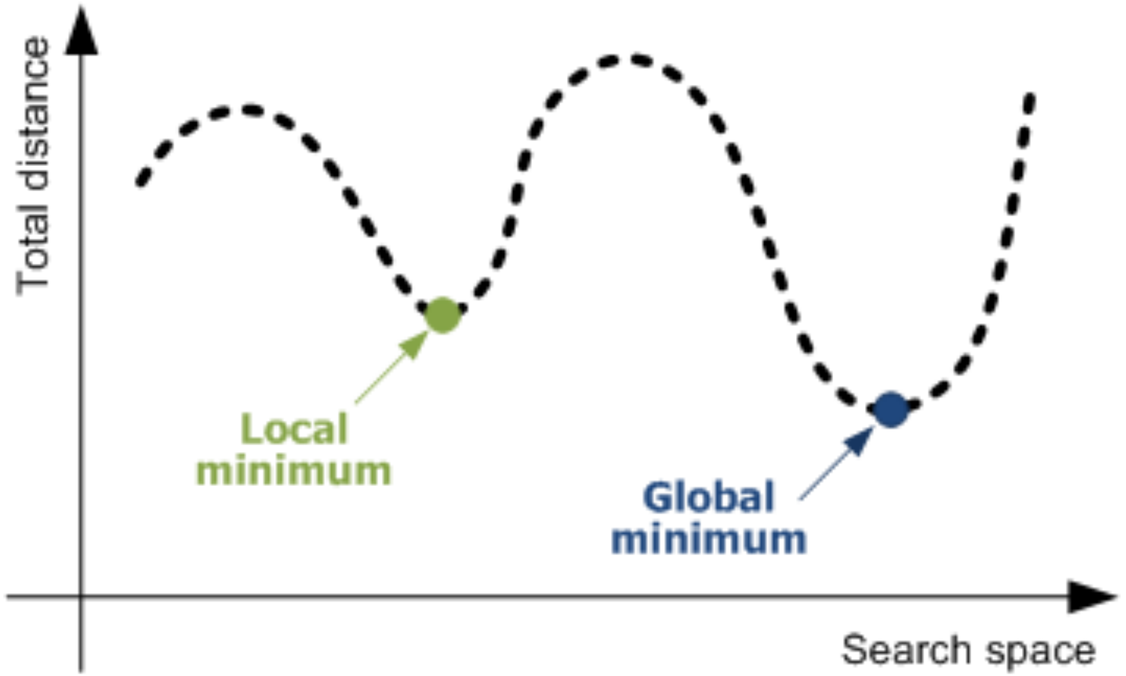


$\lambda = 0.03$

최적화 기법의 직관적 이해
by 다크 프로그래머. http://darkpgmr.tistory.com/149

Gradient Descent - Problem 2

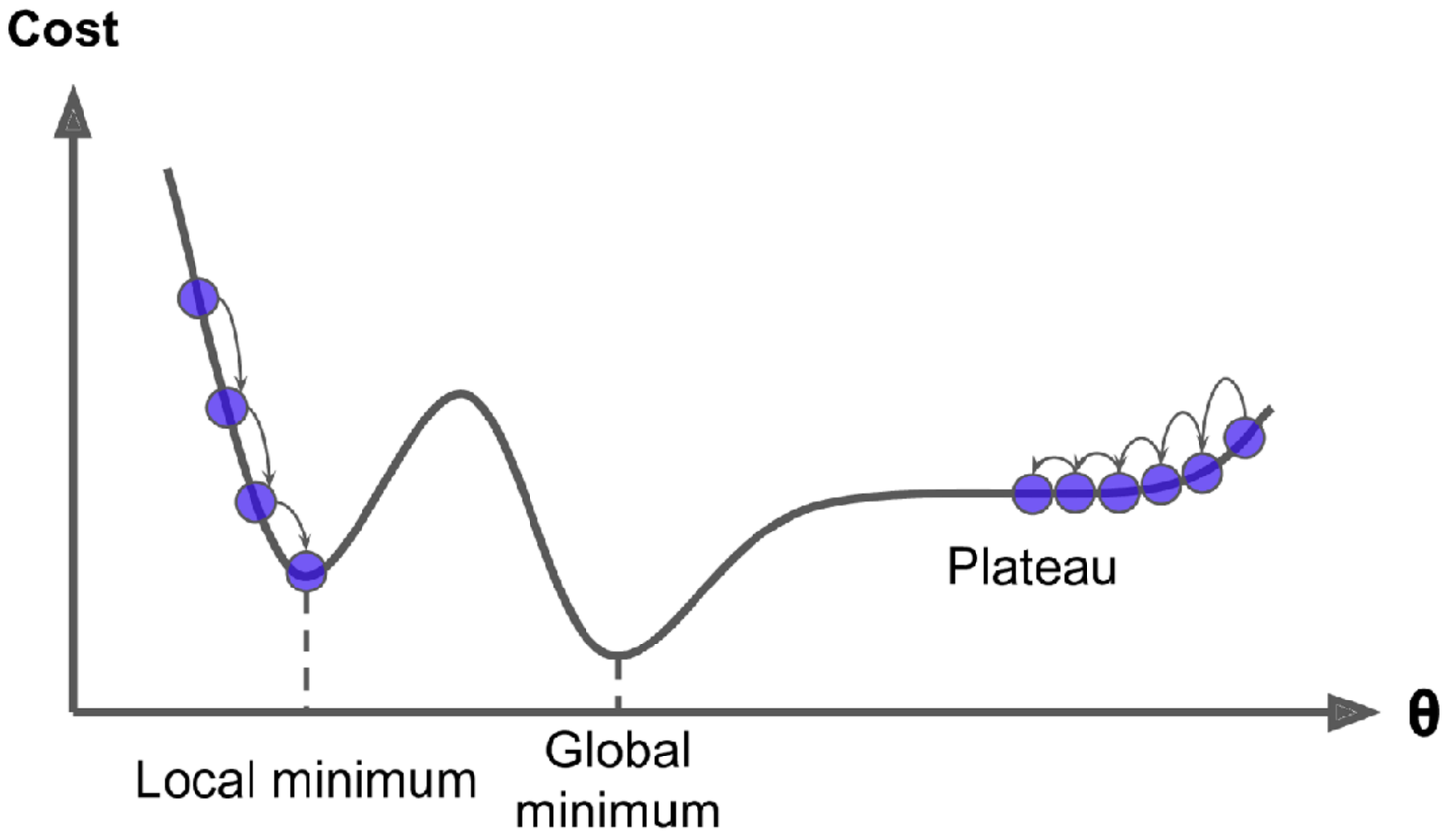
local minima 현상
gradient를 타고 내려가지만 종착점이 global minima라는 보장이 없다



