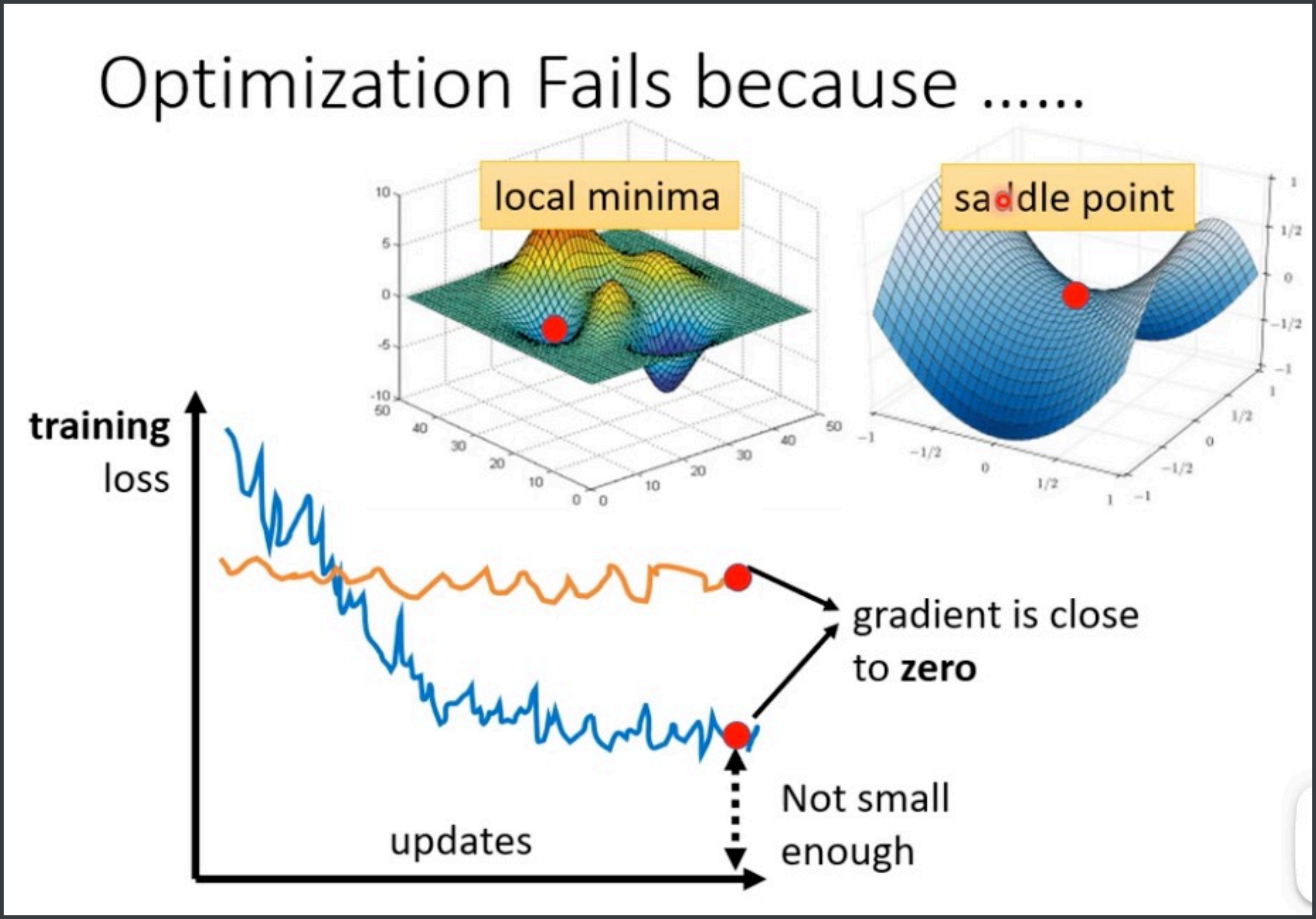
When Neural Network not work...

1) Local Minima & Saddle Point

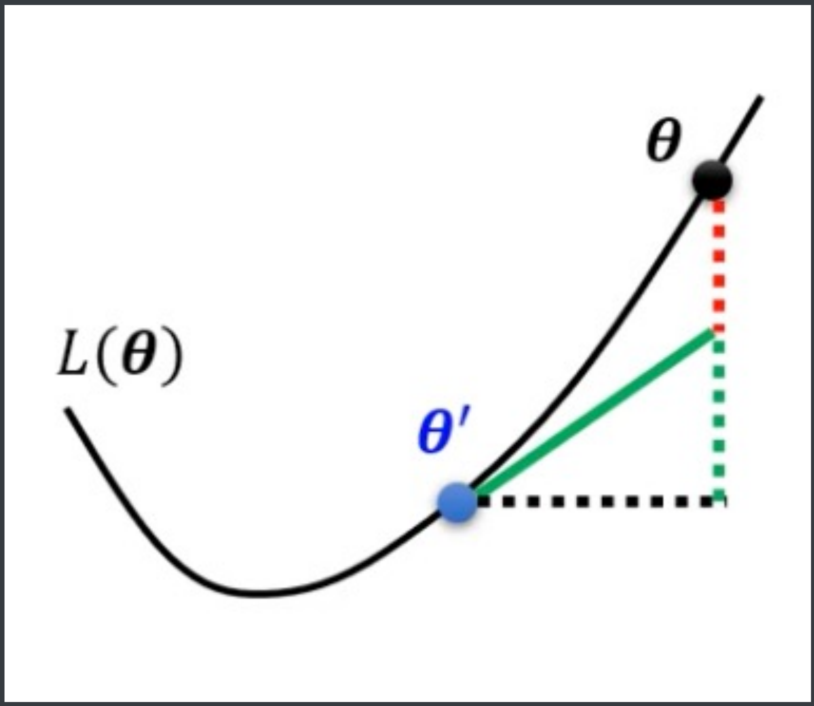


Do we know we got stuck in local minimum or saddle point?

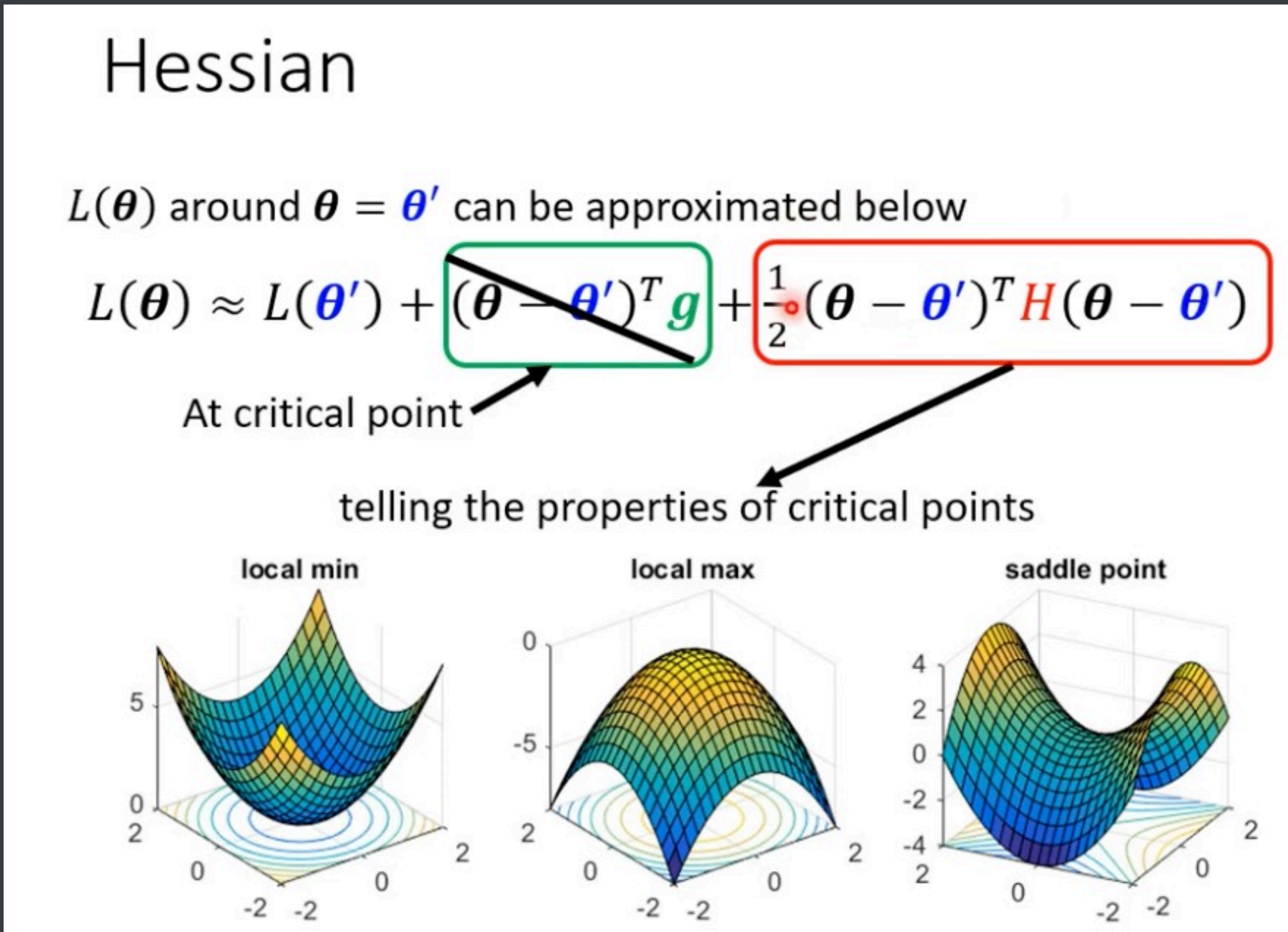
If local minima, there is no way to go. If saddle point, there's still way to go.

