Gradient Descent Algorithm

and few more optimization…

Prof. Je-Won Kang
Electronic & Electrical Engineering
Ewha Womans University

Overview

- We have some function (loss function)

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

and want

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$ ← object 함수 최소화 하는 θ_1, θ_2 원하는.

Algorithm outline:

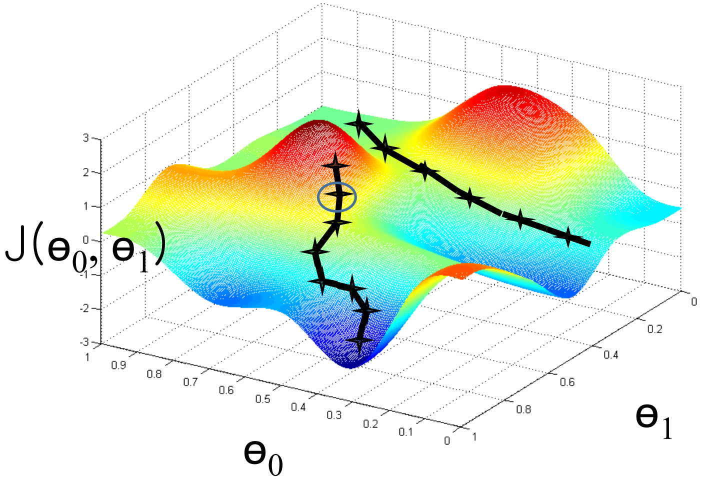
- Start with some initial parameters θ_0, θ_1
- Keep changing the parameter to reduce the loss function until we hopefully end up at a minimum.

Gradient Descent Algorithm

Key components

- **Gradient**: the derivative of vector functions (partial derivative along each dimension)
  - Direction of greatest increase (or decrease) of a function
- **The step size α** 변화정도를 설정 affects the rate at which the weight vector moves down the error surface and must be a positive number. (**hyper parameter**)
- θ is the **learnable parameters**
- The function J is **the objective function** that we want to minimize.

$$
\text{repeat until convergence } \{
$$
$$
\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)
$$
$$
\}
$$



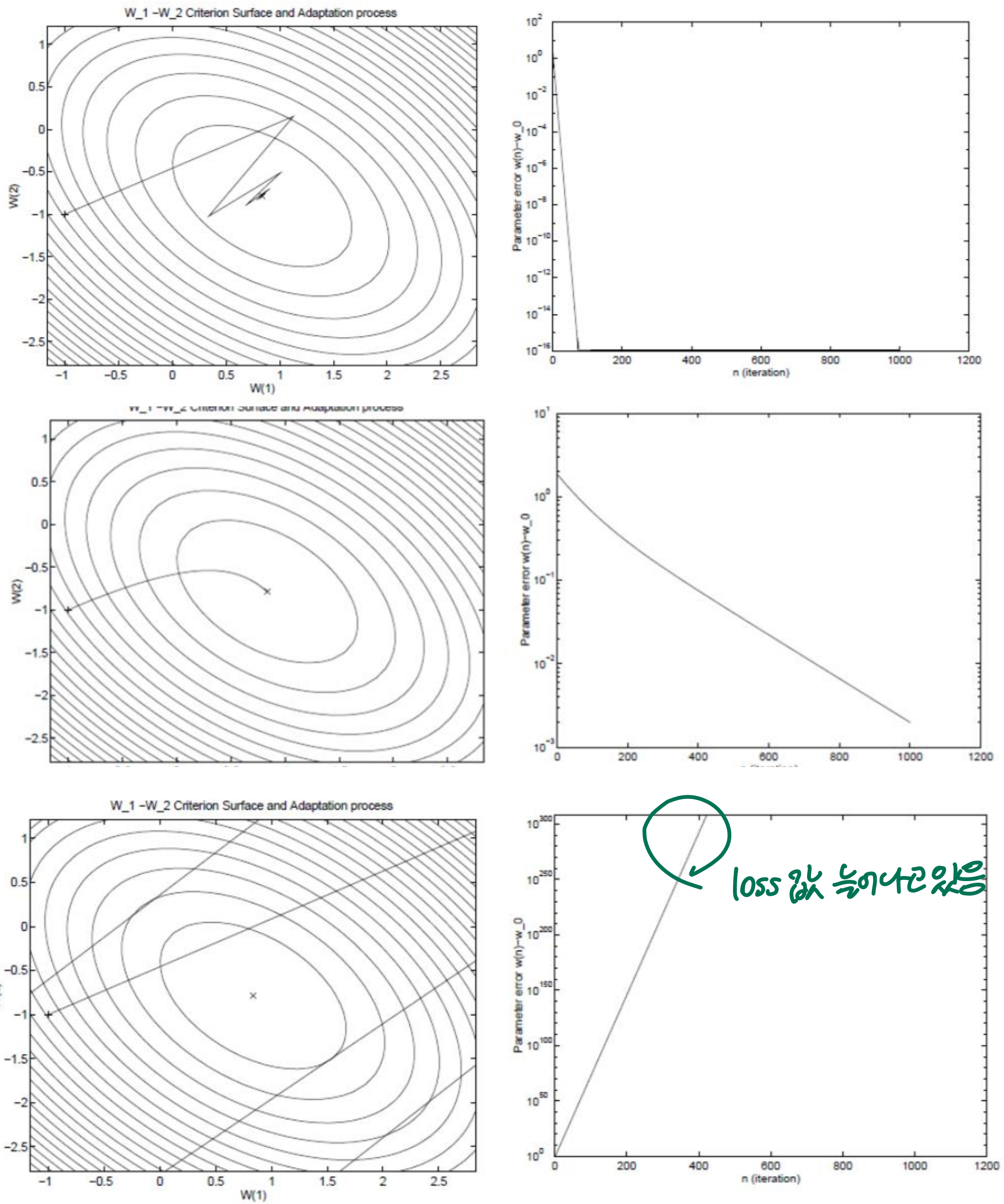
- If α is too small, gradient descent can be slow.