Gradient Descent - Problem 3

Plateau 현상

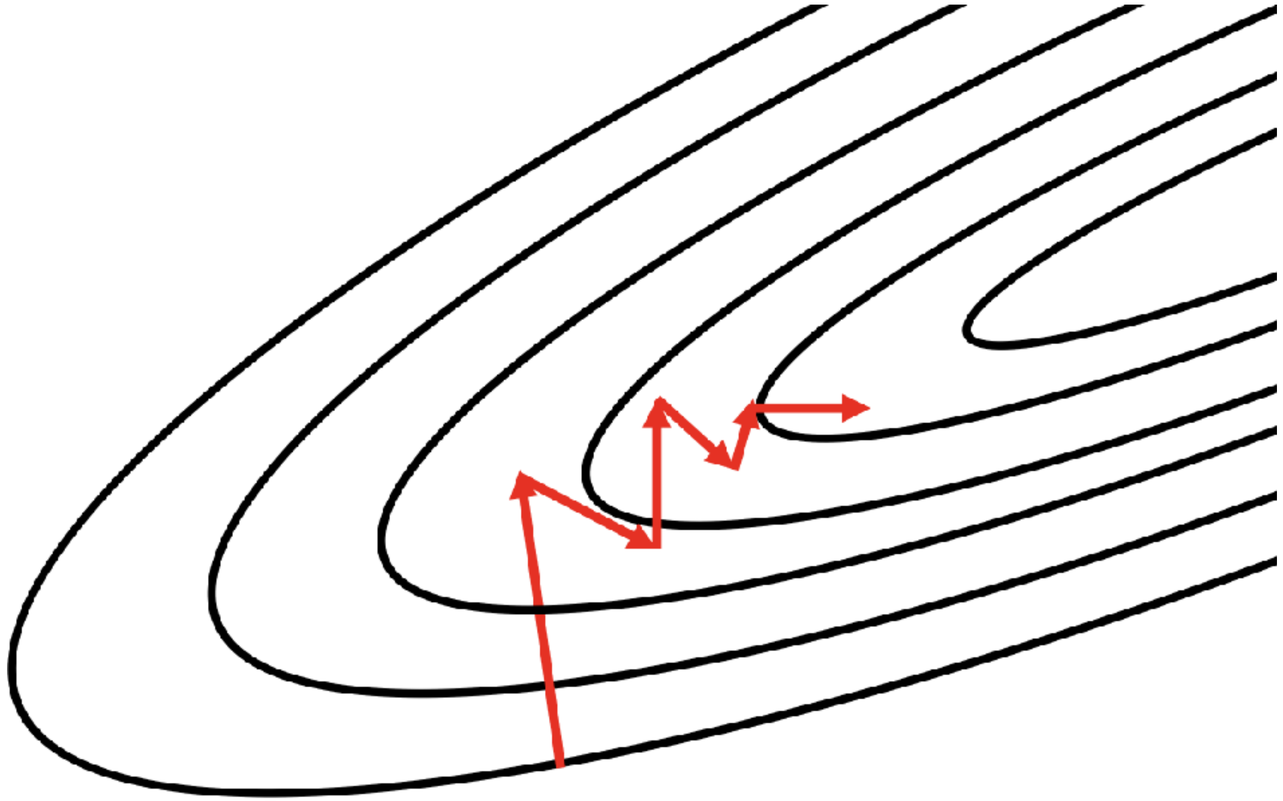
gradient가 평평해졌지만, 마찬가지로 이 부분이 global minima라고 보장할 수는 없다



