Gradient decent

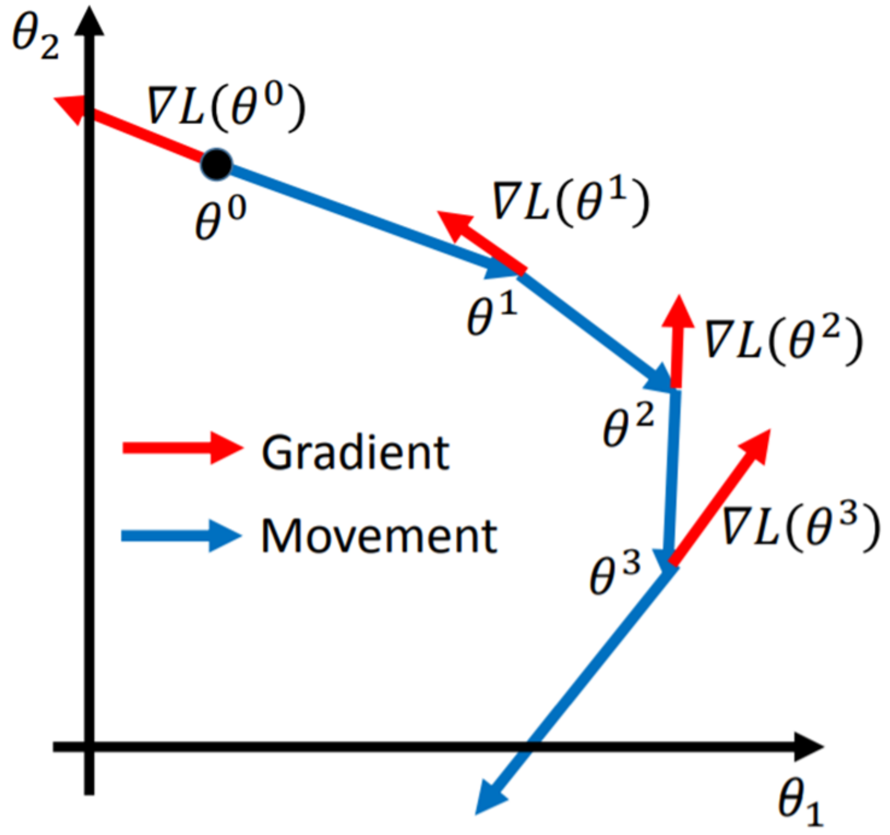
Suppose that θ has two variables $\{\theta_1, \theta_2\}$

Randomly start at $\theta^0 = \begin{bmatrix}\theta_1^0 \\ \theta_2^0\end{bmatrix}$

$$\boxed{\nabla L(\theta) = \begin{bmatrix}\partial L(\theta_1)/\partial\theta_1 \\ \partial L(\theta_2)/\partial\theta_2\end{bmatrix}}$$

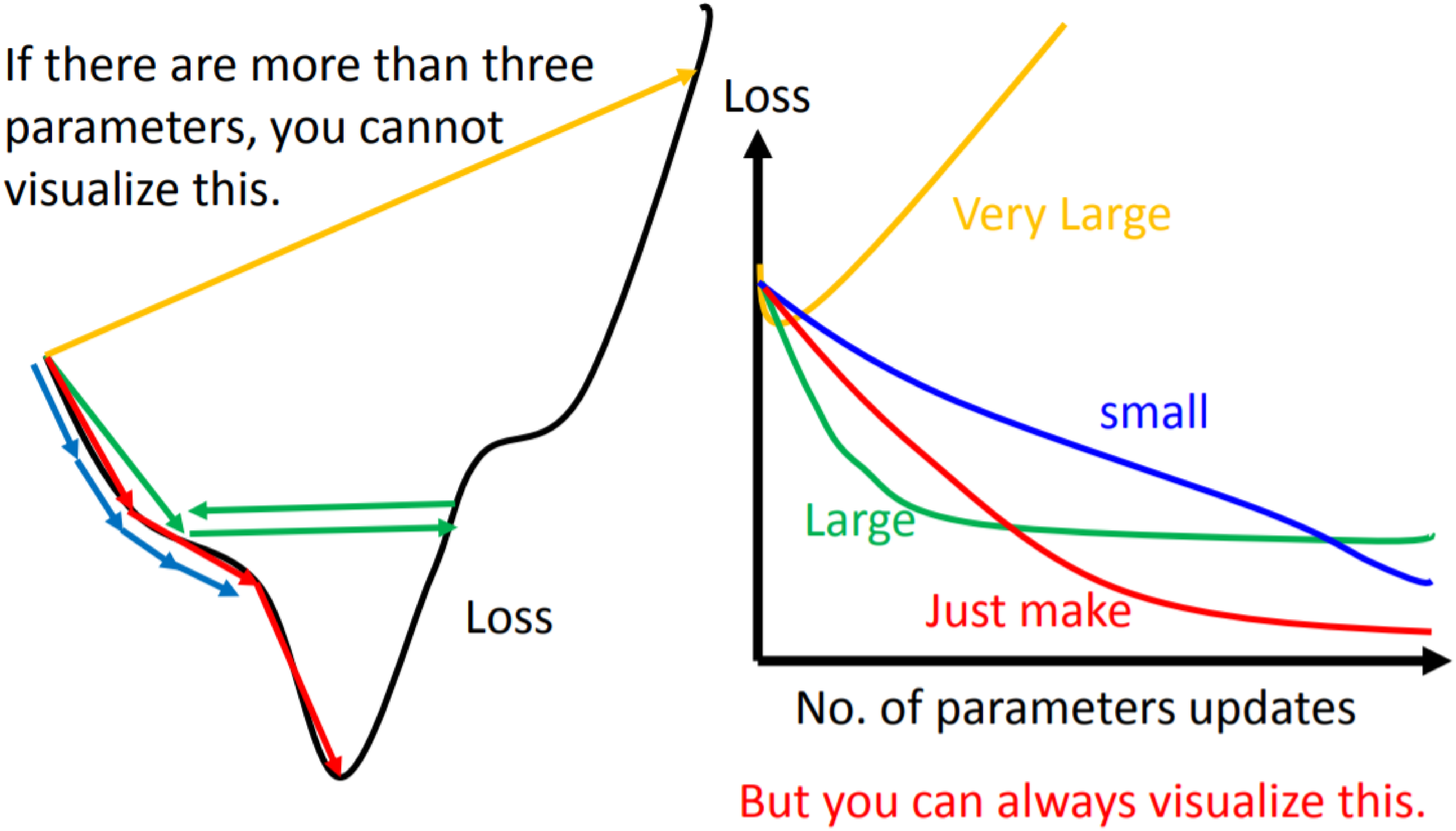
$$\begin{bmatrix}\theta_1^1 \\ \theta_2^1\end{bmatrix} = \begin{bmatrix}\theta_1^0 \\ \theta_2^0\end{bmatrix} - \eta \begin{bmatrix}\partial L(\theta_1^0)/\partial\theta_1 \\ \partial L(\theta_2^0)/\partial\theta_2\end{bmatrix} \Rightarrow \theta^1 = \theta^0 - \eta\nabla L(\theta^0)$$

$$\begin{bmatrix}\theta_1^2 \\ \theta_2^2\end{bmatrix} = \begin{bmatrix}\theta_1^1 \\ \theta_2^1\end{bmatrix} - \eta \begin{bmatrix}\partial L(\theta_1^1)/\partial\theta_1 \\ \partial L(\theta_2^1)/\partial\theta_2\end{bmatrix} \Rightarrow \theta^2 = \theta^1 - \eta\nabla L(\theta^1)$$



Reference: 李弘毅 ML Lecture 3-1 https://youtu.be/yKKNr-QKz2Q

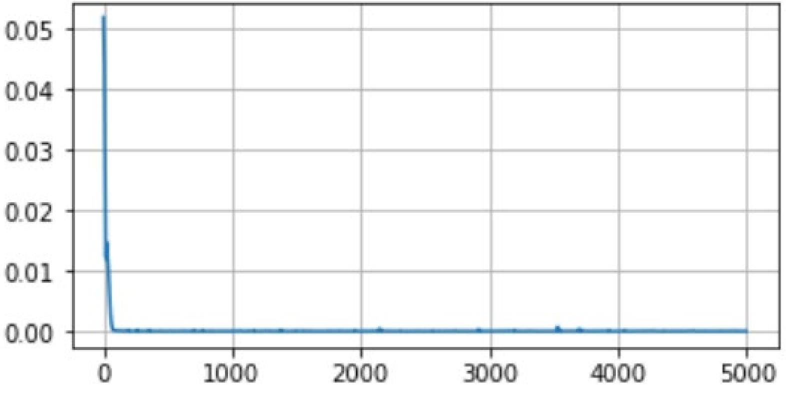
Learning rate



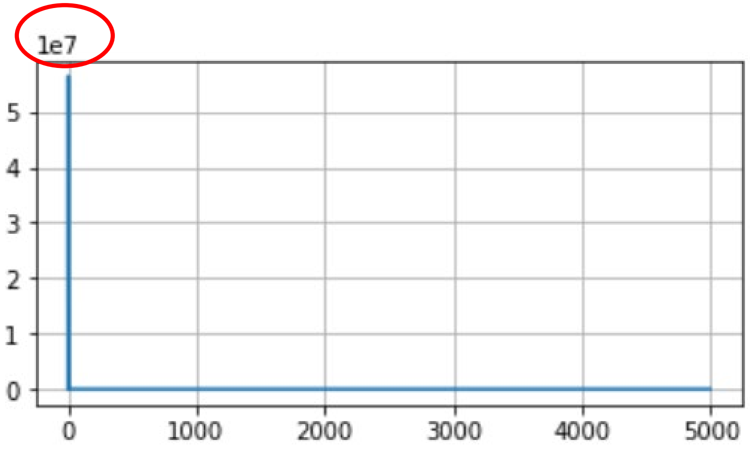
Reference: 李弘毅 ML Lecture 3-1 https://youtu.be/yKKNr-QKz2Q

Practice

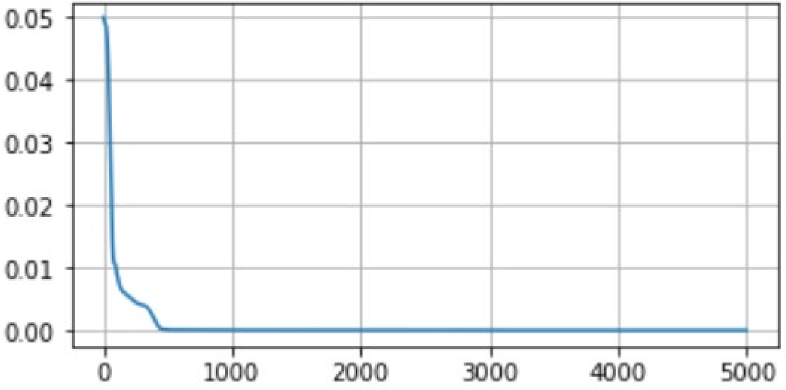
- Run "3.1. Learning rate" and explain why good *lr* is important?



