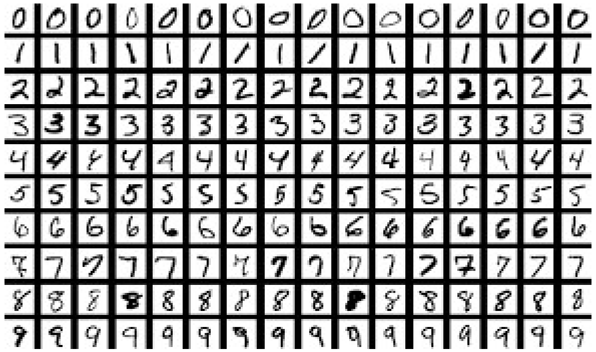
Machine Learning (ML) Training Basics

Saehwa Kim

Information and Communications Engineering
Hankuk University of Foreign Studies

MNIST Database

- MNIST: Modified National Institute of Standards and Technology
- Database of handwritten digits
- Total 70,000 images



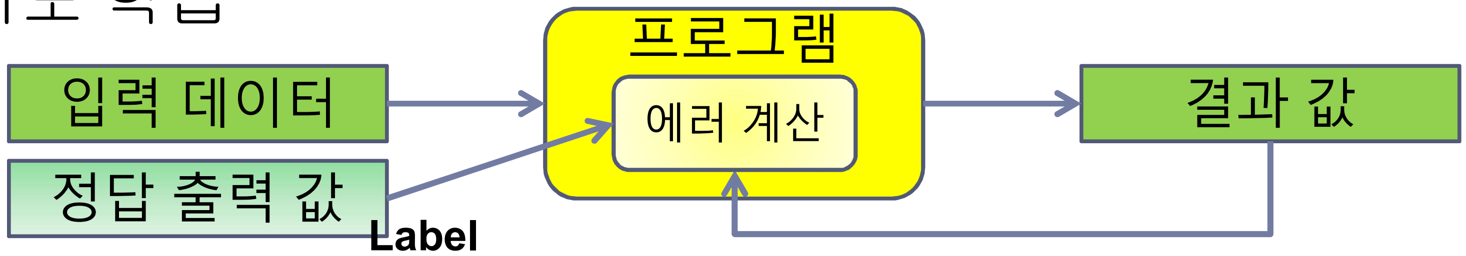
- (cf) Fashion-MNIST: a dataset of clothes images from an article

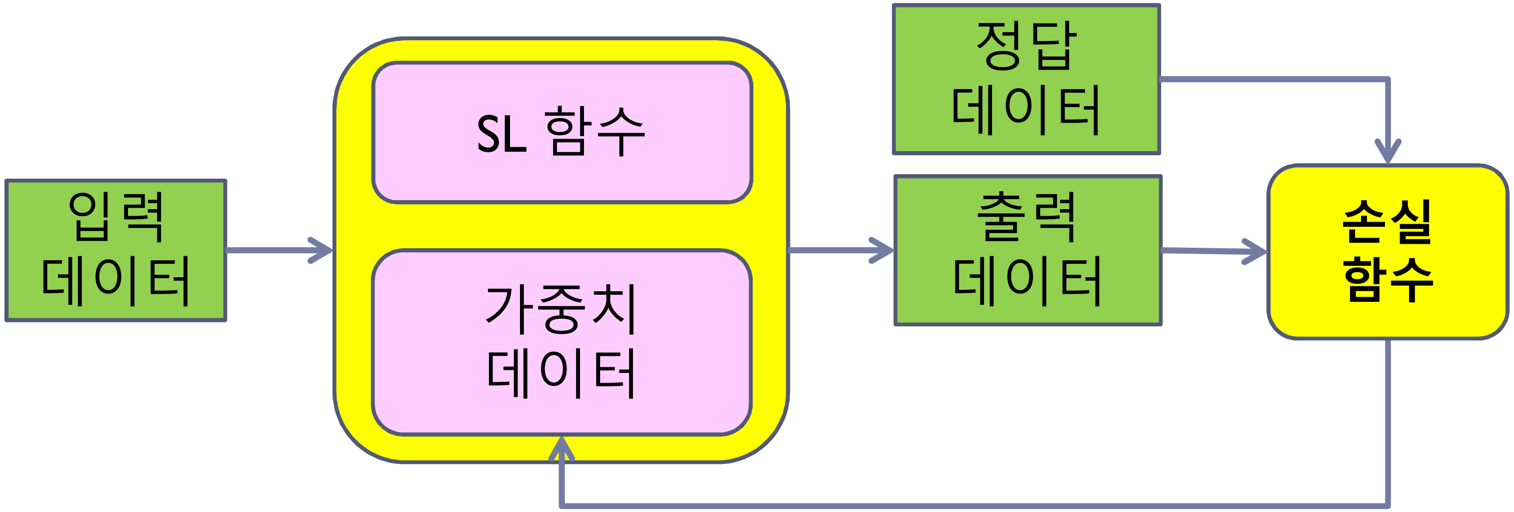
Data Set in Supervised Learning

- In supervised learning
- Training set and test set
  - Common practice
    - 80% data for training (20% data for testing)
- Data == Train data + Test data (훈련, 시험)
- Training data == True training data + Validation data (검증)
  - Holdout validation
  - Held-out set == Validation set == Development set (dev set)
- (K-fold) Cross validation
  - Uses many (K) small validation set
  - Training time is multiplied by K.
- The case where training set error is low while the test set error is high
  - Overfitting error == Generalization error == Out-of-sample error

Training in Supervised Learning (SL)

지도 학습





손실 == Loss == Error == Cost

손실 함수 == Loss function

Parameter Learning with Gradient Descent for (Simple Univariate) Linear Regression

- https://wikidocs.net/7635
- Hypothesis function h: $h_\theta(x)=\theta_0+\theta_1 x$
- Cost function J: mean-squared-error (MSE): LSE (least squared error) criterion
  - 예측값과 실제값의 차이

$\hat{y}^{(i)}$ $y^{(i)}$

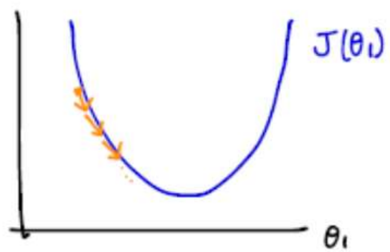
$$J(\theta_0,\theta_1)=\frac{1}{2m}\sum_{i=1}^{m}\left(\underline{\hat{y}^{(i)}-y^{(i)}}\right)^2=\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2$$

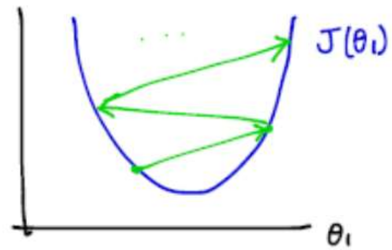
- **Gradient descent algorithm**

repeat until convergence{

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0.\theta_1) \quad \text{for } j=0, j=1$$

}





- $:=$ | "assignment" operator
- α | learning rate
- j | feature index number, should be updated simultaneously

$\bar{a} \leftarrow$ a is a vector. 보통 column vector $(N \times 1)$

$x_j^{(i)}$ ← instance idx $1 \sim m$, ← feature idx $0 \sim n$

Multiple Linear Regression

$$\hat{y}^{(i)} = \underbrace{\theta_0}_{\text{bias (편향, 절편)}} \underbrace{\overline{x_0^{(i)}}^{=\theta_0}}_{=1} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_n x_n^{(i)}$$

$$\underset{(1,1)}{\hat{y}^{(i)}} = h_{\bar{\theta}}(\bar{x}^{(i)}) = \underset{(1,n+1)}{\bar{\theta}^T} \underset{(n+1,1)}{\bar{x}^{(i)}}$$

$$\underset{(m,1)}{\bar{y}} = \underset{(m,n+1)}{\bar{X}} \underset{(n+1,1)}{\bar{\theta}}$$

[[   ]]

$$\bar{y} = \begin{bmatrix} [y^{(1)}] \\ \vdots \\ [y^{(m)}] \end{bmatrix} \quad \bar{X} = \begin{bmatrix} \bar{x}^{(1)T} \\ \vdots \\ \bar{x}^{(m)T} \end{bmatrix} = \begin{bmatrix} [1, x_1^{(1)}, \cdots, x_n^{(1)}] \\ \vdots \\ [1, x_1^{(m)}, \cdots, x_n^{(m)}] \end{bmatrix} \quad \bar{\theta} = \begin{bmatrix} [\theta_0] \\ [\theta_1] \\ \vdots \\ [\theta_n] \end{bmatrix}$$