Gradient Descent - Problem 4

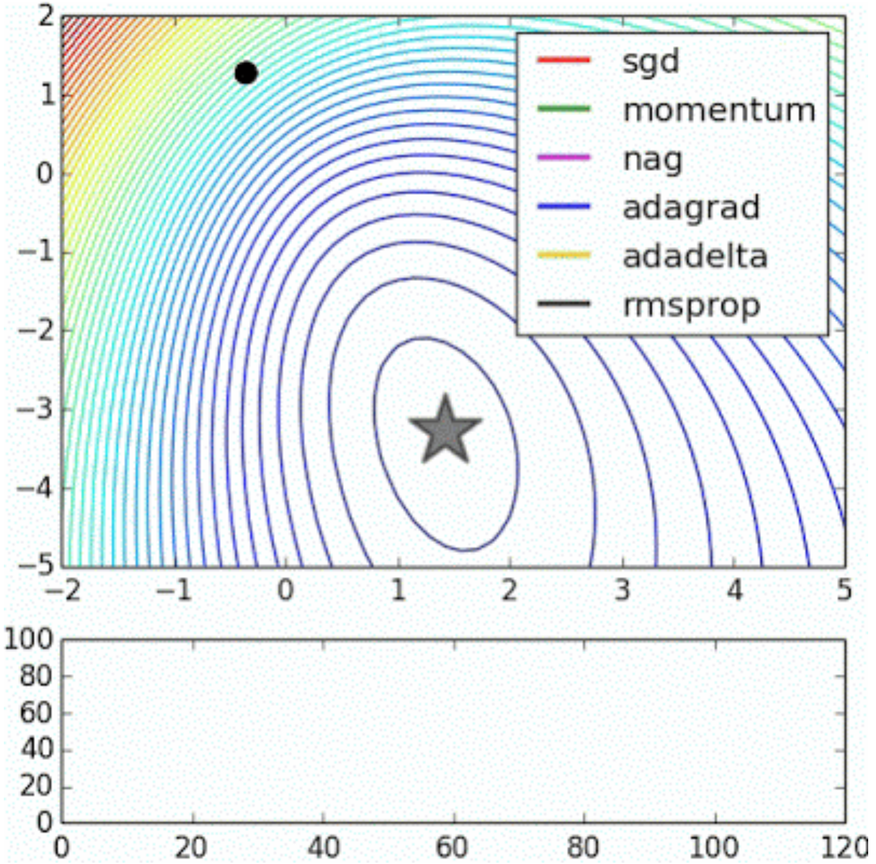
zig-zag 현상

각 dimension의 gradient가 서로 다르면 update가 zig-zag로 일어나게 된다



Gradient Descent - Solution

앞서 설명한 문제를 해결하기 위한 다양한 아이디어가 존재한다
오늘은 이 아이디어들에 대해 살펴볼 것

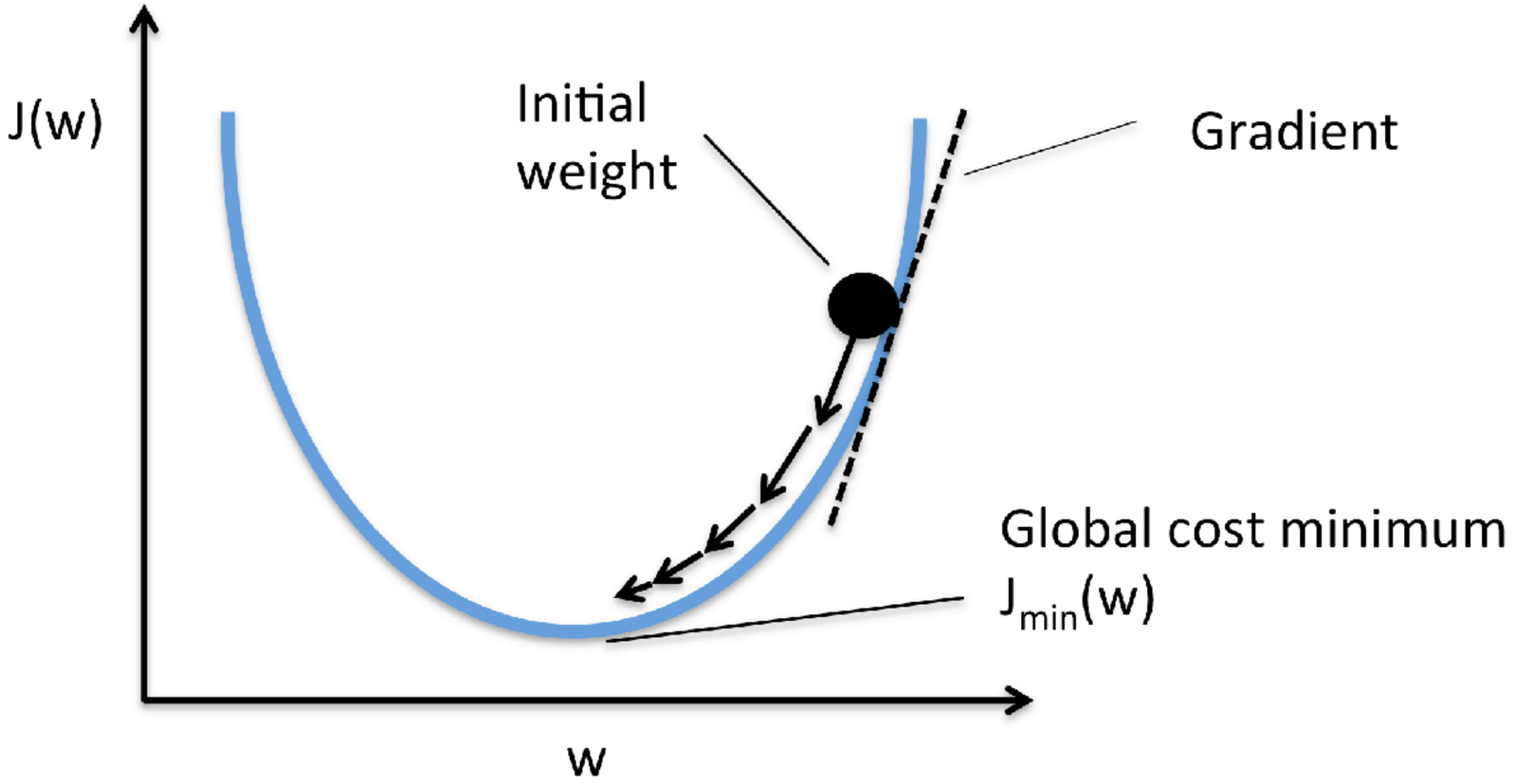


Alec Radford's animations for optimization algorithms
https://goo.gl/aHDhcT

The list of optimization techniques

Stochastic Gradient Descent(SGD)

가장 기본적인 Gradient Descent 알고리즘
앞서 설명한 단점을 전부 포함하고 있다



공식

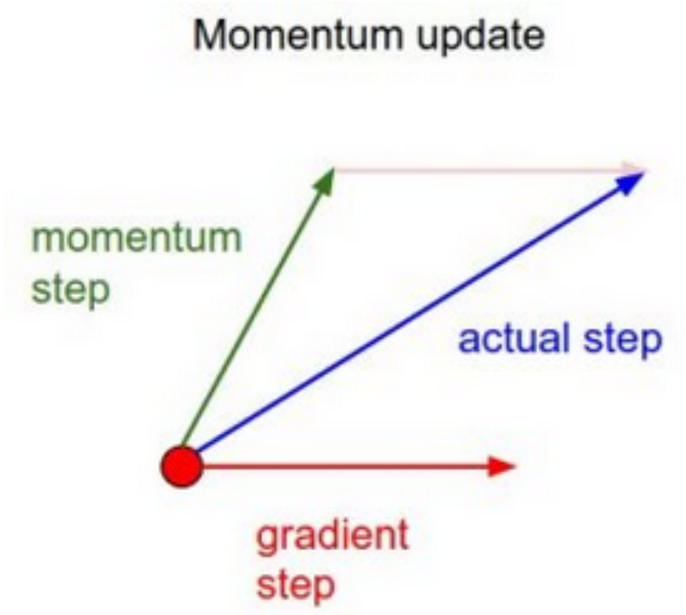
$$\theta = \theta - \alpha * \frac{dJ(\theta)}{d\theta}$$

코드

```
dw1 = X.T.dot(d1)
w1 = w1 - learning_rate * dw1
```

Momentum

Gradient Descent에 가속도(Momentum)를 붙인다
이렇게 하면 1) 극소점에 빠르게 다다를 수 있고, 2) local minima나 plateau를 뛰어넘을 수 있다



Before

$$\theta = \theta - \alpha * \frac{dJ(\theta)}{d\theta}$$

```
dw1 = X.T.dot(d1)
w1 = w1 - learning_rate * dw1
```

After

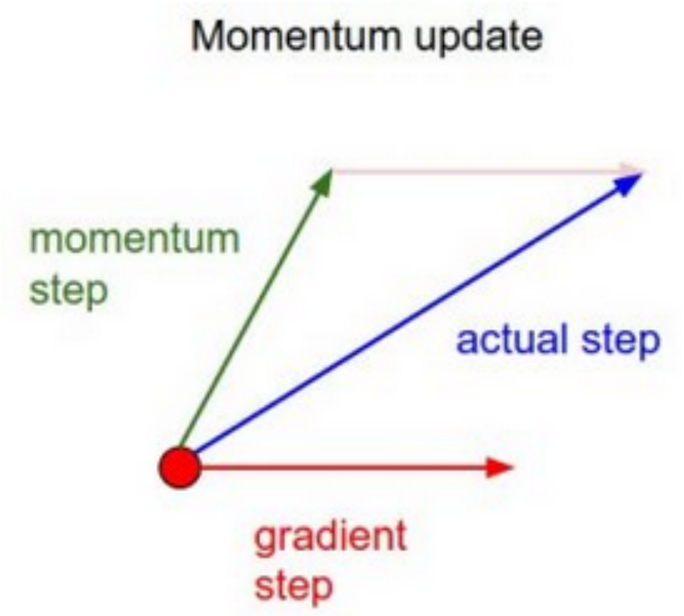
$$vdw = \gamma * vdw + \nu * \frac{dJ(\theta)}{d\theta}$$