lr = 0.005



lr = 1

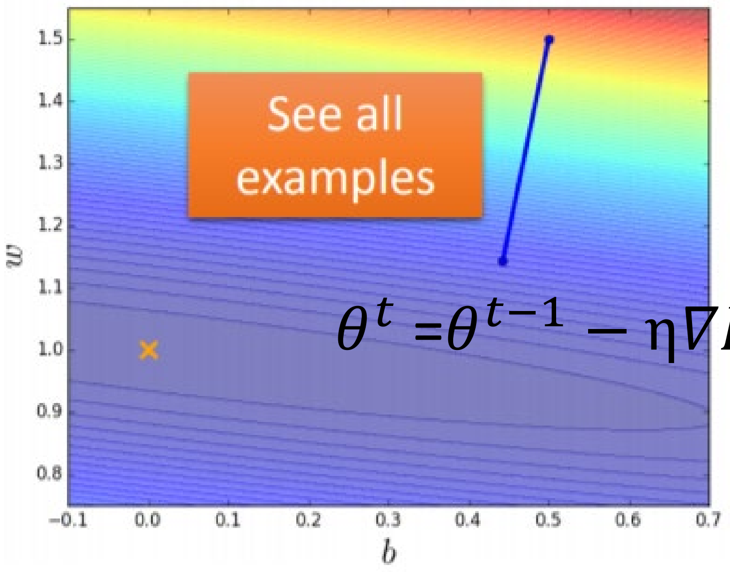


lr = 0.0005

Stochastic gradient decent

Gradient Descent

Update after seeing all examples



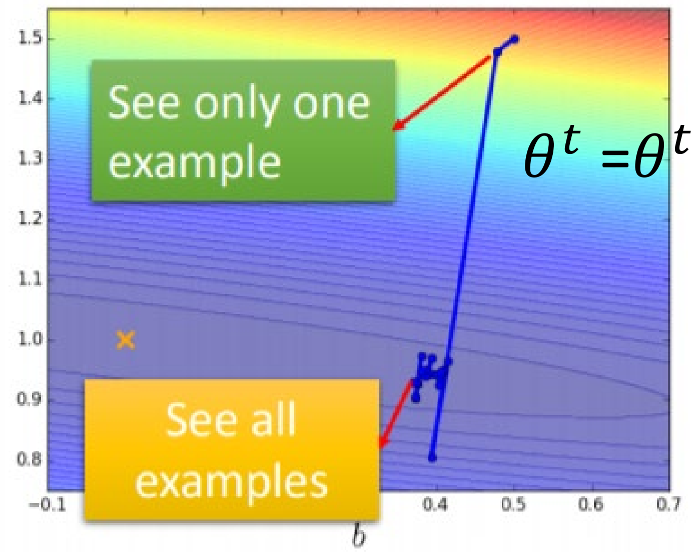
$$\theta^{t} = \theta^{t-1} - \eta \nabla L(\theta^{t-1})$$

$$\nabla L(\theta) = \begin{bmatrix} \partial L / \partial w_1 \\ \vdots \\ \partial L / \partial w_K \end{bmatrix}$$

Stochastic Gradient Descent

Update for each example

If there are 20 examples, 20 times faster.



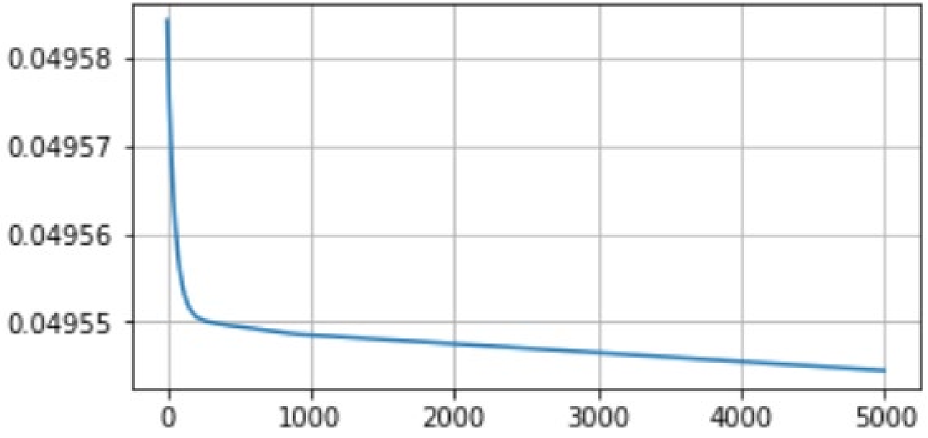
$$\theta^{t} = \theta^{t-1} - \eta \nabla L^{n} (\theta^{t-1})$$

$$\nabla L(\theta) = \begin{bmatrix} \partial L^{n} / \partial w_1 \\ \vdots \\ \partial L^{n} / \partial w_K \end{bmatrix}$$

Reference: 李弘毅 ML Lecture 3-1 https://youtu.be/yKKNr-QKz2Q

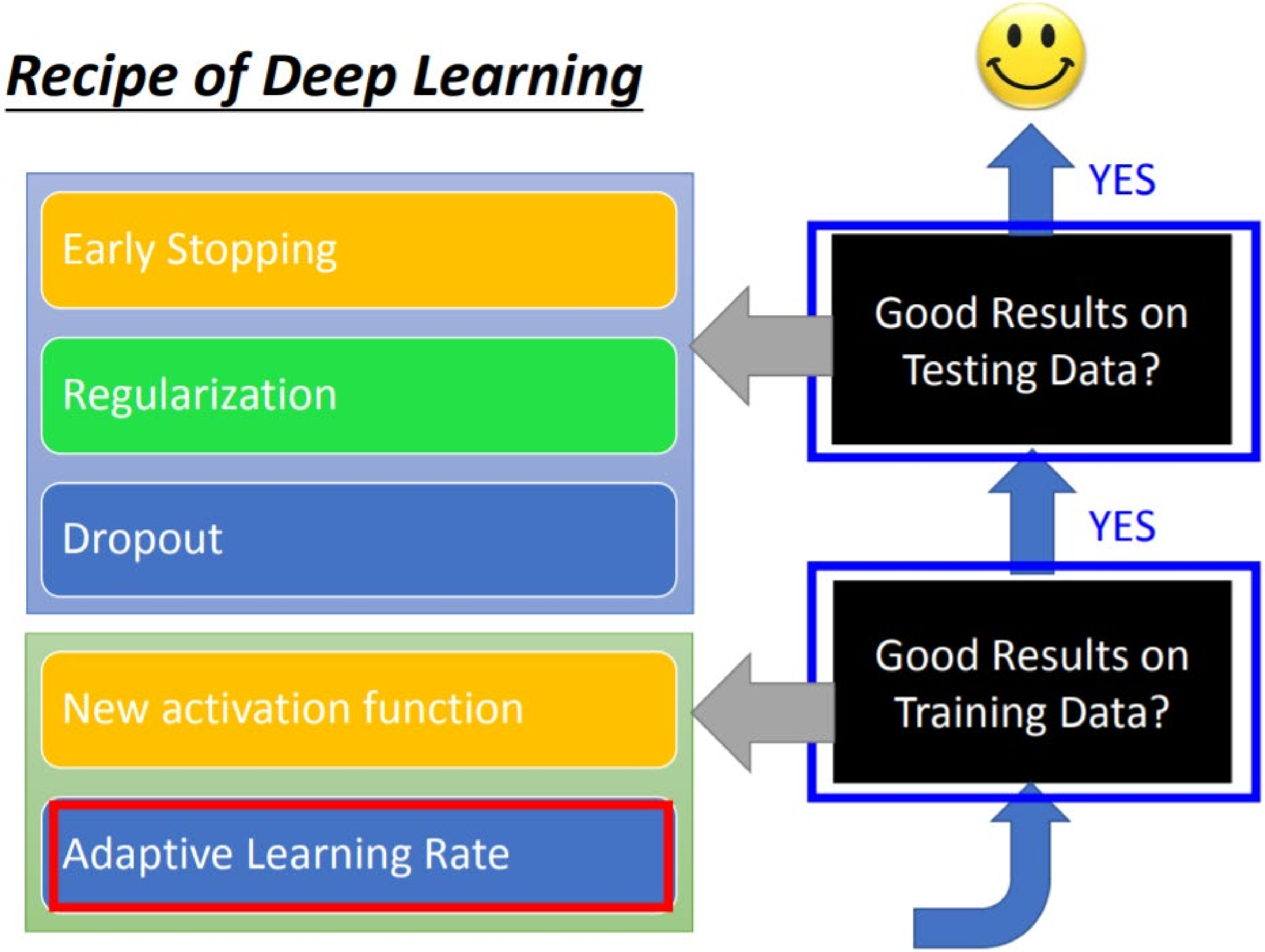
Practice

- Run "3.2. Optimizer" and discuss your loss plot.