Gradient Descent Algorithm for Multiple Linear Regression

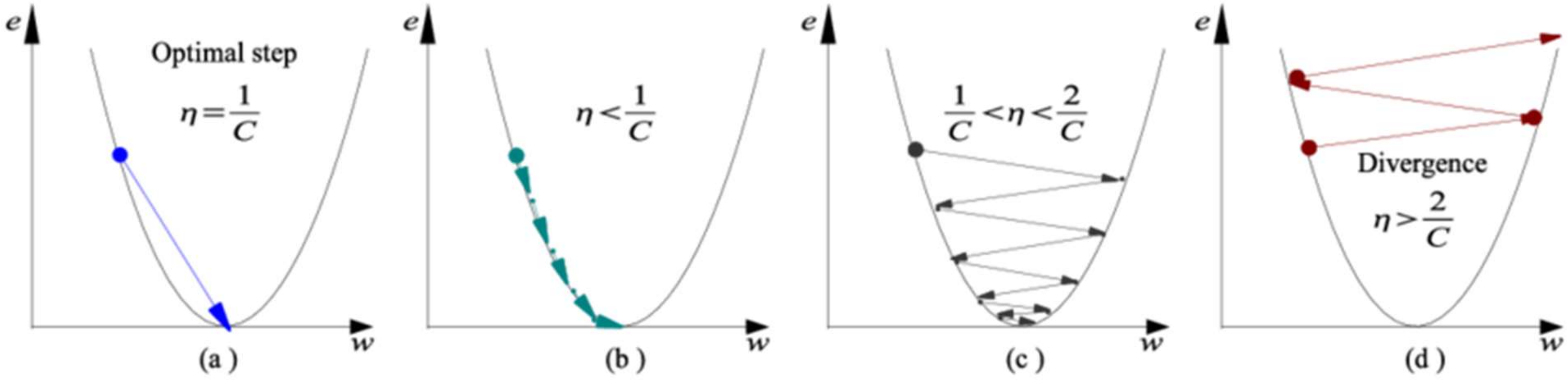
$$\overline{\theta} := \overline{\theta} - \underbrace{\alpha \nabla_{\overline{\theta}} MSE(\overline{\theta})}$$

direction

step size

$\nabla_a b = \frac{\partial b}{\partial a}$

$\boxed{\text{cf. Gradient Ascent Algorithm for RL. (reward)}}$



이미지 출처: ttps://www.researchgate.net/figure/Convergence-Conditions-in-Gradient-Descent-Algorithm_fig2_224324276

Batch Gradient Descent vs. Stochastic Gradient Descent

- Batch Gradient Descent
  - 모든 training data 활용(하여 θ update)

- Stochastic Gradient Descent (SGD)
  - "임의의 하나"의 training data 활용(하여 θ update)
  - 이걸 True SGD라고 부르기도 함

- Mini-batch Gradient Descent
  - "임의의 일부(mini-batch)" training data 활용(하여 θ update)
  - 이걸 SGD라고 부르는 사람들도 있음

Batch Gradient Descent

$$MSE(\bar{X}, h_{\bar{\theta}}) = MSE(\bar{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\Big(\overbrace{\bar{\theta}^T \bar{x}^{(i)} - y^{(i)}}^{Err^{(i)}}\Big)^2$$

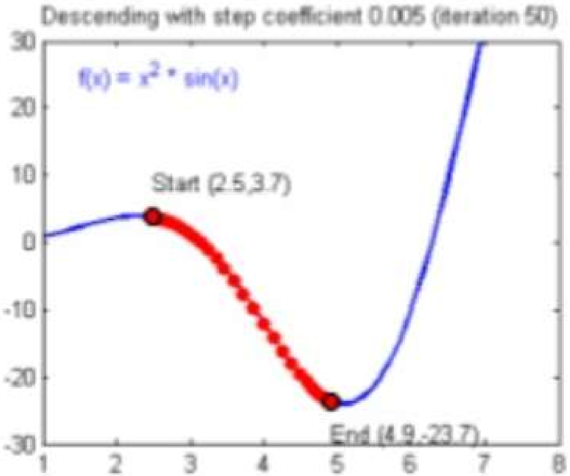
$$\frac{\partial}{\partial \theta_j} MSE(\bar{\theta}) = \frac{2}{m}\sum_{i=1}^{m}\left(\bar{\theta}^T \bar{x}^{(i)} - y^{(i)}\right) \cdot \bar{x}_j^{(i)}$$

$$\bar{y} = \bar{X}\bar{\theta}$$

$$MSE(\bar{\theta}) = \frac{1}{m}(\bar{X}\bar{\theta} - \bar{y})^T \overbrace{(\bar{X}\bar{\theta} - \bar{y})}^{\overline{Err}}$$

$$\underbrace{\frac{\partial}{\partial \bar{\theta}} MSE(\bar{\theta})}_{(n+1)\times 1} = \frac{2}{m} \cdot \underbrace{\bar{X}^T}_{(n+1)\times m} \overbrace{(\bar{X}\bar{\theta} - \bar{y})}^{\overline{Err}\ (m\times 1)}$$

BGD vs. (Mini-Batch) SGD (1)



뉴럴넷은 loss(or cost) function을 가지고 있습니다. 쉽게 말하면 "틀린 정도"

현재 가진 weight 세팅(내 자리)에서,
내가 가진 데이터를 다 넣으면
전체 에러가 계산됩니다.

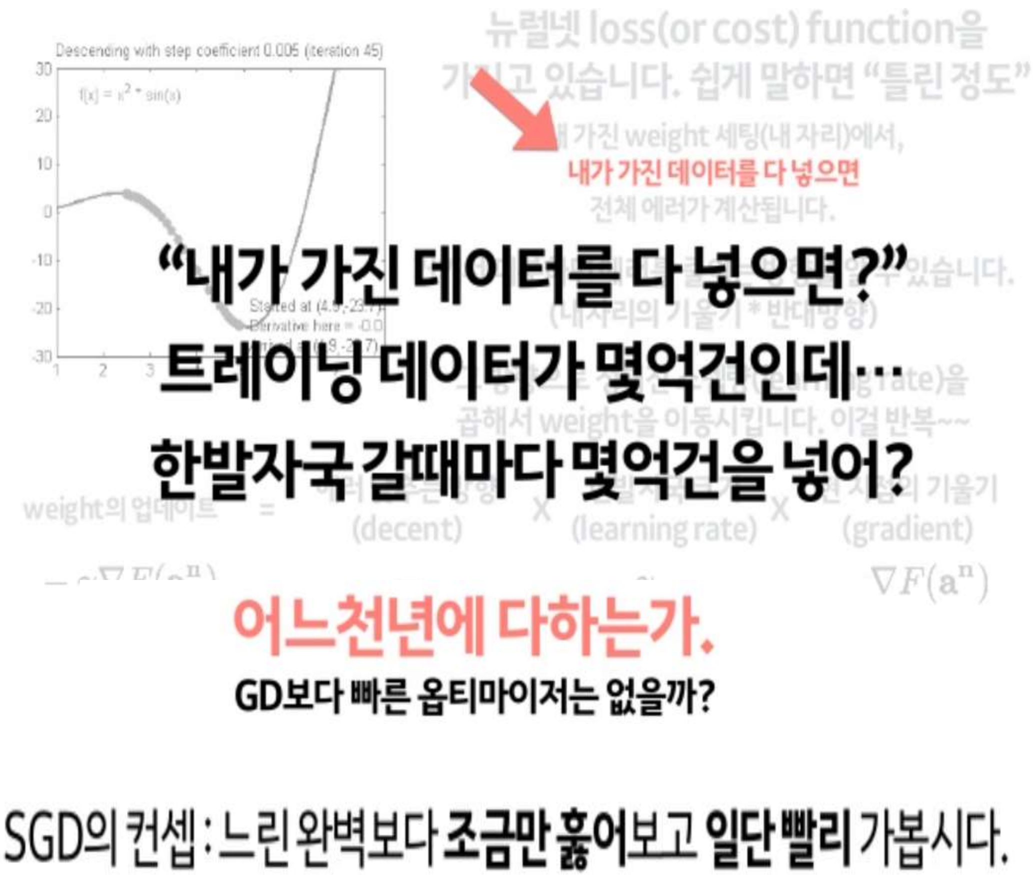
거기서 미분하면 에러를 줄이는 방향을 알 수 있습니다.
(내자리의 기울기 * 반대방향)

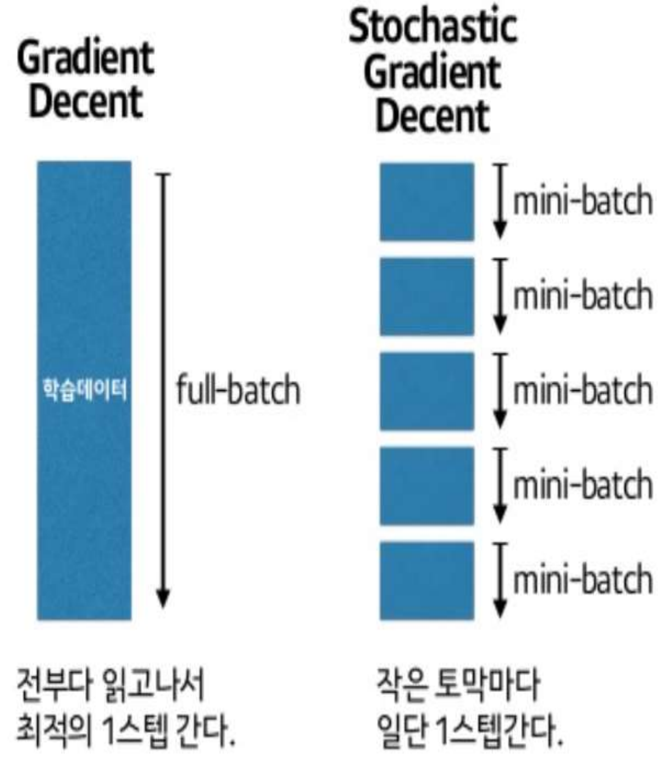
그 방향으로 정해진 스텝량(learning rate)을
곱해서 weight을 이동시킵니다. 이걸 반복~~

weight의 업데이트	=	에러 낮추는 방향 (decent)	X	한발자국 크기 (learning rate)	X	현 지점의 기울기 (gradient)
$-\gamma\nabla F(\mathbf{a}^n)$		$-$		γ		$\nabla F(\mathbf{a}^n)$