- If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

중간값 →

α 작을때 →
수렴속도 느림
but 안정적

α 큼 →
최소지점 찾기어렵



loss 값 늘어나고있음

Image from Adaptive Filter Theory, by Simon O. Haykin

Batch gradient descent

θ_0 1번 업데이트에 전체 샘플 m개 모두 고려

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

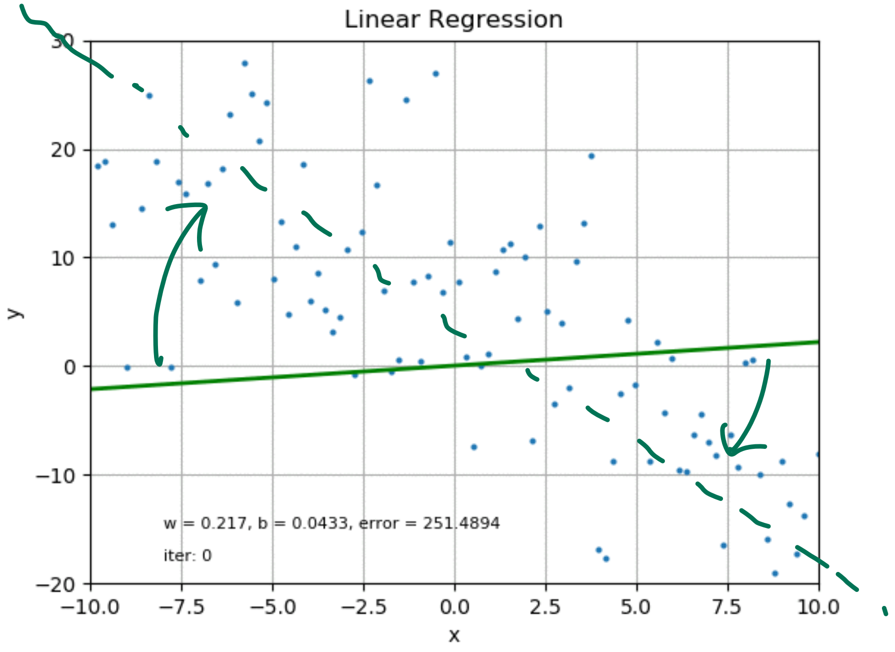
}

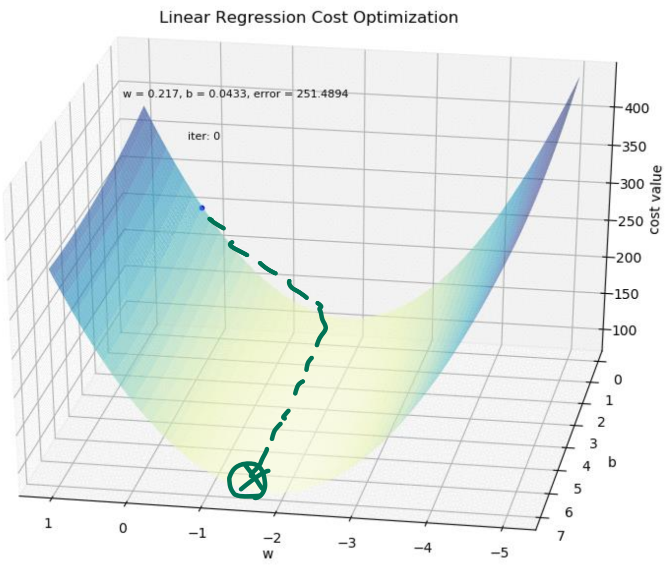
Linear regression model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

What if the number of sample sizes m is increasing?





Images from https://medium.com/swlh/from-animation-to-intuition-linear-regression-and-logistic-regression-f641a31e1caf

해결, but noise 영향 받기 쉬운 단점
방법으로

Stochastic gradient descent (SGD)