Tayler Series Approximation

$$L(\theta) \ \ around \ \ \theta = \theta' \ \ can\ be\ approximated\ below$$
$$L(\theta) \approx L(\theta') + (\theta - \theta') + \frac{1}{2}(\theta - \theta')^T g + \frac{1}{2}(\theta - \theta')^T H(\theta - \theta')$$
$$Gradient\ g\ is\ a\ vector$$
$$g = \triangledown L(\theta') \quad g_i = \frac{\partial L(\theta')}{\partial \theta_i}$$
$$Hessian\ H\ is\ a\ matrix$$
$$H_{ij} = \frac{\partial^2}{\partial \theta_i \partial \theta_j} L(\theta')$$



$$At\ critical\ point \ \ (\theta - \theta')^T g = 0$$
$$\frac{1}{2}(\theta - \theta')^T H(\theta - \theta')\ tells\ the\ properties\ of\ critical\ points$$



From the formula above, we can understand whether it's a local min, local max or saddle point.

$$v = \theta - \theta'$$

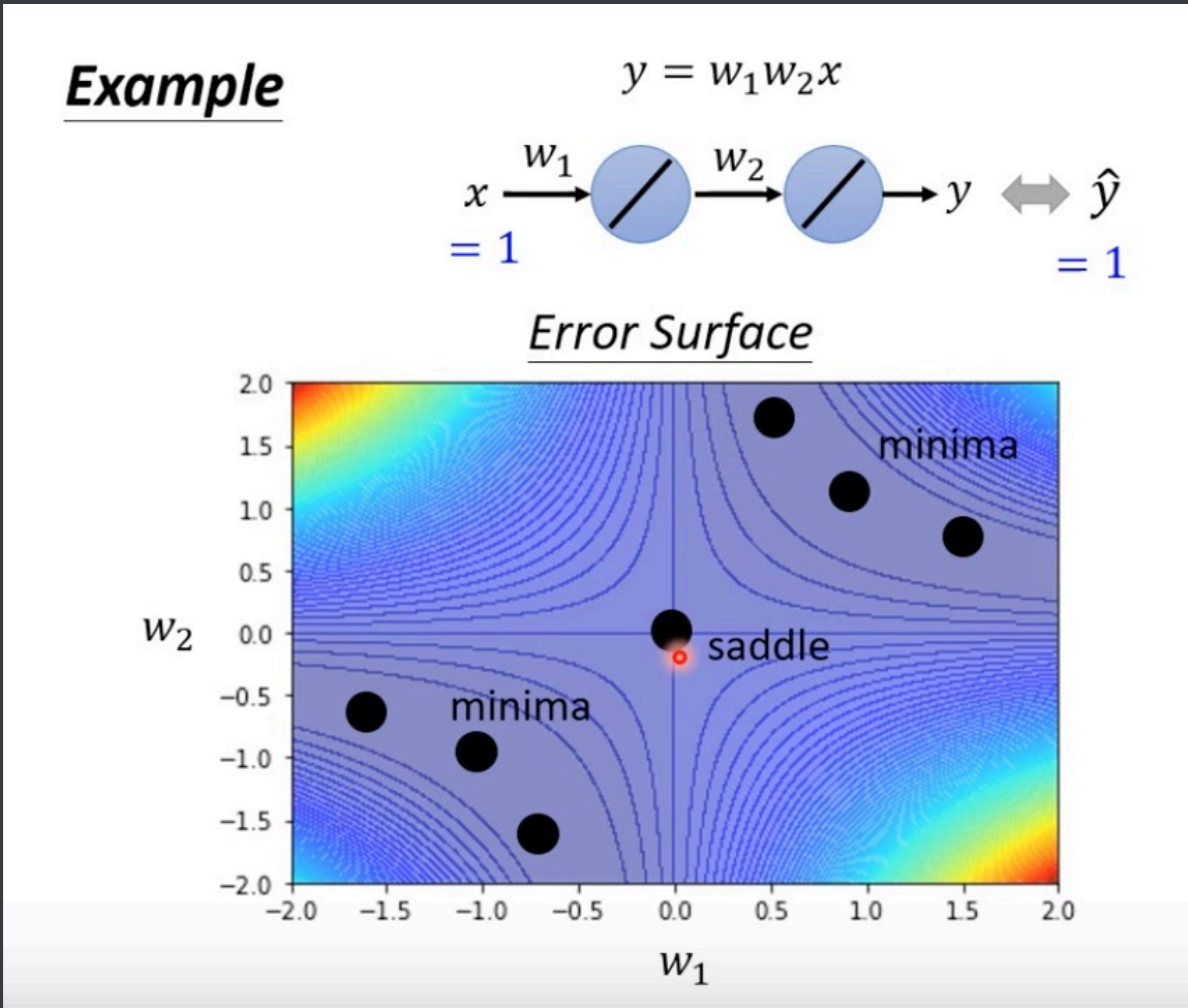
$$\begin{gathered} For\ all\ v\ :\ v^T H\ v > 0 \rightarrow Around\ \theta' : L(\theta) > L(\theta')\ \rightarrow \theta\ is\ a\ Local\ minima \\ =\ H\ is\ positive\ definite\ =\ All\ eigen\ values\ are\ positive. \end{gathered}$$

$$\begin{gathered} For\ all\ v\ :\ v^T H\ v < 0 \rightarrow Around\ \theta' : L(\theta) < L(\theta')\ \rightarrow \theta\ is\ a\ Local\ maxima \\ = H\ is\ negative\ definite\ =\ All\ eigen\ values\ are\ negative. \end{gathered}$$

$$\begin{gathered} Sometimes\ v^T H\ v > 0, sometimes\ v^T H\ v < 0\ \rightarrow \theta\ is\ a\ saddle\ point \\ Some\ eigen\ values\ are\ positive,\ and\ some\ are\ negative. \end{gathered}$$

Example:

$$y = w_1 w_2 x$$



Calculation:

$$L = (\hat{y} - w_1 w_2 x)^2 = (1 - w_1 w_2)^2$$

$$critical\ point\ :\ w_1 = 0, w_2 = 0$$

$$Gradient\ :$$

$$\frac{\partial L}{\partial w_1} = 2(1 - w_1 w_2)(-w_2)$$
$$\frac{\partial L}{\partial w_2} = 2(1 - w_1 w_2)(-w_1)$$

$$Hassian\ Matrix :$$

$$\frac{\partial^2 L}{\partial w_1^2} = 2(-w_2)(-w_2) \quad \frac{\partial^2 L}{\partial w_1 \partial w_2} = -2 + 4w_1 w_2$$

$$\frac{\partial^2 L}{\partial w_2 \partial w_1} = -2 + 4w_1 w_2 \quad \frac{\partial^2 L}{\partial w_2^2} = 2(-w_1)(-w_1)$$

Substite the value of the critical point (w1 = 0, w2 = 0) into Hessian:

$$H = \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix}$$

Calculate the egin value of H:

$$\lambda_1 = 2,\ \lambda_2 = -2$$

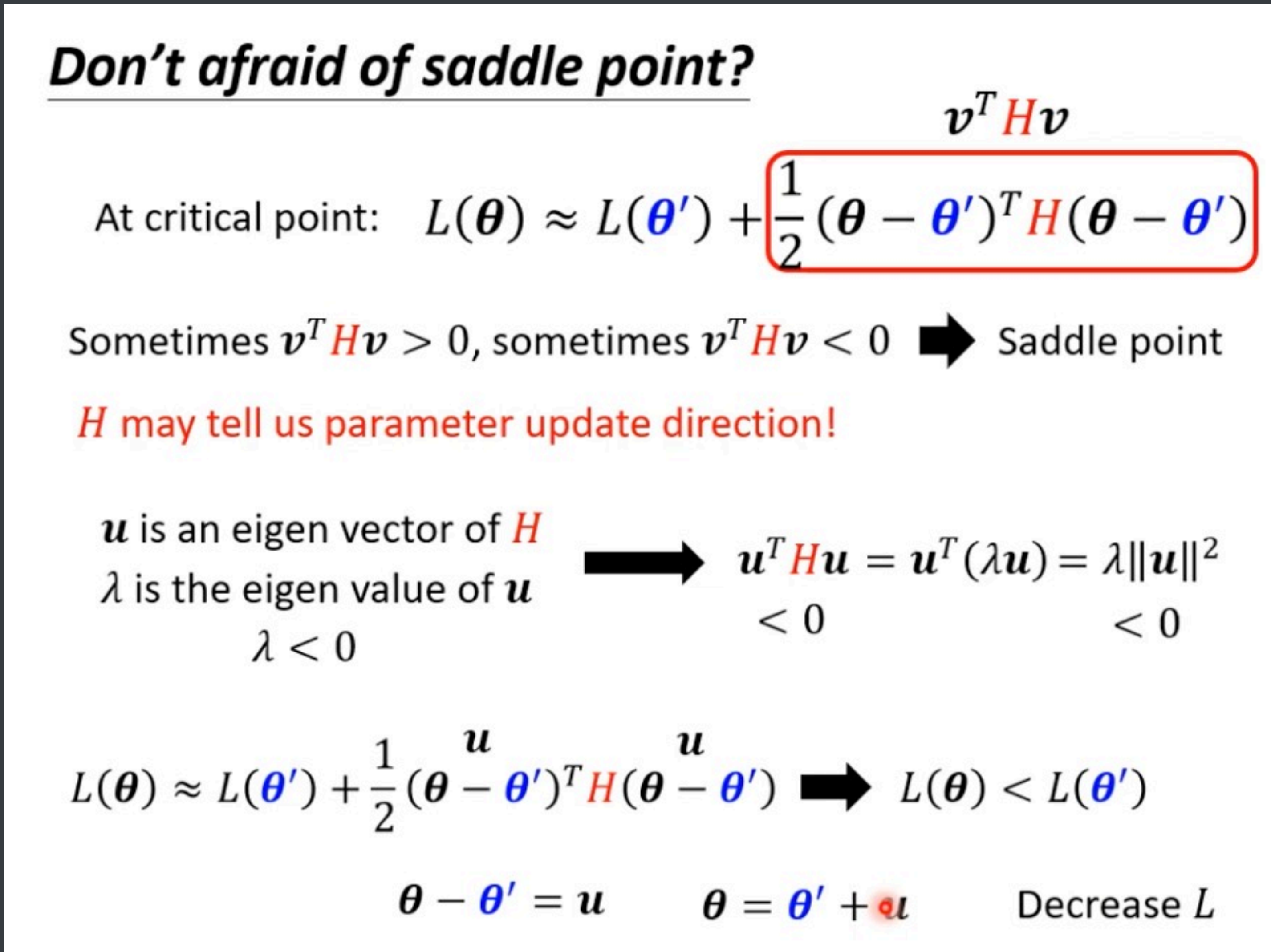
Conclusion: Saddle point

If it's a Saddle Point

H may tell us parameter update direction

$$u\ is\ an\ eigen\ vector\ of\ H$$
$$\lambda\ is\ the\ eigen\ value\ of\ u$$

$$\rightarrow u^T H u = u^T (\lambda u) = \lambda \|u\|^2$$



calculate the negative eigen value and calculate the corresponding eigen vector, plus theta'.

