Machine learning handbook

Optimizer

Taeyang Yang

Update: July 17, 2018

BCILAB, UNIST

1

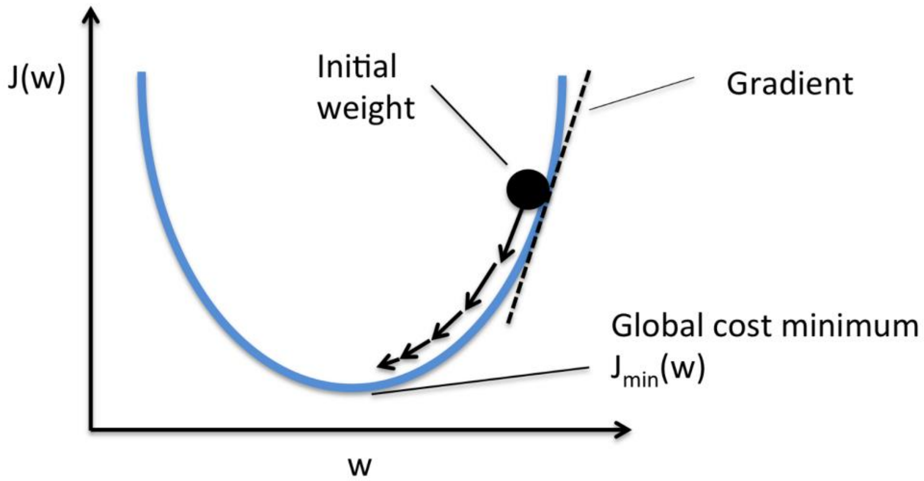
Gradient descent algorithm

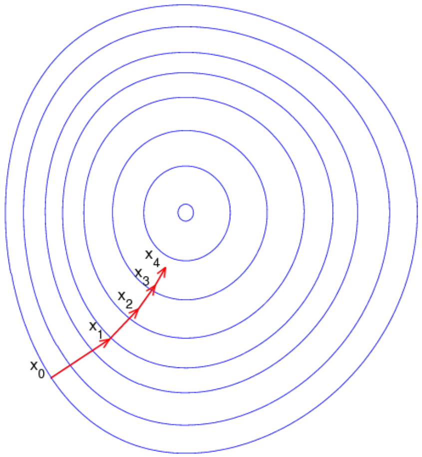
- Gradient Descent (GD)
- Full-batch Gradient Descent
- Stochastic Gradient Descent (SGD)
- Mini-batch (Stochastic) Gradient Descent

Gradient Descent (GD): algorithm

- Update rule

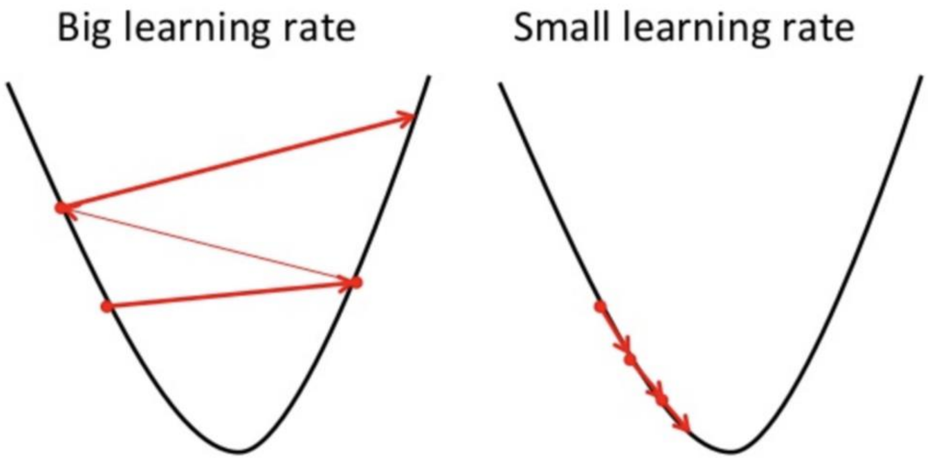
$$\theta = \theta - \eta * \frac{\partial}{\partial \theta} J(\theta)$$



