Workspace 'Workspace' in 'small-gradient-v7 (1)'
Page 1 (row 1, column 1)

假设有 Optimization Fails.

local minima　　　saddle point

Solution: use Hessian metric

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \boxed{\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H(\boldsymbol{\theta} - \boldsymbol{\theta}')}$$

其中红色部分就是 H (Hessian)

具体来说：

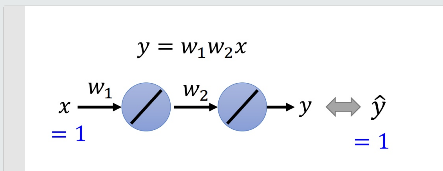
可以用 H 来解释

local minima.

local maxima

{ saddle point.

eg: 假设现在有个很烂的F.



作用就只是 times w_1 / w_2.

if input $x=1$
output $y=1$ } $w_1 \cdot w_2 = ?$

然后看 Error surface

发现这样的其实蛮好找到

但如果今天的 Error surface

其实是一个不那么容易看出的

图？

Workspace 'Workspace' in 'small-gradient-v7 (1)'
Page 3 (row 3, column 1)

用 Hessian 矩阵来求特征值

⇓ 可以得到结论.

那接下来?

可以材材看一个例子

放过说可以用高 dimensional 的角度去进入.

三维的物体

如果我们的维数很高

那么是否就会有更少的 local minima 呢?

结论是: Yes!

所以在现在这种大degree
大data set 的背景下,
local minima 其实不多..
更多可能是 saddle point.

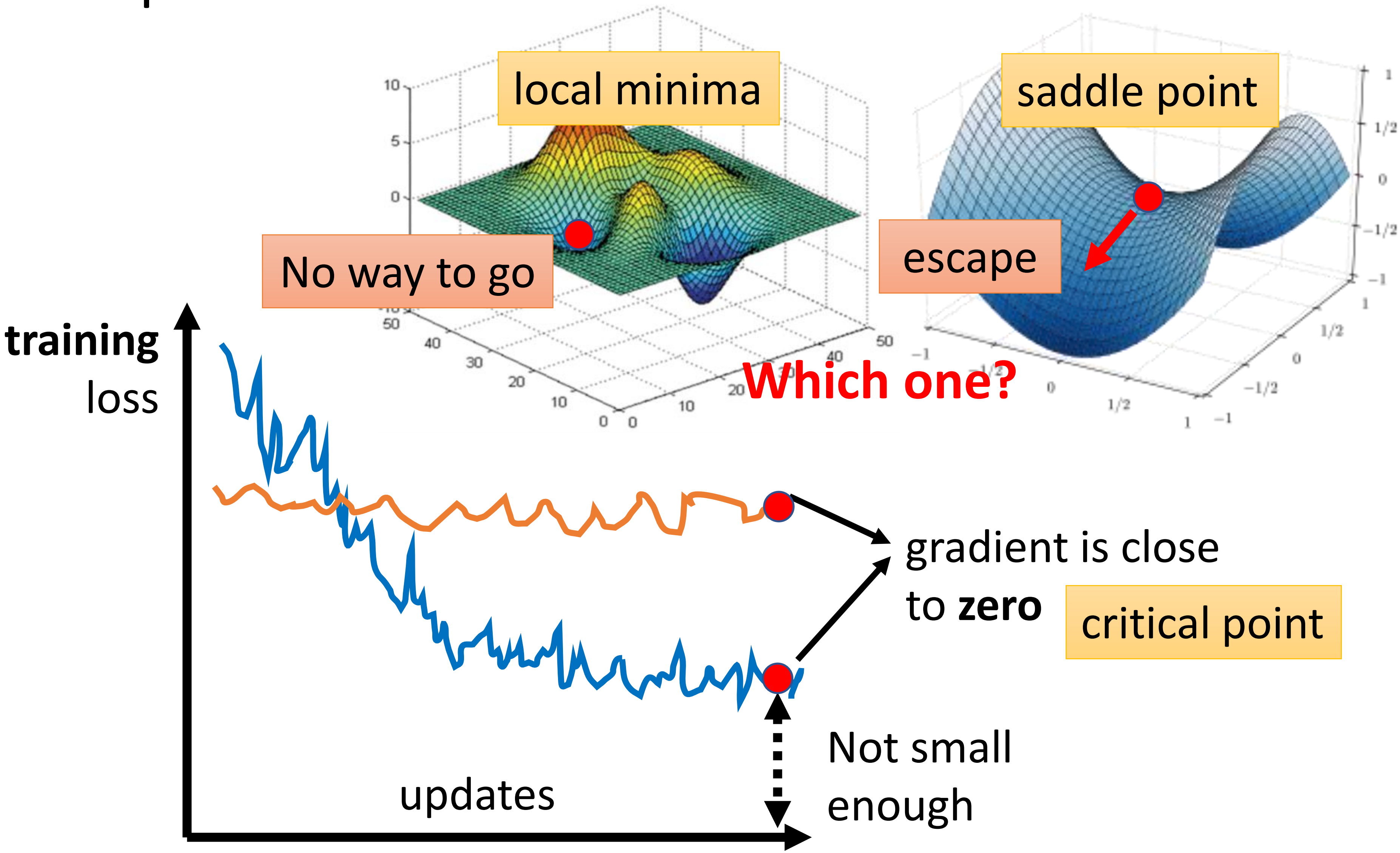
$\Downarrow$

easy to solve!

