

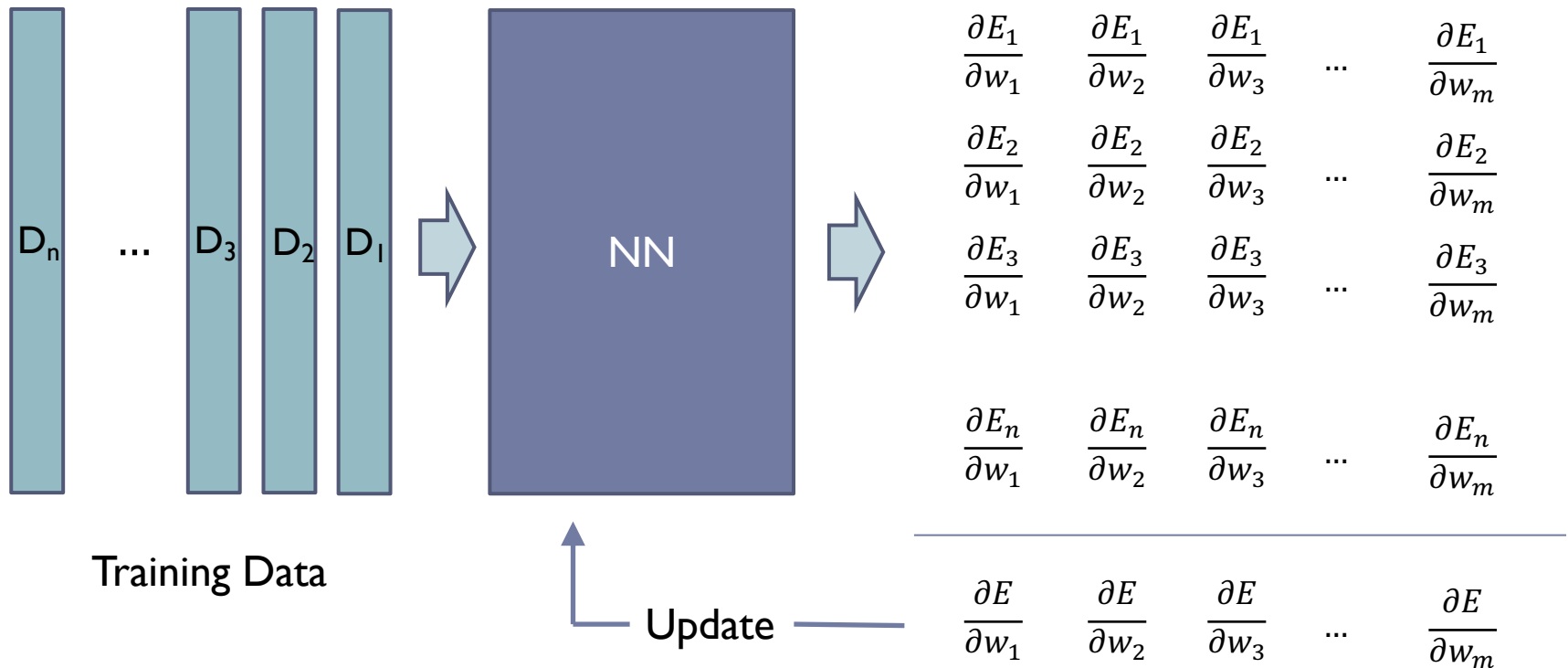
Gradient Descent Methods

차례

- ▶ **Stochastic Gradient Descent Method**
- ▶ **Momentum**
- ▶ **Adaptive Learning Rates**
 - ▶ Adagrad
 - ▶ RMSprop
 - ▶ Adam

Gradient Descent Method

► Error Back Propagation (Batch mode)



Batch Gradient Descent

▶ Algorithm

- ▶ For one update, gradients are calculated for the whole dataset
- ▶ Batch gradient descent is guaranteed to converge to a local minimum
- ▶ Redundant computations for large redundant dataset

Repeat

$$\alpha = 0$$

for $n = 1$ to N (for all training data)

$$\alpha += \frac{\partial E_n}{\partial w}$$

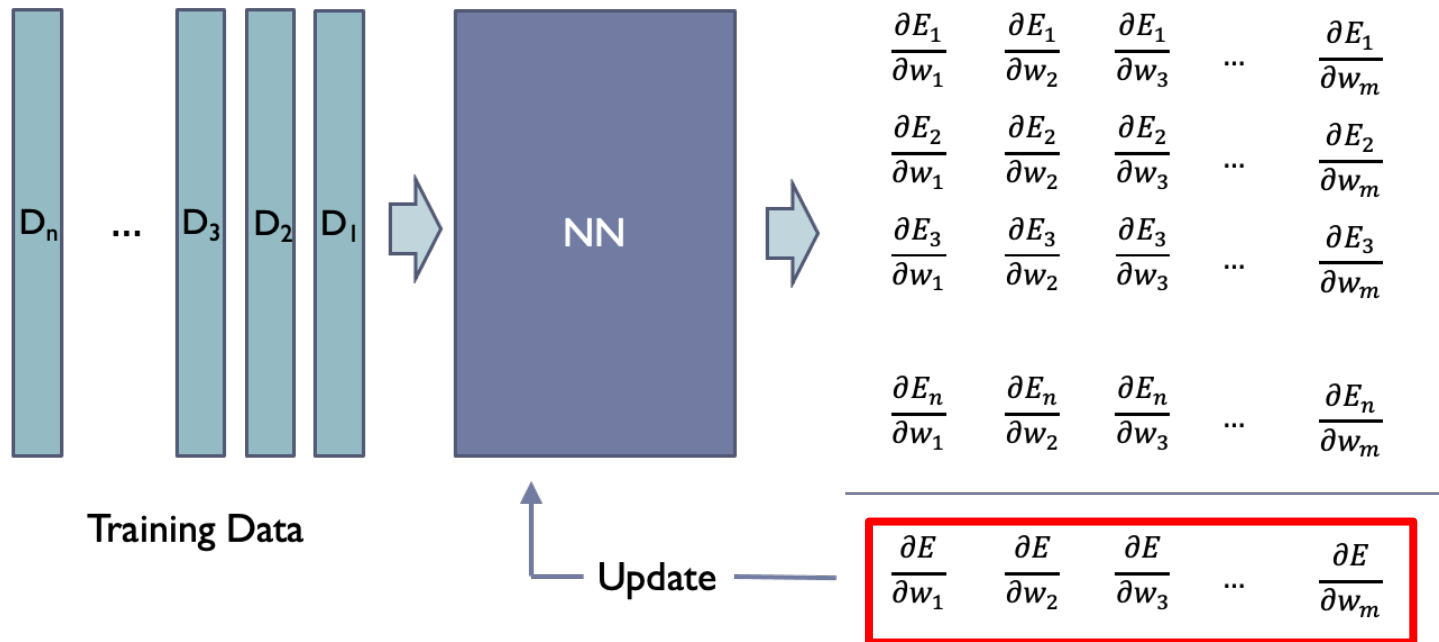
end

$$w = w - \eta \alpha$$

Until end condition satisfied

Gradient Descent Method

▶ Error Back Propagation (Batch mode)