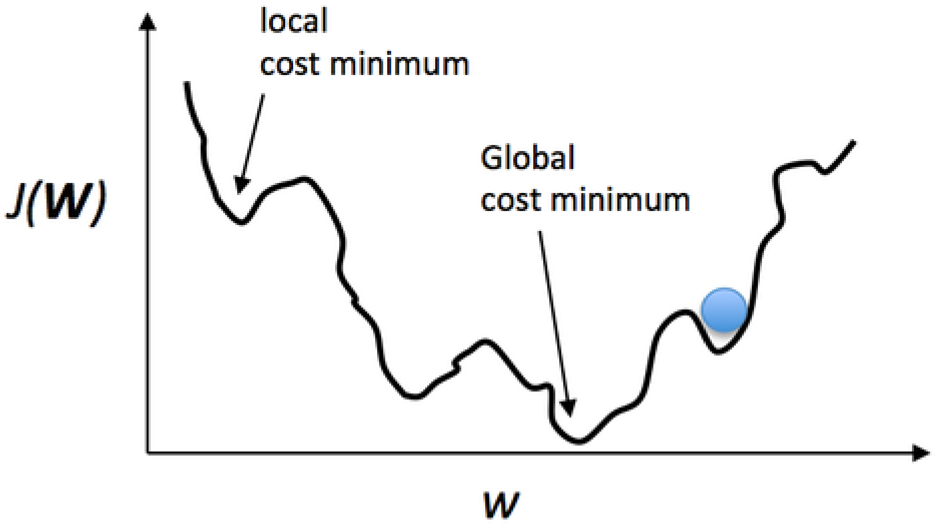


Gradient Descent (GD): Learning rate

$$\theta = \theta - \eta * \frac{\partial}{\partial\theta} J(\theta)$$

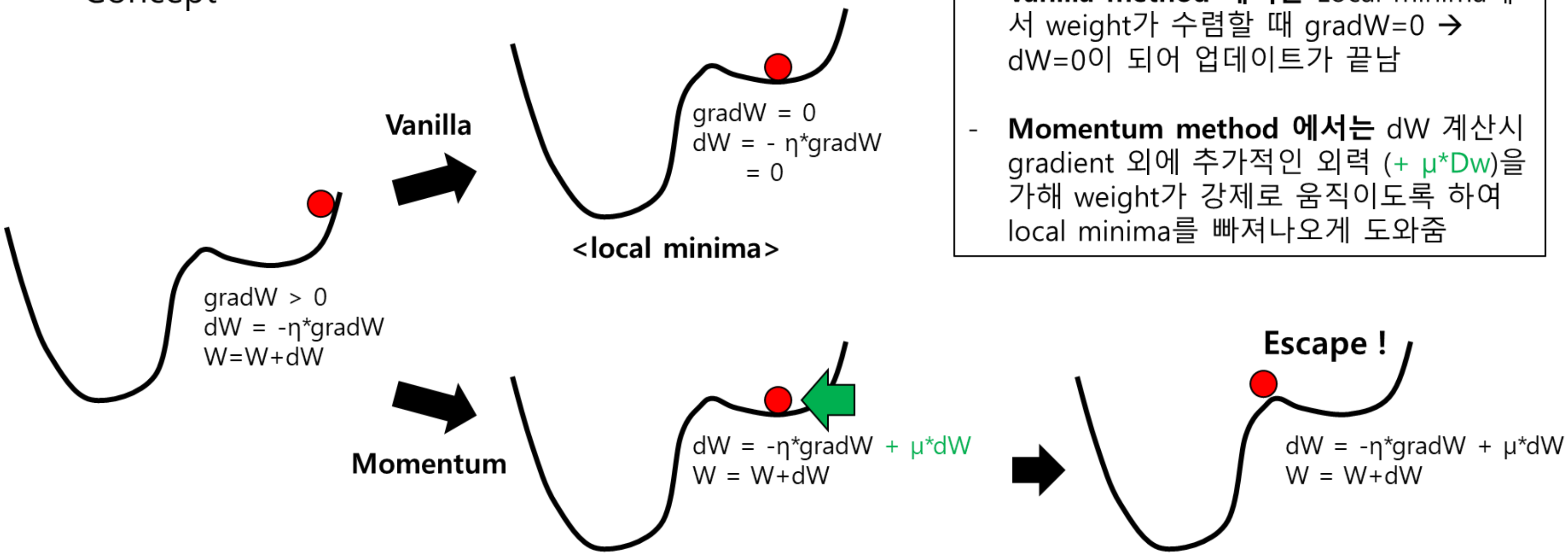


Gradient Descent (GD): Local minimum problem



Gradient Descent (GD): Momentum

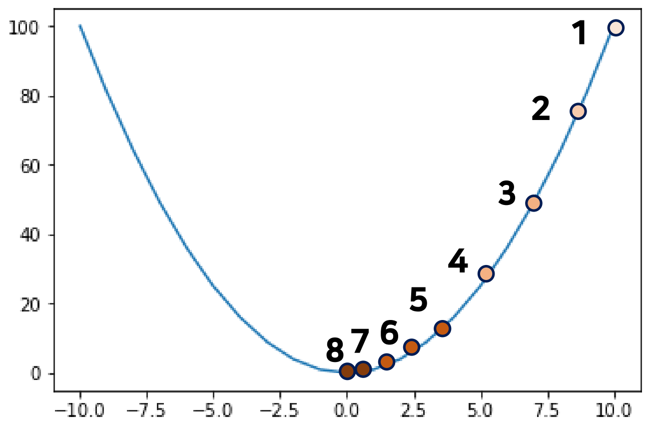
- Concept



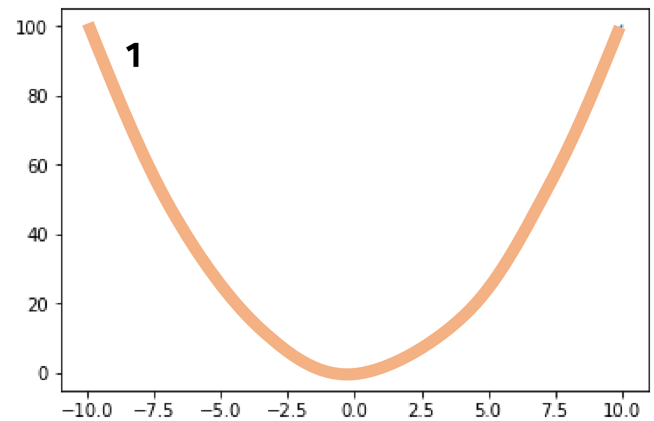
- **Vanilla method 에서는** Local minima에서 weight가 수렴할 때 gradW=0 → dW=0이 되어 업데이트가 끝남
- **Momentum method 에서는** dW 계산시 gradient 외에 추가적인 외력 (+ μ*Dw)을 가해 weight가 강제로 움직이도록 하여 local minima를 빠져나오게 도와줌

