Loss functions,
Optimizers

한양대학교 AILAB 석사과정 엄희송

Index

1. Meaning of loss function

Backpropagation 의 기본 개념은 정답과 현재 예측 값의 차이를 통해 기울기를 계산해 network 의 학습에 사용하는 것.

Loss function 은 그 차이를 어떻게 계산할 것인지 결정하는 함수.

의미로는 model 이 현재 실제 값에 비해 얼마나 잘 예측하는지를 수치화한 것으로 생각할 수 있음.

Ex) 실제 값: 500 / 모델이 예측한 값 : 350 … Loss ? 500 - 350 = 150 으로 생각할 수 있음.

2. Common loss functions

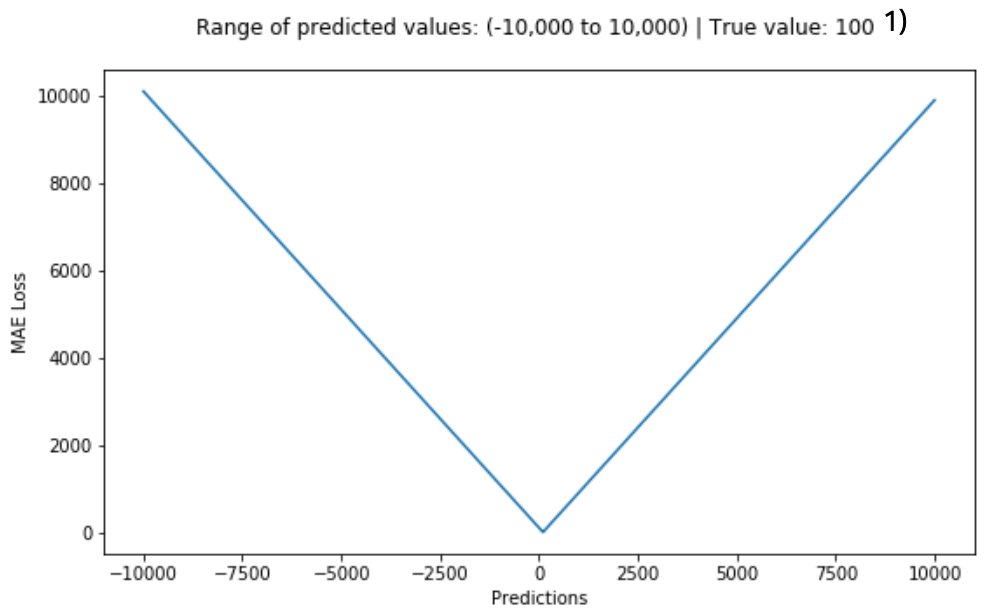
Mean Absolute Error (MAE)

L1 loss

실제 값에서 예측 값을 뺀 것의
절대값의 평균

Dataset 에 존재하는 outlier 에 대해 robust
minima 근처에서 수렴이 어렵다

$$MAE = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}$$



1)

1), 2) 5 Regression Loss Functions All Machine Learners Should Know, https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0

2. Common loss functions

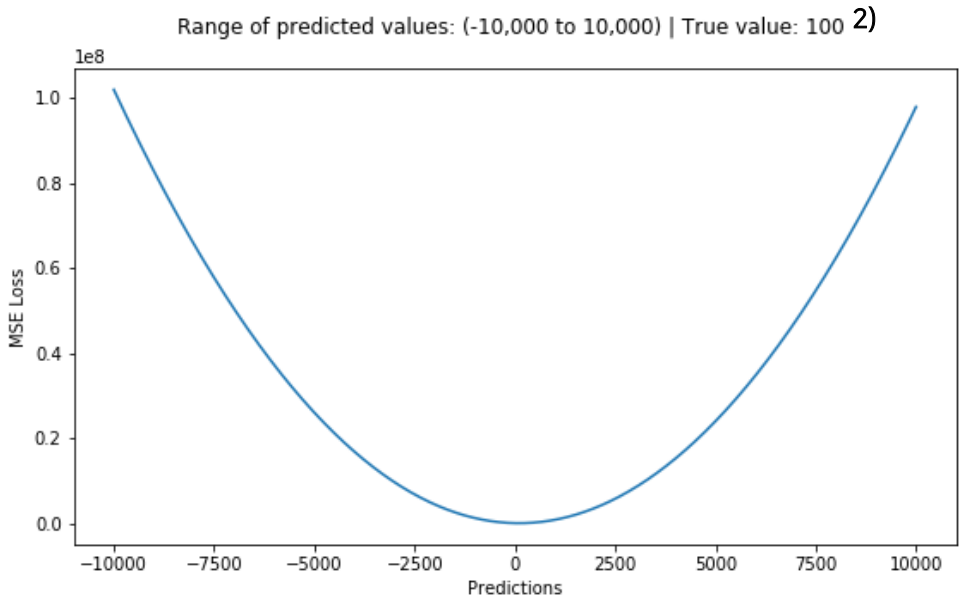
Mean Square Error (MAE)