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

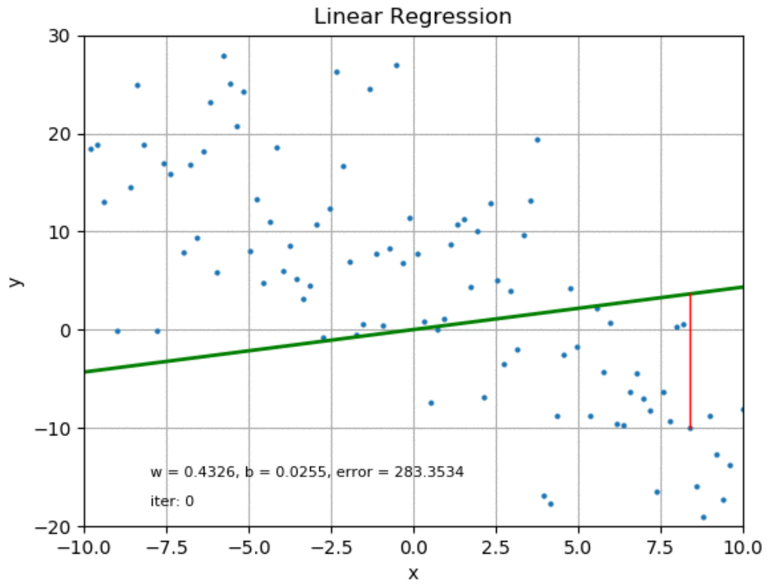
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