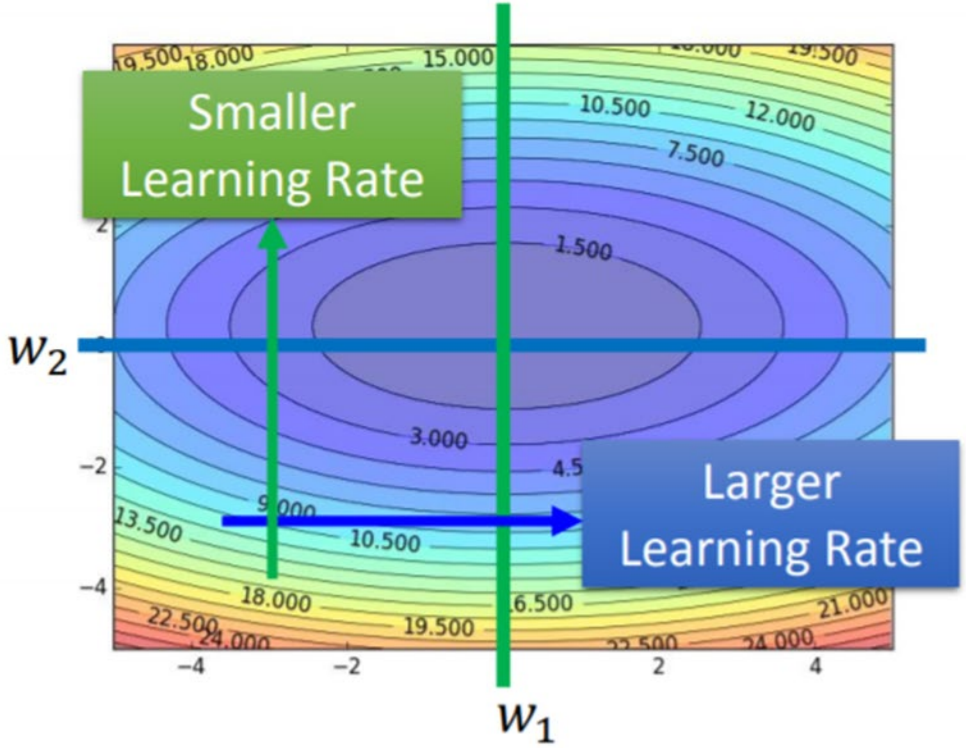
lr = 0.005, SGD

Adaptive learning rate



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Adagrad



Adagrad

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

Use first derivative to estimate second derivative

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Adagrad

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t \qquad g^t = \frac{\partial L(\theta^t)}{\partial w}$$

$$w^1 \leftarrow w^0 - \frac{\eta^0}{\sigma^0} g^0 \qquad \sigma^0 = \sqrt{(g^0)^2}$$

$$w^2 \leftarrow w^1 - \frac{\eta^1}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\frac{1}{2}[(g^0)^2 + (g^1)^2]}$$

$$w^3 \leftarrow w^2 - \frac{\eta^2}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\frac{1}{3}[(g^0)^2 + (g^1)^2 + (g^2)^2]}$$

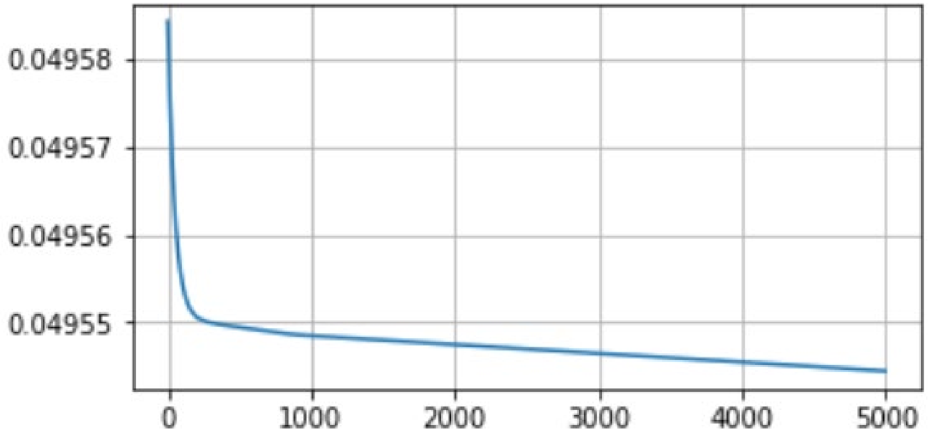
$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^i)^2}$$

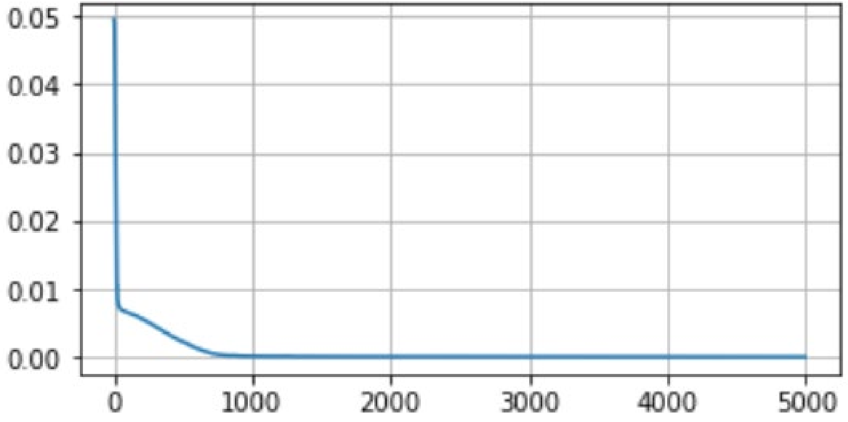
Reference: 李弘毅 ML Lecture 3-1 https://youtu.be/yKKNr-QKz2Q

Practice

- Run "3.2. Optimizer" and discuss your loss plot.



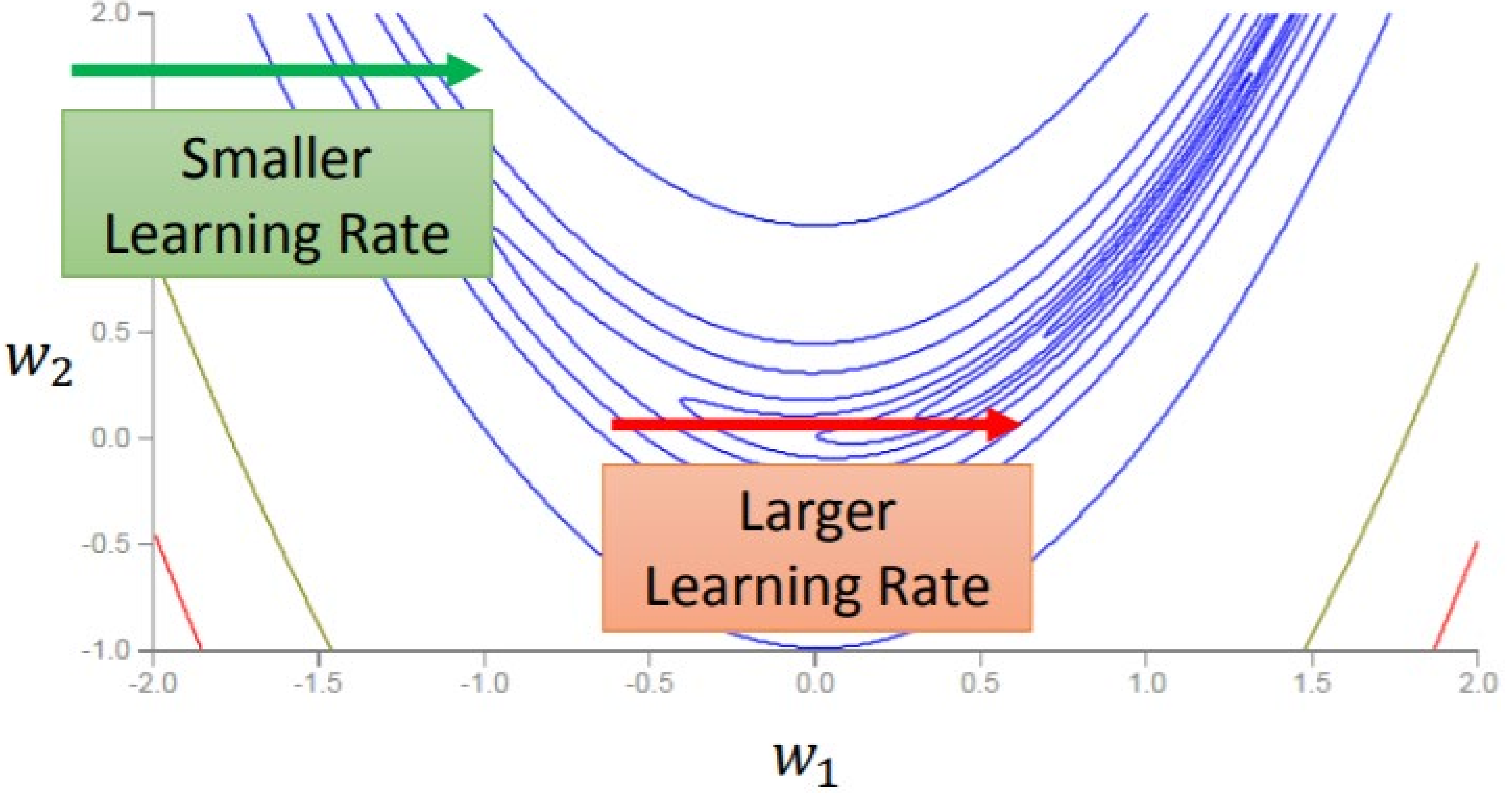
SGD



Adagrad

RMS Prop

Error Surface can be very complex when training NN.



Reference: 李弘毅 ML Lecture 3-1 https://youtu.be/yKKNr-QKz2Q

RMS Prop

$$w^1 \leftarrow w^0 - \frac{\eta}{\sigma^0} g^0 \qquad \sigma^0 = g^0$$

$$w^2 \leftarrow w^1 - \frac{\eta}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\alpha(\sigma^0)^2 + (1-\alpha)(g^1)^2}$$

$$w^3 \leftarrow w^2 - \frac{\eta}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\alpha(\sigma^1)^2 + (1-\alpha)(g^2)^2}$$

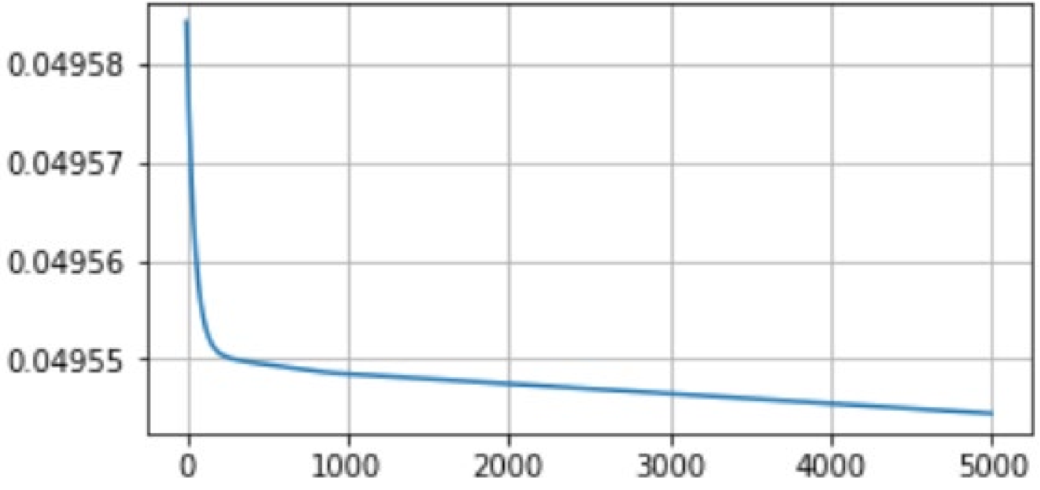
$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1-\alpha)(g^t)^2}$$