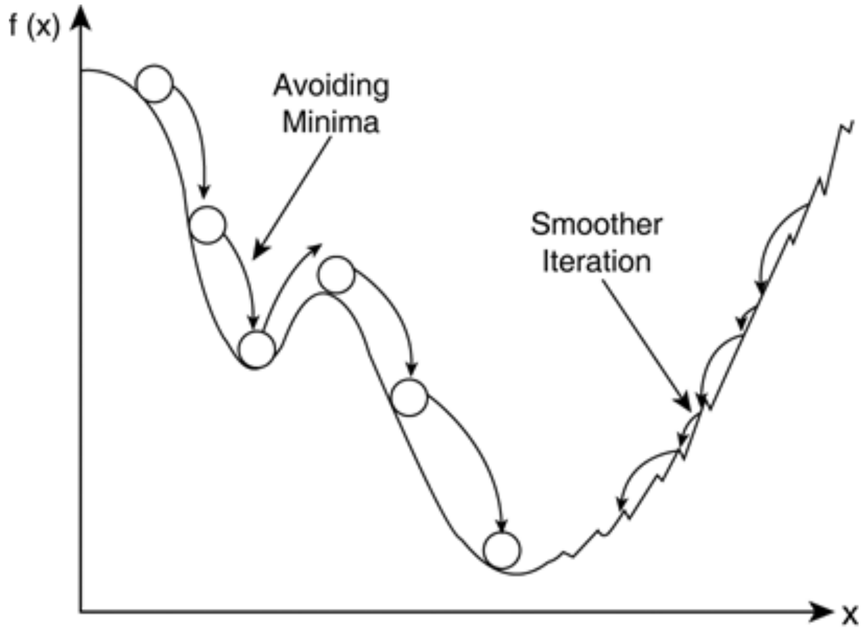
```
dw1 = mu * dw1 - learning_rate * X.T.dot(d1)
w1 = w1 + dw1
```

$$\theta = \theta - \alpha * vdw$$

Momentum

Gradient Descent에 가속도(Momentum)를 붙인다
이렇게 하면 1) 극소점에 빠르게 다다를 수 있고, 2) local minima나 plateau를 뛰어넘을 수 있다

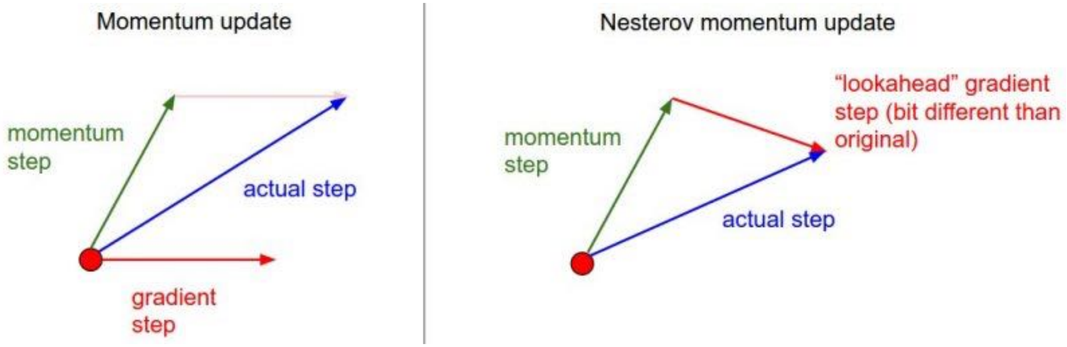




weight update에 가속도가 붙은 상태이기 때문에
local minima를 뛰어넘을 수 있다.

Nesterov Momentum

Momentum을 계산할 때 가속도를 먼저 붙인 뒤 gradient를 계산한다
Momentum이 global optima를 매 번 추월한다면, Nesterov Momentum이 더 효과적이다



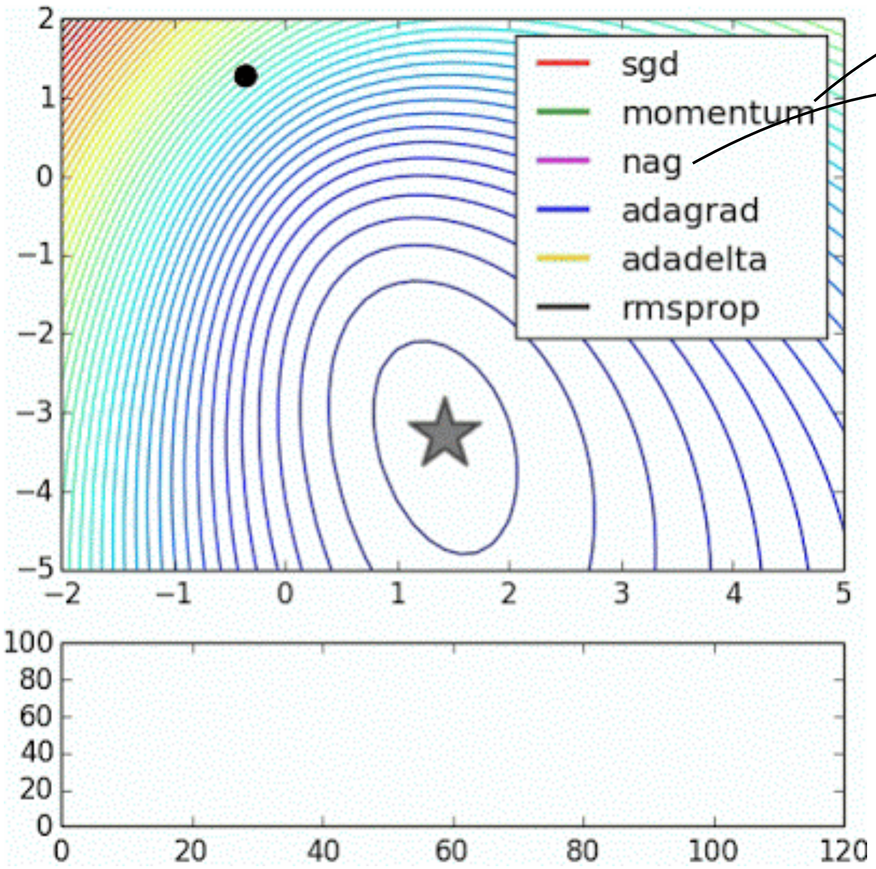
Before(Momentum)

```
dw1 = mu * dw1 - learning_rate * X.T.dot(d1)
w1 = w1 + dw1
```

After(Nesterov Momentum)

```
dw1_prev = dw1
dw1 = mu * dw1 - learning_rate * X.T.dot(d1)
w1 = w1 + (-mu * dw1_prev) + (1 + mu) * dw1
```