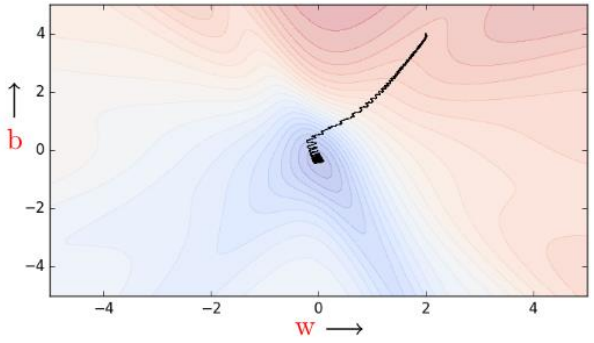
}

SGD : m=1

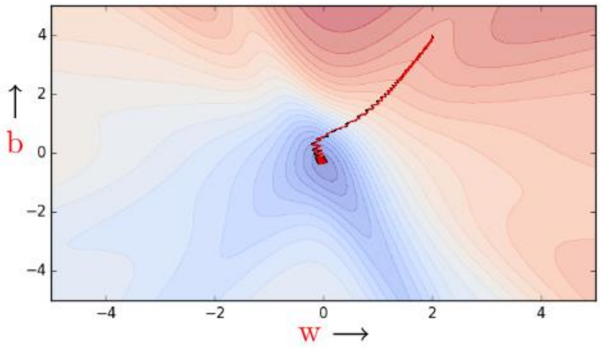
1< m < N : mini-batch SGD

VS SGD : alleviate randomness
VS GD : less time in converging





Many oscillations with greedy decision making

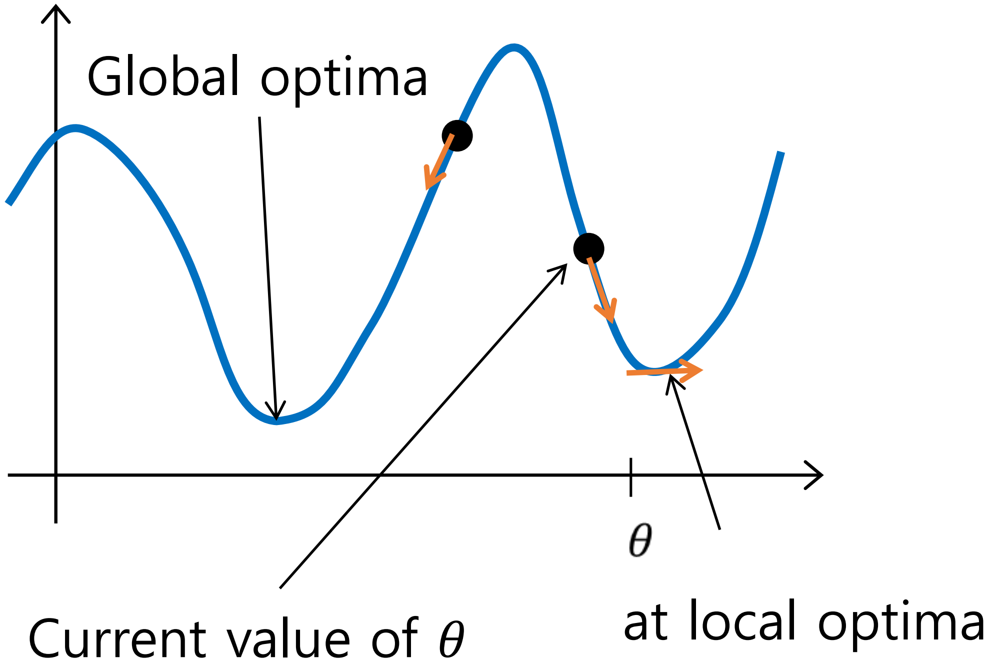


Less oscillations can be seen for m=2

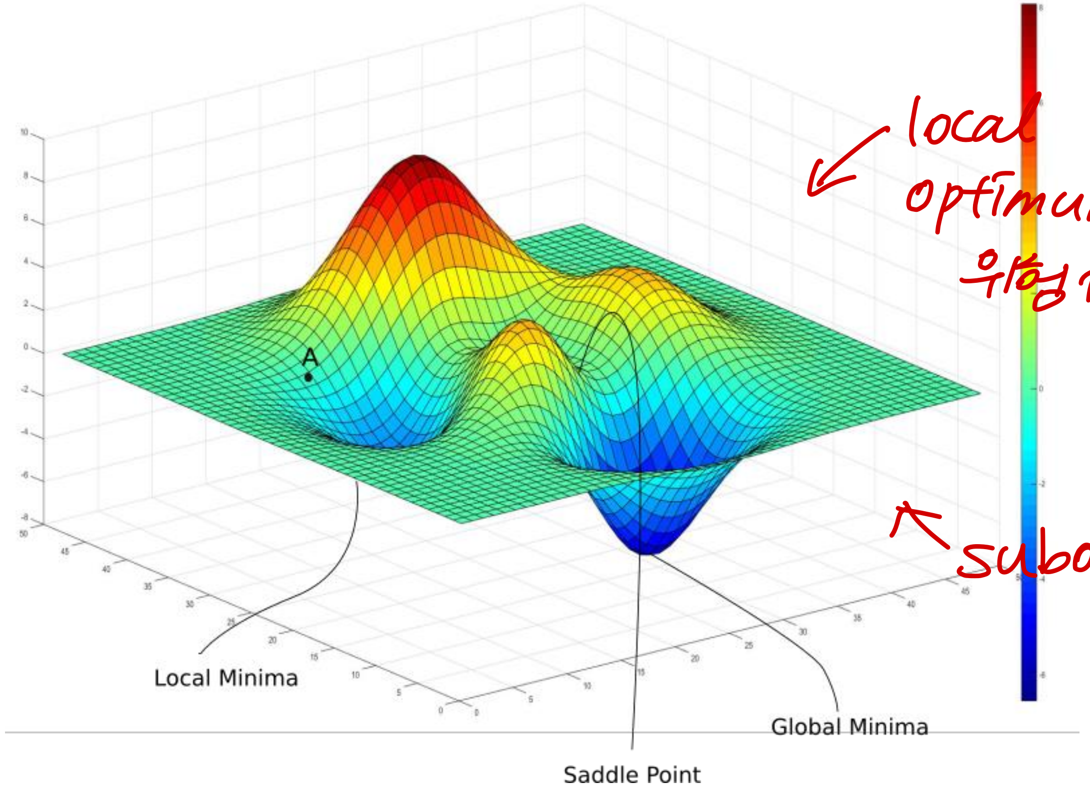
Images from https://towardsdatascience.com/why-visualize-gradient-descent-optimization-algorithms-a393806eee2
Images from https://medium.com/swlh/from-animation-to-intuition-linear-regression-and-logistic-regression-f641a31e1caf

Limitation : Local Optimum

$$\theta_{new} \leftarrow \theta_{old} - \alpha \frac{\partial}{\partial\theta} J(\theta)$$



Critical points with zero slope : $\nabla_x f(x)$=0
gives no information about which direction to move



local optimum에 위험함

suboptimal 문제

Cannot guarantee global minimum but attempt to find a good local minimum

Images from https://bdtechtalks.com/2020/04/27/deep-learning-mode-connectivity-adversarial-attacks/gradient-descent-local-minima/

Some ideas to avoid local minimum
Method of momentum

- SGD : very popular but tends to be slow and difficult to reach the minimum
- **Method of momentum**

  suboptimal 문제 해결 방법.

  - Designed to speed up learning in high curvature and small/noise gradients
  - Exponentially weighted moving average of past gradients (low pass filtering)

$$v_t = \begin{cases} g_1, & t = 1 \\ \rho v_{t-1} + (1-\rho) g_t, & t > 1 \end{cases}$$	v_t: Exponentially weighted moving average at time t g_t: observation gradient at time t ρ (0~1): degree of weighting decrease (smoothing factor)

현재

c.f $v_t = \rho^k v_{t-k} + (1-\rho)[g_t + \rho g_{t-1} + \ldots + \rho^{k-1} g_{t-k+1}]$

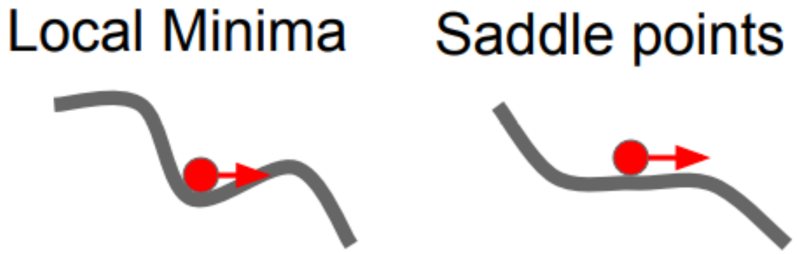
과거

먼 과거의 gradient는 더 작아짐.