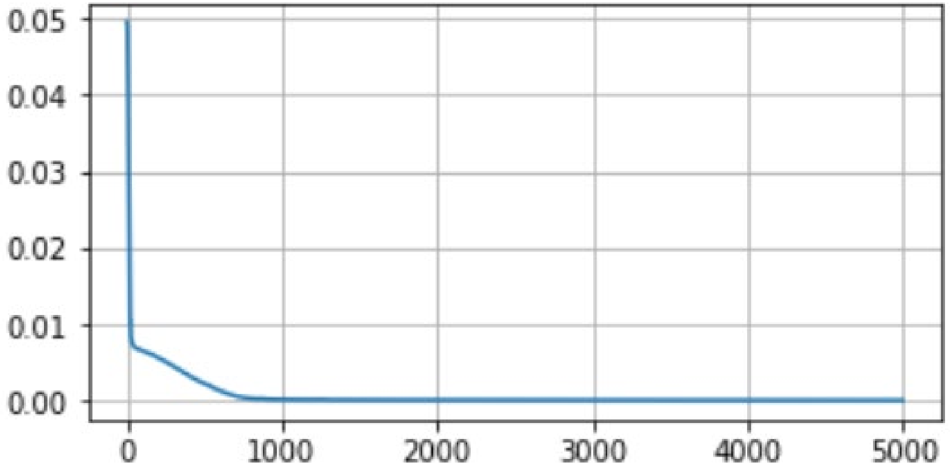
Root Mean Square of the gradients
with previous gradients being decayed

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

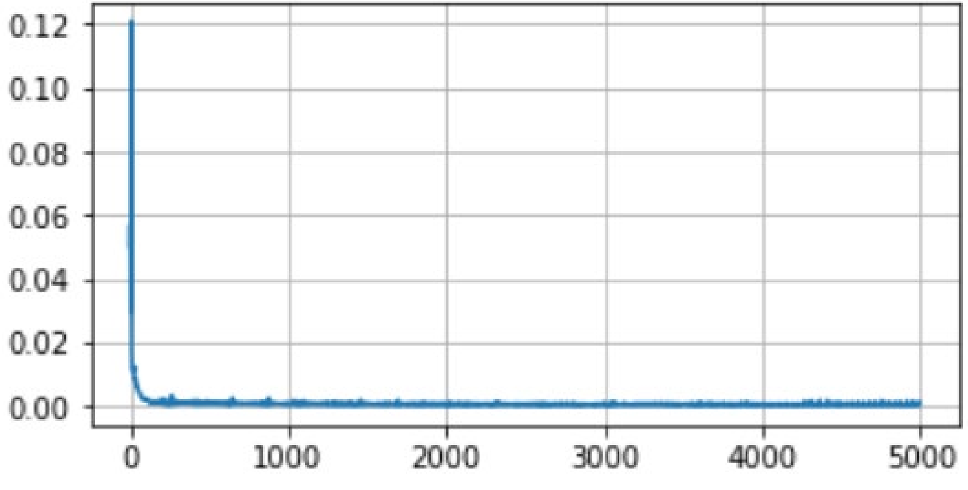
Run "3.2. Optimizer"