출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

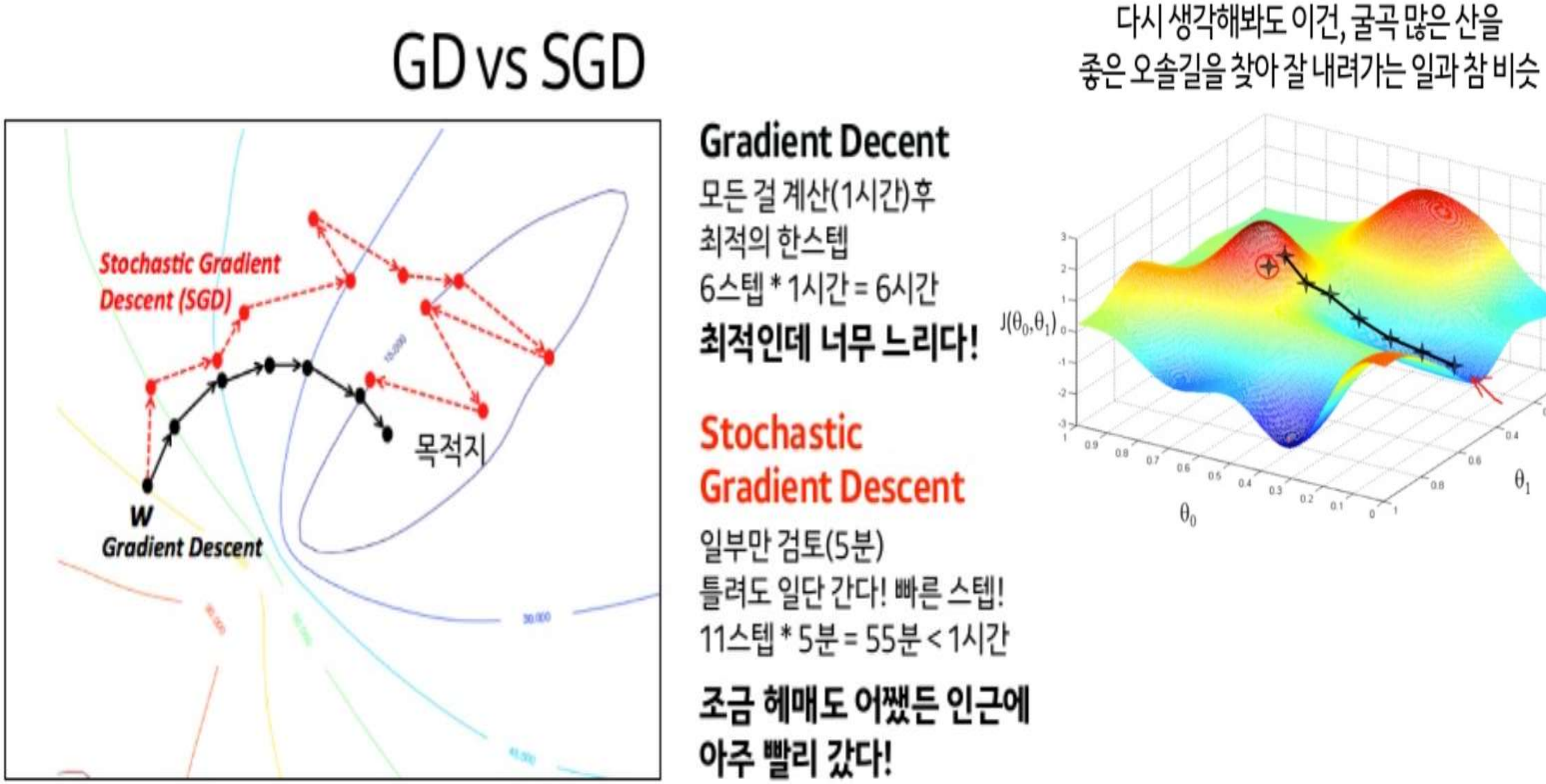
BGD vs. (Mini-Batch) SGD (2)





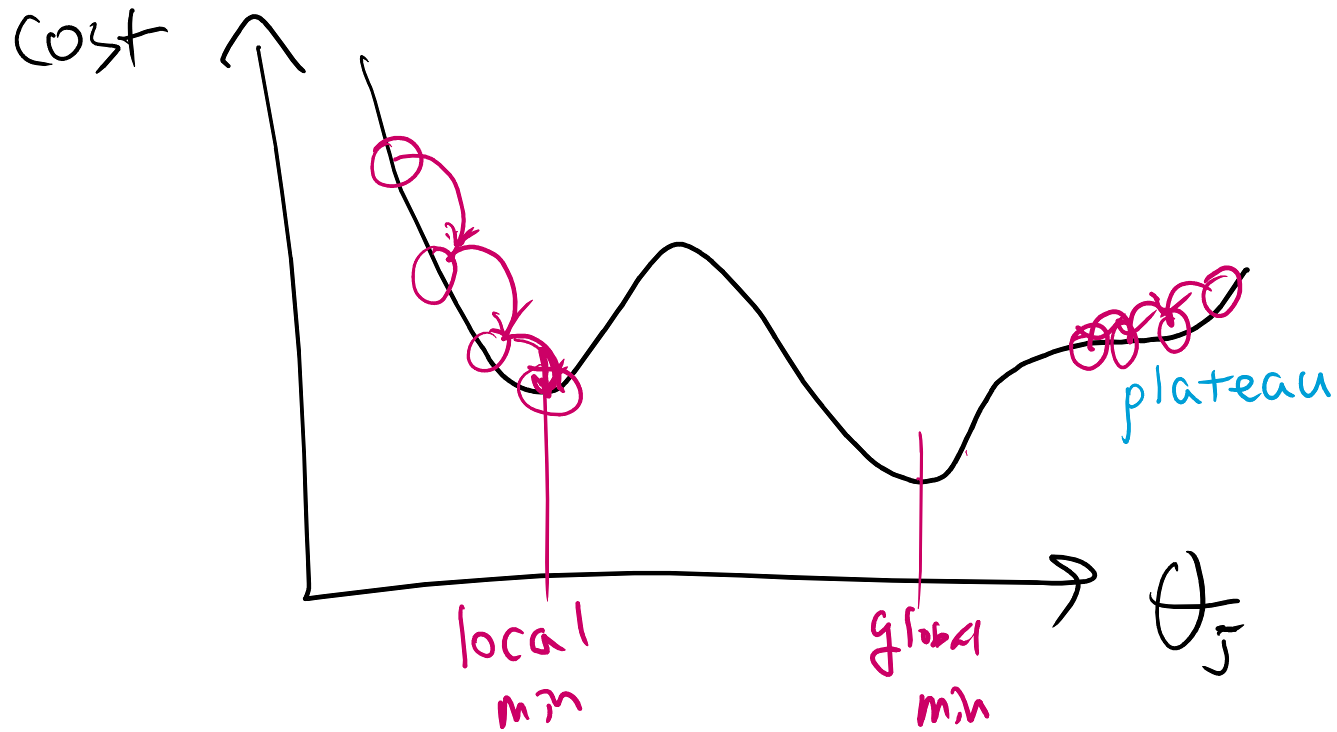
출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

BGD vs. (Mini-Batch) SGD (3)



출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

Gradient Descent Pitfalls



- 선형회기에서는 MSE cost function이 볼록함수여서 항상 최솟값 찾는 것이 가능

global

Feature Scaling

- Min-max scaling
  - Makes values ranging from 0 to 1
  - V' = (V-min)/(max-min)
- Standardization
  - Achieves zero mean and unit variance. (0, 1)
  - V' = (V-mean)/standard_dev
  - Much less effected by outliers (이상치)