Momentum + Nesterov Momentum



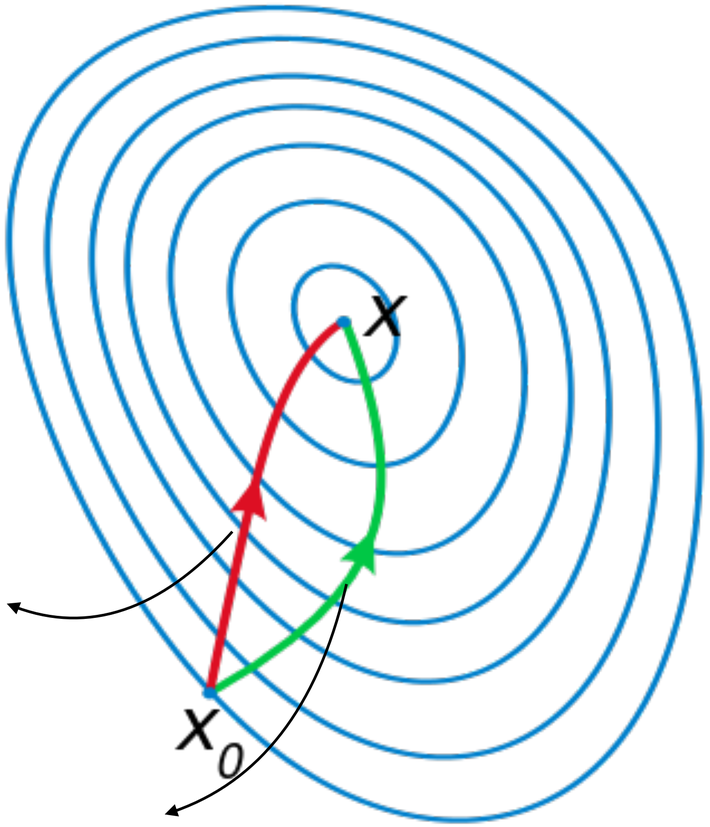
Momentum

Nesterov Accelerated Gradient(NAG)
= Nesterov Momentum

Alec Radford's animations for optimization algorithms
https://goo.gl/aHDhcT

AdaGrad

모든 weight update량의 제곱을 누적한 뒤 나눠준다
weight update가 휘거나 zigzag로 업데이트되는 현상을 막을 수 있다



After
(AdaGrad)

Before
(SGD, Momentum, Nesterov Momentum)

Before(SGD)

```
dw1 = X.T.dot(d1)
w1 = w1 - learning_rate * dw1
```

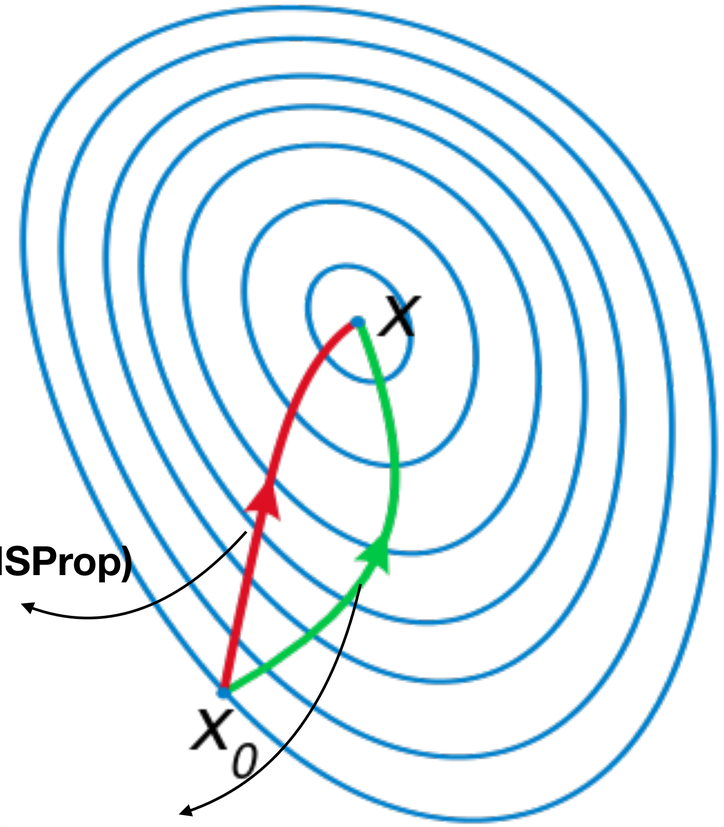
After(AdaGrad)

```
dw1 = X.T.dot(d1)
cache_dw1 = dw1 ** 2
w1 = w1 - learning_rate * dw1 / np.sqrt(cache_dw1 + 0.0000001)
```

Adaptive Subgradient Methods for Online Learning and Stochastic Optimization
Duchi et al., 2011. https://goo.gl/WWqubD

RMSProp

누적량을 decay 옵션으로 조절한다
Adagrad와 유사하지만, weight update가 누적되도 속도가 크게 줄어들지 않는다



After
(AdaGrad, RMSProp)

Before
(SGD, Momentum, Nesterov Momentum)

Before(AdaGrad)

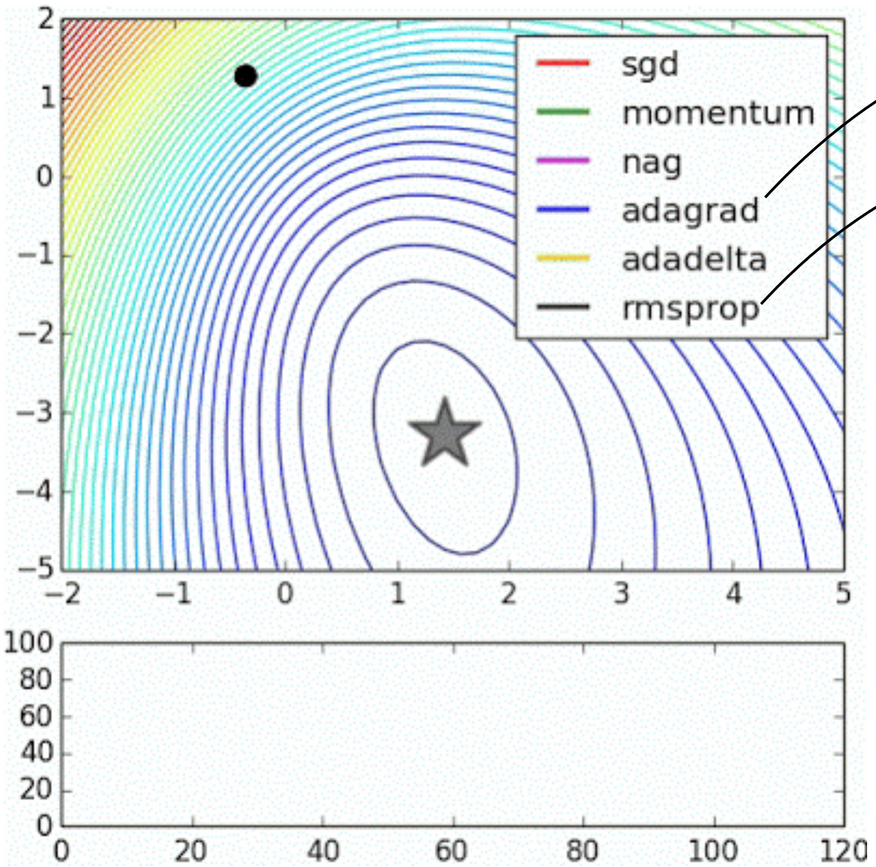
```
dw1 = X.T.dot(d1)
cache_dw1 = dw1 ** 2
w1 = w1 - learning_rate * dw1 / np.sqrt(cache_dw1 + 0.0000001)
```

After(RMSProp)

```
dw1 = X.T.dot(d1)
cache_dw1 = decay * cache_dw1 + (1 - decay) * dw1 ** 2
w1 = w1 - learning_rate * dw1 / np.sqrt(cache_dw1 + 0.0000001)
```

rmsprop: Divide the gradient by a running average of its recent magnitude
Tieleman and Hinton, 2012. https://goo.gl/CJ7HyJ