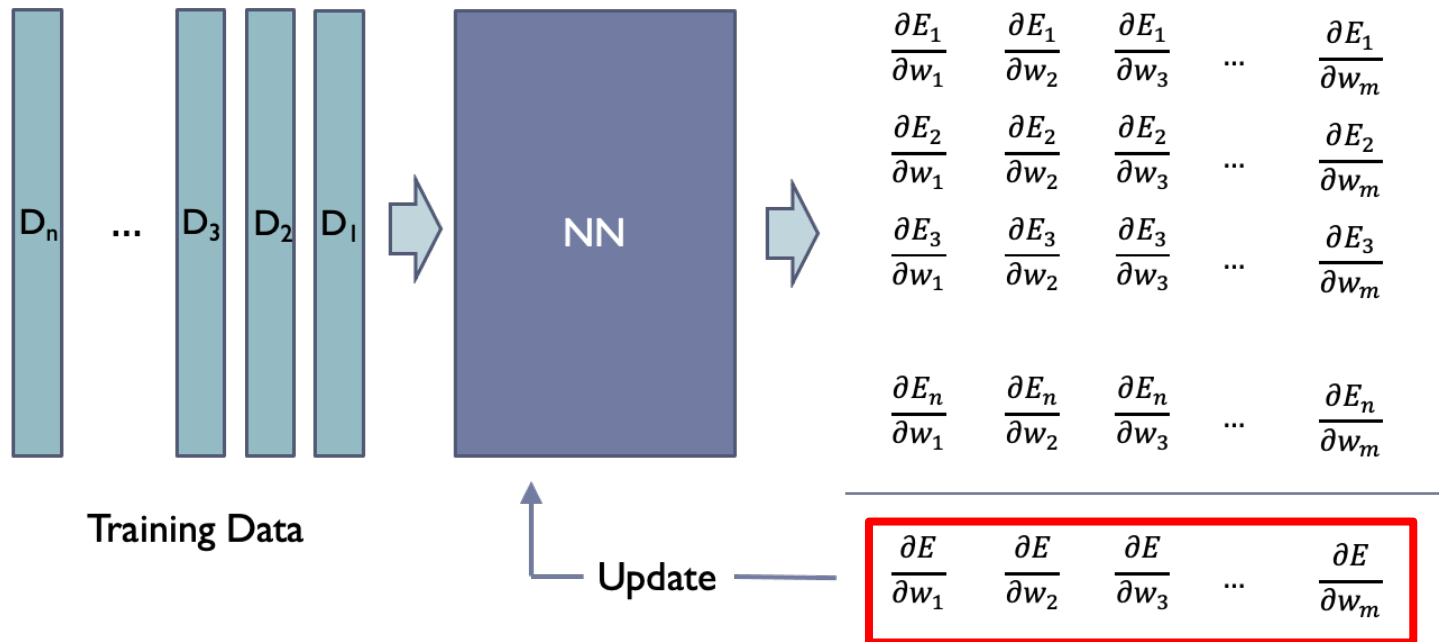
Exact Gradients!! However, too much cost for them !!

Do we need EXACT gradients?

How about ESTIMATING the gradient with cheap cost?

Gradient Descent Method

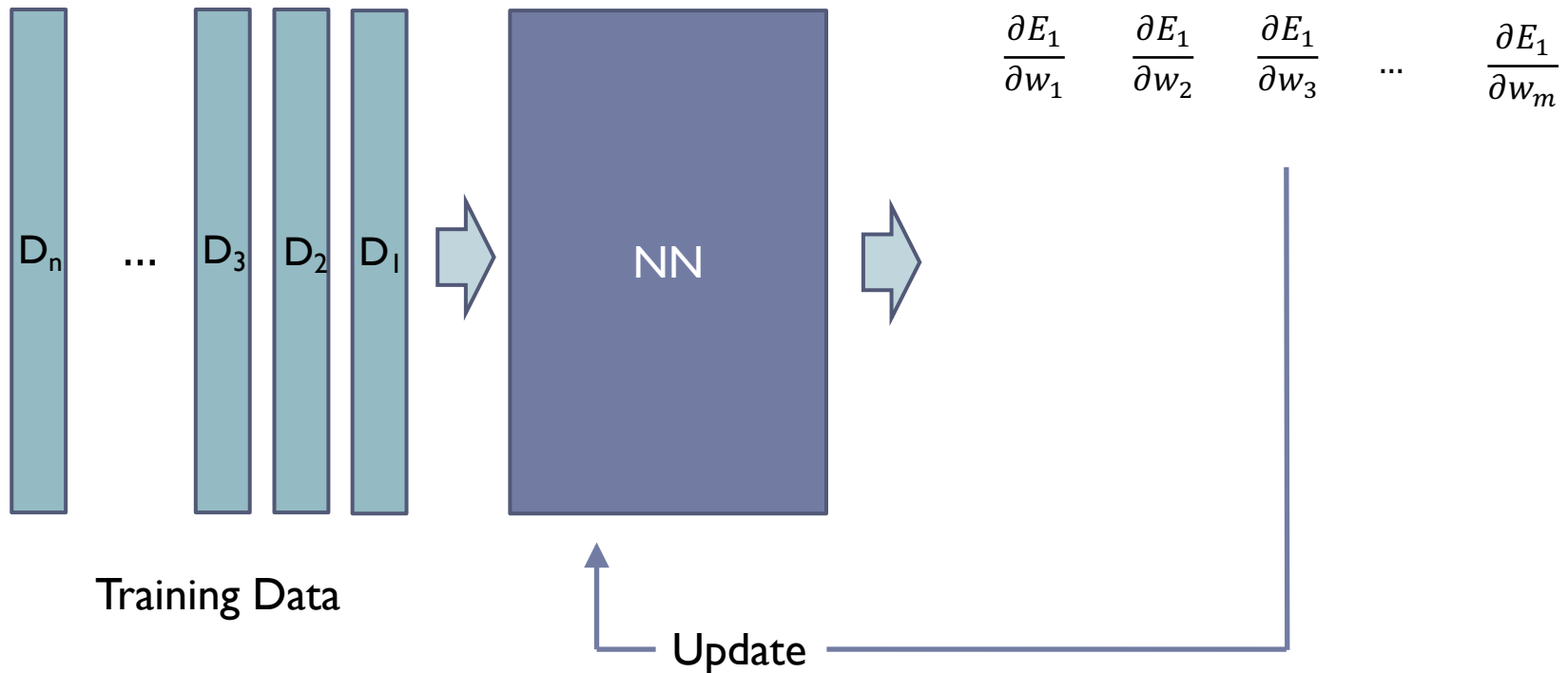
▶ Error Back Propagation (Batch mode)



$$\frac{\partial E}{\partial w_1} = \frac{1}{n} \sum_{i=1}^n \frac{\partial E_i}{\partial w_1} \quad \frac{\partial E}{\partial w_1} \approx \frac{\partial E_1}{\partial w_1} \approx \frac{\partial E_2}{\partial w_1} \approx \dots \approx \frac{\partial E_n}{\partial w_1}$$