Gradient Descent with and without Feature Scaling

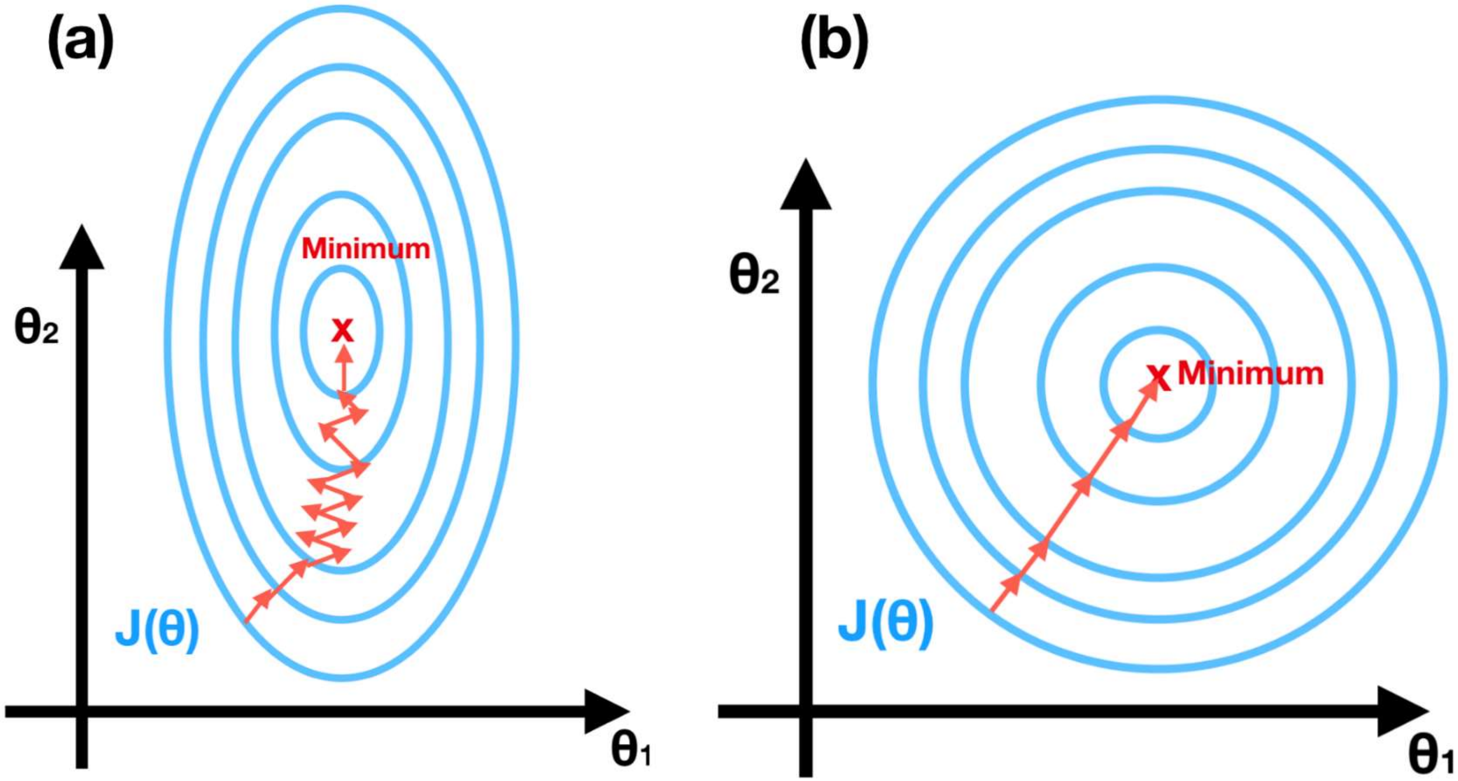
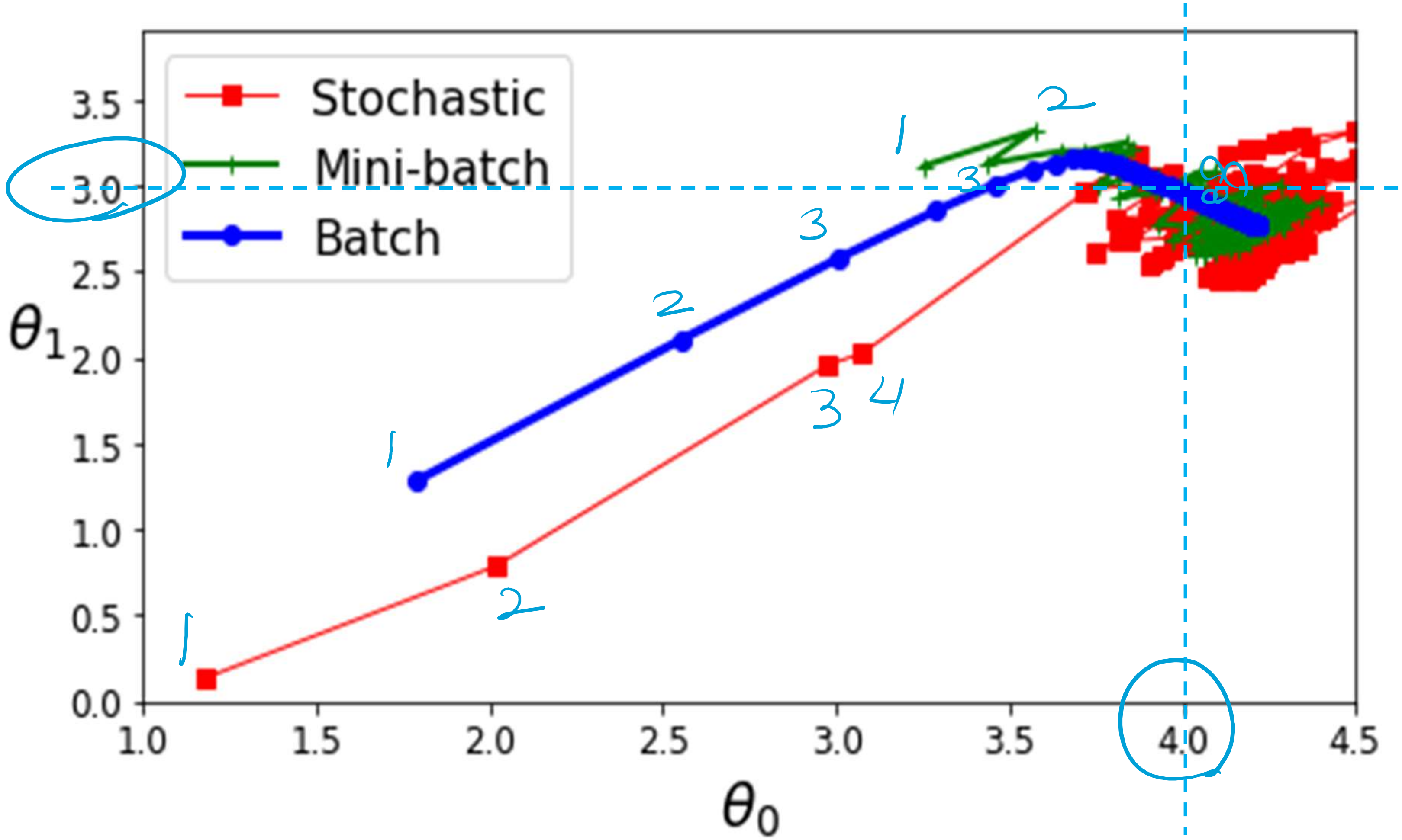


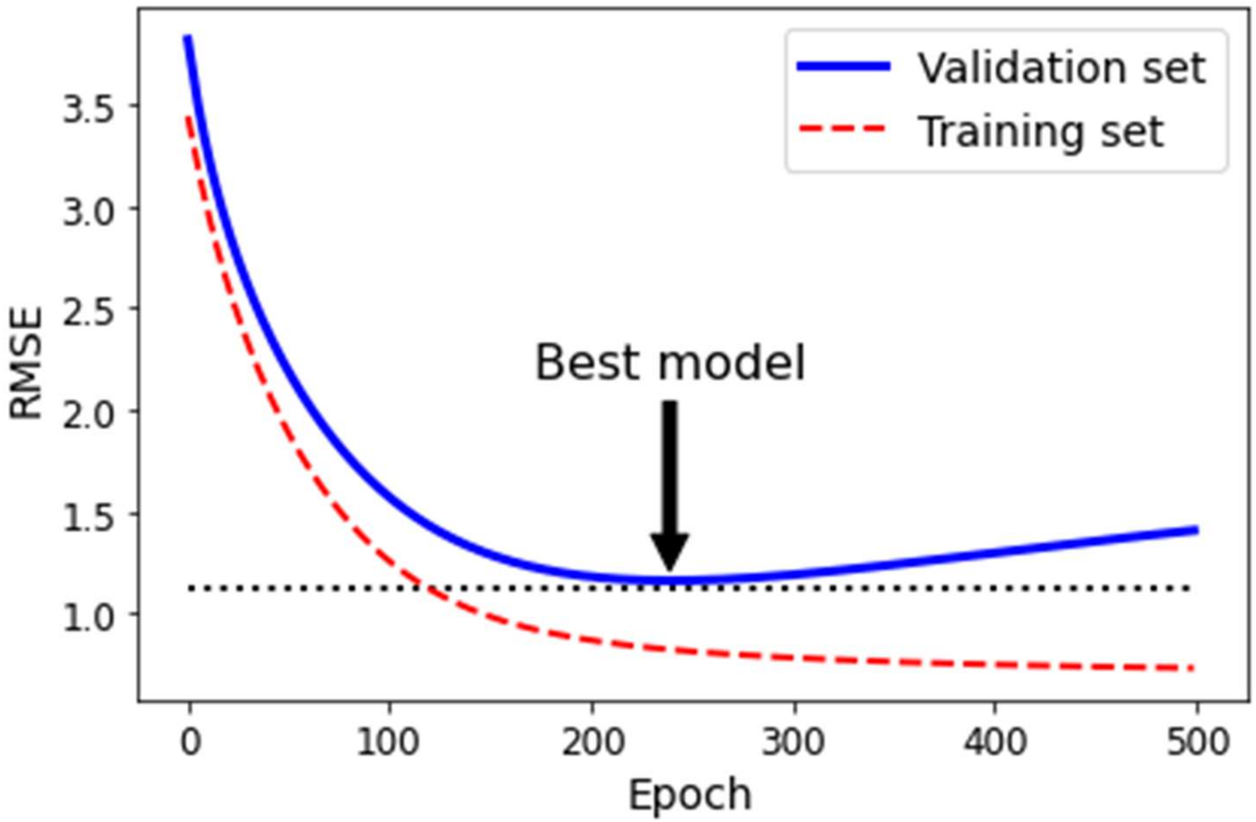
그림 출처: https://medium.com/@mlgomez230/optimization-techniques-in-machine-learning-5d06725942b