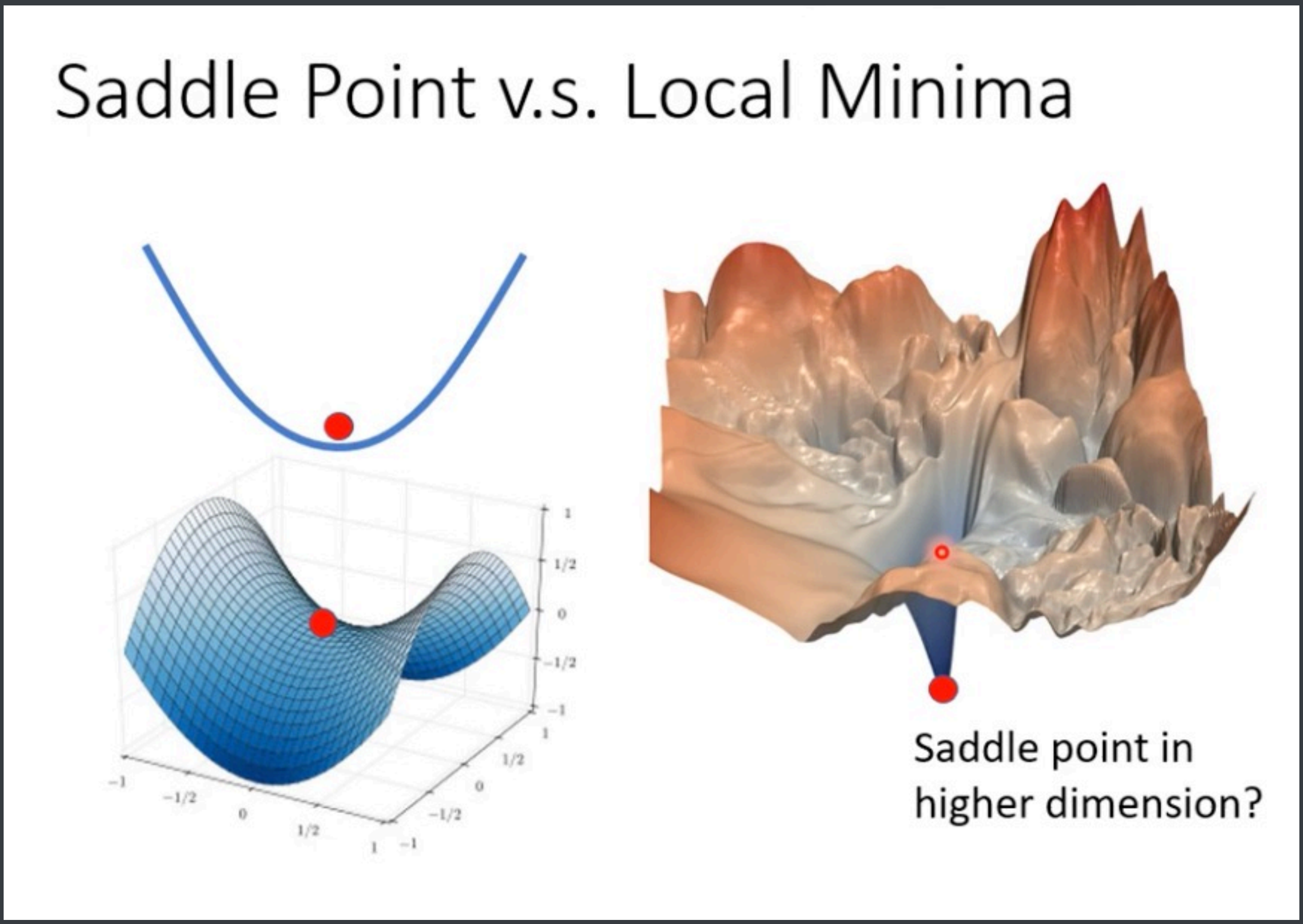
Example:

$$\lambda_2 = -2 \ has \ eigenvector \ u = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

update the parameter along the direction of u

(In practice, nearly not possible to calculate H - computational resources)

Saddle Point vs Local Minima

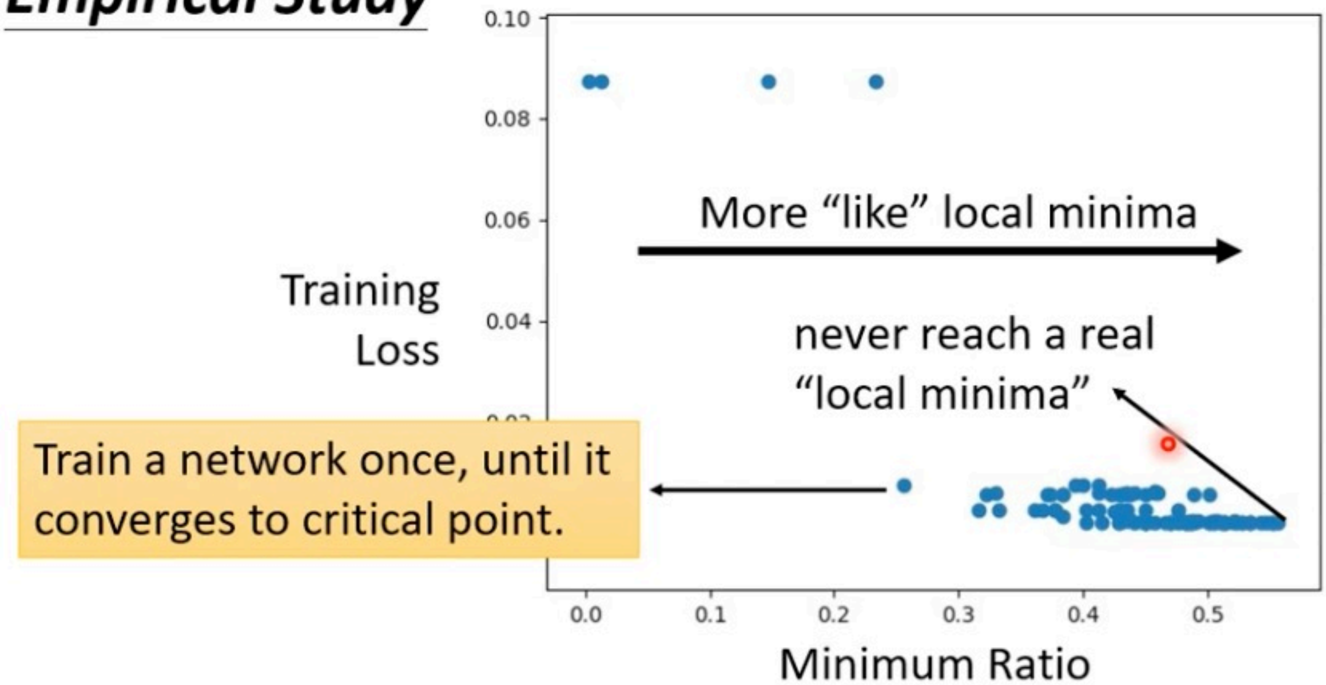


Looking in a two dimensinoal space, it's a local minima

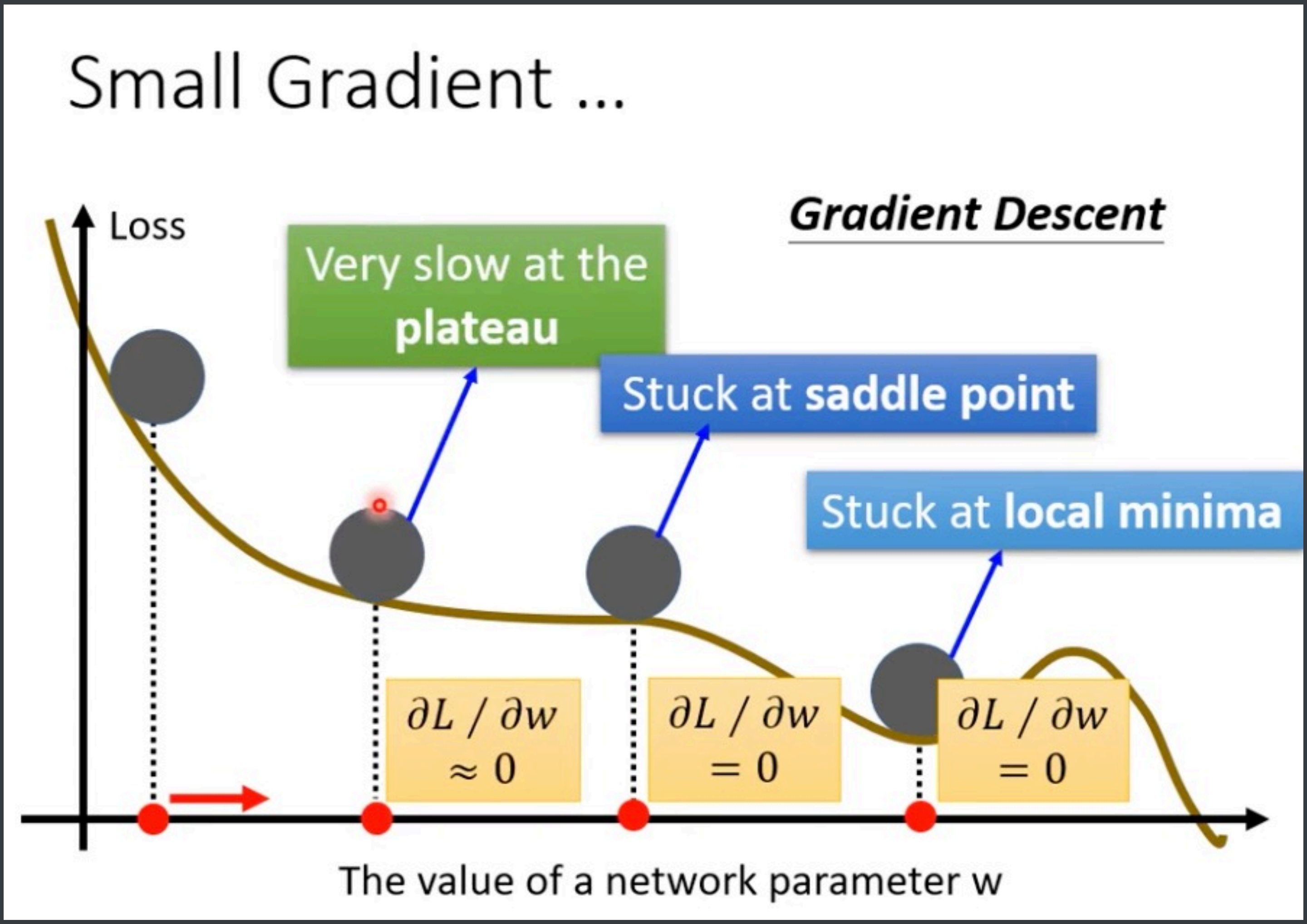
From a three dimensional space, it's a saddle point