SGD

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

$$let \;\; g = \nabla_\theta J(\theta)$$



Images from "The Frontier of SGD and Its Variants in Machine Learning" (2017)

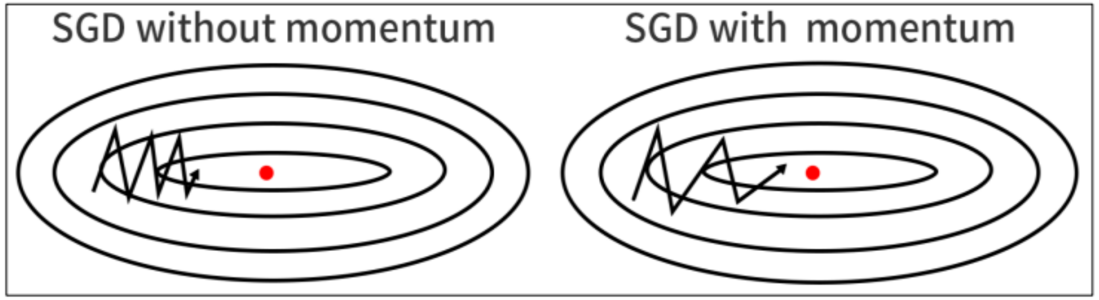
SGD + momentum : Use a velocity as a weighted moving average of previous gradients

과거값 반영해주는것.

$$v \leftarrow \rho v - \alpha g$$

$$\theta \leftarrow \theta + v$$

gradient 값이 0 이 되더라도 학습계속 진행할수 있음.



A parameter is updated by linear combination of gradient and velocity

Some ideas to avoid local minimum

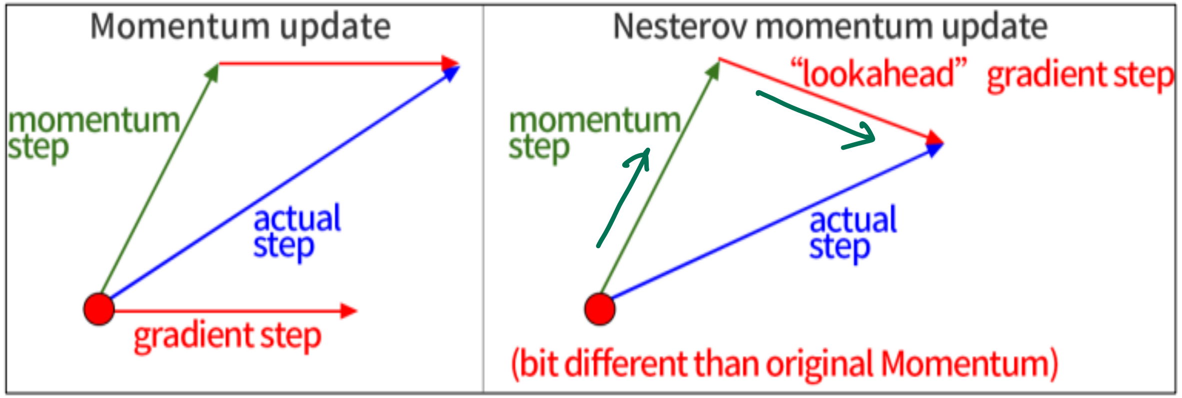
Method of momentum

더 newest

- **Nesterov Momentum** ⇒ gradient 먼저 평가하고 update 함
  - Difference from standard momentum: where gradient g is evaluated (i.e. "lookahead" gradient step)

$$v \leftarrow \rho v_t - \alpha \nabla_\theta J(\theta + \rho v)$$

$$\theta \leftarrow \theta + v$$



Images from "The Frontier of SGD and Its Variants in Machine Learning" (2017)

Some ideas to avoid local minimum

per-parameter adaptive learning rates

- **AdaGrad** : Adapts an individual learning rate of each direction
  - Slow down the learning rate when an accumulated gradient is large
  - Speed up the learning rate when an accumulated gradient is small
- Allows an automatic tuning of the learning rate per parameter

각 방향으로
learning rate를
적응적으로 조절하여
학습 효율 높이는 것

$$let\ \ g = \nabla_\theta J(\theta)$$

gradient · gradient 값이 누적

$$r \leftarrow r + g \cdot g$$

값 점점 커짐

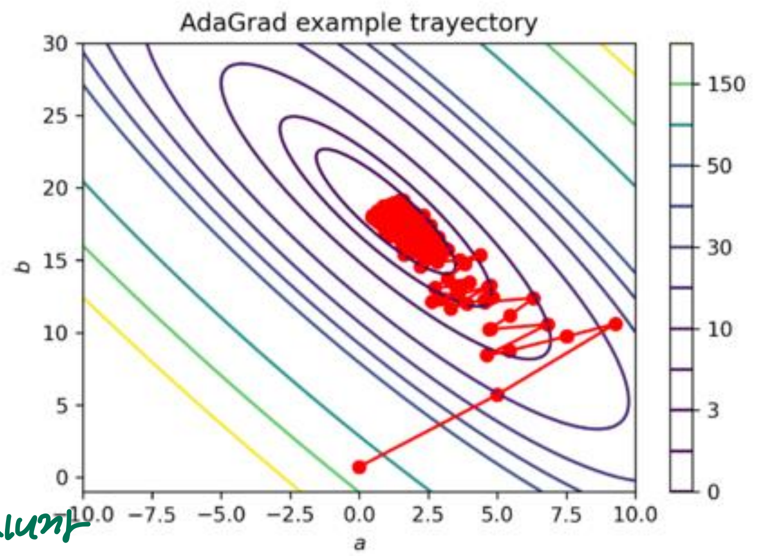
$$\Delta\theta \leftarrow \frac{\alpha}{\sqrt{r+\varepsilon}} \cdot g$$

값 점점
작아짐

$$\theta \leftarrow \theta - \Delta\theta$$

learning rate
조절 됨.