Adagrad + RMSProp



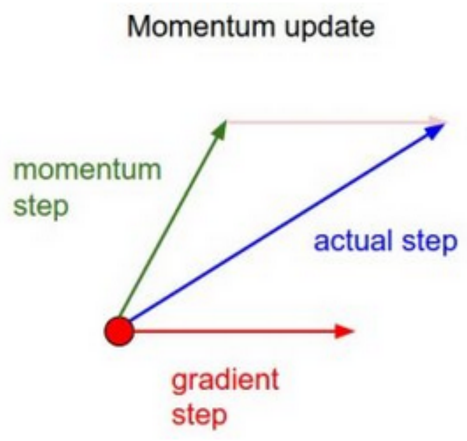
adagrad

rmsprop

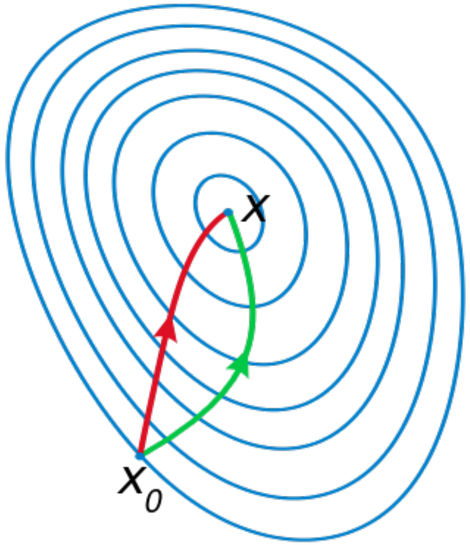
Alec Radford's animations for optimization algorithms
https://goo.gl/aHDhcT

Adam

RMSProp과 Momentum을 섞어서 사용한다
양 쪽의 장점을 모두 보유하고 있다. 가장 보편적으로 쓰이는 weight update 방식



+



Before(RMSProp)

```
dw1 = X.T.dot(d1)
cache_dw1 = decay * cache_dw1 + (1 - decay) * dw1 ** 2
w1 = w1 - learning_rate * dw1 / np.sqrt(cache_dw1 + 0.0000001)
```

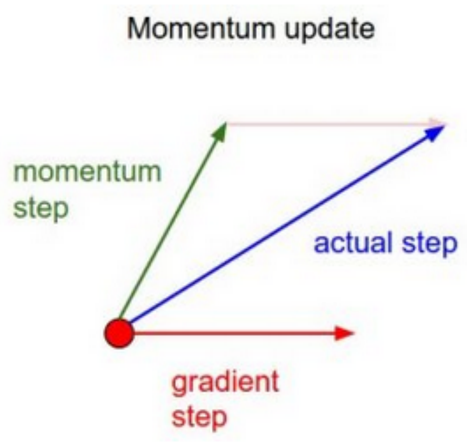
After(Adam)

```
dw1 = X.T.dot(d1)
dw1m = beta1 * dw1m + (1 - beta1) * dw1
dw1v = beta2 * dw1v + (1 - beta2) * (dw1 ** 2)
w1 = w1 - learning_rate * dw1m / np.sqrt(dw1v + 0.0000001)
```

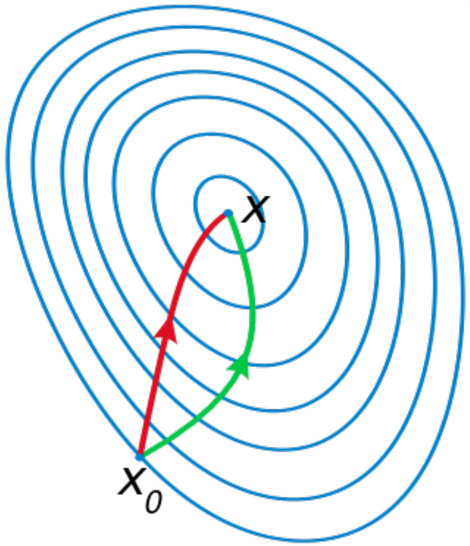
Adam: A Method for Stochastic Optimization
Kingma and Ba, 2014. https://goo.gl/Dct3n6

Adam

adam을 사용할 경우 초반에 weight update가 느려지는 단점이 있기 때문에
warm start 라는 방식을 활용하여 초반부에 weight update량을 강제로 늘려주기도 한다



+



Before(Adam)

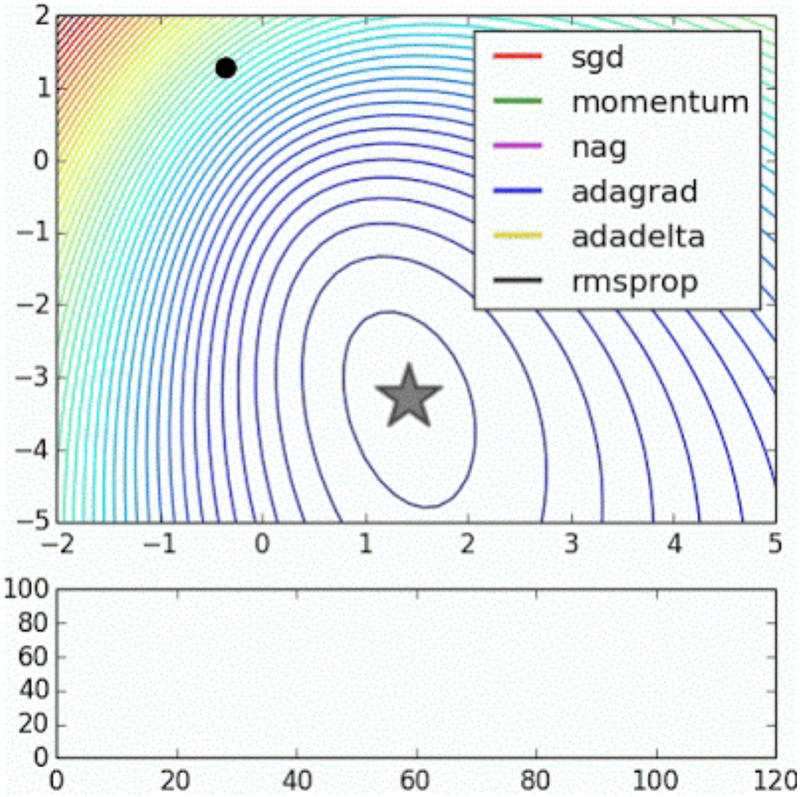
```
dw1 = X.T.dot(d1)
dw1m = beta1 * dw1m + (1 - beta1) * dw1
dw1v = beta2 * dw1v + (1 - beta2) * (dw1 ** 2)
w1 = w1 - learning_rate * dw1m / np.sqrt(dw1v + 0.0000001)
```