Path of Gradient Descent Parameters



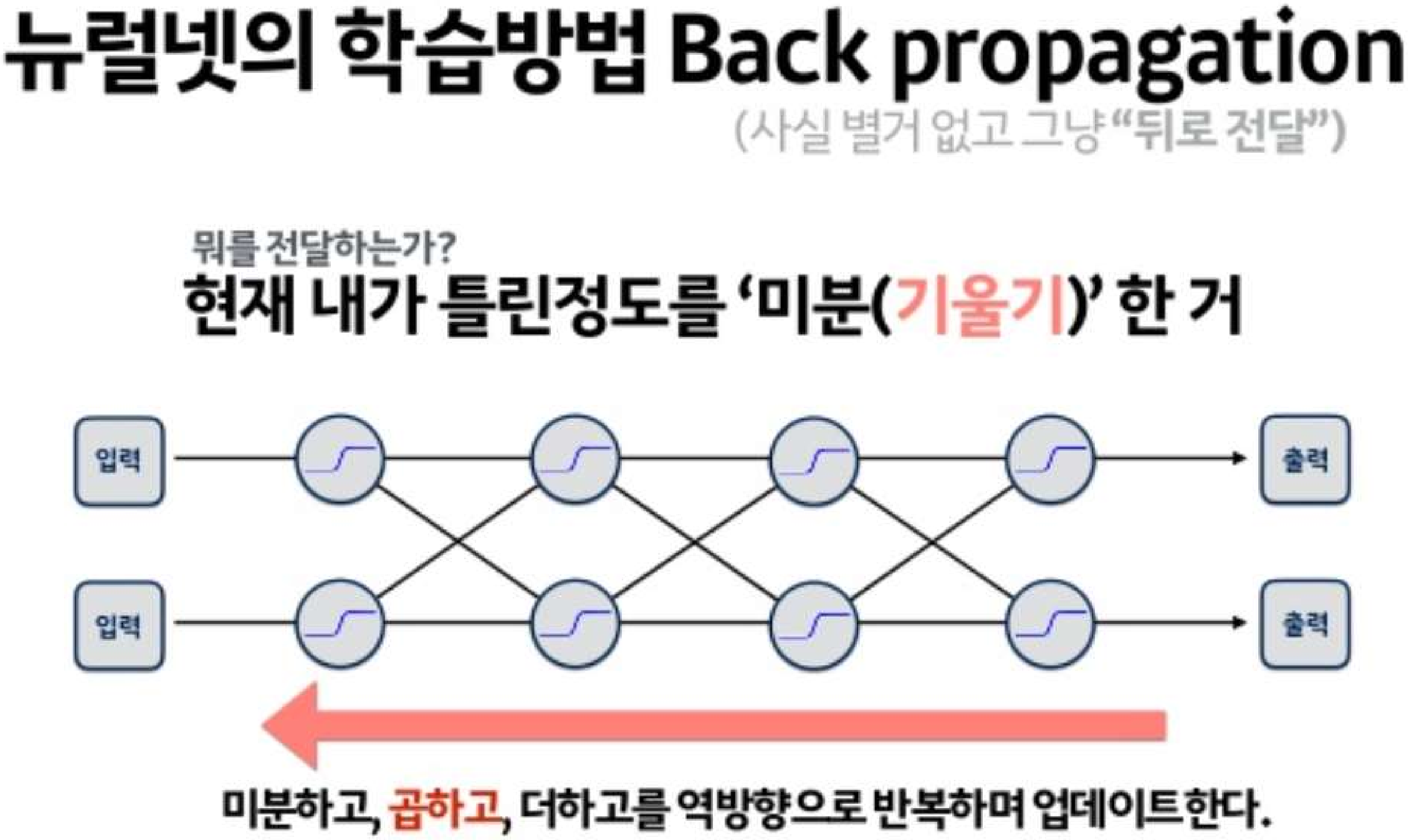
출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

Early Stopping



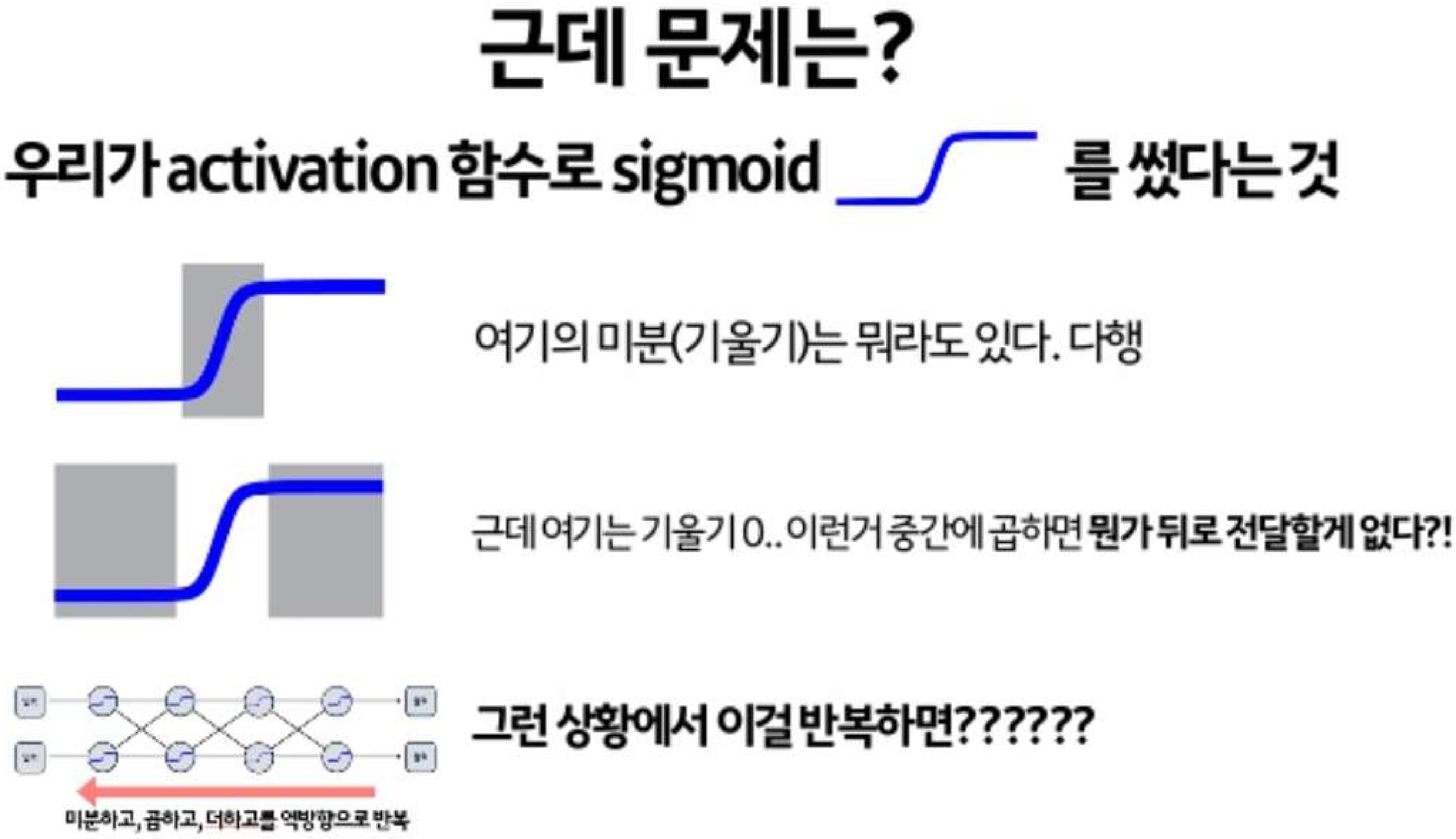
출처: Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Media

Vanishing Gradients Problem (1)