SGD

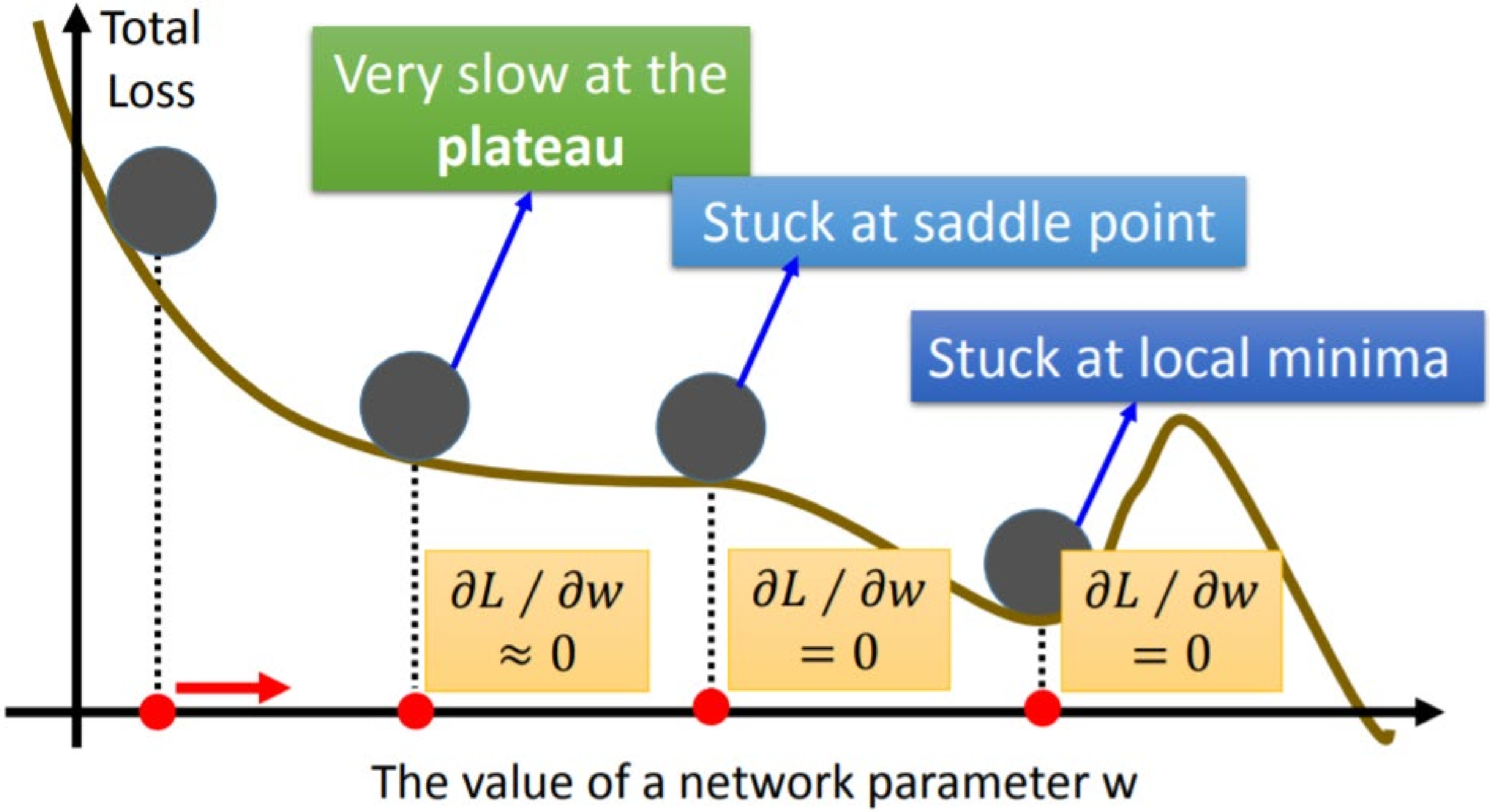


Adagrad



RMS Prop

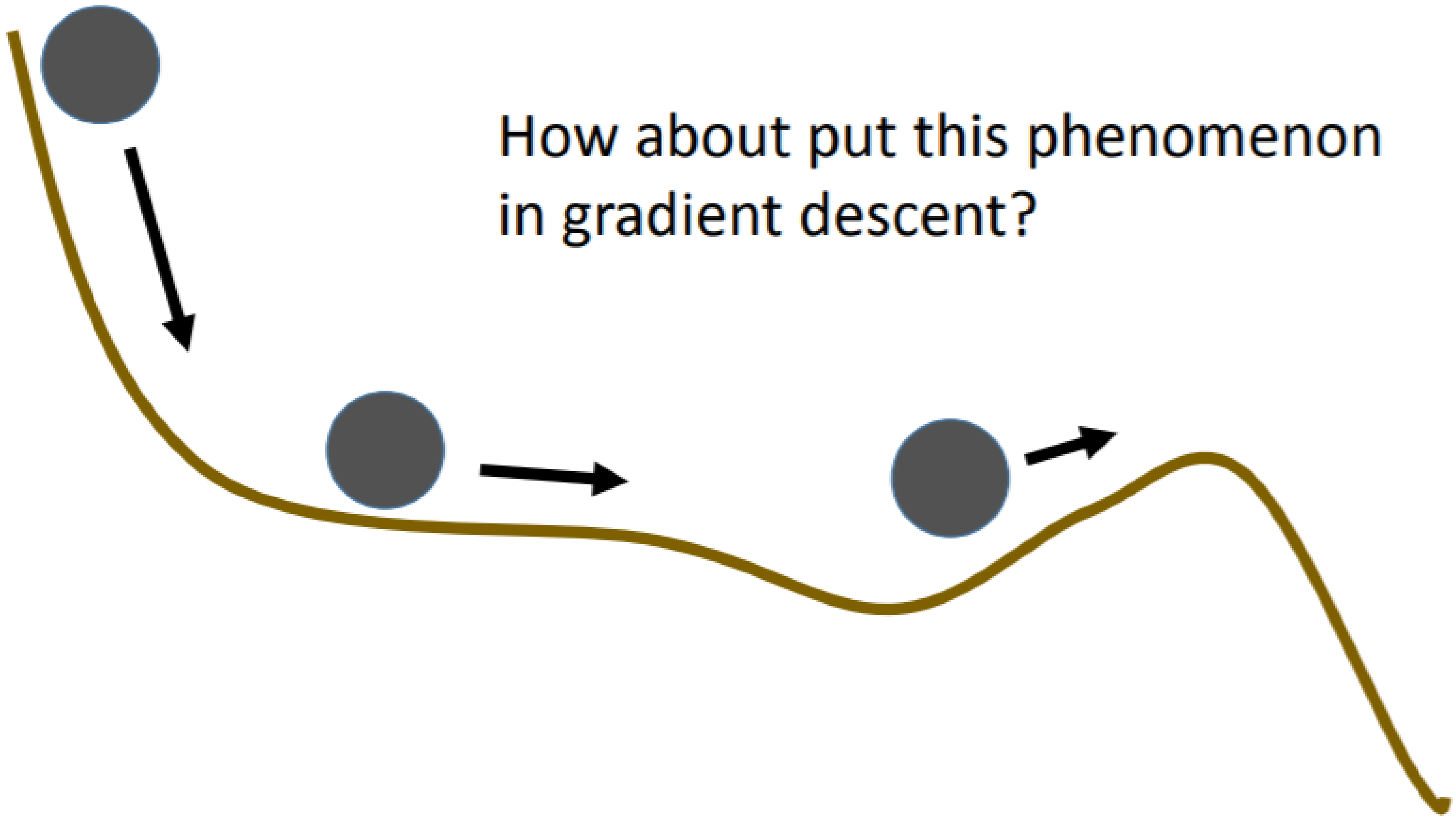
Hard to find optimal parameters



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

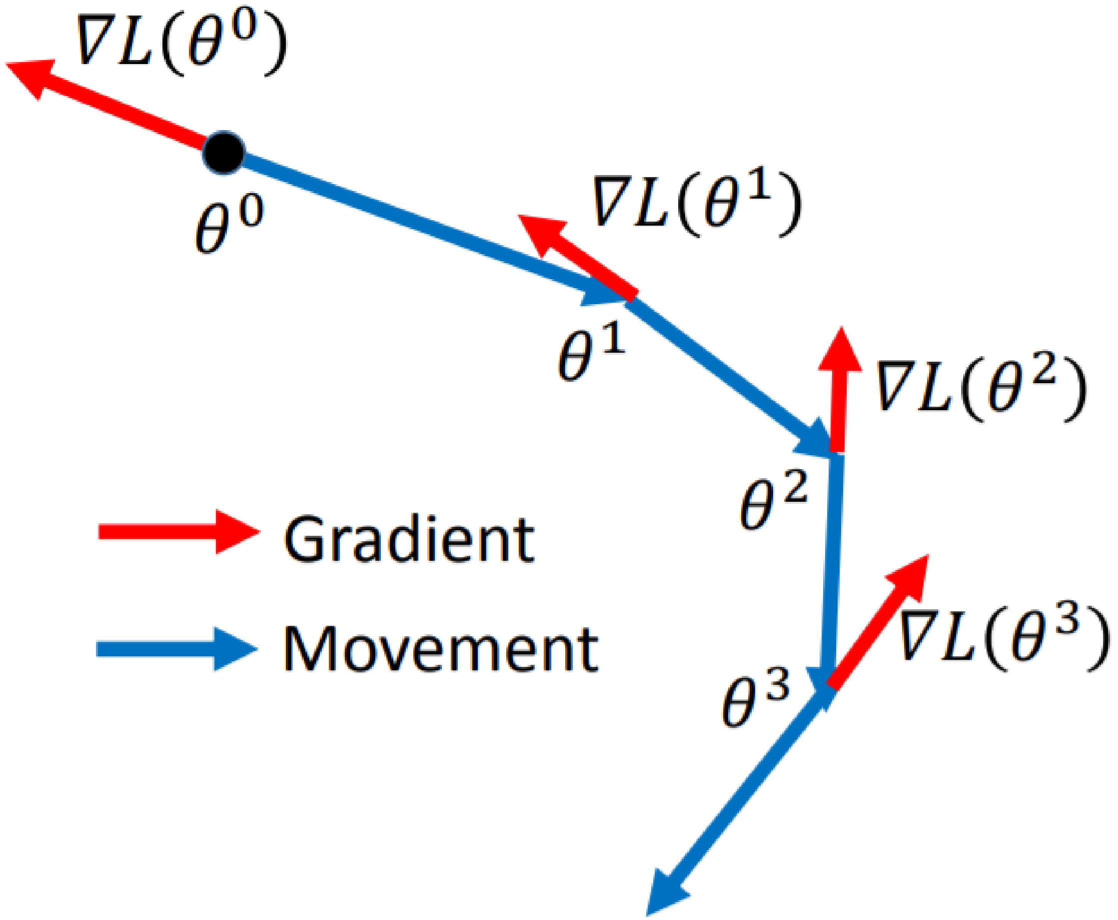
Momentum

- Momentum



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Vanilla gradient decent



Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta\nabla L(\theta^0)$

Compute gradient at θ^1

Move to $\theta^2 = \theta^1 - \eta\nabla L(\theta^1)$

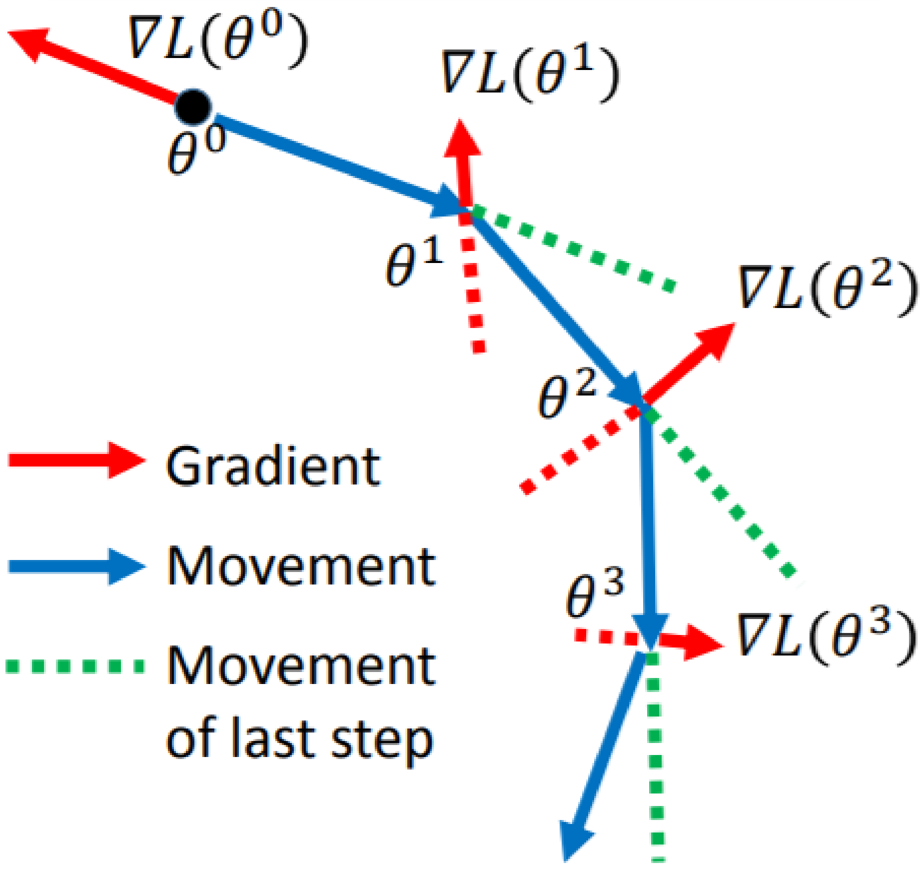
⋮

Stop until $\nabla L(\theta^t) \approx 0$

Reference: 李弘毅 ML 9-1 https://youtu.be/xki61j7z-30

Momentum

Movement: movement of last step minus gradient at present



Start at point θ^0

Movement $v^0=0$

Compute gradient at θ^0

Movement $v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$

Move to $\theta^1 = \theta^0 + v^1$

Compute gradient at θ^1

Movement $v^2 = \lambda v^1 - \eta \nabla L(\theta^1)$

Move to $\theta^2 = \theta^1 + v^2$

Movement not just based on gradient, but previous movement.

Reference: 李弘毅 ML 9-1 https://youtu.be/xki61j7z-30

Momentum

Movement: movement of last step minus gradient at present

v^i is actually the weighted sum of all the previous gradient: $\nabla L(\theta^0)$,$\nabla L(\theta^1)$, … $\nabla L\left(\theta^{i-1}\right)$