Comparison

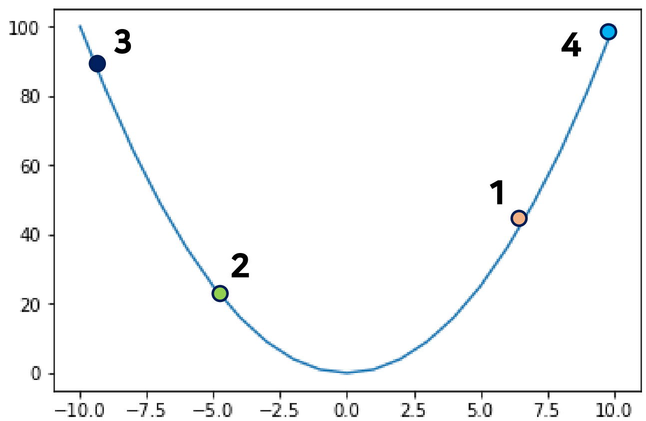
Gradient descent



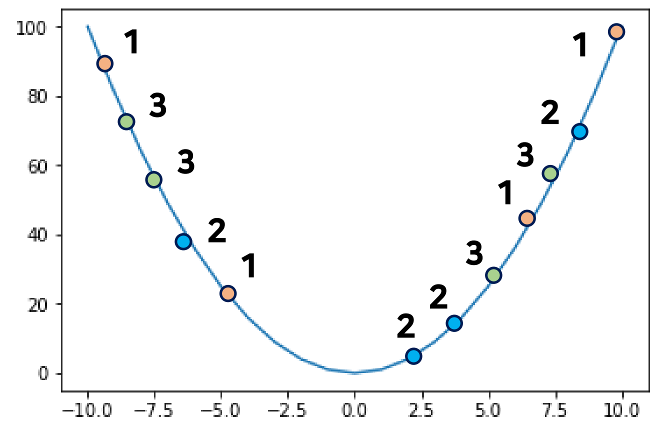
Full-batch gradient descent



Stochastic gradient descent



Mini-batch gradient descent



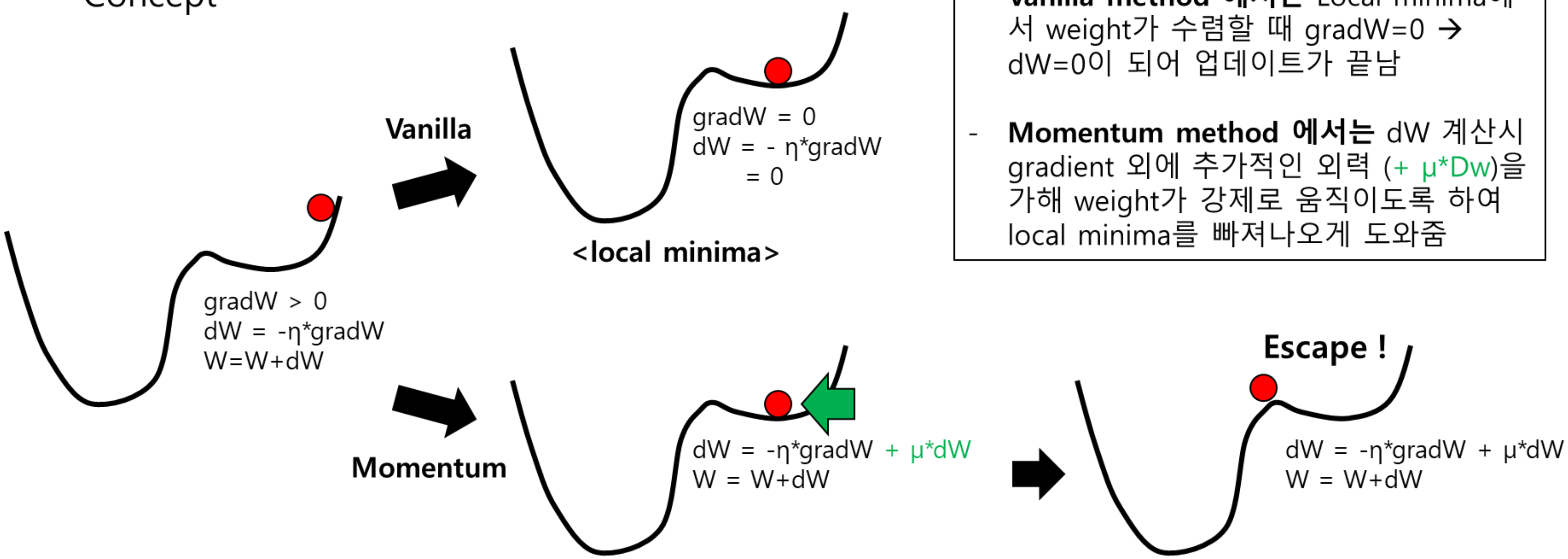
2

Further materials

- Momentum
- Regularization (Weight decay)
- Learning rate decay
- Early stop

Momentum

- Concept



- **Vanilla method 에서는** Local minima에서 weight가 수렴할 때 gradW=0 → dW=0이 되어 업데이트가 끝남
- **Momentum method 에서는** dW 계산시 gradient 외에 추가적인 외력 (+ μ*Dw)을 가해 weight가 강제로 움직이도록 하여 local minima를 빠져나오게 도와줌

Momentum

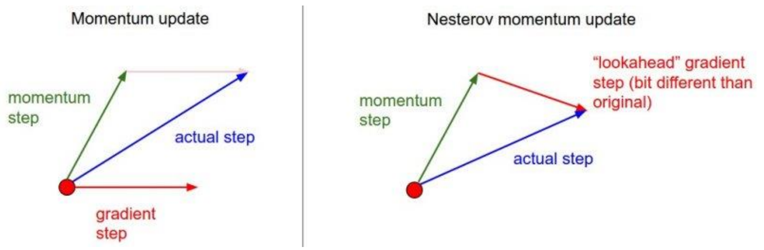
- Parameters update methods
  - **Vanilla update**

$$\Delta Weight_t = -\eta * Gradient_t$$
$$Weight_t = Weight_{t-1} + \Delta Weight_t$$

  - **Momentum update**
    - 직전 w 업데이트 ($\Delta Weight_{t-1}$) * momentum(μ)를 현재 w 업데이트에 추가

$$\Delta Weight_t = -\eta * Gradient_t + \mu * \Delta Weight_{t-1}$$
$$Weight_t = Weight_{t-1} + \Delta Weight_t$$

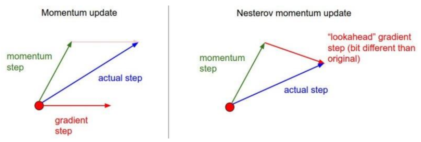
  - **Nesterov momentum update**
    - Momentum update와 비슷하지만, momentum를 더한 위치에서 Gradient 계산



Bengio, Y., Boulanger-Lewandowski, N., & Pascanu, R. (2013, May). ***Advances in optimizing recurrent networks.*** *In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on (pp. 8624-8628). IEEE.*

Ilya Sutskever's thesis*, http://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf*