The number of feature determines the 'dimension' of the error surface

Empirical Study



$$\text{Minimum ratio} = \frac{\text{Number of } \textbf{Positive} \text{ Eigen values}}{\text{Number of Eigen values}}$$

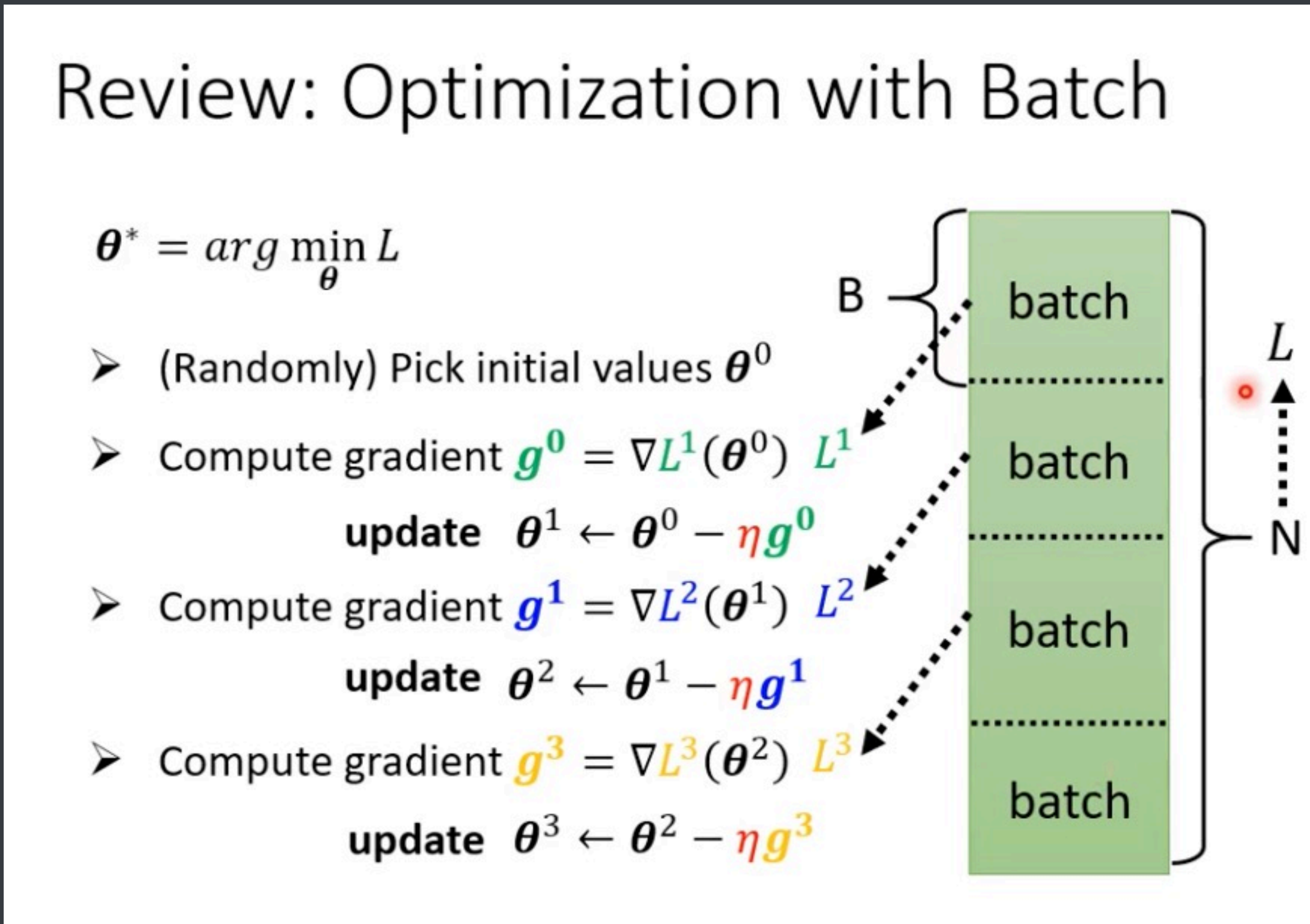


2） Batch and Momentum

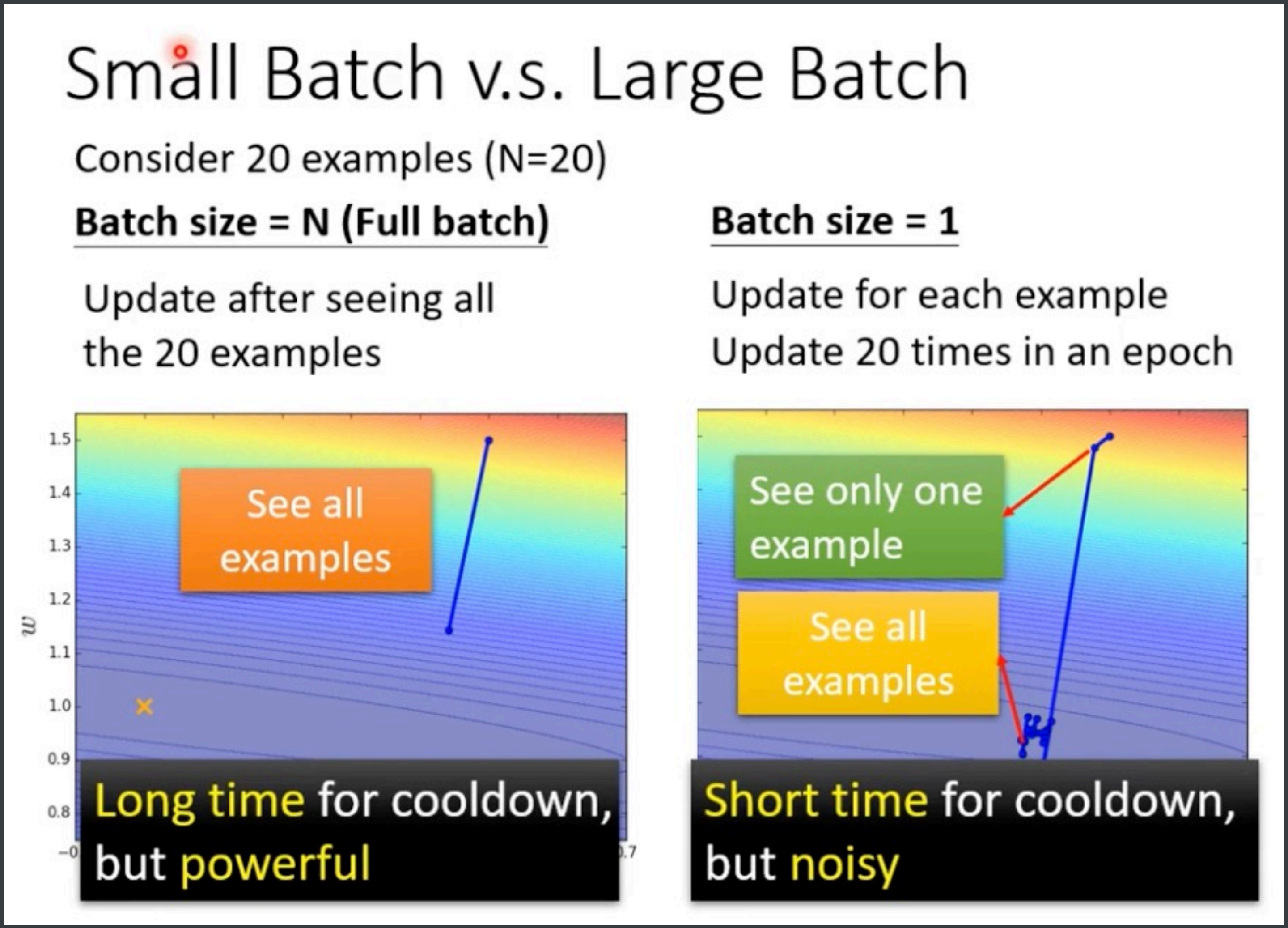
Batch

Why batch?

$$\theta^* = argmin\ L$$



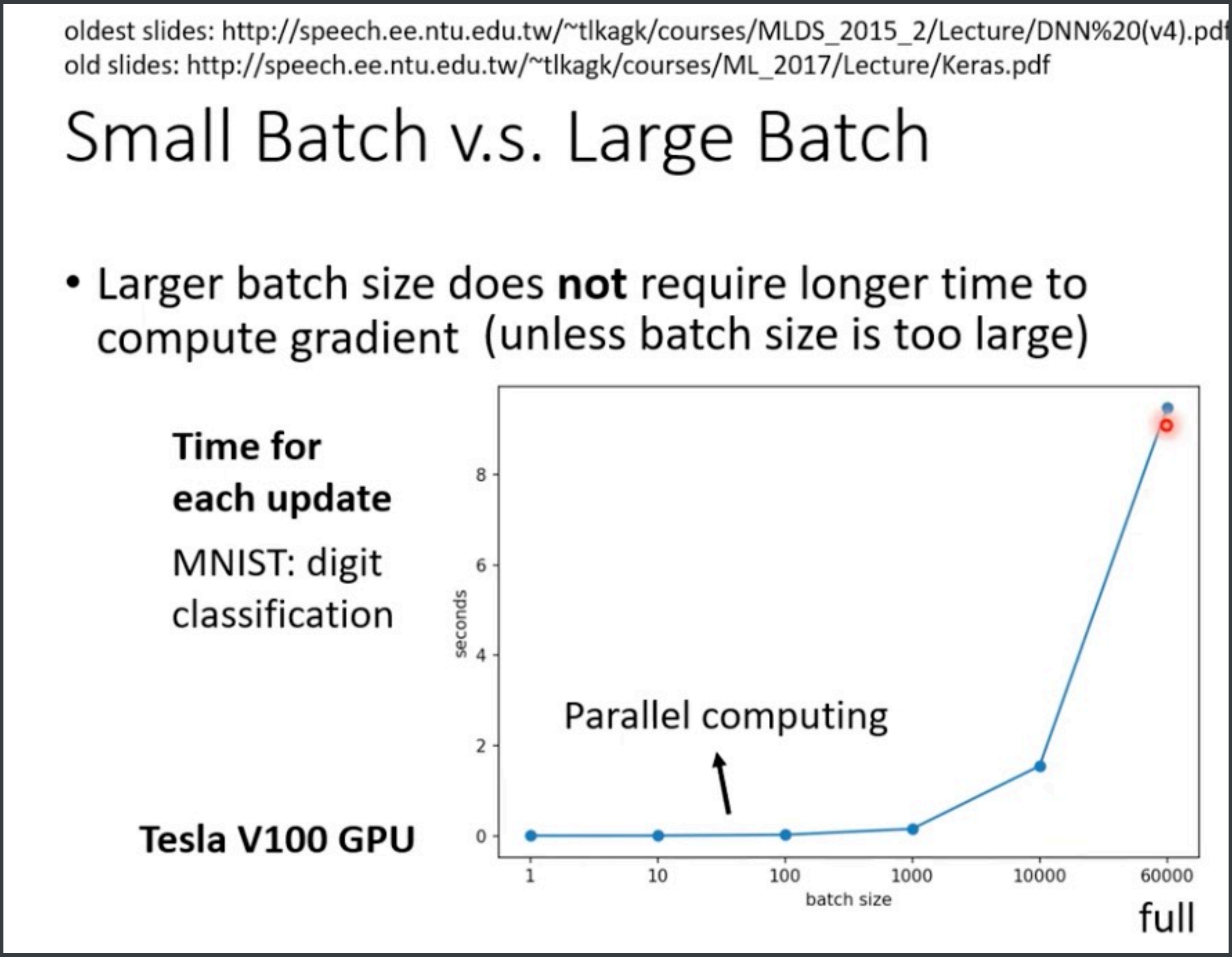
1 epoch = see all the batches once >>> Shuffle after each epoch



LHS: slow but powerful （Sniper rifle)

RHS: fast but 'noisy' (tututu(?)