출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

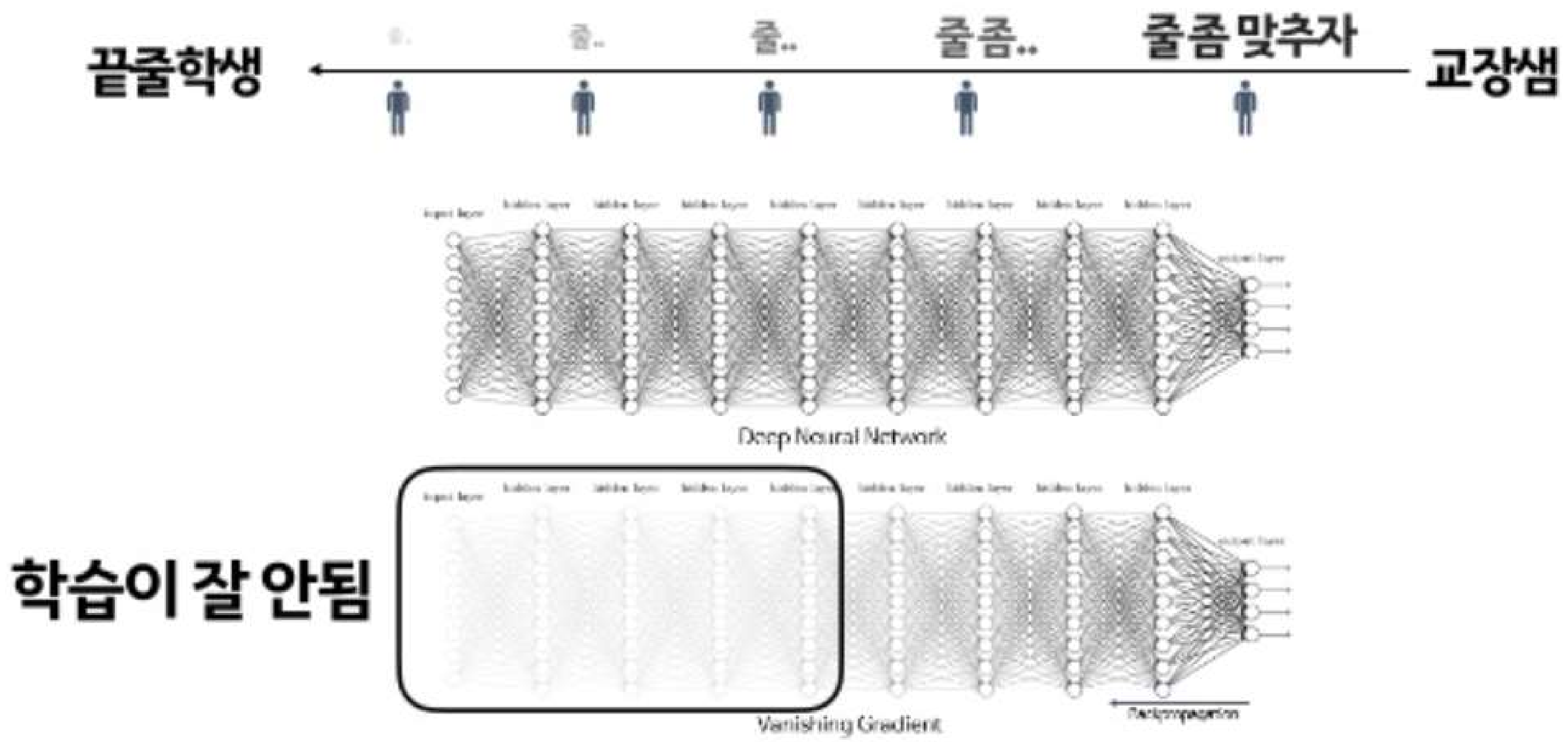
Vanishing Gradients Problem (2)



출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

Vanishing Gradients Problem (3)

Vanishing gradient 현상 : 레이어가 깊을 수록 업데이트가 사라져간다. 그래서 fitting이 잘 안됨(underfitting)

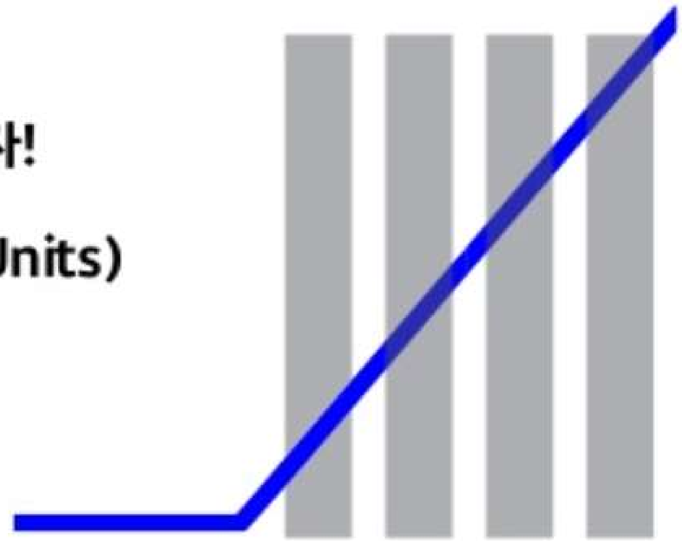


출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

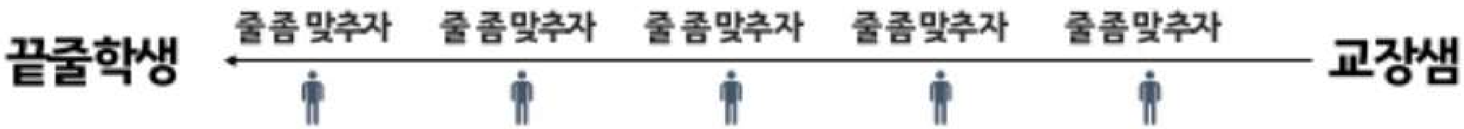
Vanishing Gradients Problem (4)

사그라드는 sigmoid대신
죽지않는 activation func을 쓰자!

→ **ReLU** (Rectified Linear Units)



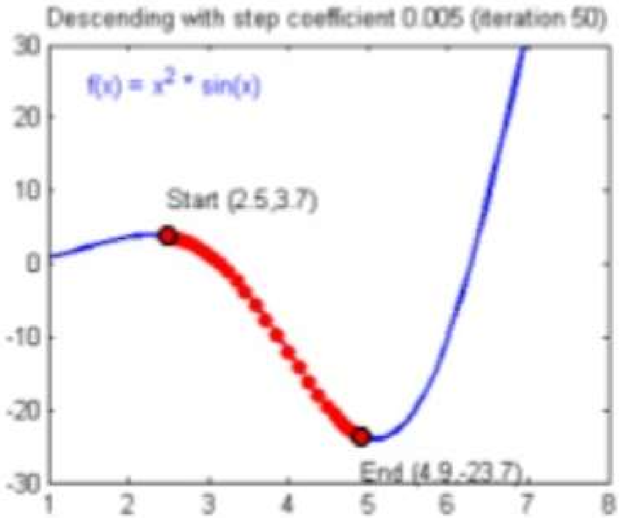
이녀석은 양의구간에서전부 미분값(1)이 있다!



끝 줄 학생까지 이야기가 전달이 잘 되고 위치를 고친다!

출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

기본 Optimizer: (Mini-Batch) SGD (1)



뉴럴넷은 loss(or cost) function을 가지고 있습니다. 쉽게 말하면 "틀린 정도"

현재 가진 weight 세팅(내 자리)에서,
내가 가진 데이터를 다 넣으면
전체 에러가 계산됩니다.

거기서 미분하면 에러를 줄이는 방향을 알 수 있습니다.
(내자리의 기울기 * 반대방향)

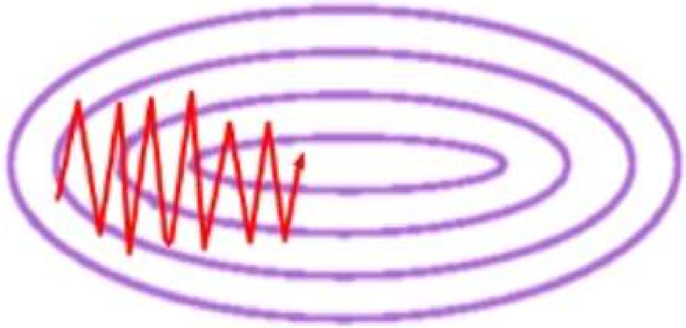
그 방향으로 정해진 스텝량(learning rate)을
곱해서 weight을 이동시킵니다. 이걸 반복~~

weight의 업데이트 =	에러 낮추는 방향 (decent)	X	한발자국 크기 (learning rate)	X	현 지점의 기울기 (gradient)
$-\gamma\nabla F(\mathbf{a}^n)$	$-$		γ		$\nabla F(\mathbf{a}^n)$

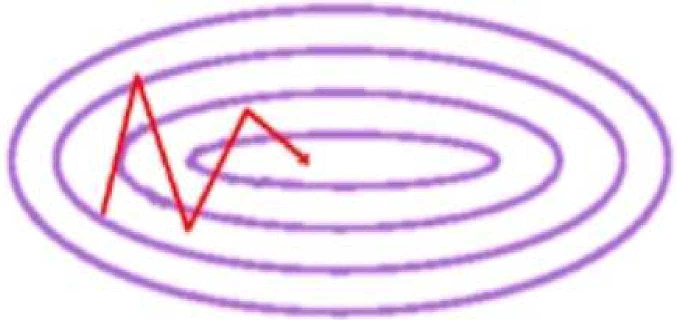
출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

기본 Optimizer: (Mini-Batch) SGD (2)

근데 미니 배치를 하다보니 와리가리(?) 방향 문제가 있다.



딱 봐도 더 잘 갈 수 있는데
훨씬 더 헤매면서 간다.