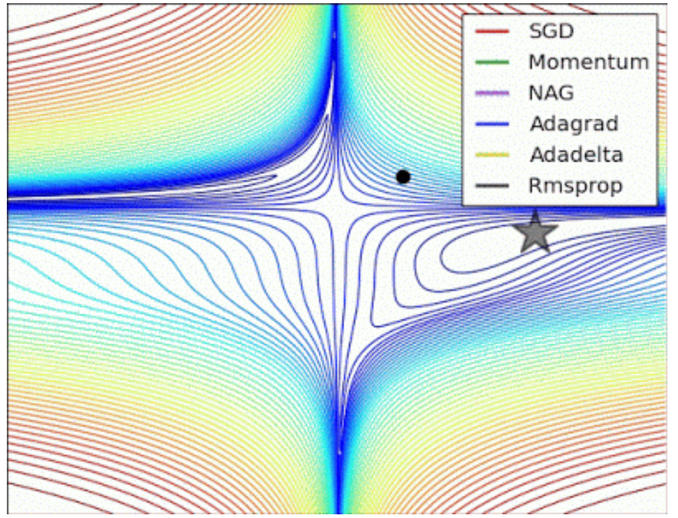
After(Adam w/ Warm Start)

```
dw1 = X.T.dot(d1)
dw1m = beta1 * dw1m + (1 - beta1) * dw1
dw1v = beta2 * dw1v + (1 - beta2) * (dw1 ** 2)
dw1mb = dw1m / (1 - beta1 ** (1 + epoch))
dw1vb = dw1v / (1 - beta2 ** (1 + epoch))
w1 = w1 - learning_rate * dw1mb / np.sqrt(dw1vb + 0.0000001)
```

Adam: A Method for Stochastic Optimization
Kingma and Ba, 2014. https://goo.gl/Dct3n6