Quadratic loss, L2 loss

실제 값에서 예측 값을 뺀 것의
제곱의 평균

학습 속도가 더 빠르고 수렴이 안정적
outlier 에 민감

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



2)

1), 2) 5 Regression Loss Functions All Machine Learners Should Know, https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0

Cross entropy error

두 확률 분포 사이의 차이가 얼마나 되는지를
수치화 (~ KL divergence)

$$H(P,Q) = -\sum P(x)\log Q(x)$$

$$KL(p|q) := -\sum_{i=1}^{N} p_i \log q_i - \left(-\sum_{i=1}^{N} p_i \log p_i\right) = -\sum_{i=1}^{N} p_i \log\left(\frac{q_i}{p_i}\right).$$

$$L = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

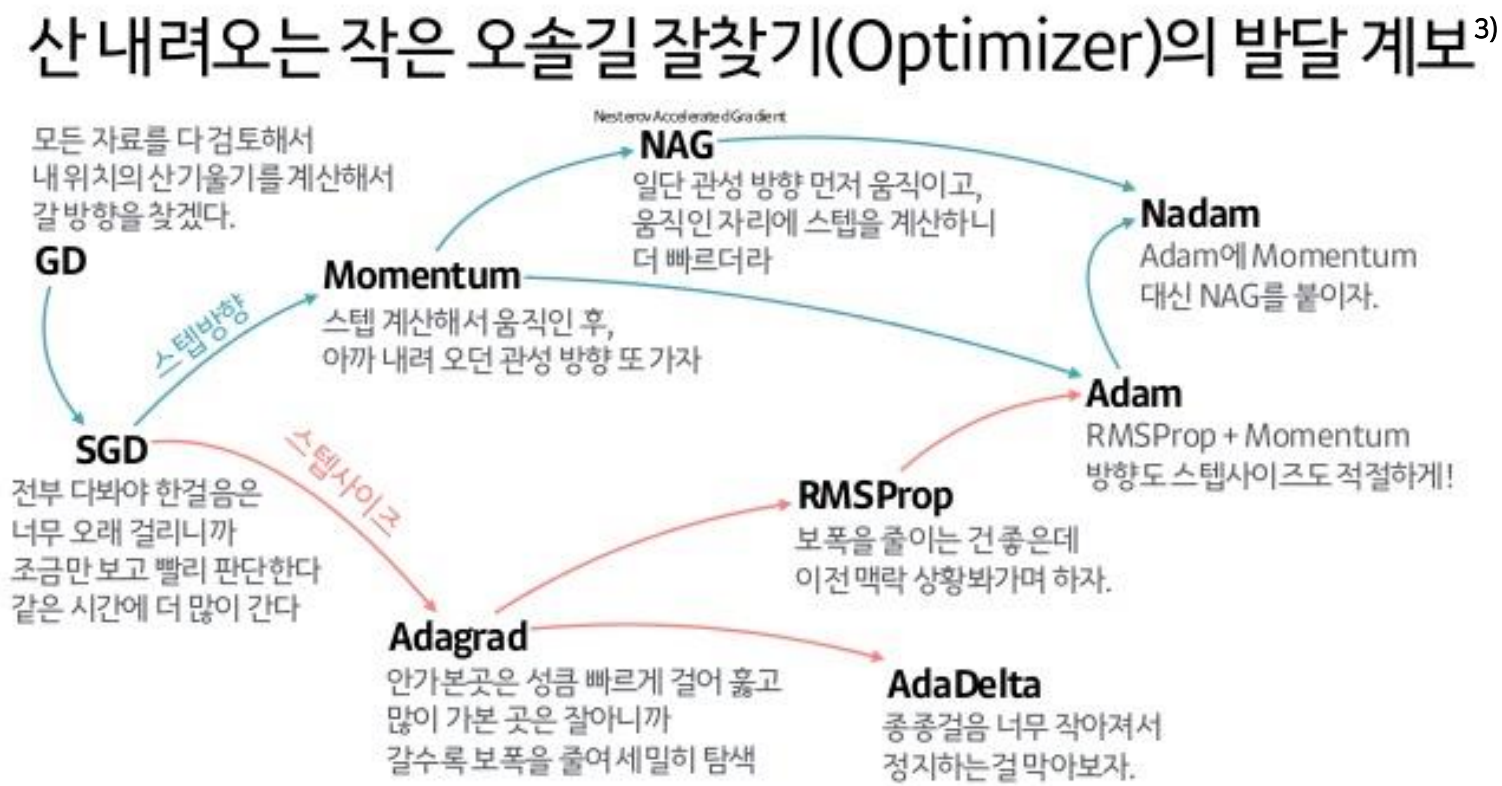
$$\hat{Y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} \infty \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0, \quad cost=0$$