Momentum: Nesterov momentum update



- Parameters update methods
  - **Nesterov momentum update**
    - Momentum update와 비슷하지만, momentum를 더한 위치에서 Gradient 계산

$$Weight\_ahead_t = Weight_t + \mu * \Delta Weight_{t-1}$$

- $Weight\_ahead_t$ 이용해서 $Gradient\_ahead_t$ 계산

$$\Delta Weight_t = -\eta * Gradient_{ahead_t} + \mu * \Delta Weight_{t-1}$$
$$Weight_t = Weight_{t-1} + \Delta Weight_t$$

- 정리된 표현

$$\Delta Weight_t = -\eta * Gradient_t + \mu * \Delta Weight_{t-1}$$
$$Weight_t = Weight_{t-1} + \Delta Weight_t + \mu * \Delta Weight_t - \mu * \Delta Weight_{t-1}$$
$$= Weight_{t-1} + (1 + \mu) * \Delta Weight_t - \mu * \Delta Weight_{t-1}$$

Regularization (Weight decay)

- Cost function에 weight에 대한 Penalty를 부여하여, Weight 들의 크기에 제한을 줌
  - 극단적인 weight 값 방지 → Overfitting 방지

- 종류
  - **L1 regularization (Lasso)**: Manhattan distance

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n}|\theta_j|$$

  - **L2 regularization (Ridge)**: Euclidean distance
    - 더 많은 Penalty ← 제곱이니까 !

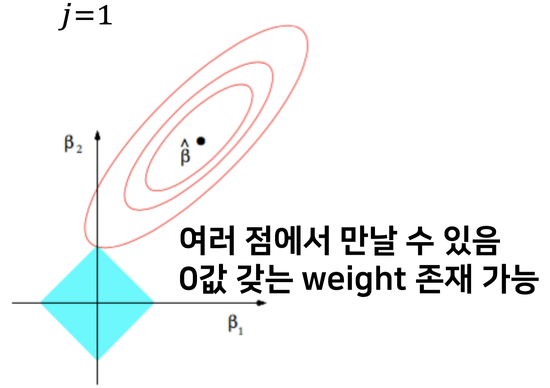
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n}\theta_j^2$$

Regularization (Weight decay): L1 vs L2

L1 regularization (Lasso)

- Unstable solution (여러 접점)
- Always on solution
- Sparse solution (weight = 0 도 존재함)
- Feature selection (weight = 0 → 덜 중요)

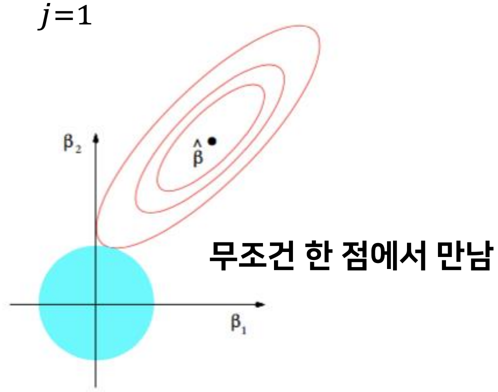
$$\sum_{j=1}^{n} |\theta_j| \leq s$$



L2 regularization (Ridge)

- Stable solution
- Only one solution
- Non-sparse solution

$$\sum_{j=1}^{n} \theta_j^2 \leq s$$



S가 크면 파란 면적 (weight가 가질 수 있는 범위)이 넓어짐

Regularization (Weight decay)

- $\Delta\theta$ 식에 바로 적용할 수도 있다 !
  - Weight 각각의 일정 비율 $(\lambda, lambda)$ 을 곱해준 값을 gradient 값에서 빼준다.
  - 이 때, learning rate 곱하기 전에 빼주기 !
  - 매우 작은 값을 사용한다 (RBM에서는 0.0002)

  - 예시 1: Vanilla update에서,

$$\Delta\theta = \eta(\frac{\partial E}{\partial \theta} - \lambda\theta)$$

  - 예시 2: Momentum update에서,

$$\Delta\theta = \eta\left(\frac{\partial E}{\partial \theta} - \lambda\theta\right) + \mu\Delta\theta$$

Learning rate decay

- 일정 주기로 Learning rate를 감소
- 특정 epoch마다 learning rate를 감소 → Hyper-parameter 설정의 어려움 생김
- SGD 에서는 필수적으로 사용하는 알고리즘
- **지수감소**

$$\eta = \eta_0 e^{-kt}$$

- **1/t감소**

$$\eta = \frac{\eta_0}{(1 + kt)}$$

Early stop

- 특정 값 이하로 cost function이 줄어들지 않거나, 변동이 없을 경우 종료조건 설정
- Hyper-parameter 설정 필요