→ 이게 문제임. learning rate 계속 작아지니까
학습이 일어나지 않게 됨



Some ideas to avoid local minimum

per-parameter adaptive learning rates

- **RMSProp** : attempts to fix the drawbacks of AdaGrad, in which the learning rate becomes infinitesimally small and the algorithm is no longer able learning when the accumulated gradient is large.
- (**Remedy**) Gradient accumulation by weighted decaying

AdaGrad

$$r \leftarrow r + g \cdot g$$

$$\Delta\theta \leftarrow \frac{\alpha}{\sqrt{r+\varepsilon}} \cdot g$$

$$\theta \leftarrow \theta - \Delta\theta$$

RMSProp

AdaGrad문제해결

$$r \leftarrow \rho r + (1-\rho) g \cdot g$$

극단적으로
학습속도 줄어드는거
막아줌.

$$\Delta\theta \leftarrow \frac{\alpha}{\sqrt{r+\varepsilon}} \cdot g$$

$$\theta \leftarrow \theta - \Delta\theta$$

Some ideas to avoid local minimum
per-parameter adaptive learning rates

- **Adam (adaptive moment estimation) : RMSProp + momentum**

$$let\ \ g = \nabla_\theta J(\theta)$$

① Compute the first moment from momentum)

$$s \leftarrow \rho_1 s + (1-\rho_1) g \cdot g$$

② Compute the second moment from RMSProp

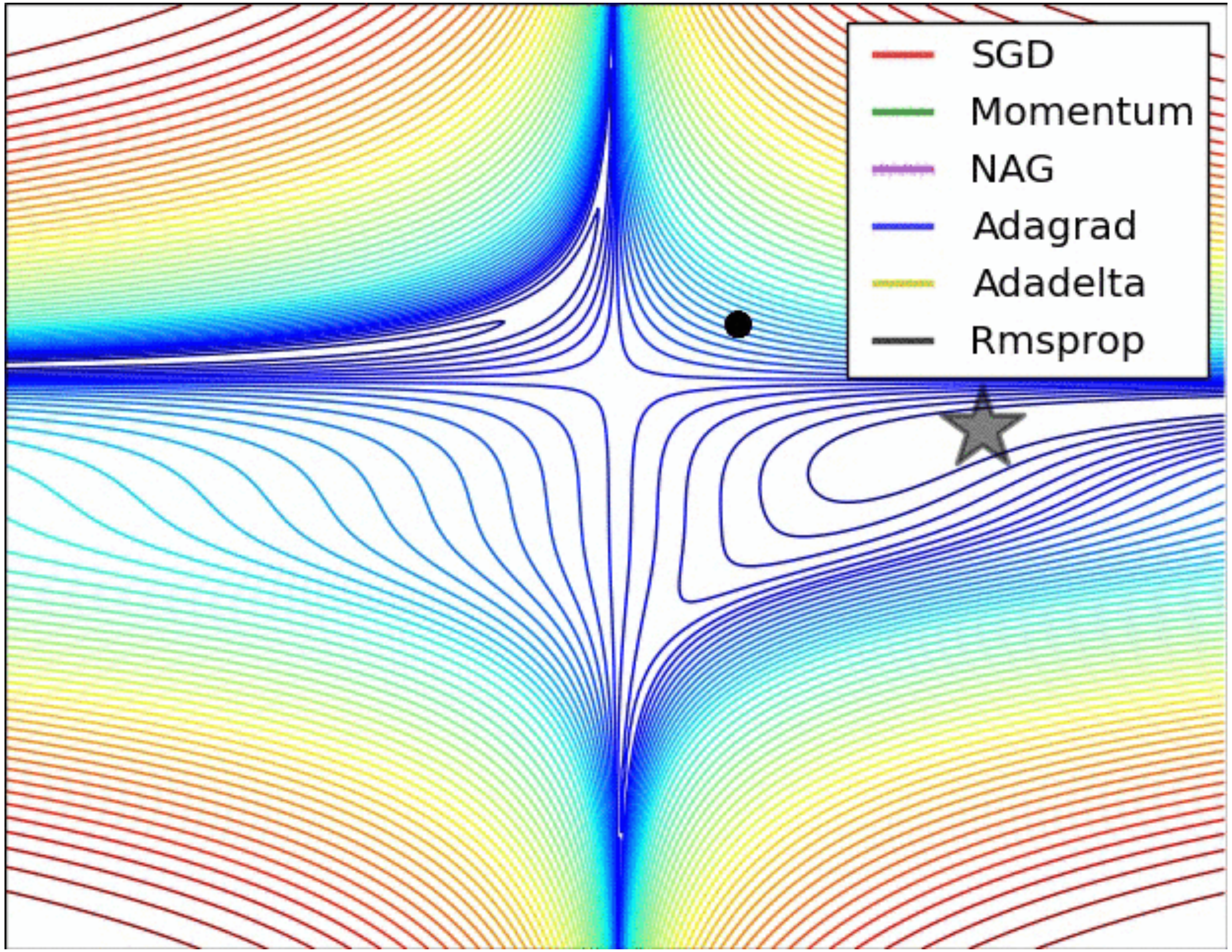
$$r \leftarrow \rho_2 r + (1-\rho_2) g \cdot g$$