Considering parallel running:

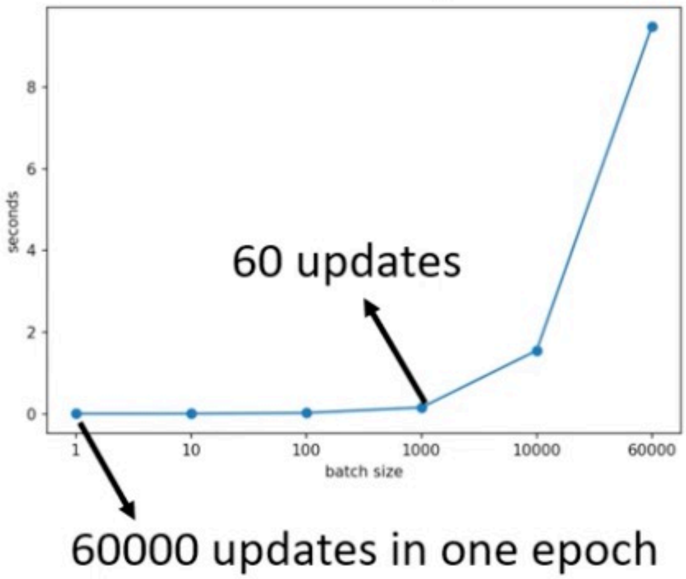


Consider significant time can be spent on updating:

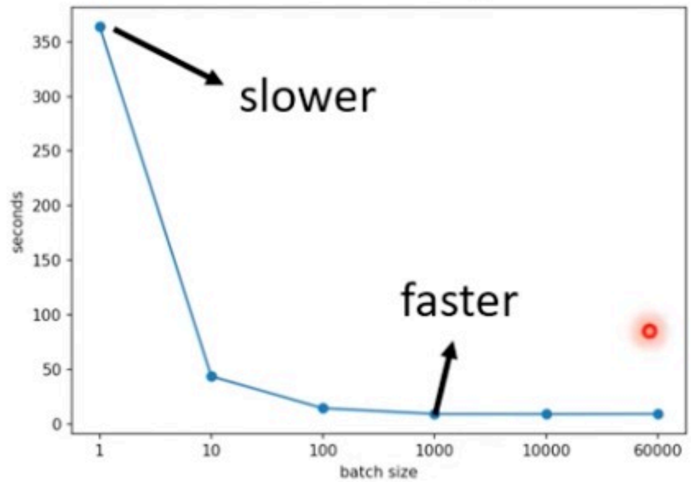
Small Batch v.s. Large Batch

- Smaller batch requires longer time for one epoch (longer time for seeing all data once)

Time for one **update**



Time for one **epoch**

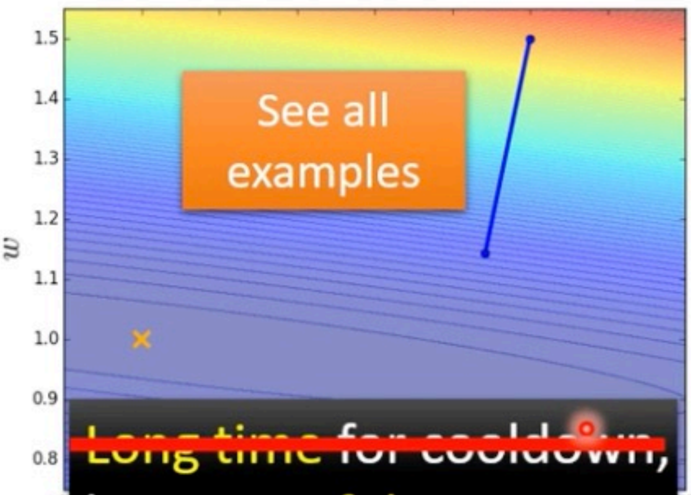


Small Batch v.s. Large Batch

Consider 20 examples (N=20)

Batch size = N (Full Batch)

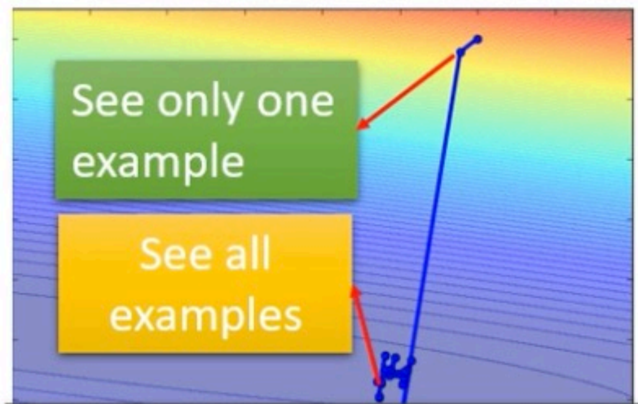
Update after seeing all the 20 examples



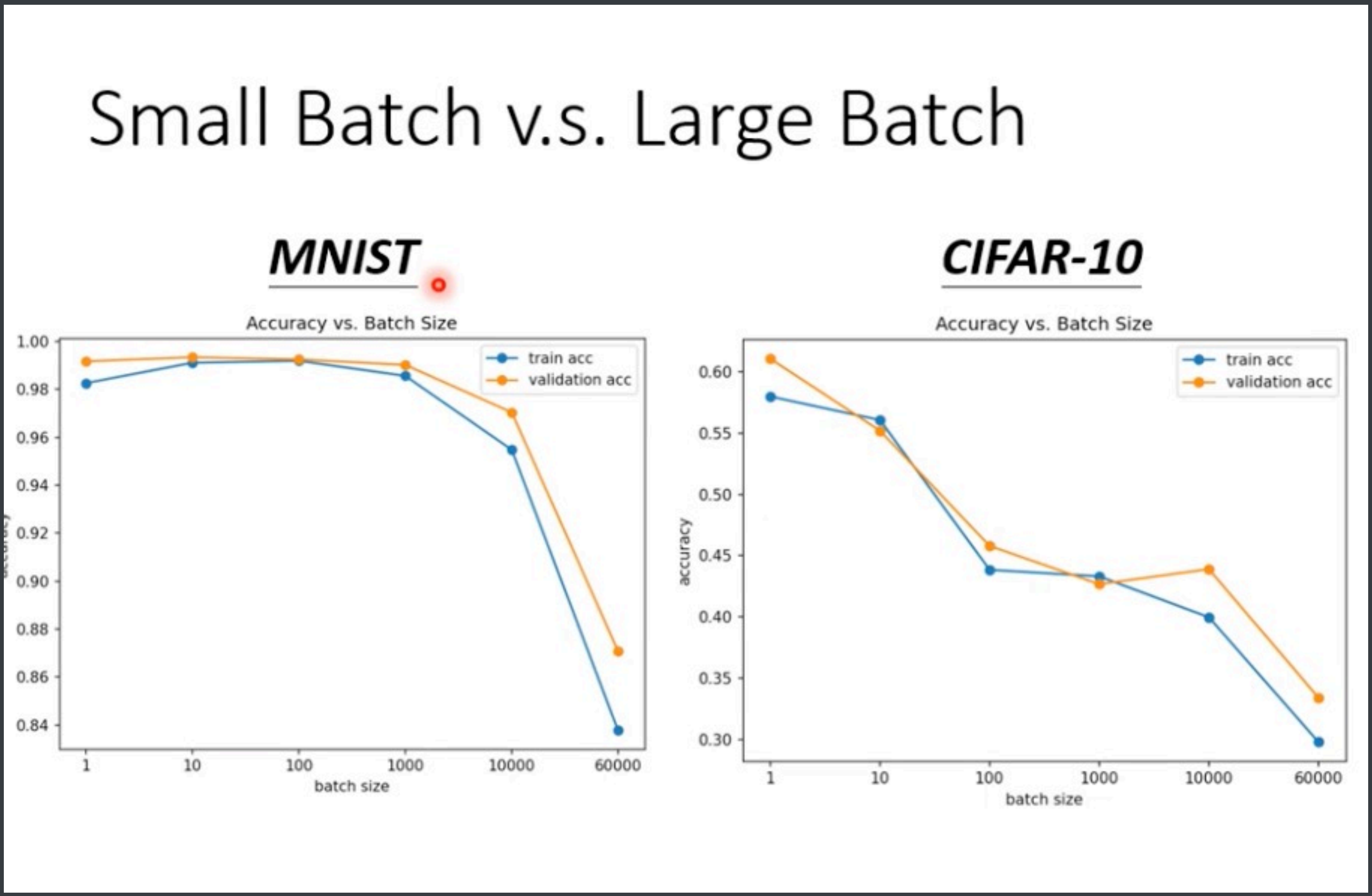
~~Long time for cooldown,~~ but powerful