$v^0 = 0$

$v^1 = - \eta \nabla L(\theta^0)$

$v^2 = - \lambda\, \eta \nabla L(\theta^0) - \eta \nabla L(\theta^1)$

$\vdots$

Start at point θ^0

Movement v^0=0

Compute gradient at θ^0

Movement $v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$

Move to $\theta^1 = \theta^0 + v^1$

Compute gradient at θ^1

Movement $v^2 = \lambda v^1 - \eta \nabla L(\theta^1)$

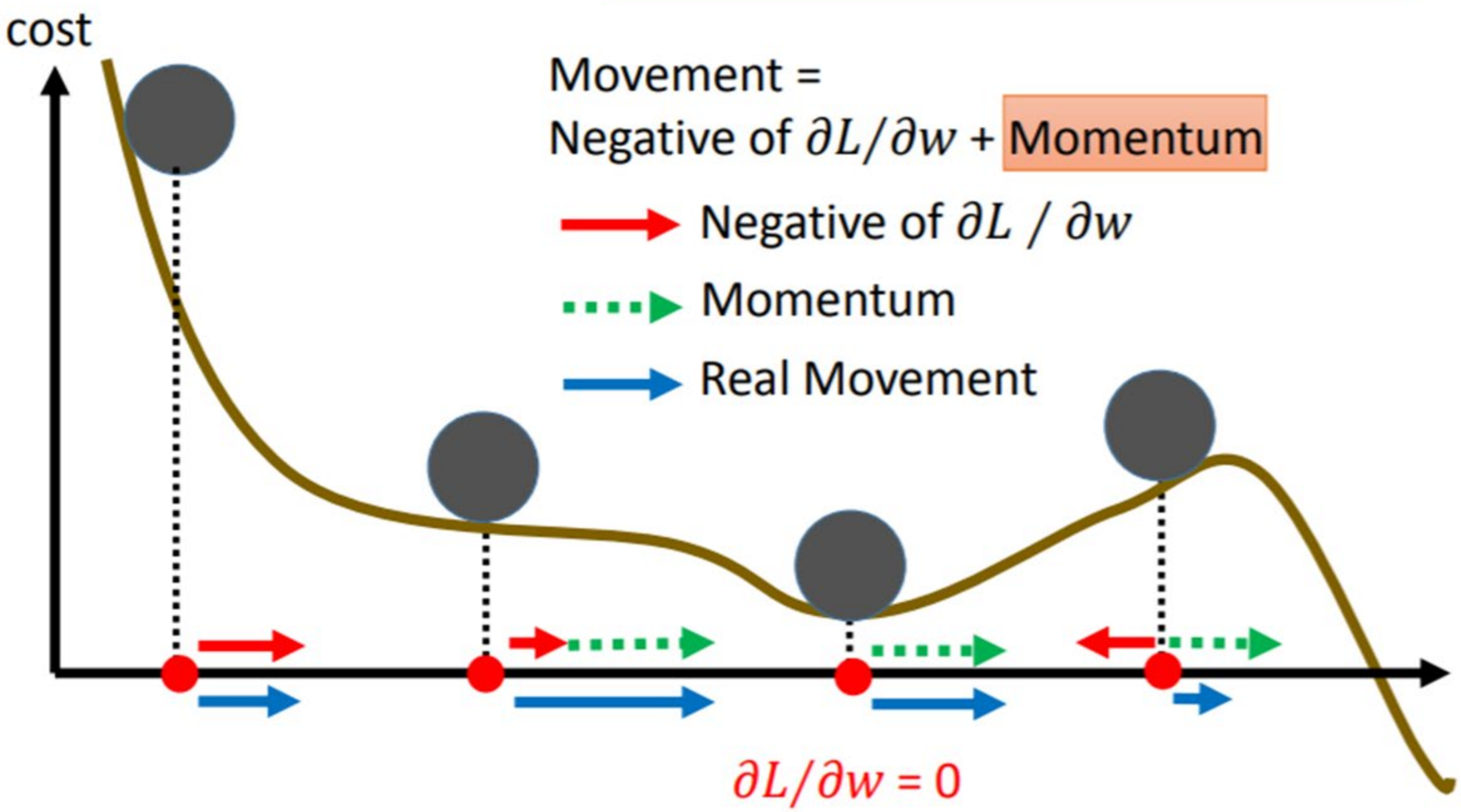
Move to $\theta^2 = \theta^1 + v^2$

Movement not just based on gradient, but previous movement

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Momentum

Still not guarantee reaching global minima, but give some hope ……



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Adam

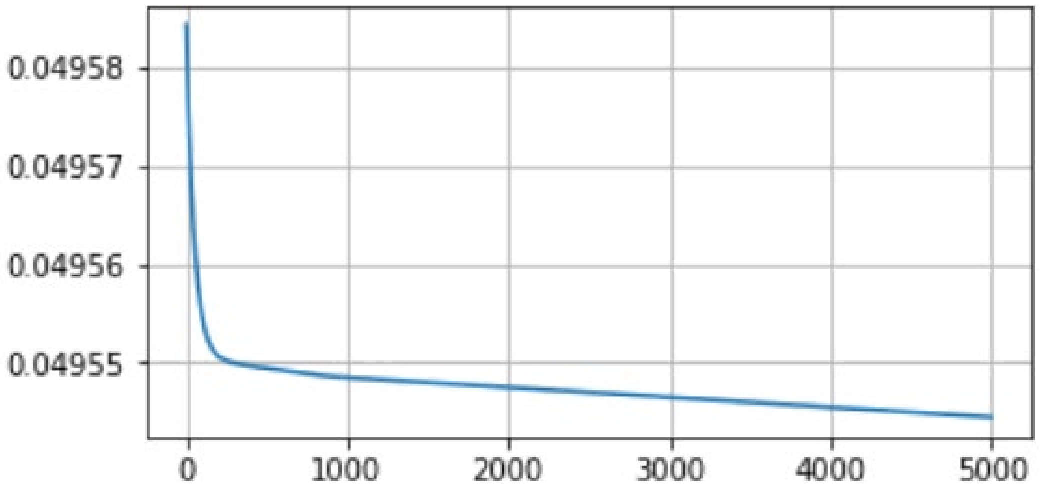
RMSProp + Momentum

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

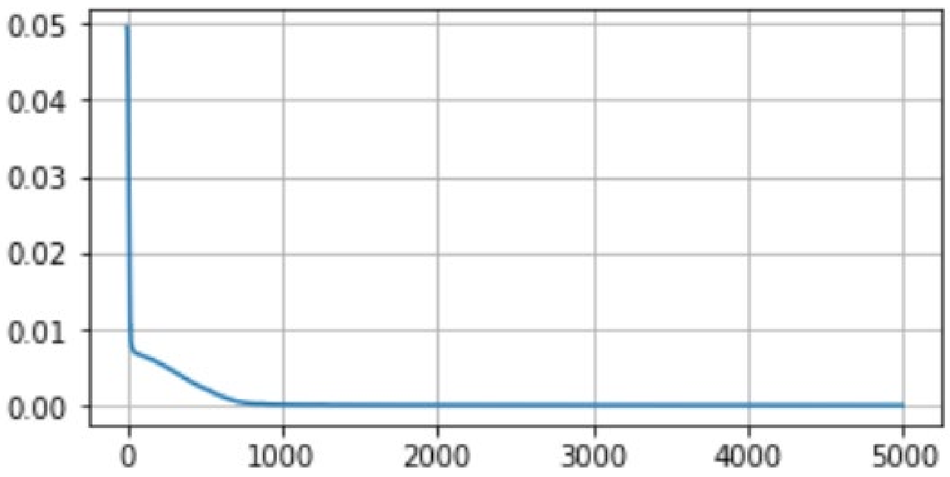
Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector) → for momentum
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector) → for RMSprop
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

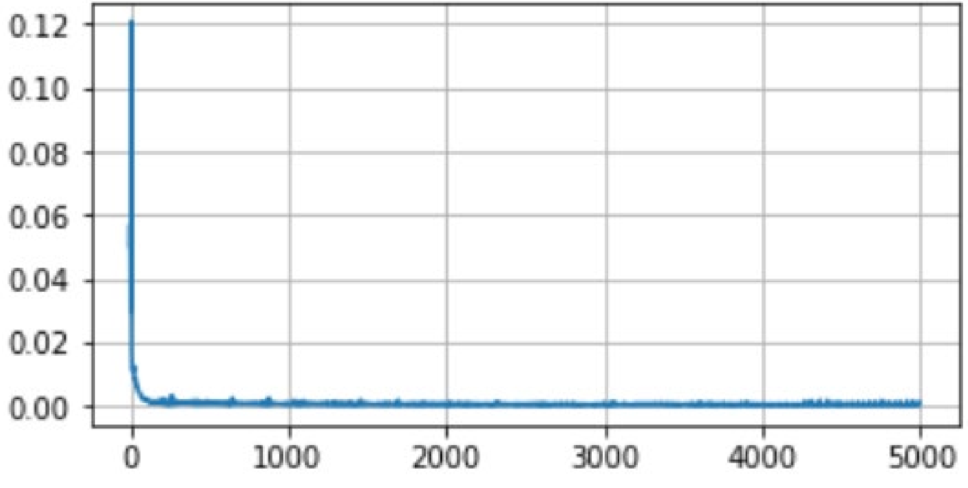
Run "3.2. Optimizer"



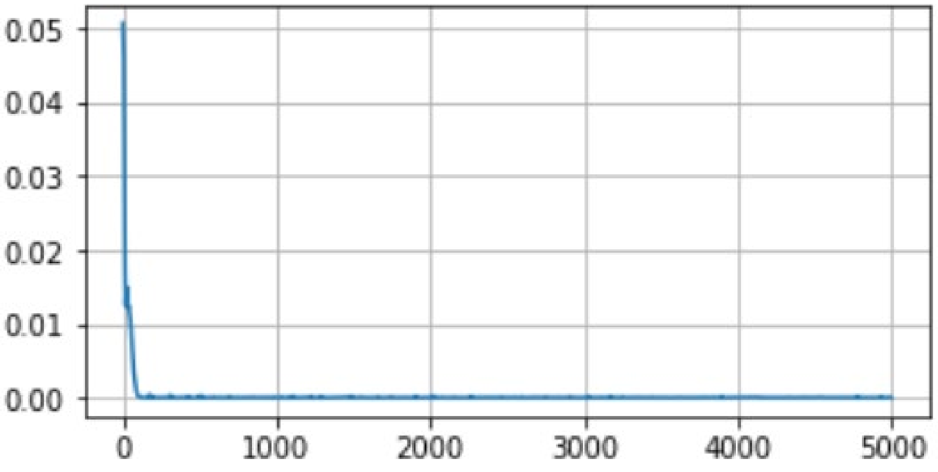
SGD



Adagrad

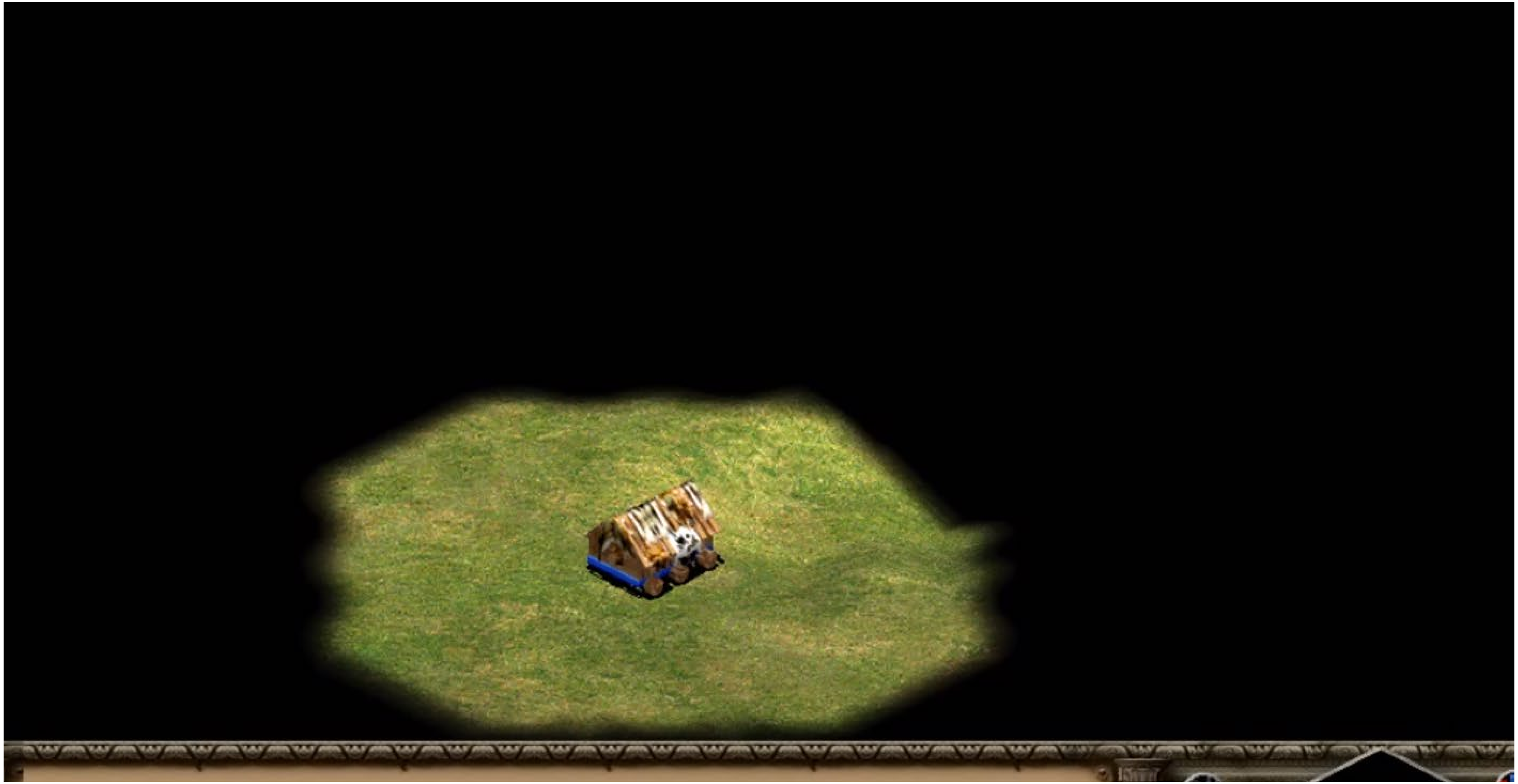


RMS Prop



Adam

Why gradient decent can find local minimum?