③ Correct the bias

$$s' \leftarrow \frac{s}{1-\rho_1} \qquad r' \leftarrow \frac{r}{1-\rho_2}$$

④ Update the parameters

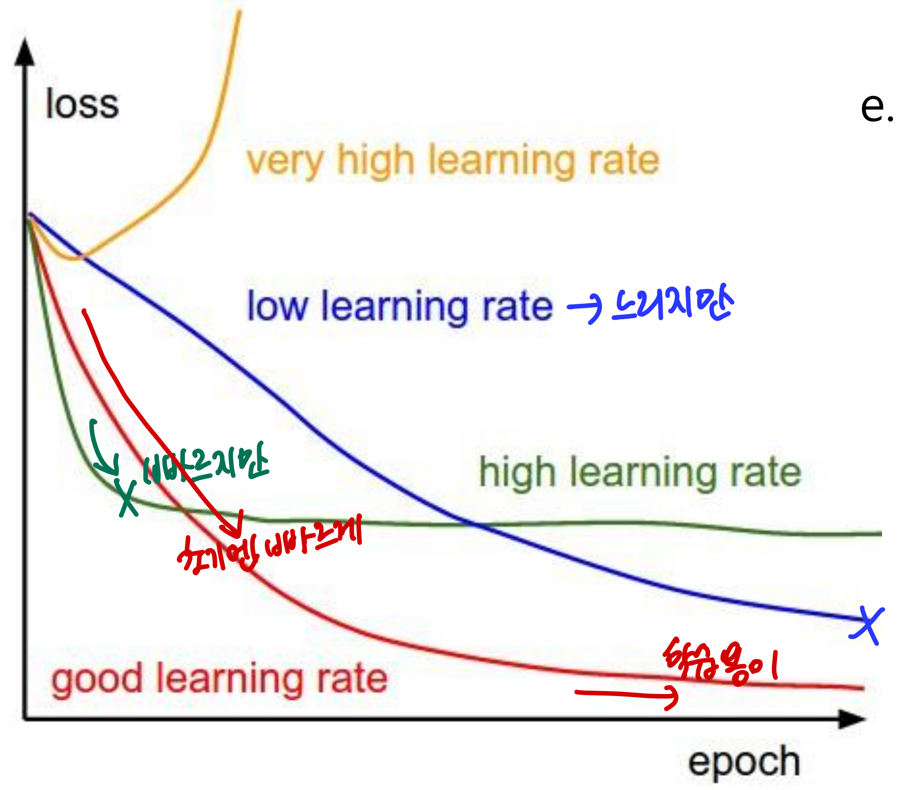
$$\theta \leftarrow \theta - \alpha \frac{s'}{\sqrt{r' + \varepsilon}}$$



https://towardsdatascience.com/why-visualize-gradient-descent-optimization-algorithms-a393806eee2

Learning rate scheduling

- Learning rate: key hyper parameter for gradient-based algorithms ; need to gradually <u>decrease</u> learning rate over time