Batch size = 1

Update for each example
Update 20 times in an epoch



~~Short time for cooldown,~~ but noisy

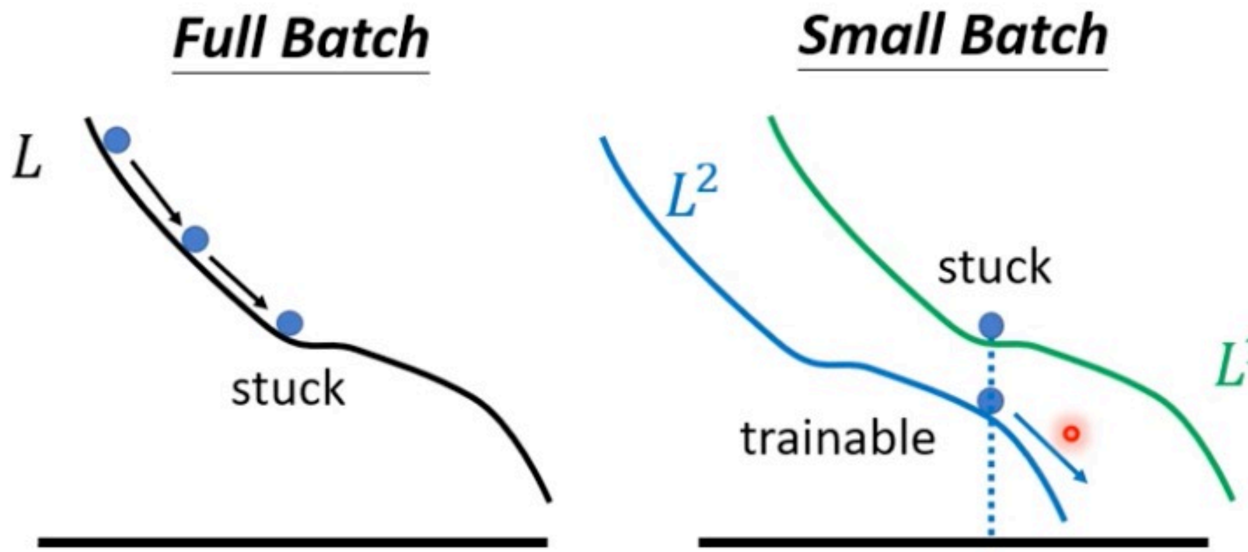


Smaller batch size has better performance

What's wrong with large batch size? Optimization Fails

Small Batch v.s. Large Batch

- Smaller batch size has better performance
- "Noisy" update is better for training



https://arxiv.org/abs/1609.04836

Small Batch v.s. Large Batch

- Small batch is better on testing data?

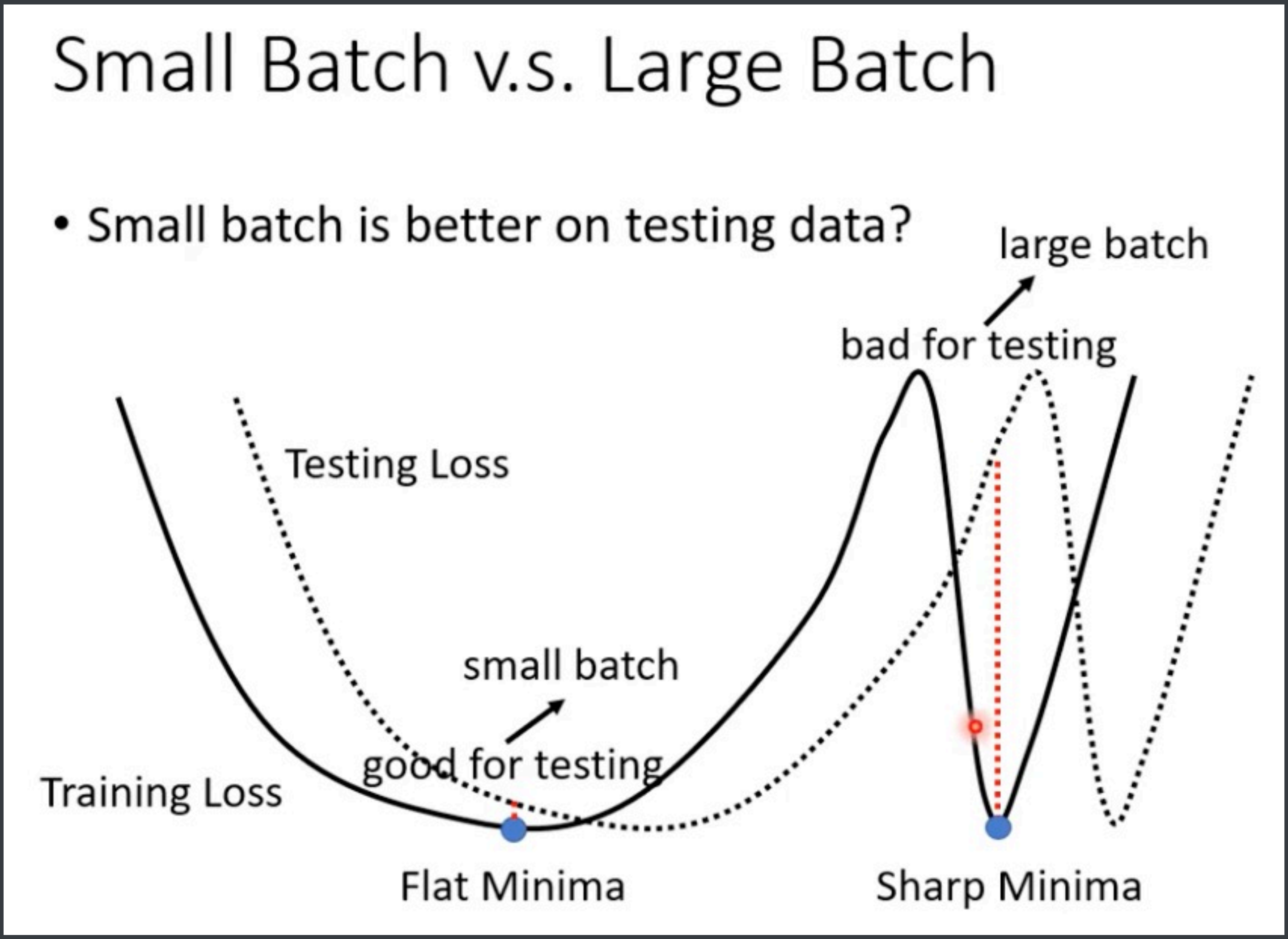
SB = 256

LB = 0.1 x data set

Name	Network Type	Data set
F_1	Fully Connected	MNIST (LeCun et al., 1998a)
F_2	Fully Connected	TIMIT (Garofolo et al., 1993)
C_1	(Shallow) Convolutional	CIFAR-10 (Krizhevsky & Hinton, 2009)
C_2	(Deep) Convolutional	CIFAR-10
C_3	(Shallow) Convolutional	CIFAR-100 (Krizhevsky & Hinton, 2009)
C_4	(Deep) Convolutional	CIFAR-100

	Training Accuracy		Testing Accuracy	
Name	SB	LB	SB	LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
C_2	$99.99\% \pm 0.04\%$	$99.99\% \pm 0.01\%$	$89.24\% \pm 0.12\%$	$87.26\% \pm 0.07\%$
C_3	$99.56\% \pm 0.44\%$	$99.88\% \pm 0.30\%$	$49.58\% \pm 0.39\%$	$46.45\% \pm 0.43\%$
C_4	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

Flat Minima is better than Sharp Minima



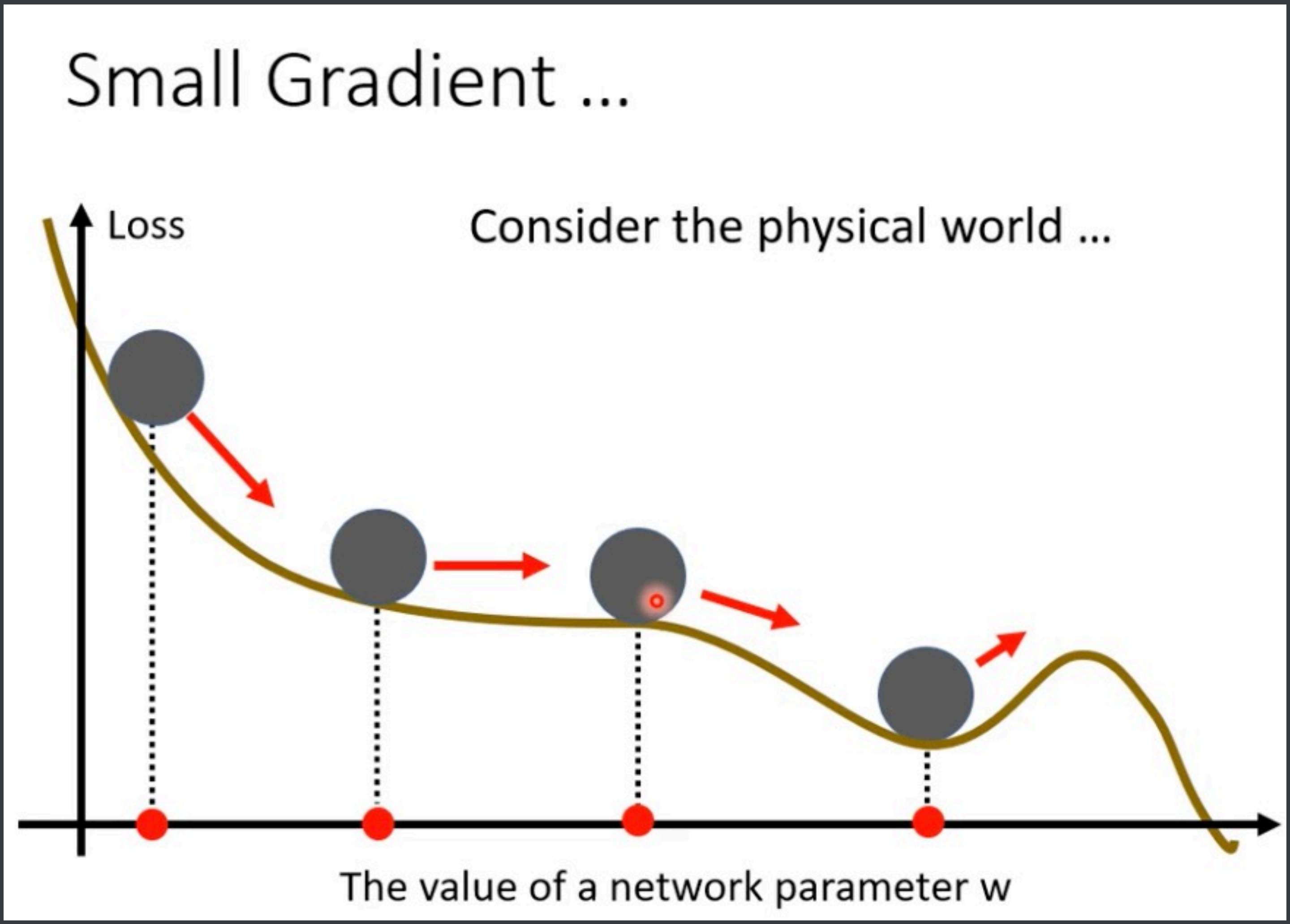
The difference between train and test set is large at sharp minima