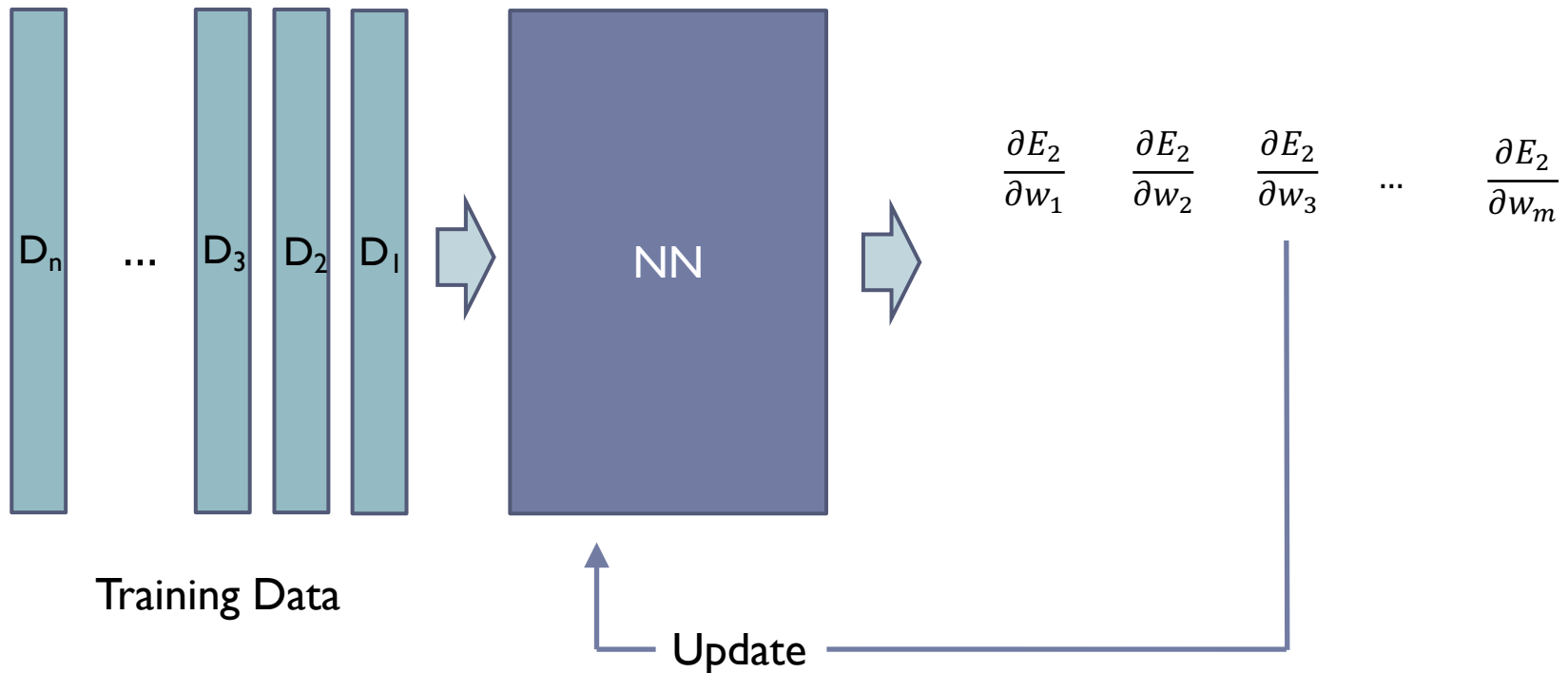
Stochastic Gradient Descent

► Why not?



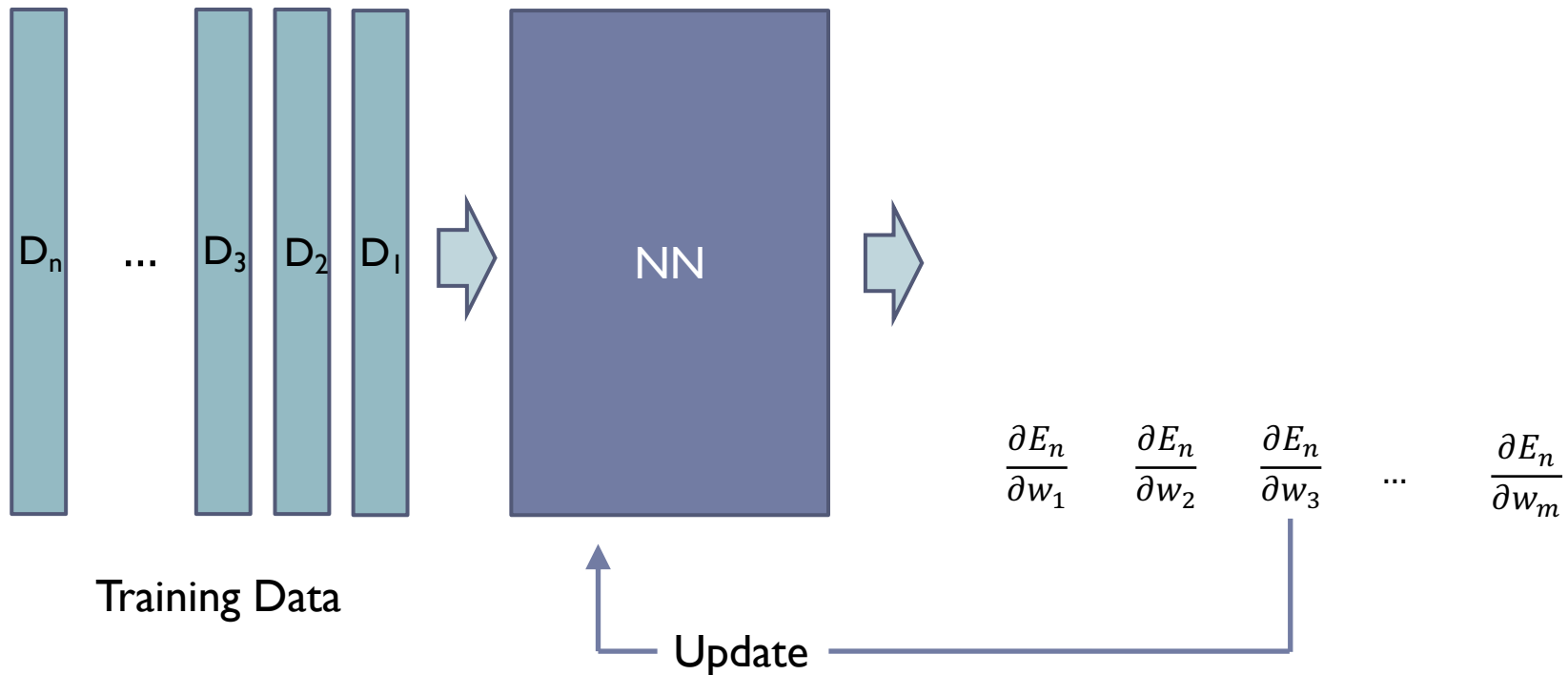
Stochastic Gradient Descent

► Why not?



Stochastic Gradient Descent

► Why not?