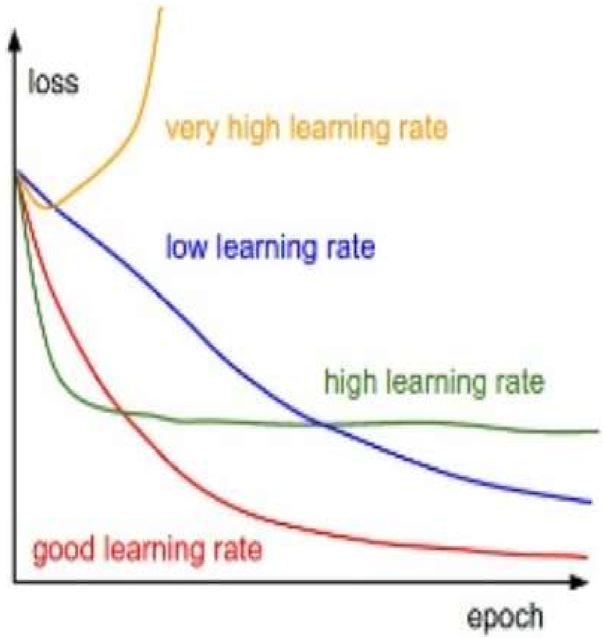


훑기도 잘 훑으면서,
좀 더 휙휙 더 좋은 방향으로 갈 순 없을까?

출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

기본 Optimizer: (Mini-Batch) SGD (3)

스텝사이즈(learning rate)도 문제가 된다.



보폭이 너무 작으면 오래 헤매고(파란라인)

보폭이 너무 크면, 오솔길을 지나친다(녹색라인)

출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

Faster Optimizers (1)

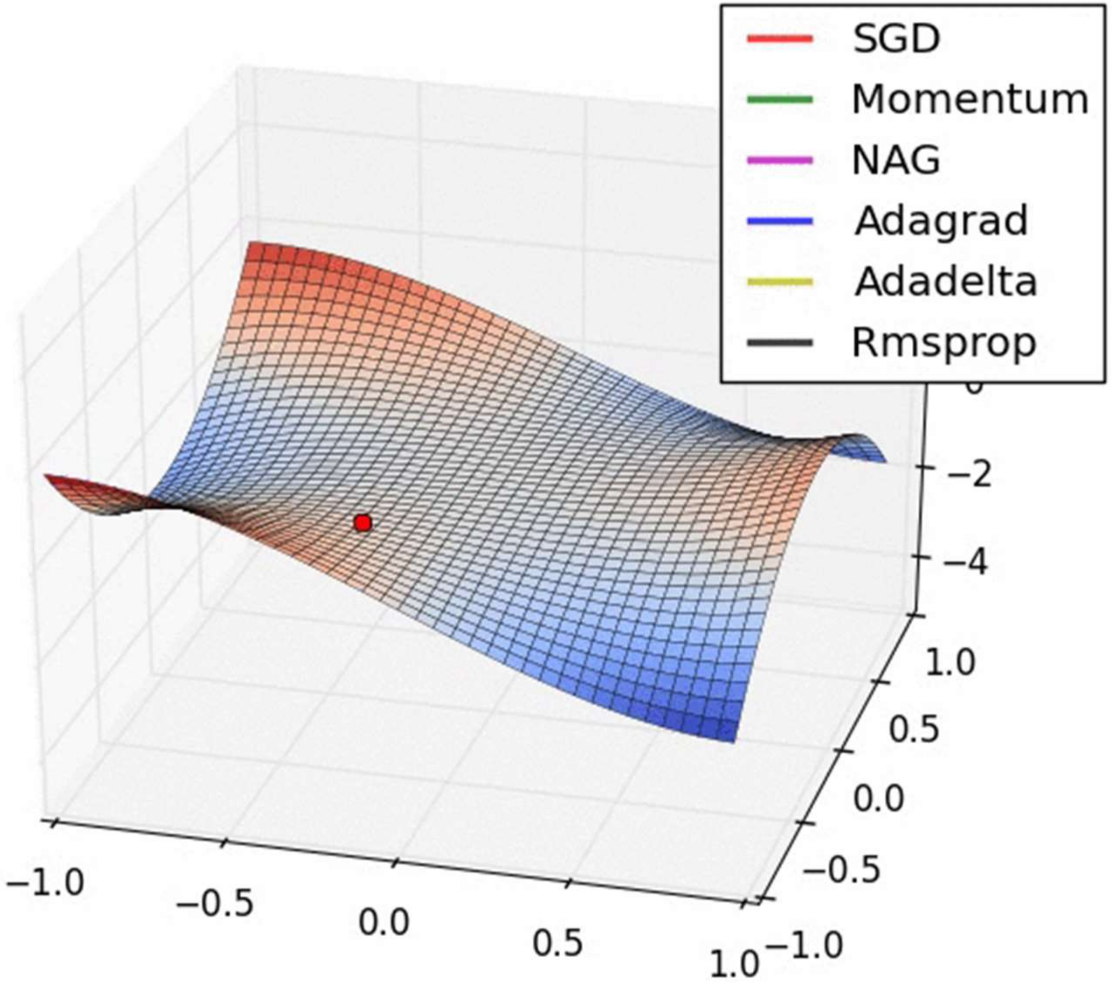
$-\gamma \nabla F(\mathbf{a}^{\mathbf{n}})$	**산을 잘 타고 내려오는 것은**
$\nabla F(\mathbf{a}^{\mathbf{n}})$	어느 **방향**으로 발을 디딜지
γ	얼마 **보폭**으로 발을 디딜지

두가지를 잘잡아야 빠르게 타고 내려온다.

SGD를 더 개선한 멋진 optimizer가 많다!
SGD의 개선된 후계자들

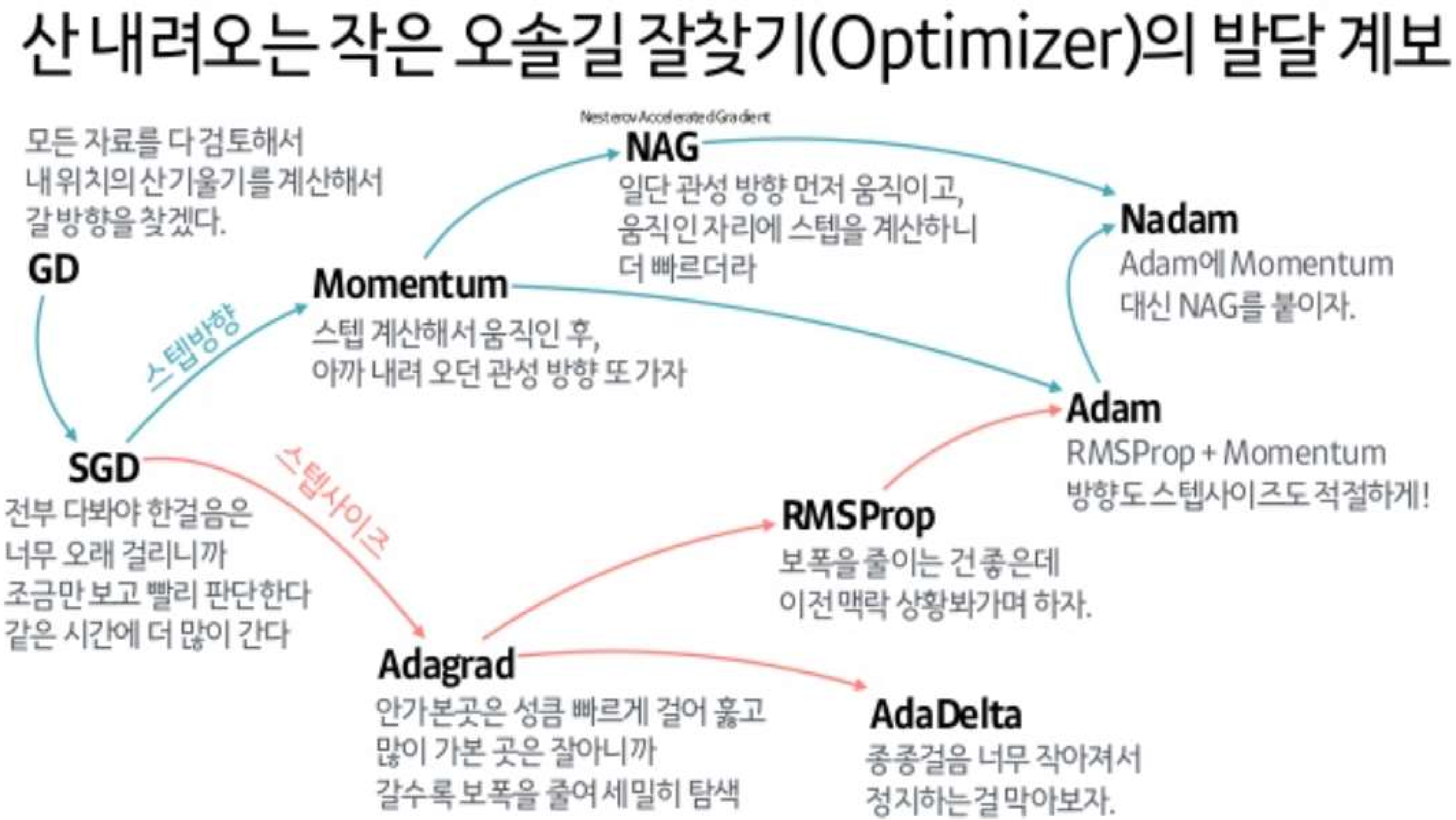
출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