Small Batch v.s. Large Batch

	Small	Large
Speed for one update (no parallel)	Faster	Slower
Speed for one update (with parallel)	Same	Same (not too large)
Time for one epoch	Slower	Faster WIN
Gradient	Noisy	Stable
Optimization	Better WIN	Worse
Generalization	Better WIN	Worse

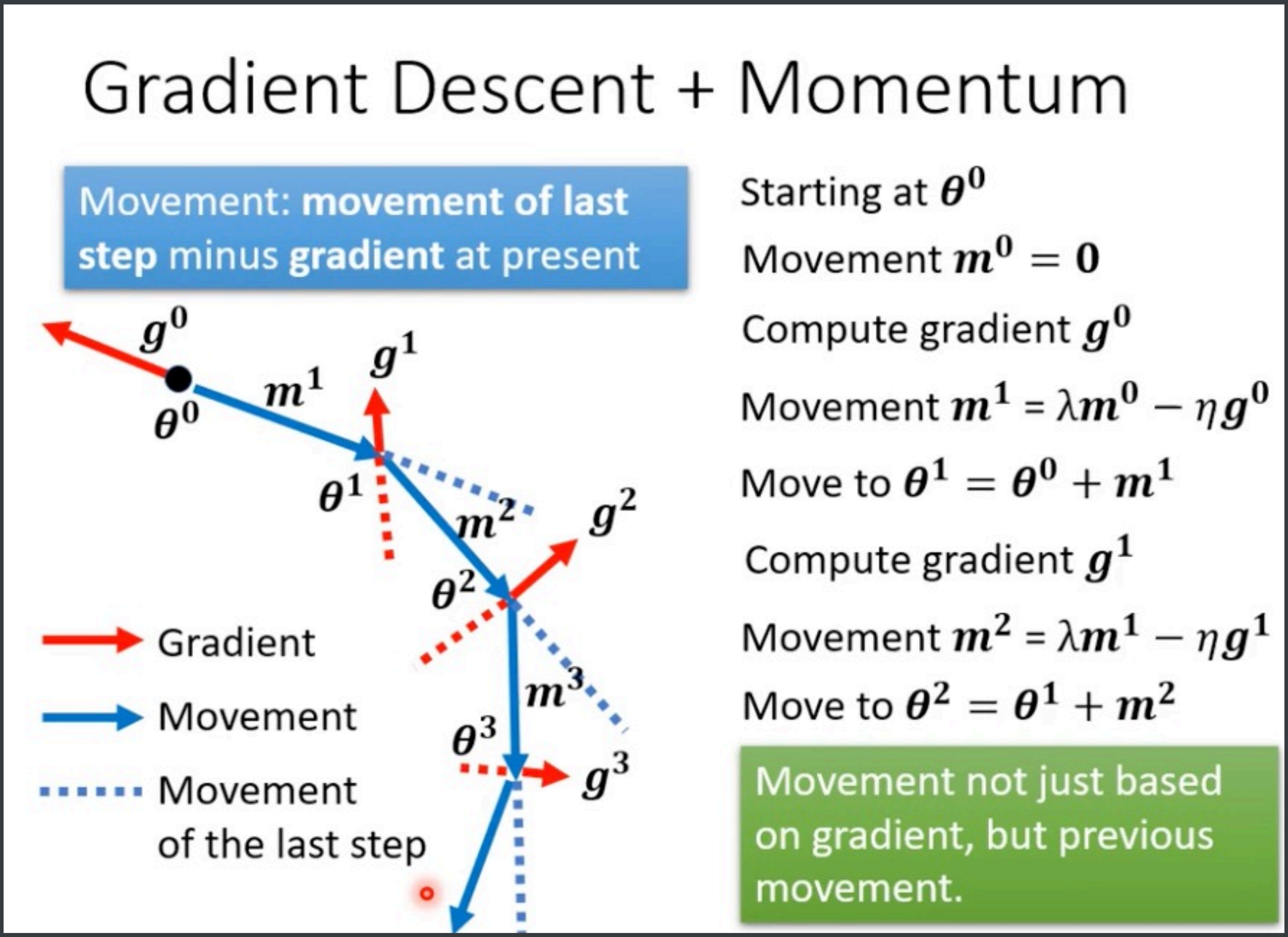
We want to train the batch size as a hyper-parameter

Momentum



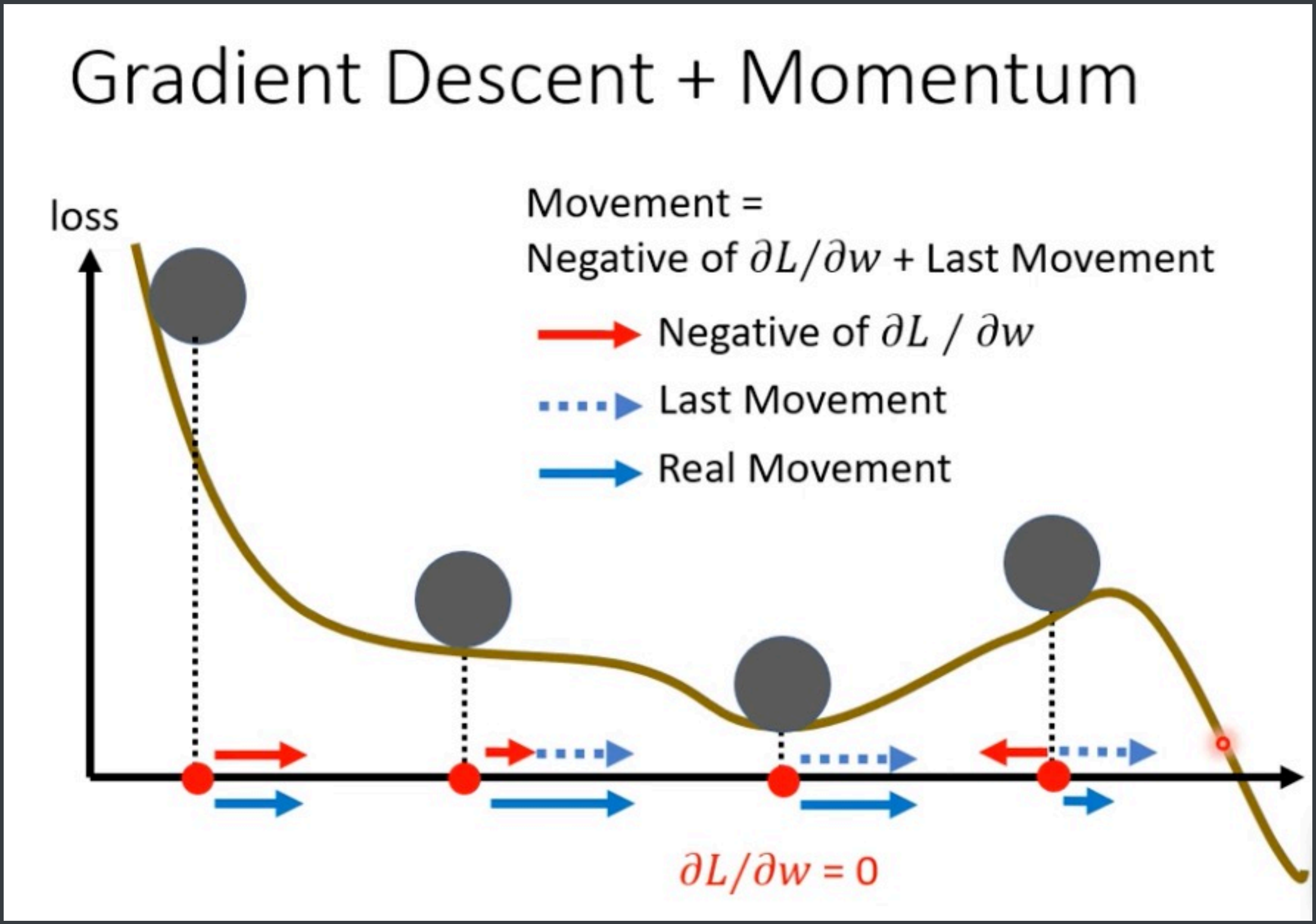
(Vanilla) Gradient Descent + Momentum

Movement: movement of last step minus gradient at present



$$m^i \text{ is the weighted sum of all the previous gradient } : g^0, g^1, \ldots, g^{i-1}$$

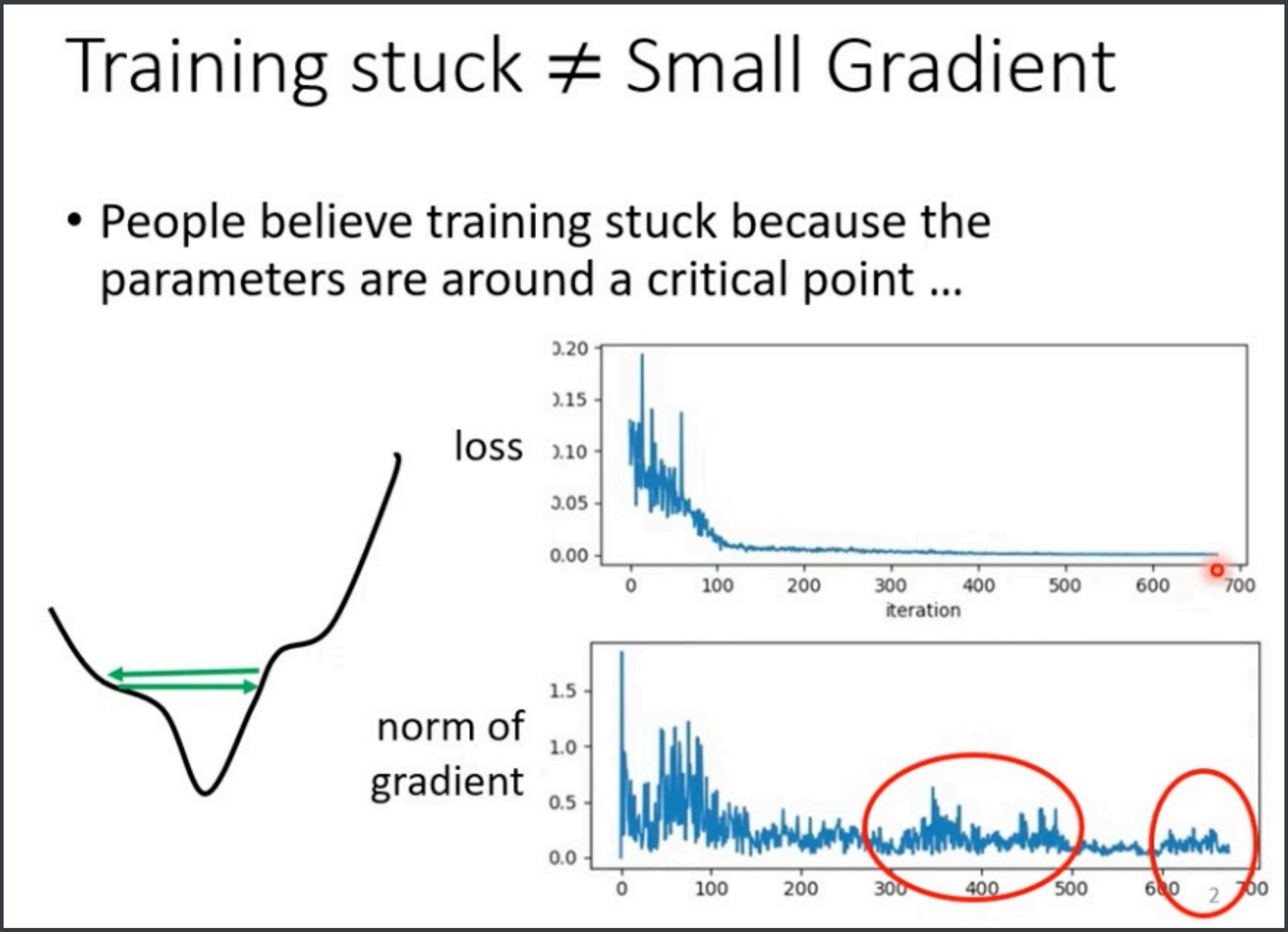
$$m^0 = 0$$
$$m^1 = -\eta g^0$$
$$m^2 = -\lambda \eta g^0 - \eta g^1$$
$$\vdots$$