Stochastic Gradient Descent

▶ Stochastic Gradient Descent

- ▶ For one update, gradients are calculated for one sample
- ▶ Usually faster and can be used to learn online
- ▶ Fluctuations: Maybe good or maybe bad
- ▶ With a small learn rate, show similar performance

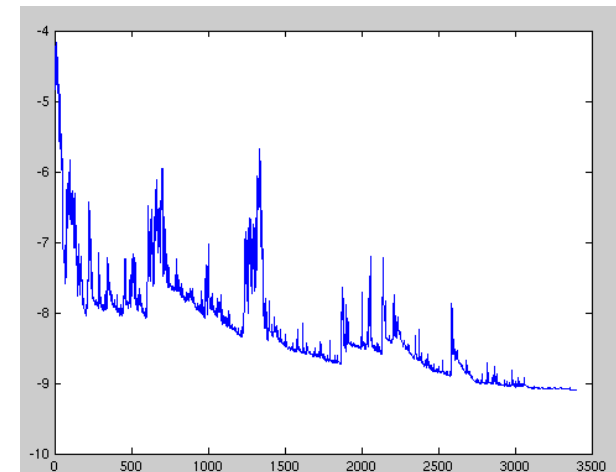
Repeat

for $n = 1$ to N (for all training data)

$$w = w - \eta \frac{\partial E_n}{\partial w}$$

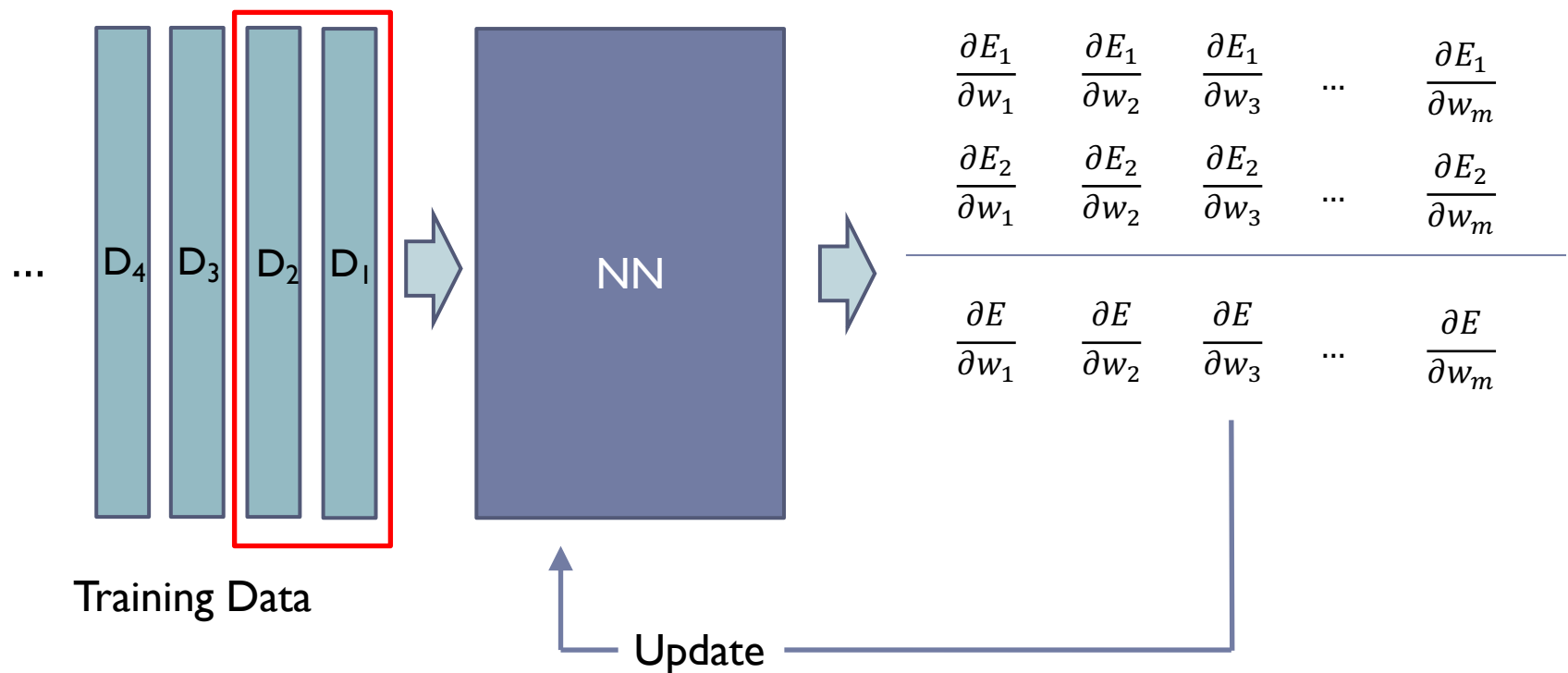
end

Until end condition satisfied



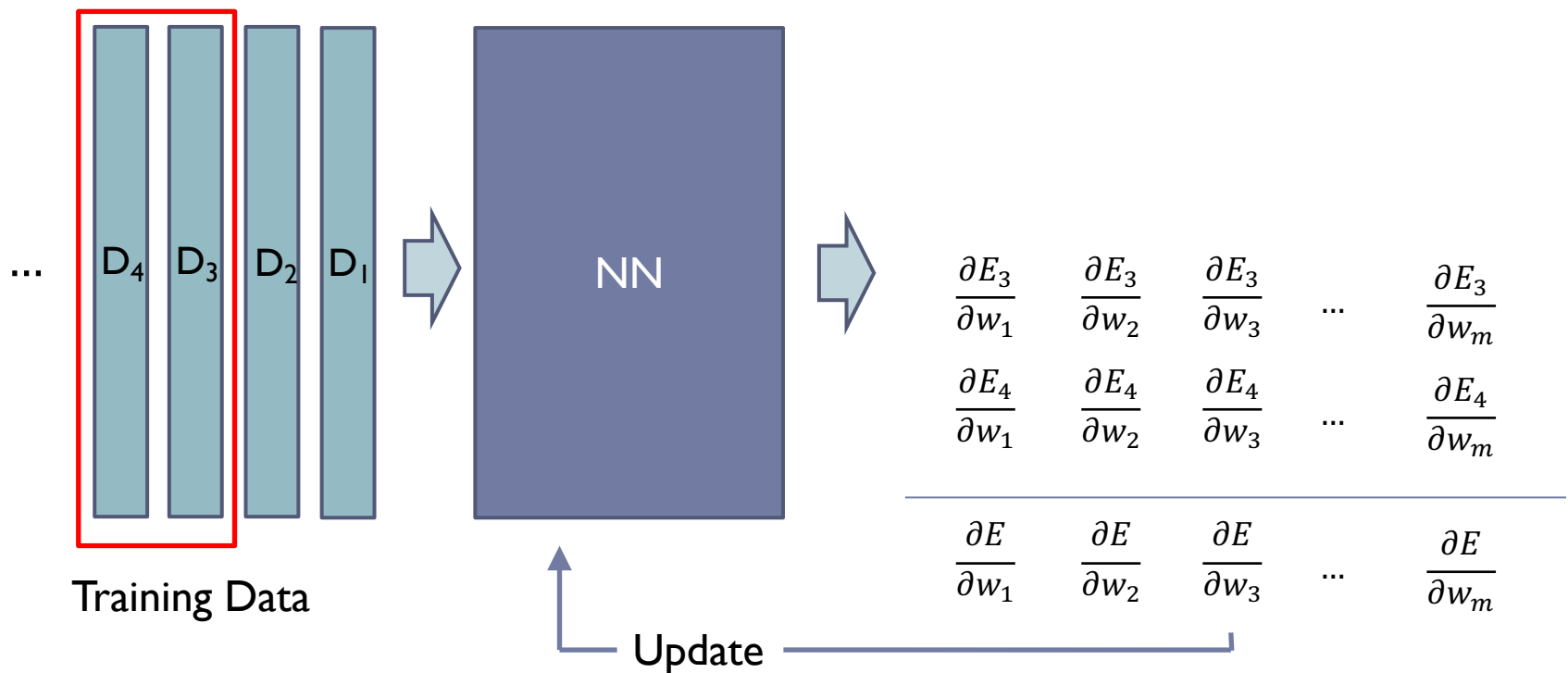
Stochastic Gradient Descent

▶ Mini-batch Gradient Descent Method



Stochastic Gradient Descent

▶ Mini-batch Gradient Descent Method



Stochastic Gradient Descent

▶ Mini-batch Gradient Descent

- ▶ For an update, gradients are calculated for a batch

Repeat

for $b = 1$ to B (for all batches)