$$\hat{Y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \infty, \quad cost=\infty$$

3)

3) Softmax classifier 의 cost 함수, https://pythonkim.tistory.com/20

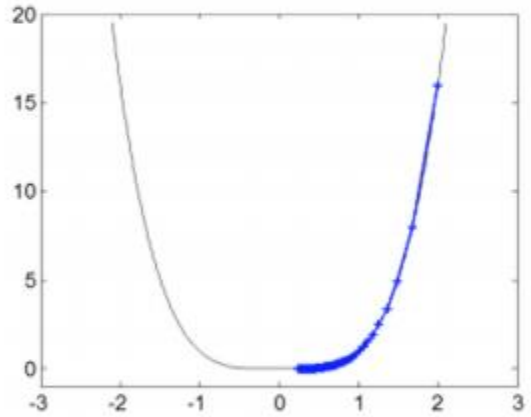
Loss function 을 통해 구한 차이를 사용해 기울기를 구하고
network의 parameter 의 학습에 어떻게 반영할 것인지를 결정하는 방법



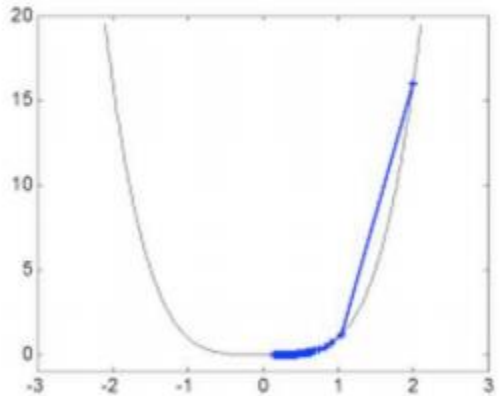
3) Optimizer의 발달 계보. 49p, https://www.slideshare.net/yongho/ss-79607172.

Gradient Descent – 1회의 학습 step 시에, 현재 모델(~parameter)의 모든 data에 대해서 예측 값에 대한 loss 의 미분을 learning rate 만큼 보정해서 반영한다.