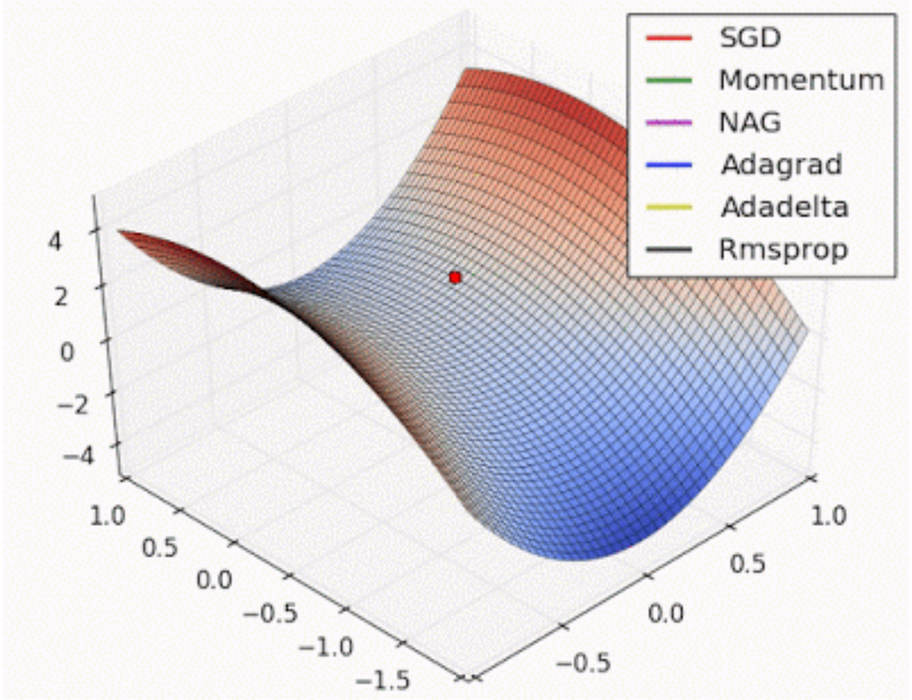
Comparison of optimization algorithms

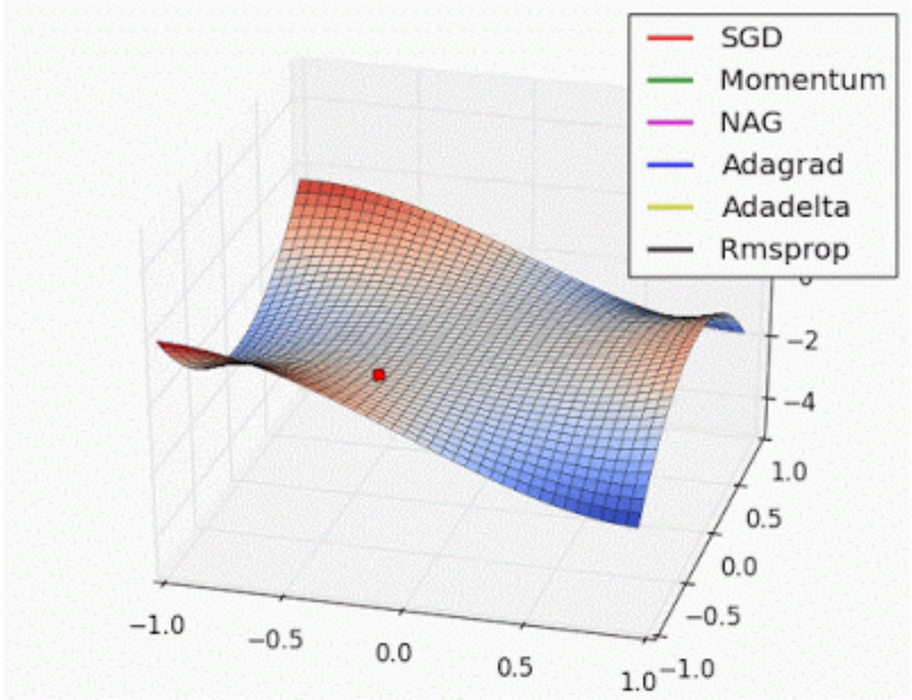




Alec Radford's animations for optimization algorithms
https://goo.gl/aHDhcT

Comparison of optimization algorithms

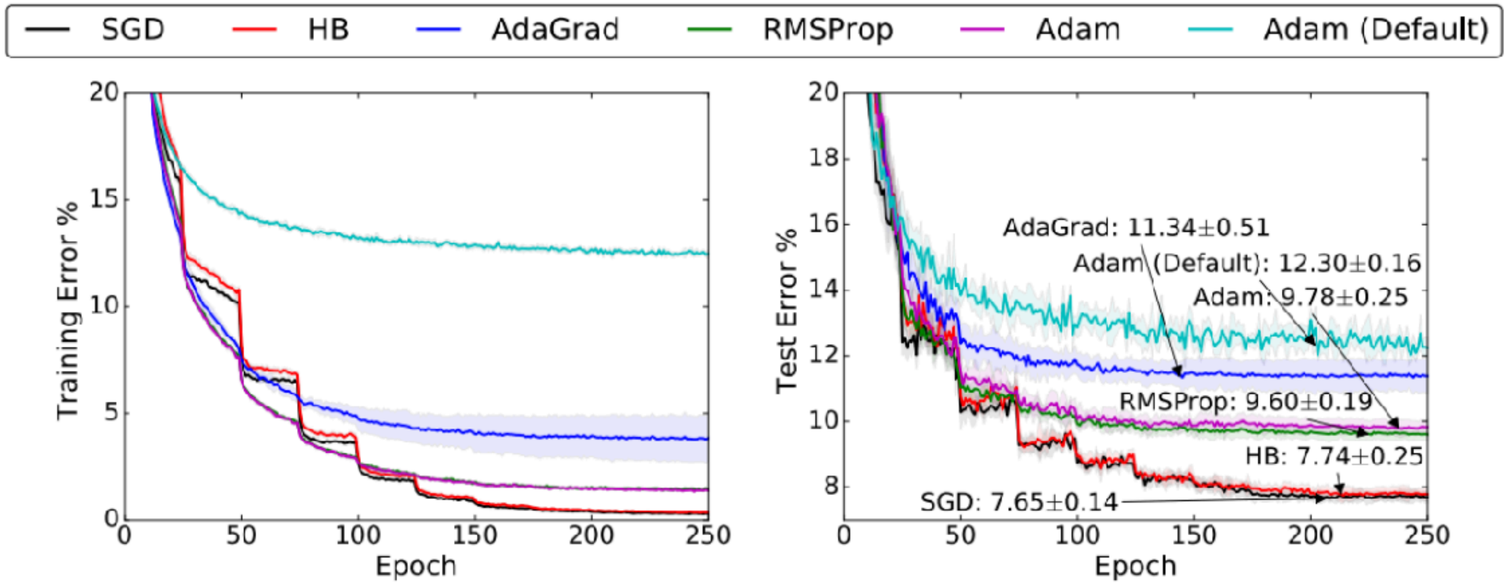




Alec Radford's animations for optimization algorithms
https://goo.gl/aHDhcT

Comparison of optimization algorithms

예전에는 adam을 사용하는 것을 권장하였지만, 최근에 adaptive method가 성능이 좋지 않다는 의견이 나오고 있다
그러므로 momentum + nesterov momentum을 사용하는 것을 권장



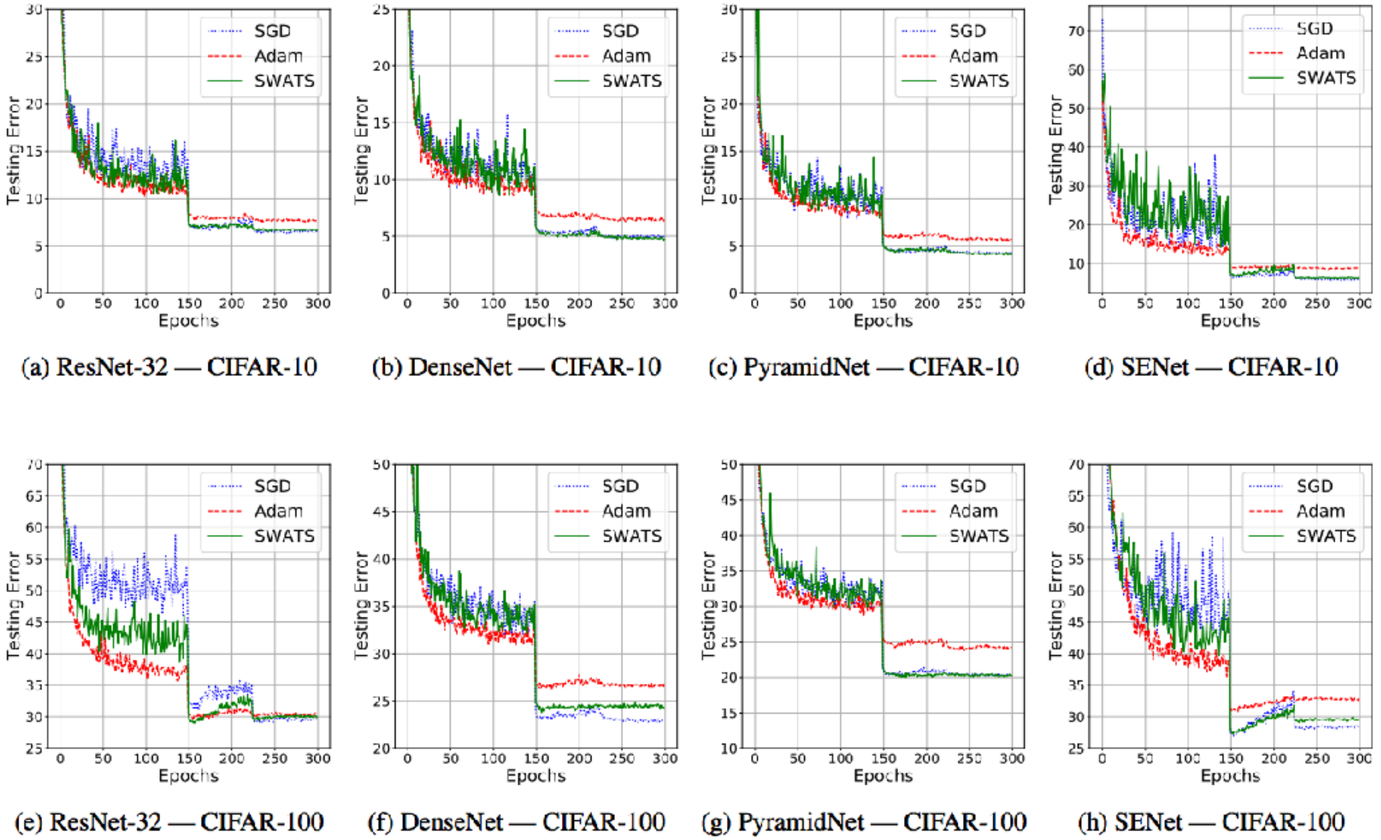
(a) CIFAR-10 (Train)

(b) CIFAR-10 (Test)

The Marginal Value of Adaptive Gradient Methods in Machine Learning
Wilson et al, 2017. https://goo.gl/5Se2jg

SWATS

초반에는 adam으로 학습하다가 후반부에 자연스럽게 SGD로 switch하는 전략
아직 실험적인 알고리즘이며, 실제 성능도 SGD와 큰 차이가 없는 것으로 보인다



(a) ResNet-32 — CIFAR-10 (b) DenseNet — CIFAR-10 (c) PyramidNet — CIFAR-10 (d) SENet — CIFAR-10

(e) ResNet-32 — CIFAR-100 (f) DenseNet — CIFAR-100 (g) PyramidNet — CIFAR-100 (h) SENet — CIFAR-100

Improving Generalization Performance by Switching from Adam to SGD
Keskar and Socher, 2017. https://goo.gl/zb1FNf

TL; DR

Optimization 알고리즘은 크게 두 가지 방향으로 발전되어가고 있다.

- **가속도를 준다.** 이를 통해 plateau와 local minima를 뛰어넘어 global minima에 도달할 수 있다. (=**Momentum**, **Nesterov Momentum**)
- **weight마다 update 변화량을 다르게 준다.** 이를 통해 zigzag로 업데이트되는 현상을 막을 수 있다. (=**Adam**, **RMSProp**)

가장 많이 쓰이는건 adam이었으나, 최근에는 adaptive method 가 성능이 안 좋다는 것이 증명되었다.
그러므로 **SGD + Momentum(or Nesterov Momentum)을 기본으로 하자.**

Q & A