$$\alpha = 0$$

for $n = 1$ to N_b (for all training data in batch b)

$$\alpha += \frac{\partial E_n}{\partial w}$$

end

$$w = w - \eta \alpha$$

end

Until end condition satisfied

Stochastic Gradient Descent

▶ Usual Batch Size

- ▶ Dependent on datasets from several thousands to several tens

▶ Advantage

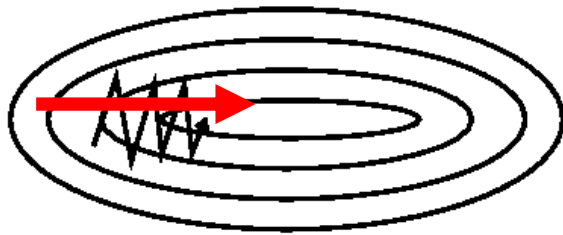
- ▶ Good estimation of real gradient
- ▶ High throughput: may use the large number of cores at once in a GPU.
- ▶ Faster convergence: Good estimation + High throughput

▶ Disadvantage

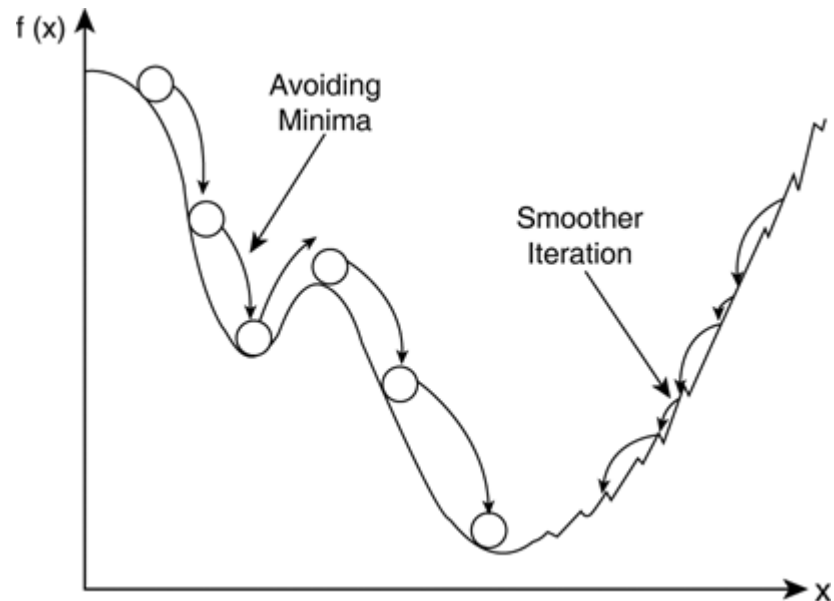
- ▶ Inaccurate: dataset with large variances

Momentum

- ▶ Simple gradient method depends on the current position
 - ▶ Hard to avoid local minimum
 - ▶ Gradient can vary much