Conclusion Remarks

- Critical points have zero gradients
- Critical points can be either saddle points or local minima
  - Can be determined by the Hessian matrix
  - It's possible to escape saddle points along the direction of eigenvectors of the Hessian matrix
  - Local minima may be rare
- Smaller batch size and momentum help escape critical points

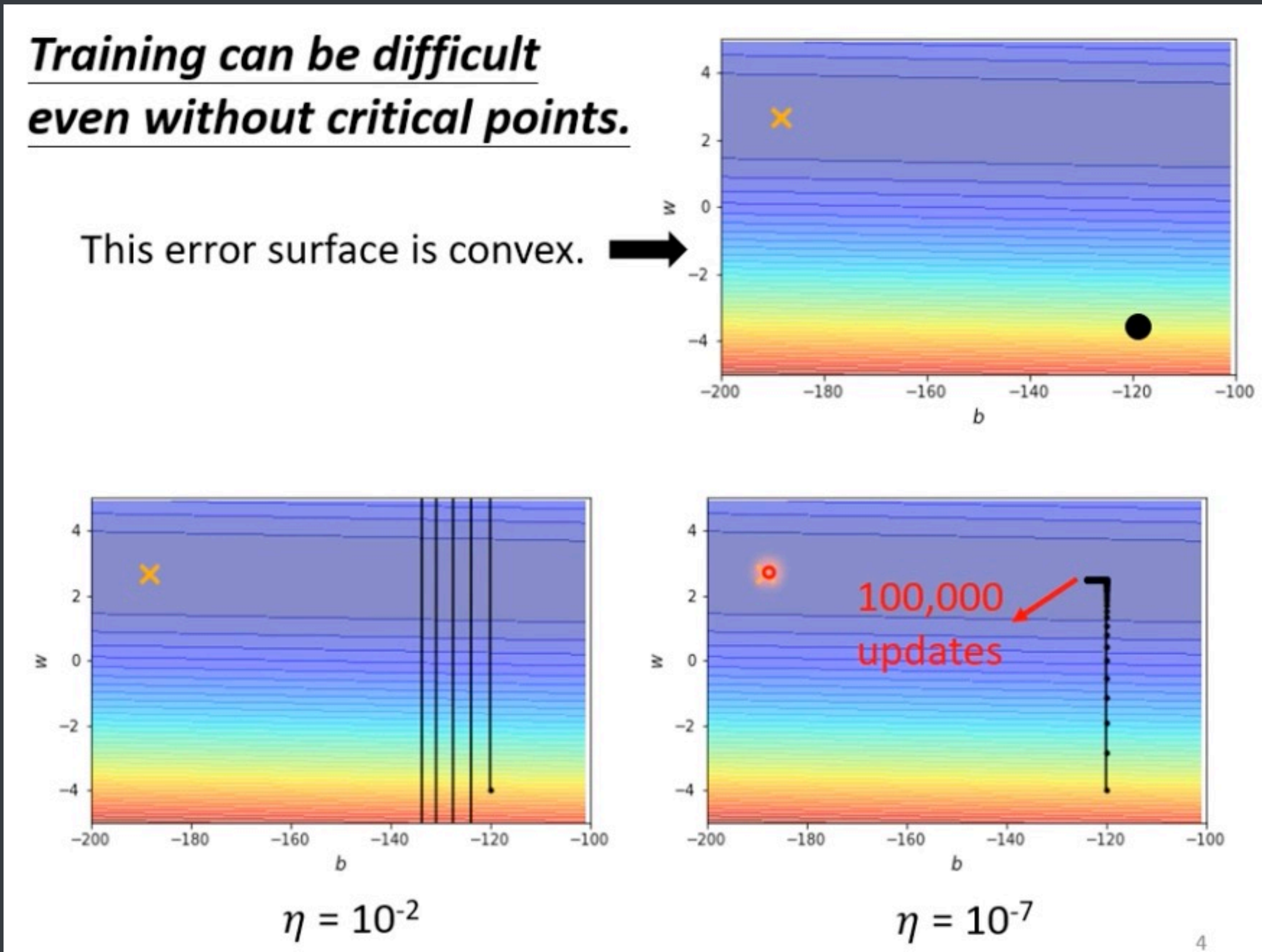
3) Adaptive Learning Rate



In this case, loss cannot go down and it is not because gradient become 0

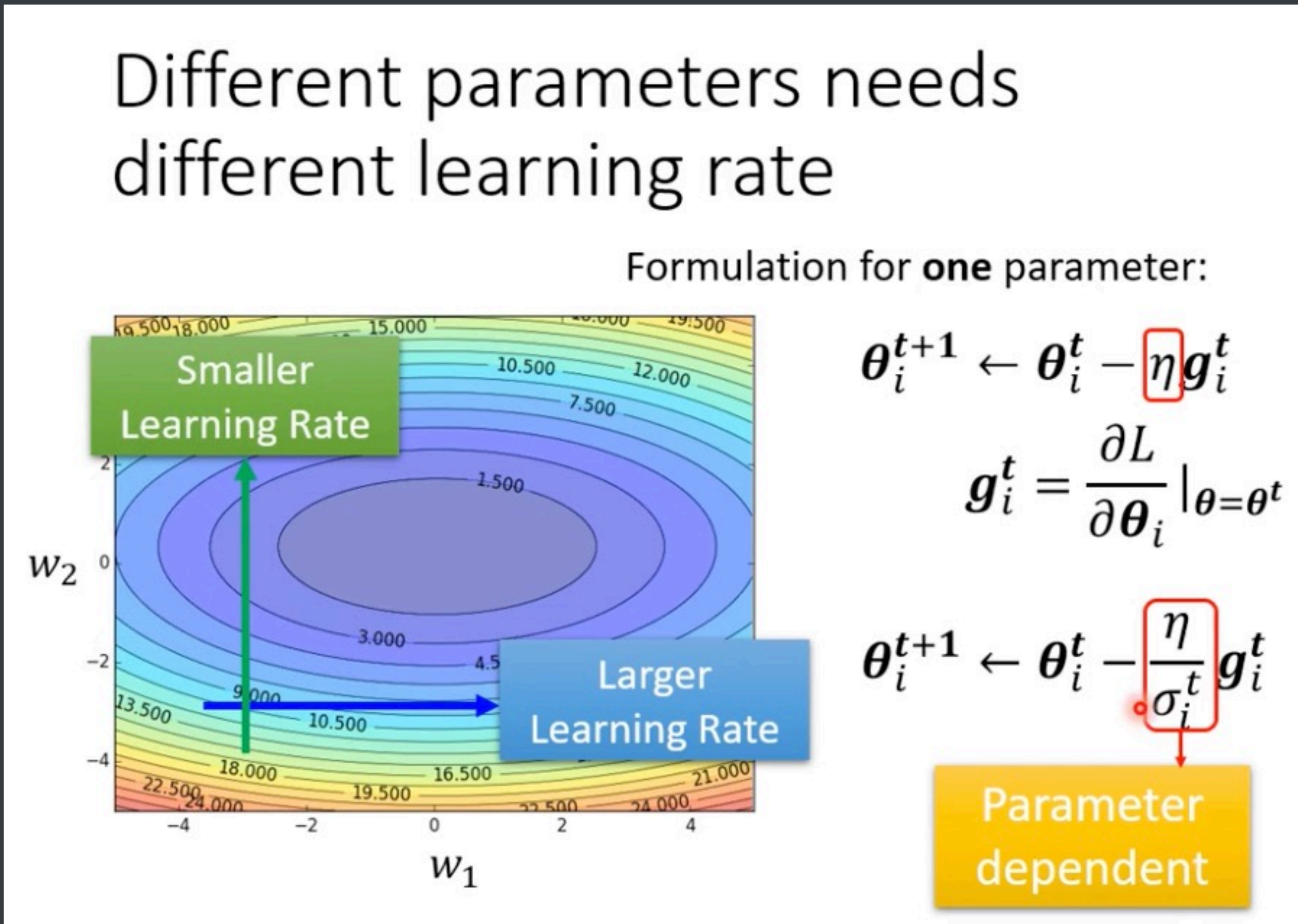
In many cases, loss cannot go down before actually reaching a local minimum.

Training can be difficult even without critical points...



Even with a small lerning rage (10^-7), the training cannot step forward

Learning rate cannot be one-size-fits-all: Different parameters needs different learning rate



$$\theta_i^{t+1} \leftarrow \theta_i^{y} - \eta g_i^t$$

$$g_i^t = \frac{\partial L}{\partial \theta_i}|_{\theta=\theta^t}$$

$$\theta_i^{t+1} \leftarrow \theta_i^{t} - \frac{\eta}{\sigma^t} g_i^t$$

What are the possible forms of sigma? The parameter dependent rate

- Root Mean Square

$$\theta_i^1 \leftarrow \theta_i^0 - \frac{\eta}{\sigma_i^0} g_i^0 \ ; \ \ \sigma_i^0 = \sqrt{(g_i^0)^2} = |g_i^0|$$