Faster Optimizers (2)



그래프 출처: https://imgur.com/NKsFHJb

Faster Optimizers (3)



출처: https://www.slideshare.net/yongho/ss-79607172, 하용호@kakao

Bias and Variance Tradeoff (1)

- http://scott.fortmann-roe.com/docs/BiasVariance.html

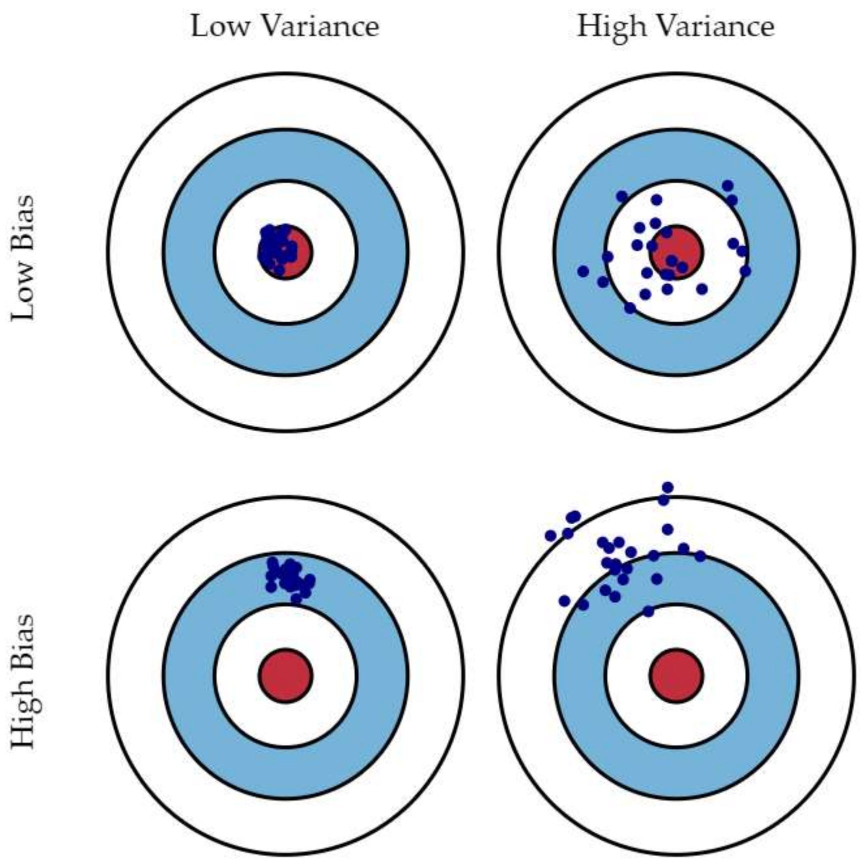


Fig. 1 Graphical illustration of bias and variance.

The center of the target is a model that perfectly predicts the correct values. (As we move away from the bulls-eye, our predictions get worse.)

Bias and Variance Tradeoff (2)

- https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9
- Suppose that we have independent variables *x* that affect the value of a dependent variable *y*

$$y = f(x) + \epsilon$$

- Noise is modeled by random variable ϵ with zero mean and variance $\sigma\epsilon^2$

$$\mathbb{E}[\epsilon] = 0, \mathrm{var}(\epsilon) = \mathbb{E}[\epsilon^2] = \sigma_\epsilon^2$$

- In the linear regression, mean square error

$$\mathrm{MSE} = \mathbb{E}[(y - \hat{f}(x))^2]$$

$$\mathrm{bias}[\hat{f}(x)] = \mathbb{E}[\hat{f}(x)] - f(x)$$

$$\mathrm{var}(\hat{f}(x)) = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]$$

$$\mathbb{E}[\mathbb{E}[(y - \hat{f}(x))^2]] = \mathbb{E}[\mathrm{bias}[\hat{f}(x)]^2] + \mathbb{E}[\mathrm{var}(\hat{f}(x))] + \sigma_\epsilon^2$$

$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$

$$\begin{aligned}
\mathbb{E}[(y - \hat{f}(x))^2] &= \mathbb{E}[(f(x) + \epsilon - \hat{f}(x))^2] && (1)\\
&= \mathbb{E}[(f(x) - \hat{f}(x))^2] + \mathbb{E}[\epsilon^2] + 2\mathbb{E}[(f(x) - \hat{f}(x))\epsilon] \\
&= \mathbb{E}[(f(x) - \hat{f}(x))^2] + \underbrace{\mathbb{E}[\epsilon^2]}_{=\sigma_\epsilon^2} + 2\mathbb{E}[(f(x) - \hat{f}(x))]\underbrace{\mathbb{E}[\epsilon]}_{=0} && (2)\\
&= \mathbb{E}[(f(x) - \hat{f}(x))^2] + \sigma_\epsilon^2 && (3)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}[(f(x) - \hat{f}(x))^2] &= \mathbb{E}\left[\left((f(x) - \mathbb{E}[\hat{f}(x)]) - (\hat{f}(x) - \mathbb{E}[\hat{f}(x)])\right)^2\right] && (4)\\
&= \mathbb{E}\left[\left(\mathbb{E}[\hat{f}(x)] - f(x)\right)^2\right] + \mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)^2\right] \\
&\quad - 2\mathbb{E}\left[\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\right] && (5)\\
&= \underbrace{(\mathbb{E}[\hat{f}(x)] - f(x))^2}_{=\mathrm{bias}[\hat{f}(x)]} + \underbrace{\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)^2\right]}_{=\mathrm{var}(\hat{f}(x))} \\
&\quad - 2\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\right)\right] && (6)\\
&= \mathrm{bias}[\hat{f}(x)]^2 + \mathrm{var}(\hat{f}(x)) \\
&\quad - 2\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)\left(\mathbb{E}[\hat{f}(x)] - \mathbb{E}[\hat{f}(x)]\right) && (7)\\
&= \mathrm{bias}[\hat{f}(x)]^2 + \mathrm{var}(\hat{f}(x)) && (8)
\end{aligned}$$

Bias and Variance Tradeoff (3)

- http://scott.fortmann-roe.com/docs/BiasVariance.html

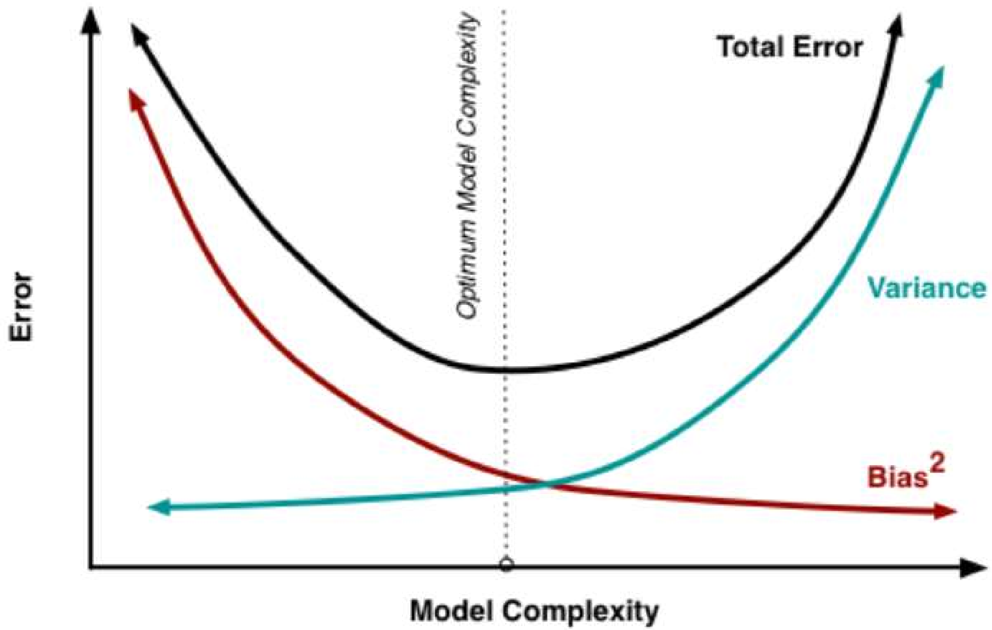


Fig. 6 Bias and variance contributing to total error.

$$\frac{dBias}{dComplexity} = -\frac{dVariance}{dComplexity}$$

편향 분산

Bias-Variance Tradeoff (4)

https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

- In parameter estimation of a model
  - A lower bias ←→ A higher variance
- Higher bias → Underfitting (too simple)
  - Miss the relevant relations between features and target outputs
- Higher variance → Overfitting (too complex)
  - Sensitivity to small fluctuations in the training set
  - Do not ignore the random noise in the training data
- Both bias and variance decrease when increasing the width of a neural network.
- Bias-variance decomposition
  - Regularization with the expected generalization error

* ksaehwa:
  - Bias : similarity를 찾음
  - Variance : difference를 찾음

Bias-Variance Tradeoff (5)

- https://medium.com/@mp32445/understanding-bias-variance-tradeoff-ca59a22e2a83
- Examples of low-bias and high-variance machine learning algorithms
  - Support Vector Machines
  - Decision Trees
  - k-Nearest Neighbors
- Examples of high-bias and low-variance machine learning algorithms
  - Linear Regression
  - Logistic Regression