Reference: 李弘毅 ML Lecture 3-2 https://youtu.be/1_HBTJyWgNA

Why gradient decent can find local minimum?

Based on Taylor Series:

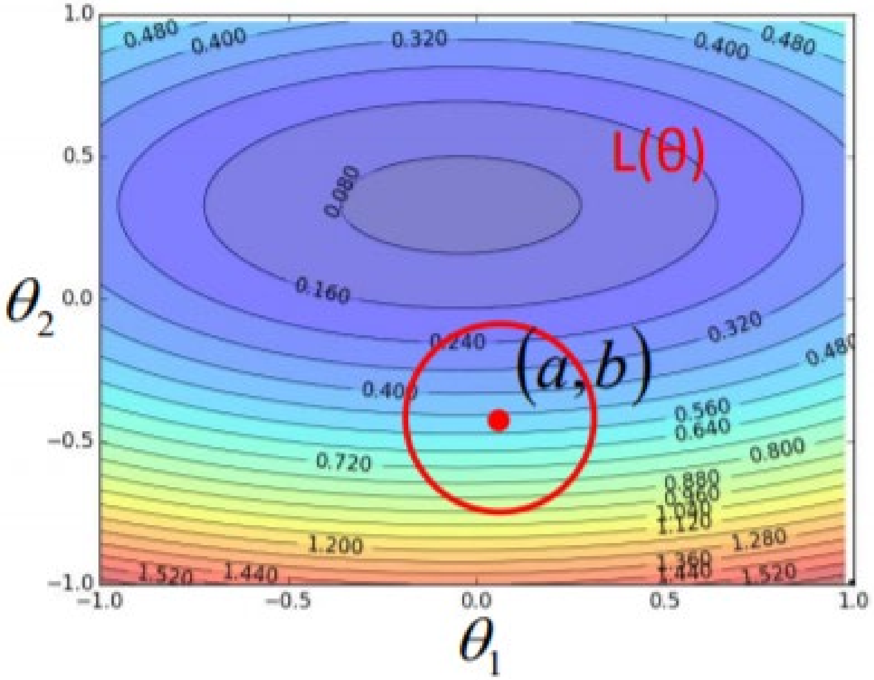
If the red circle is *small enough*, in the red circle

$$L(\theta) \approx L(a,b) + \frac{\partial L(a,b)}{\partial \theta_1}(\theta_1 - a) + \frac{\partial L(a,b)}{\partial \theta_2}(\theta_2 - b)$$

$$s = L(a,b)$$

$$u = \frac{\partial L(a,b)}{\partial \theta_1}, v = \frac{\partial L(a,b)}{\partial \theta_2}$$

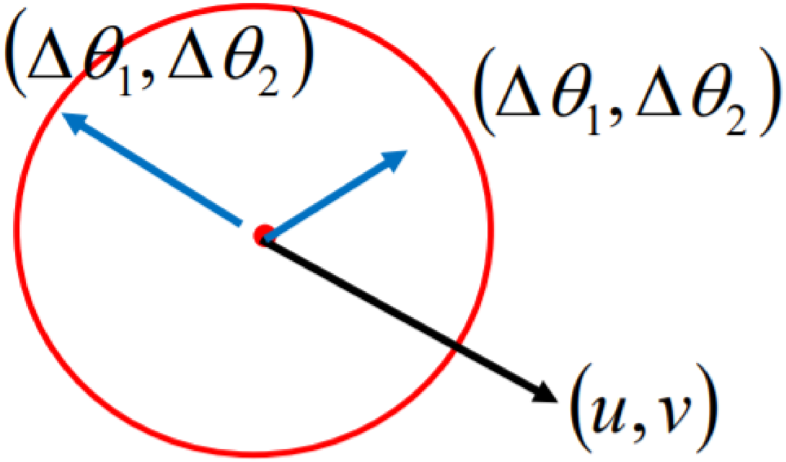
$$L(\theta)$$
$$\approx s + u(\theta_1 - a) + v(\theta_2 - b)$$



Why gradient decent can find local minimum?

Red Circle: (If the radius is small)

$$\mathrm{L}(\theta) \approx \cancel{s} + u(\underbrace{\theta_1 - a}_{\Delta\theta_1}) + v(\underbrace{\theta_2 - b}_{\Delta\theta_2})$$



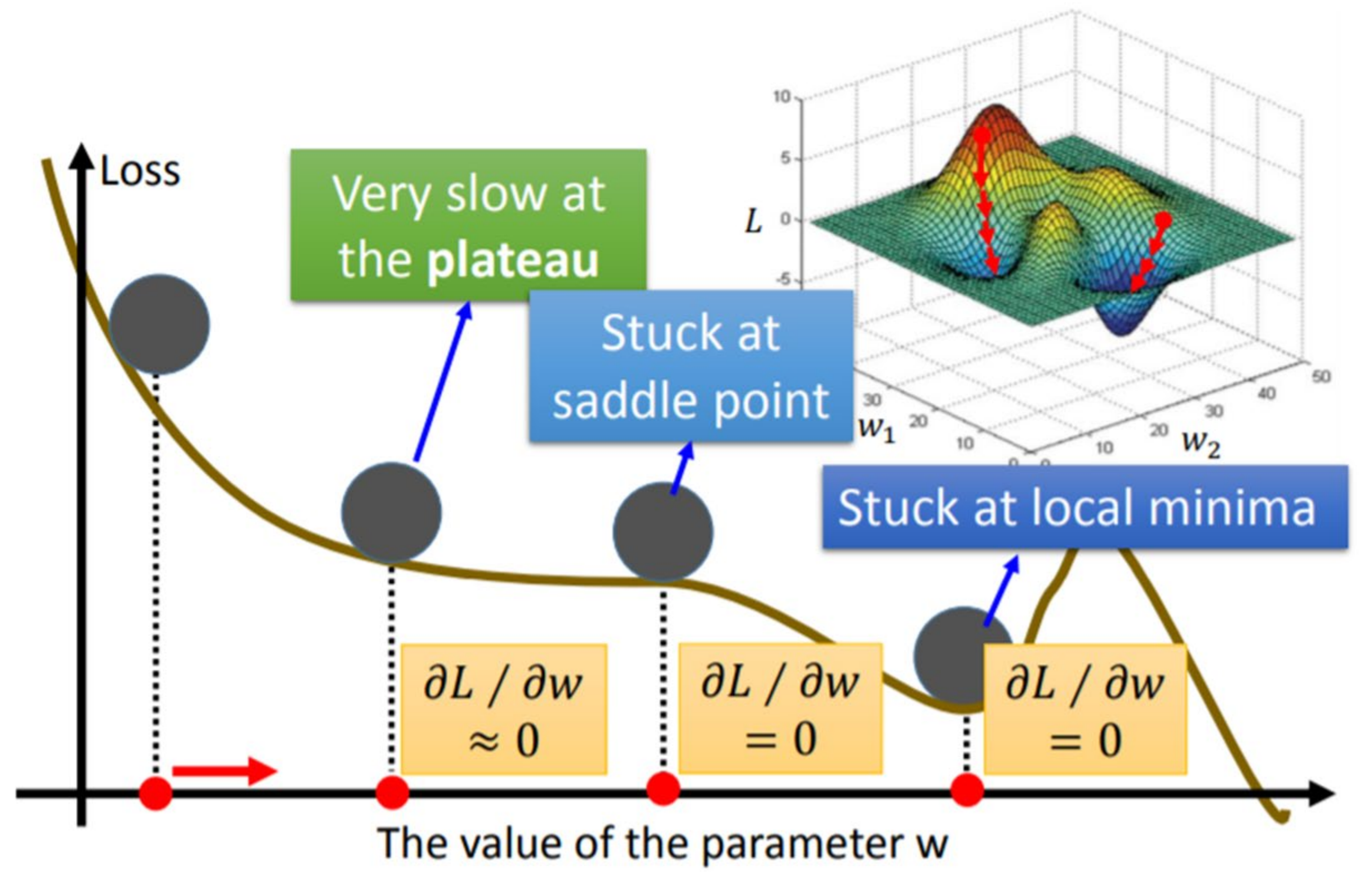
Find θ_1 and θ_2 in the red circle ***minimizing*** $\mathrm{L}(\theta)$

$$(\underbrace{\theta_1 - a}_{\Delta\theta_1})^2 + (\underbrace{\theta_2 - b}_{\Delta\theta_2})^2 \le d^2$$

To minimize $\mathrm{L}(\theta)$

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix} = -\eta \begin{bmatrix} u \\ v \end{bmatrix} \Rightarrow \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} - \eta \begin{bmatrix} u \\ v \end{bmatrix}$$

Limitation of gradient decent

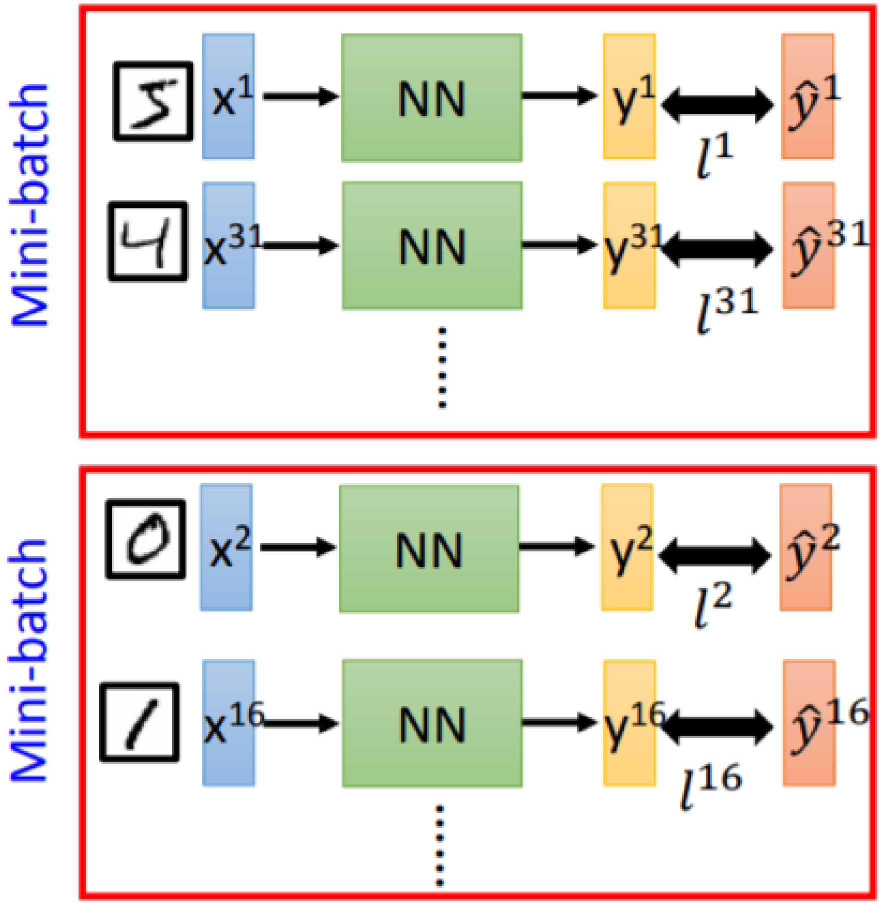


Reference: 李弘毅 ML Lecture 3-1 https://youtu.be/yKKNr-QKz2Q

Mini-batches

We do not really minimize total loss!