e.g. decay learning rate by every half of few epochs

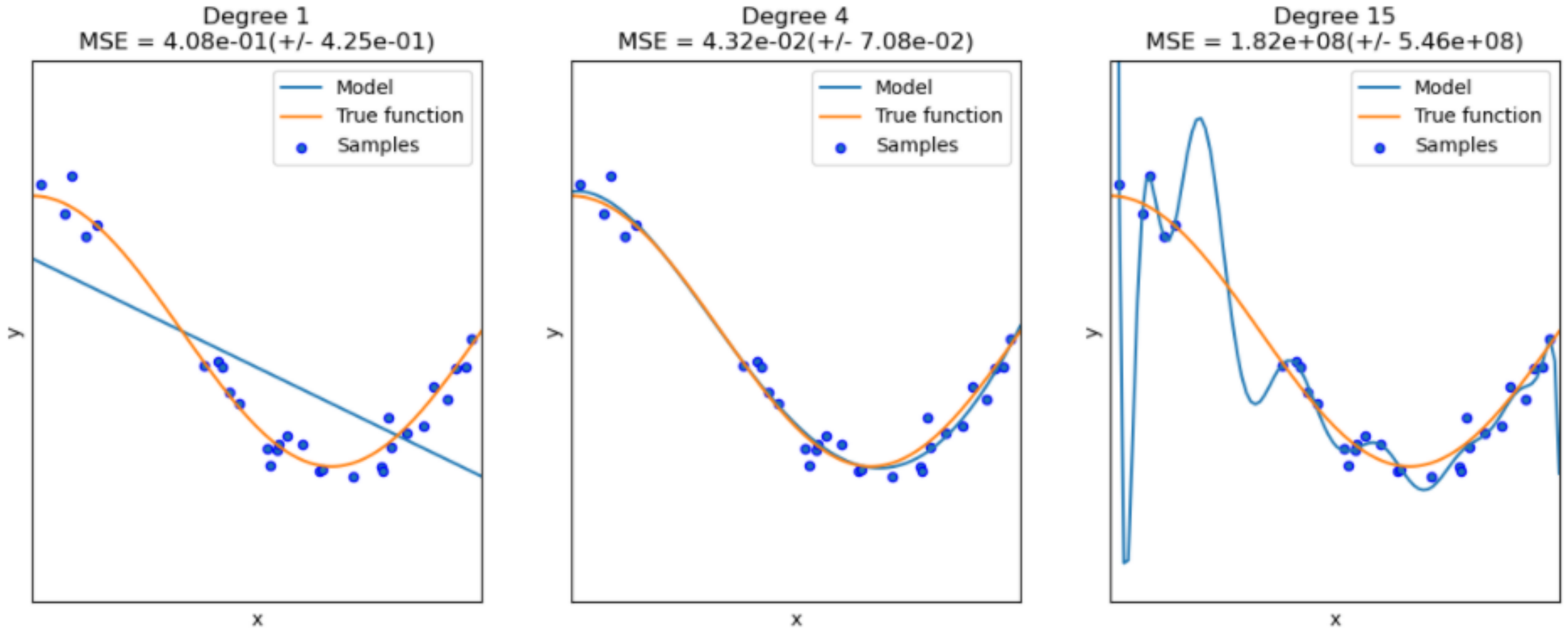
$$\alpha = \alpha_0 e^{-kt}$$

$$\alpha = \frac{\alpha_0}{(1+kt)}$$

Images from Stanford CS231n

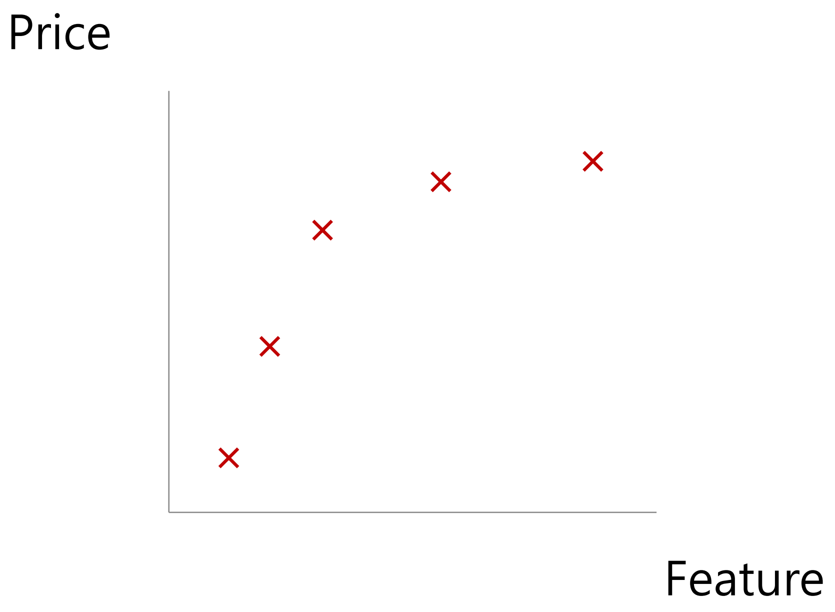
Some optimization in regression
to avoid overfitting

- If we have too many features, the hypothesis may fit the training set very well. However, it may fail to generalize to new samples



Some optimization in regression
to avoid overfitting

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$
$x_6 =$ kitchen size
$\vdots$
x_{100}



More features → more parameters → need more data ; (in practice) less data → overfitting
Furthermore, mean-squared-error is sensitive to outliers

Some optimization in regression
to avoid overfitting

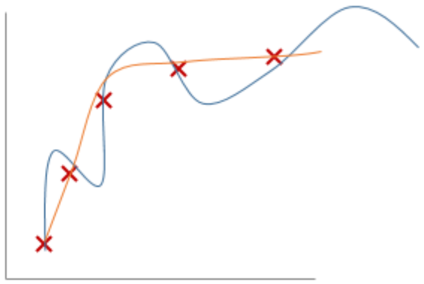
1. Reduce number of features.
   — select which features to keep.
2. Regularization. ← 복잡한 모델 사용하게 되도 학습과정에서 모델의 복잡도에 대한 penalty 줘서 모델이 overfitting 되지 않도록 해줌.
   — keep the features but reduce magnitude/values of parameters
   — Simple hypothesis and less prone to overfitting and robust to noise

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\underbrace{(h_\theta(x^{(i)}) - y^{(i)})}_{\text{mse . "fitting"}}{}^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

θ_j 가 클수록 늘어나는 오류
모델은 최대한 θ_j를 0으로 만들어 parameter 적게 만든다

$$\min_\theta J(\theta)$$

λ: regularization parameter



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Quiz

What answers are correct? Select all that apply.

A. Gradient descent produces a numerical solution but it may not achieve a global optimum

Correct. The solution can be different with an initial point of an error surface

B. Momentum of previous gradient descent can help avoid overfitting

False. Momentum can avoid local minimum and help obtain a solution close to global optimum

C. Regularization can penalize the importance of some input features to avoid overfitting

Correct. Regularization can decrease some weights to have compact sets of parameters

Summary

- Optimization in general ML/DL
  - General ideas of gradient descent algorithm
  - Mostly stochastic gradient descent and its variants using gradient estimates
  - Adam is a good default choice in most cases
- Regularization
  - Reduce magnitude/values of parameters while keeping features
  - Simple hypothesis and less prone to overfitting
  - Robust to noise

Reference

- Book: Pattern Recognition and Machine Learning (by Christopher M. Bishop)
- Book: Machine Learning: a Probabilistic Perspective (by Kevin P. Murphy)
- https://www.andrewng.org/courses/