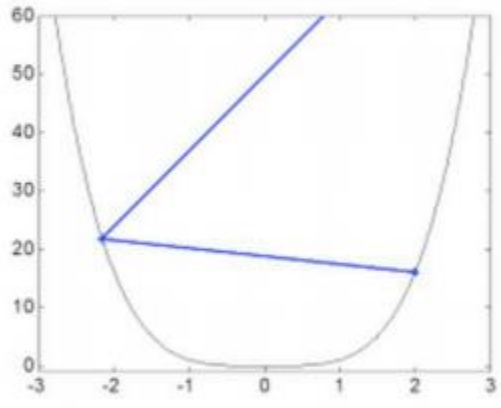
$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta)$$



$\eta = 0.01$



$\eta = 0.03$

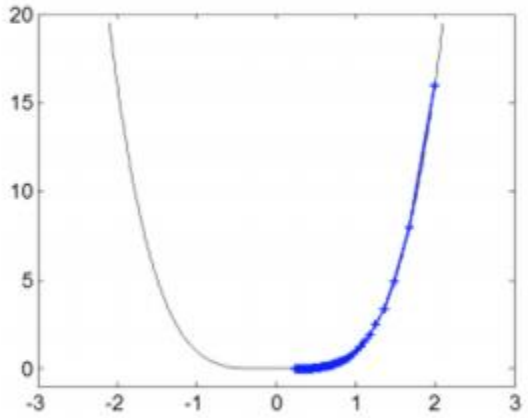


$\eta = 0.13$

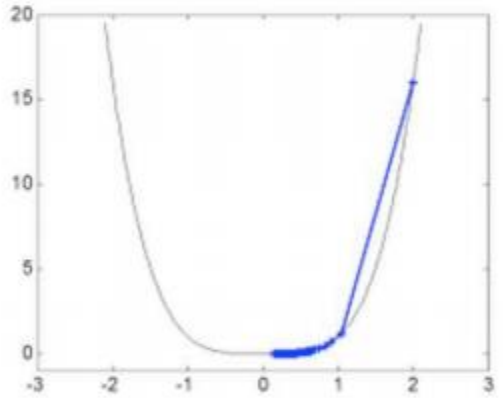
Stochastic Gradient Descent – 1 step 마다 모든 data 의 계산이 너무 느리므로,
적은 수의 data sample 이 전체 set 의 gradient 와 유사할 것이라고 가정하여 계산에 사용함.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta)$$