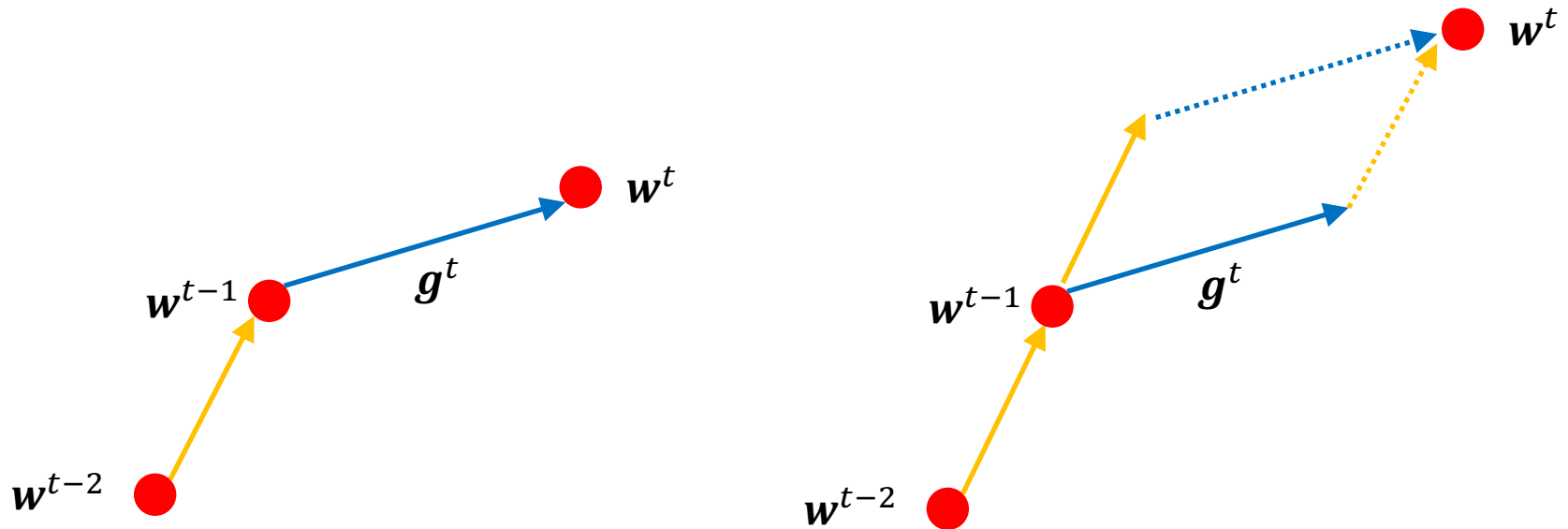


Oscillation



Momentum

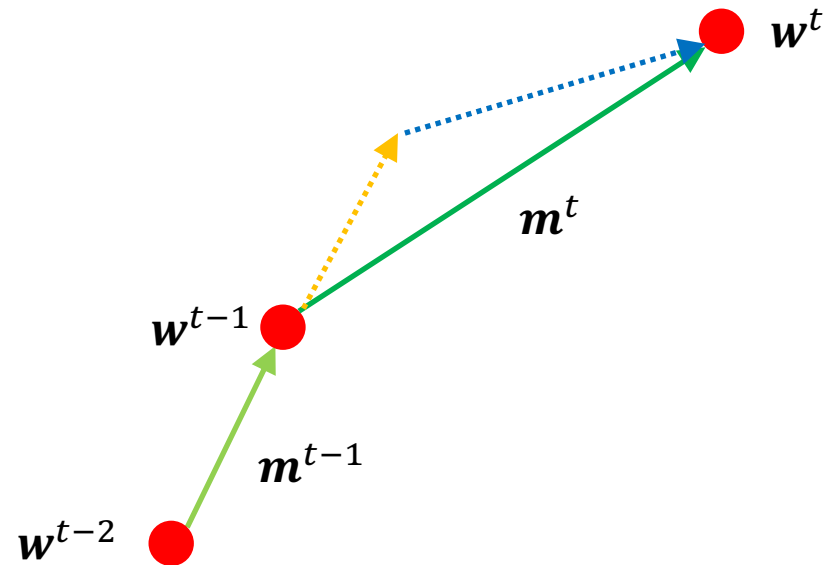
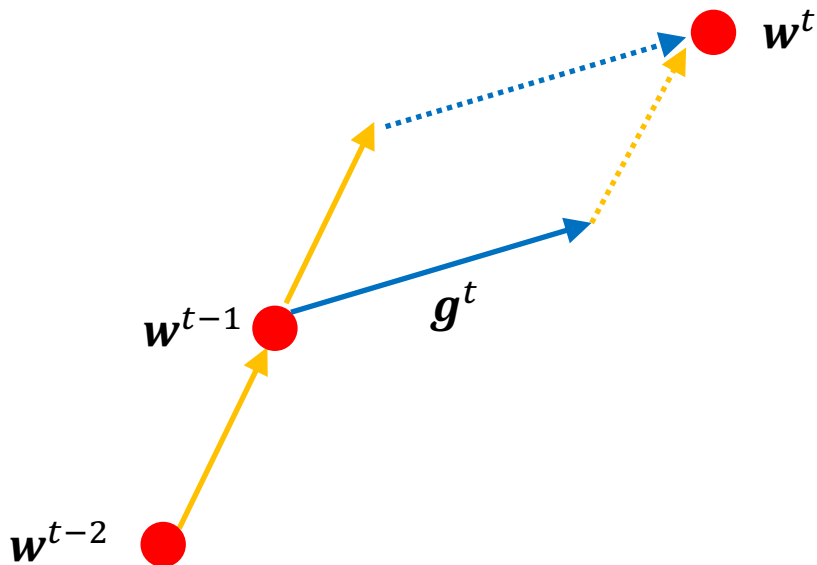
▶ How to cross the hill



Momentum

► Momentum

$$m^t = m^{t-1} + g^t$$



Momentum

▶ Momentum

g^t : 시간 t 에서의 기울기

Momentum rate

Learning rate

$$w^t = w^{t-1} - \eta g^t$$

표준 방법

$$\begin{aligned} m^t &= \gamma m^{t-1} + \eta g^t \\ w^t &= w^{t-1} - m^t \end{aligned}$$

Momentum

- ▶ Update parameters considering both the momentum and the gradient of the current position

Momentum

► Momentum

- Momentum is the exponential average of the past gradients

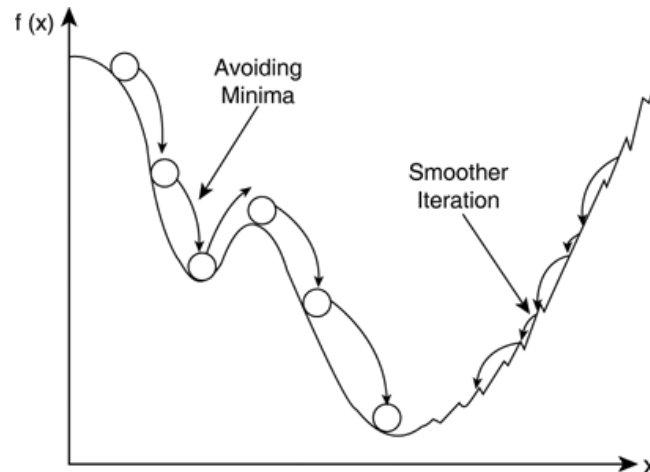
$$\mathbf{m}^t = \eta \mathbf{g}^t + \gamma \eta \mathbf{g}^{t-1} + \gamma^2 \eta \mathbf{g}^{t-2} + \dots$$

$$\mathbf{m}^t = \gamma \mathbf{m}^{t-1} + \eta \mathbf{g}^t$$

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \mathbf{m}^t$$



Oscillation

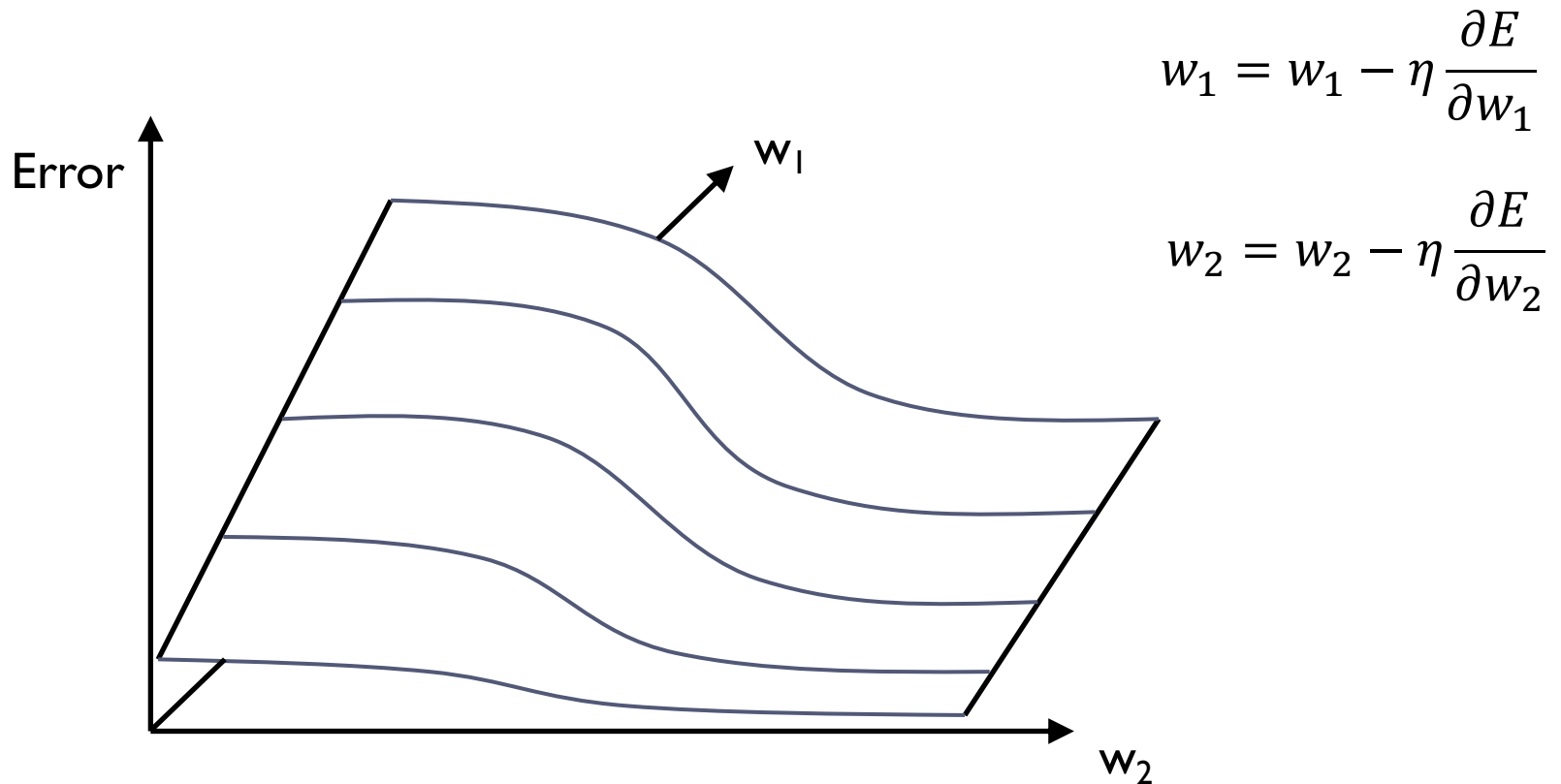


Adaptive Learning Rates

- ▶ **Why do we use the SAME learning rate for all the weights?**
 - ▶ Can we use different learning rate?
 - ▶ Hmm.. Interesting, but why do we need different learning rate?
- ▶ **Some weight are updated much and some are not**
 - ▶ What if some inputs may be 0 but some may have non-zero values in most of training data
 - ▶ OK.. Let's make a large update for the less updated parameters