Mini-batch



- Randomly initialize network parameters
- Pick the 1st batch
  $L' = l^1 + l^{31} + \cdots$
  Update parameters once
- Pick the 2nd batch
  $L'' = l^2 + l^{16} + \cdots$
  Update parameters once

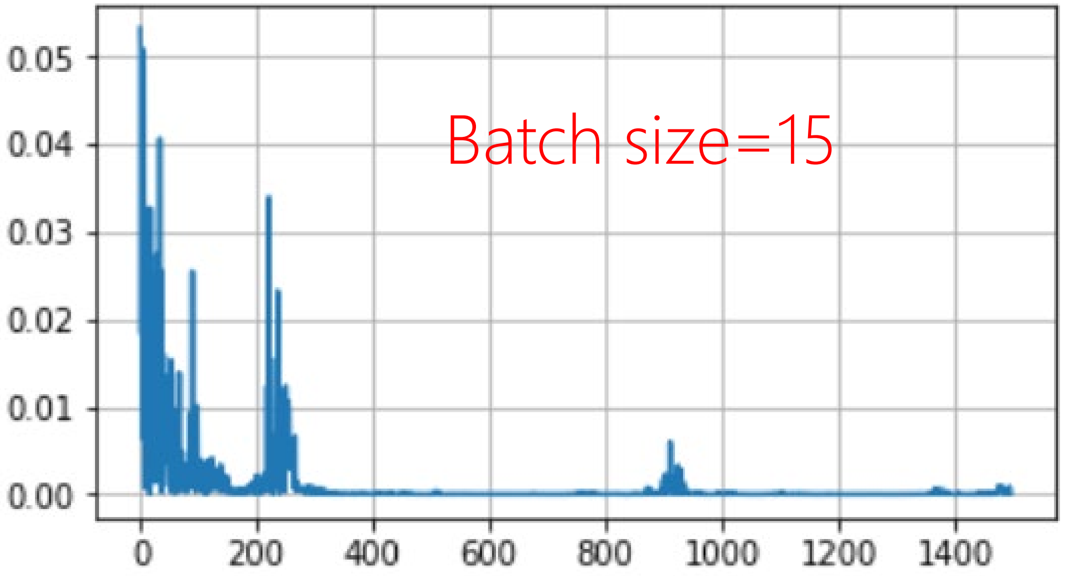
  ⋮
- Until all mini-batches have been picked

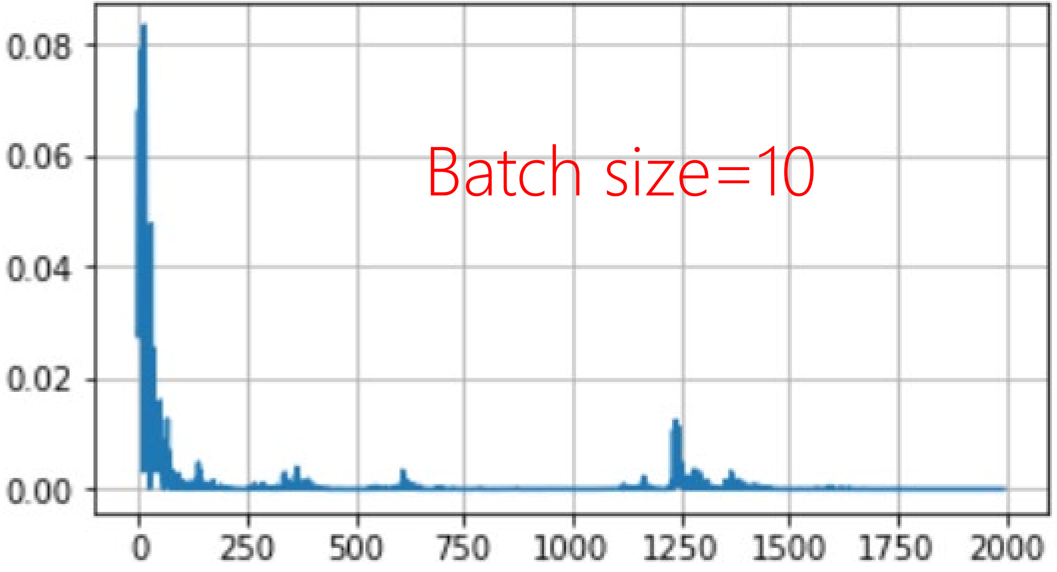
one epoch

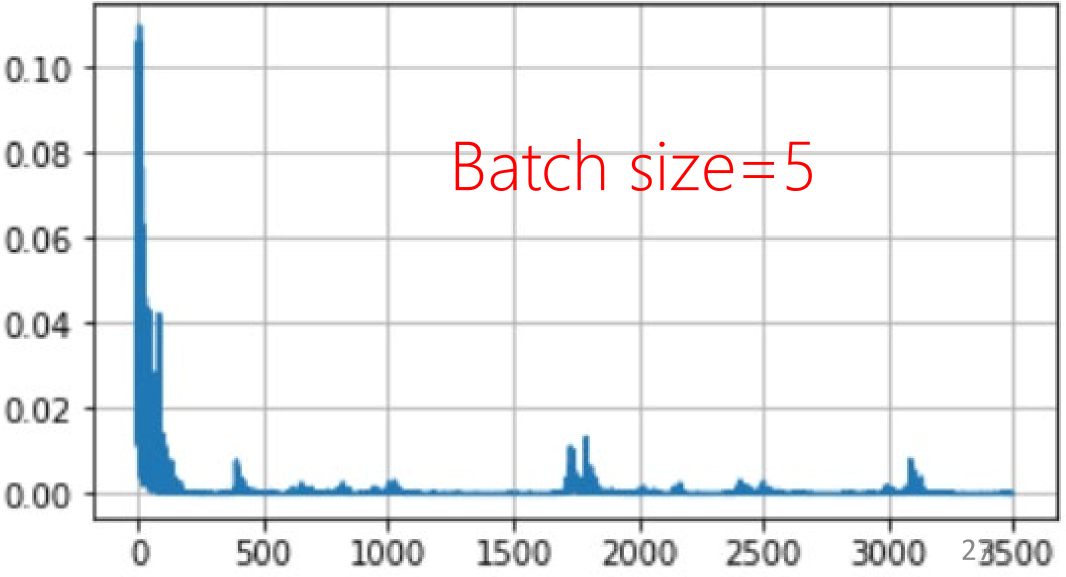
Repeat the above process

Reference: 李弘毅 ML Lecture 8-1 https://youtu.be/Lx3l4lOrquw

Run "3.3. Mini_Batch_Training"





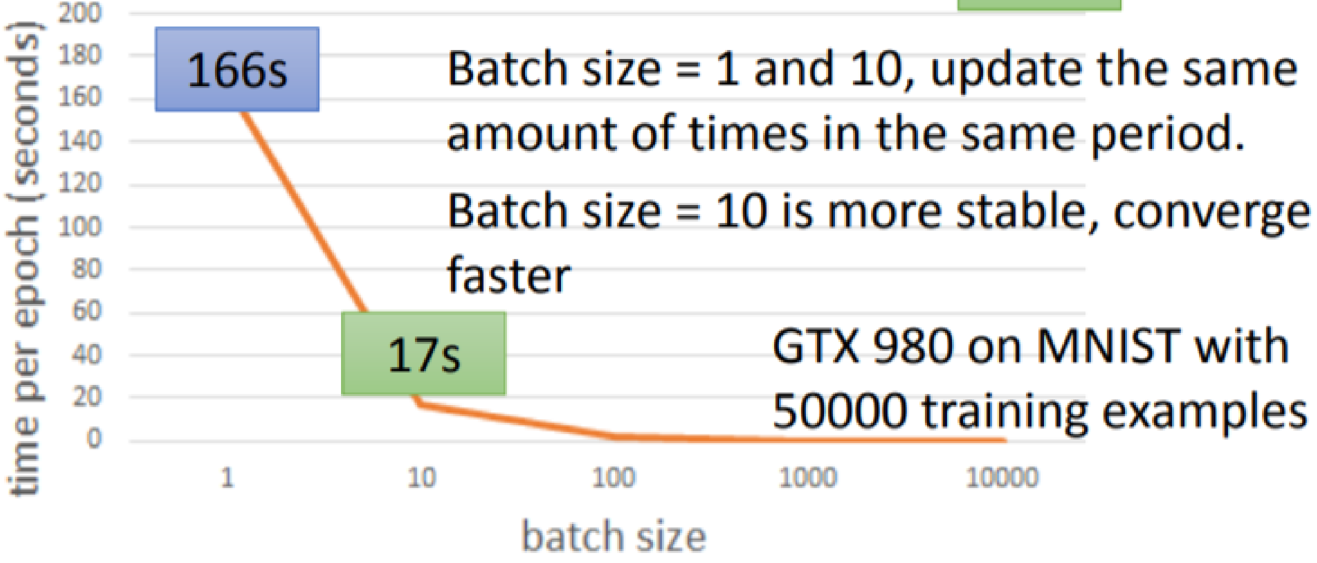


Batch size influences both speed and performance. You have to tune it.

Speed

Very large batch size can yield worse performance

- Smaller batch size means more updates in one epoch
  - E.g. 50000 examples
  - batch size = 1, 50000 updates in one epoch — 166s — 1 epoch
  - batch size = 10, 5000 updates in one epoch — 17s — 10 epoch



Reference: 李弘毅 ML Lecture 8-1 https://youtu.be/Lx3l4lOrquw