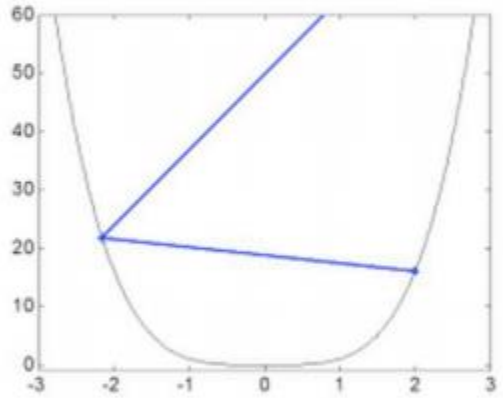
$\theta; x^{(i)}; y^{(i)}$



$\eta = 0.01$



$\eta = 0.03$

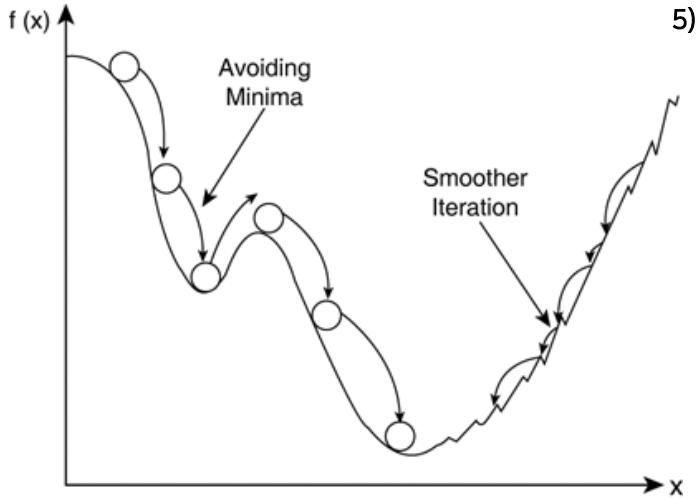


$\eta = 0.13$

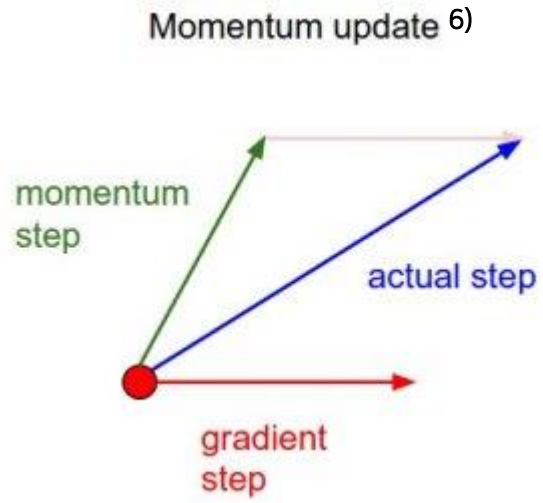
Momentum – 이전 step 의 방향(= 관성)과 현재 상태의 gradient 를 더해 현재 학습할 방향과 크기를 정함.
local minima 를 빠져 나올 수 있음.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

$$\theta = \theta - v_t$$



5)

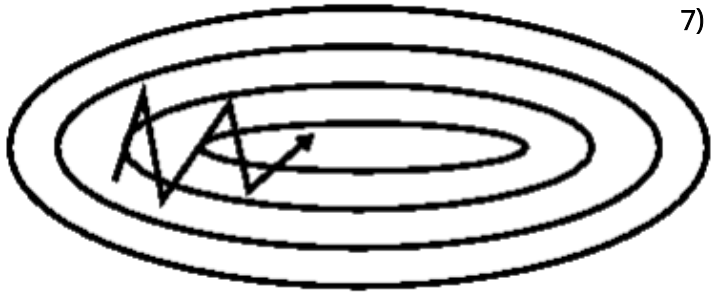


6)

5) Artificial Neural Network. https://sites.google.com/a/case.edu/hpc-upgraded-cluster/home/important-notes-for-new-users/helpful-references/deep-learning/artificial-neural-network
6) Difference between Momentum and NAG. CS231.

Momentum – 이전 step 의 방향(= 관성)과 현재 상태의 gradient 를 더해 현재 학습할 방향과 크기를 정함.
local minima 를 빠져 나올 수 있음.

- Momentum 방식은 SGD가 Oscilation 현상을 겪는 상황을 해결.
- SGD는 한번의 step에서 움직일 수 있는 step size는 한계가 있으므로 이러한 oscilation 현상이 일어날 때는 좌우로 계속 진동하면서 이동에 난항을 겪음.



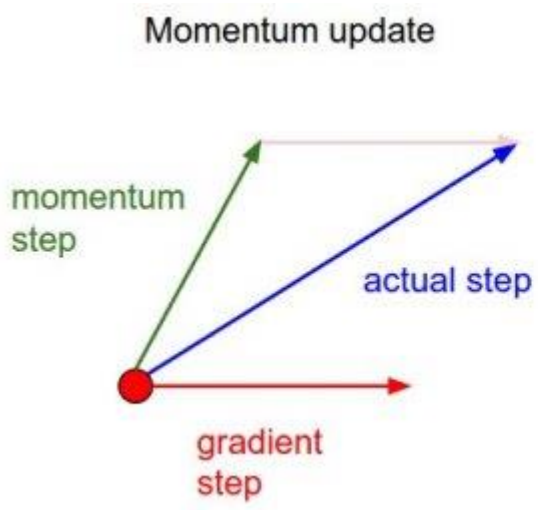
7)

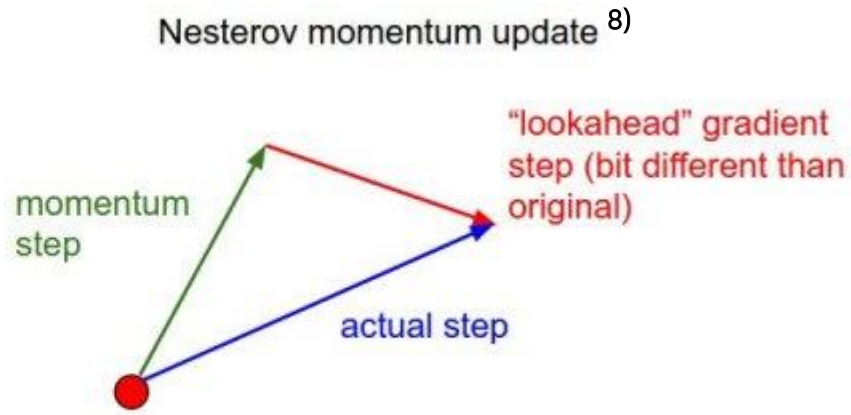
7) An overview of gradient descent optimization algorithms. http://ruder.io/optimizing-gradient-descent