Adaptive Learning Rates

- ▶ Some weight are updated much and some are not



Adaptive Learning Rates

▶ Adagrad

- ▶ Update much less-updated parameters
- ▶ Update less much-updated parameter

g_i^t : parameter i 의 시간 t 에서의 기울기

G_i^t : parameter i 가 지금까지 update된 총량

$$w_i^t = w_i^{t-1} - \eta g_i^t$$

표준 방법

$$G_i^t = G_i^{t-1} + (g_i^t)^2$$
$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$

Adagrad

Very small value

Adaptive Learning Rates

► RMSProp

- Instead of considering the total amount of updates, let's consider the amount of recent updates

g_i^t : parameter i 의 시간 t 에서의 기울기

G_i^t : parameter i 가 update된 양

$$G_i^t = G_i^{t-1} + (g_i^t)^2$$
$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$

Adagrad

$$G_i^t = \gamma G_i^{t-1} + (1 - \gamma)(g_i^t)^2$$
$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$

RMSProp

Very small value

Adaptive Learning Rates

▶ Adam

▶ RMSProp + Momentum

g_i^t : parameter i 의 시간 t 에서의 기울기

G_i^t : parameter i 가 update된 양

Momentum

$$m_i^t = \gamma m_i^{t-1} + \eta g_i^t$$

$$w_i^t = w_i^{t-1} - m_i^t$$



$$m_i^t = \beta_1 m_i^{t-1} + (1 - \beta_1) g_i^t$$

RMSProp

$$G_i^t = \gamma G_i^{t-1} + (1 - \gamma) (g_i^t)^2$$

$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t$$



$$G_i^t = \beta_2 G_i^{t-1} + (1 - \beta_2) (g_i^t)^2$$

Adaptive Learning Rates

► Adam

► RMSProp + Momentum

$$m_i^t = \beta_1 m_i^{t-1} + (1 - \beta_1) g_i^t$$

$$G_i^t = \beta_2 G_i^{t-1} + (1 - \beta_2) (g_i^t)^2$$

Unbiased expectation

$$\hat{m}_i^t = \frac{m_i^t}{1 - (\beta_1)^t} \quad \hat{G}_i^t = \frac{G_i^t}{1 - (\beta_2)^t}$$

$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{\hat{G}_i^t + \epsilon}} \hat{m}_i^t$$

Adaptive Learning Rates

► Adam

► RMSProp + Momentum

$$\begin{aligned}m_i^t &= \gamma m_i^{t-1} + \eta g_i^t \\w_i^t &= w_i^{t-1} - m_i^t\end{aligned}$$

Momentum

$$\begin{aligned}G_i^t &= \gamma G_i^{t-1} + (1 - \gamma)(g_i^t)^2 \\w_i^t &= w_i^{t-1} - \frac{\eta}{\sqrt{G_i^t + \epsilon}} g_i^t\end{aligned}$$

RMSProp

$$\begin{aligned}m_i^t &= \beta_1 m_i^{t-1} + (1 - \beta_1) g_i^t \\G_i^t &= \beta_2 G_i^{t-1} + (1 - \beta_2)(g_i^t)^2\end{aligned}$$

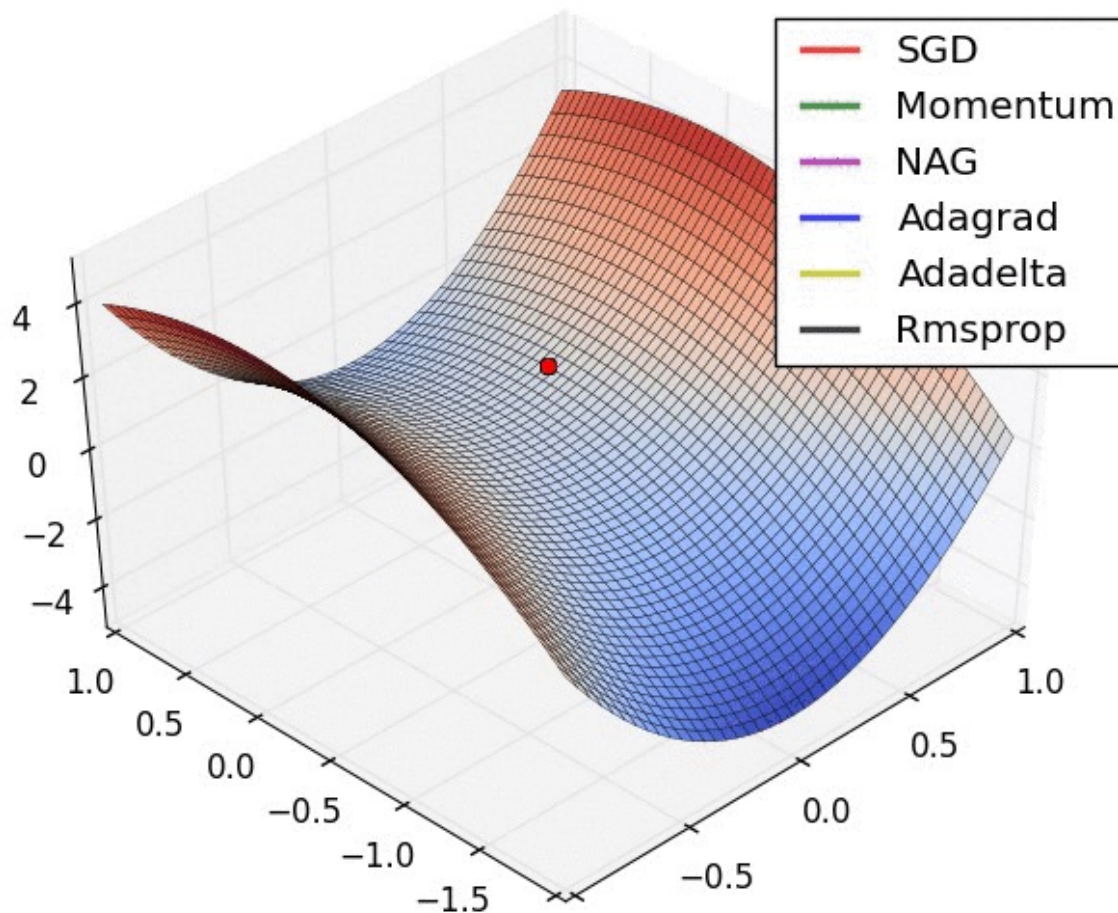
$$\hat{m}_i^t = \frac{m_i^t}{1 - (\beta_1)^t} \quad \hat{G}_i^t = \frac{G_i^t}{1 - (\beta_2)^t}$$