Nesterov Accelerated Gradient (NAG) – Momentum 과 비슷한 개념, 이전까지의 moment step 과 moment step 만큼 이동한 위치에서의 gradient 를 더함.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$





8) Difference between Momentum and NAG. CS231.

Adagrad (Adaptive Gradient) – parameter 별로 gradient 를 다르게 주는 방식.
Network 의 parameter 개수가 k 일 때, G_t 는 time step t까지 각 변수가 이동한 gradient의 sum of squares 를 저장하는 k x k dimension diagonal matrix. 제곱 및 · 연산은 element-wise 로 계산.

많이 변화한 변수들은 G에 저장된 값이 커지기 때문에 step size 가 작은 상태로, 적게 변화한 변수들은 상대적으로 step size 가 큰 상태로 학습에 반영됨.

Step size decay 를 생각하지 않아도 되지만, 학습이 오래 진행되는 경우 G값이 너무 커져서 움직이지 않는 경우도 생김.

$$G_t = G_{t-1} + (\nabla_\theta J(\theta_t))^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(G_t+\epsilon)}} \cdot \nabla_\theta J(\theta_t)$$

RMSProp – 학습이 오래 진행되면 step size가 너무 작아지는 adagrad 의 단점을 보완하기 위한 방법.
각 변수에 대한 gradient 의 제곱을 계속 더하는 것이 아니라, 지수평균으로 바꾸어 G값이 무한정 커지지 않도록 방지하면서 변화량의 상대적인 크기 차이를 유지하도록 함.
Exponential decaying learning rate.

$$G = \gamma G + (1-\gamma)(\nabla_\theta J(\theta_t))^2$$

$$\theta = \theta - \frac{\eta}{\sqrt{G+\epsilon}} \cdot \nabla_\theta J(\theta_t)$$

AdaDelta (Adaptive Delta) – RMSProp 과 유사한 점은 G에 저장할 때 지수평균을 사용.
차이점은 parameter update 시에 수학적 증명[8)]을 통해 learning rate 를 제거함.

$$G = \gamma G + (1-\gamma)(\nabla_\theta J(\theta_t))^2$$

$$\Delta_\theta = \frac{\sqrt{s+\epsilon}}{\sqrt{G+\epsilon}} \cdot \nabla_\theta J(\theta_t)$$

$$\theta = \theta - \Delta_\theta$$

$$s = \gamma s + (1-\gamma)\Delta_\theta^2$$

8) An overview of gradient descent optimization algorithms. http://ruder.io/optimizing-gradient-descent

Adam (Adaptive Moment Estimation) , Nadam
Adam의 경우 Momentum 방식과 유사하게 지금까지 계산해온 기울기의 지수평균을 저장하며,
RMSProp과 유사하게 기울기의 제곱값의 지수평균을 저장한다.
학습에 초반부에 m과 v가 0에 가깝게 bias 되어 있을 것이라고 판단해 unbiased 작업을 거친 후에 계산을 진행.
Nadam 의 경우 m 의 식을 NAG 로 변경해 사용.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta J(\theta))^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t$$

Reference

5 Regression Loss Functions All Machine Learners Should Know
https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0

자습해도 모르겠던 딥 러닝, 머리속에 인스톨 시켜드립니다.
https://www.slideshare.net/yongho/ss-79607172

Gradient Descent Optimization Algorithms 정리
http://shuuki4.github.io/deep%20learning/2016/05/20/Gradient-Descent-Algorithm-Overview.html

An overview of gradient descent optimization algorithms
http://ruder.io/optimizing-gradient-descent/index.html