$$w_i^t = w_i^{t-1} - \frac{\eta}{\sqrt{\hat{G}_i^t + \epsilon}} \hat{m}_i^t$$

Adam

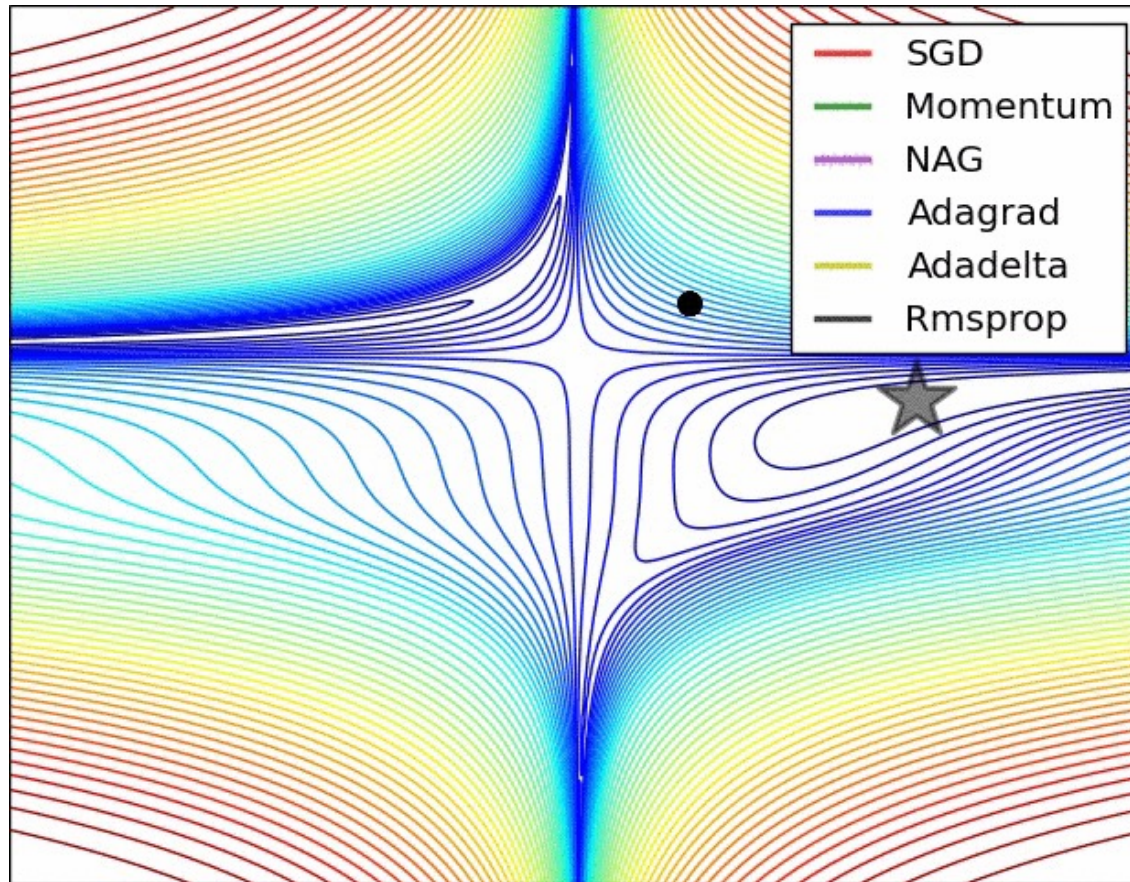
Adaptive Learning Rates

▶ Comparison: Long valley



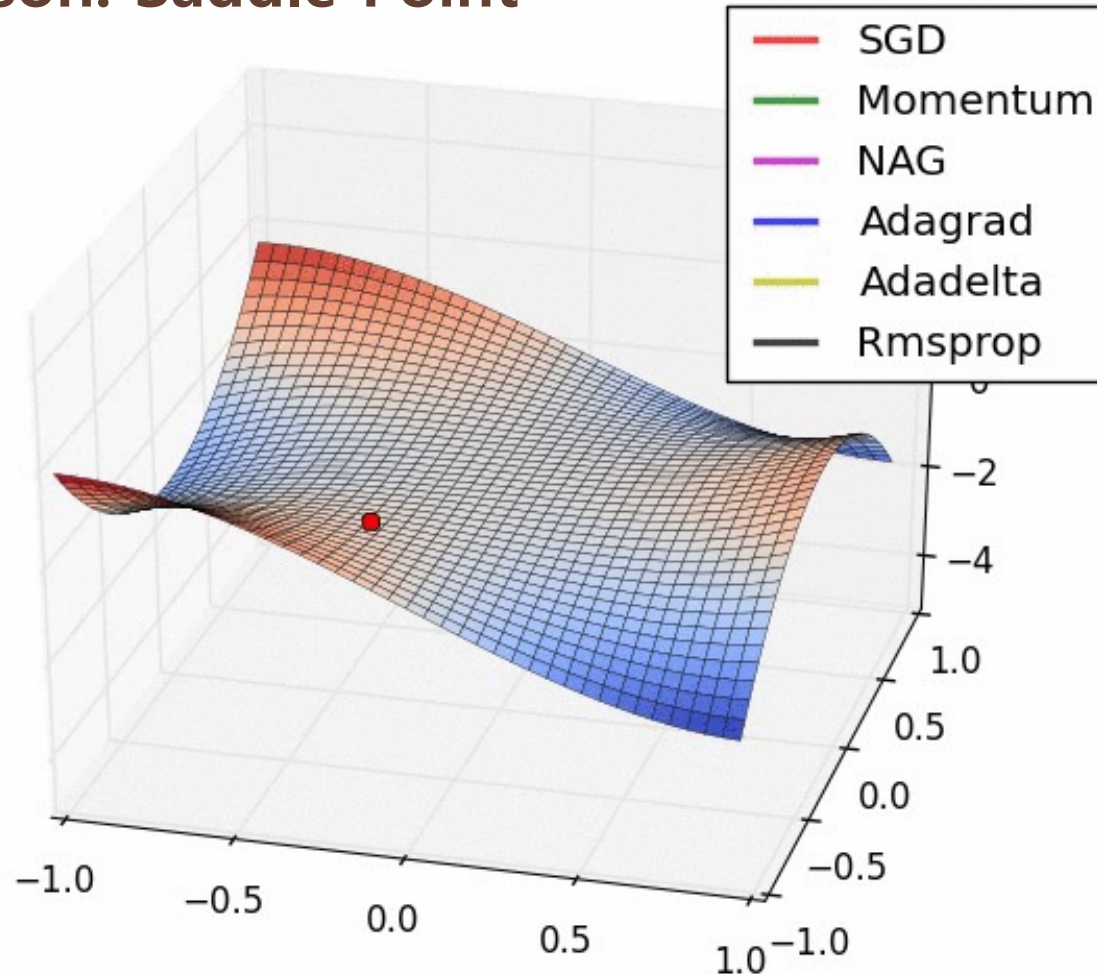
Adaptive Learning Rates

▶ Comparison: Beale's Function



Adaptive Learning Rates

▶ Comparison: Saddle Point



Question and Answer