When gradient is small …

Hung-yi Lee 李宏毅

Optimization Fails because ……



Warning of Math

Tayler Series Approximation

$L(\boldsymbol{\theta})$ around $\boldsymbol{\theta} = \boldsymbol{\theta}'$ can be approximated below

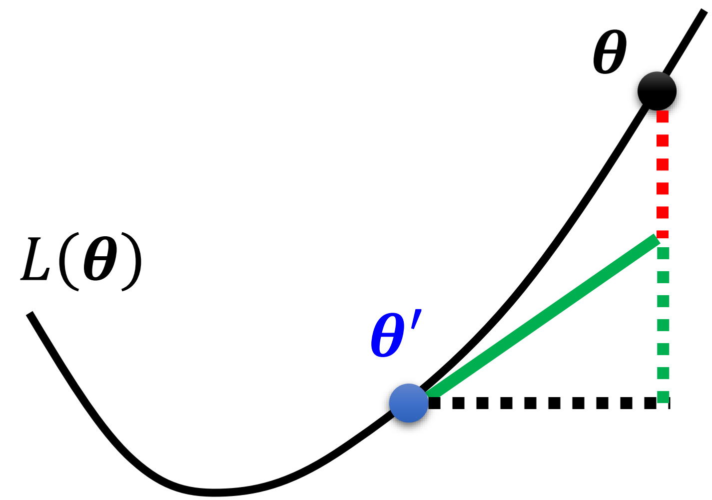
$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \boxed{(\boldsymbol{\theta} - \boldsymbol{\theta}')^T \boldsymbol{g}} + \boxed{\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H (\boldsymbol{\theta} - \boldsymbol{\theta}')}$$

Gradient $\boldsymbol{g}$ is a ***vector***

$$\boldsymbol{g} = \nabla L(\boldsymbol{\theta}') \qquad \boldsymbol{g}_i = \frac{\partial L(\boldsymbol{\theta}')}{\partial \boldsymbol{\theta}_i}$$

Hessian H is a ***matrix***

$$H_{ij} = \frac{\partial^2}{\partial \boldsymbol{\theta}_i \partial \boldsymbol{\theta}_j} L(\boldsymbol{\theta}')$$



Source of image:http://www.offconvex.org/2016/03/22/saddlepoints/

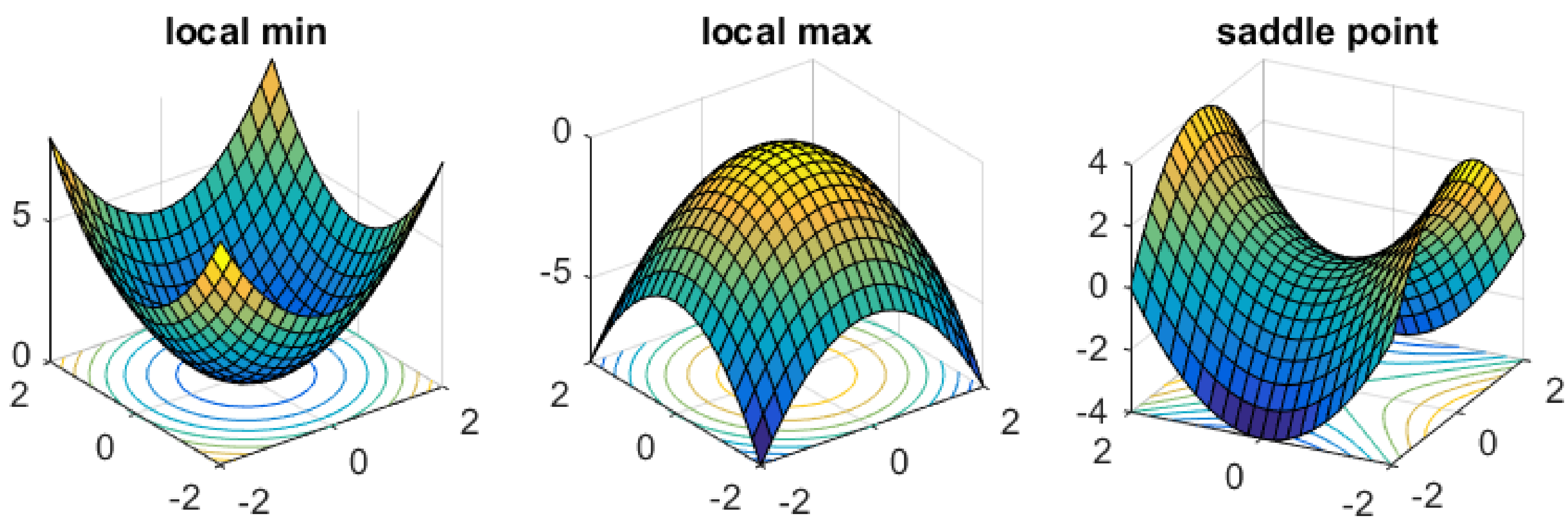
Hessian

$L(\boldsymbol{\theta})$ around $\boldsymbol{\theta} = \boldsymbol{\theta}'$ can be approximated below

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \boxed{\cancel{(\boldsymbol{\theta} - \boldsymbol{\theta}')^T \boldsymbol{g}}} + \boxed{\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H (\boldsymbol{\theta} - \boldsymbol{\theta}')}$$

At critical point

telling the properties of critical points



Hessian

At critical point:

$$\boldsymbol{v}^T H \boldsymbol{v}$$

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \boxed{\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H (\boldsymbol{\theta} - \boldsymbol{\theta}')}$$

For all $\boldsymbol{v}$

$\boldsymbol{v}^T H \boldsymbol{v} > 0$ ➡ Around $\boldsymbol{\theta}'$: $L(\boldsymbol{\theta}) > L(\boldsymbol{\theta}')$ ➡ **Local minima**

= H is positive definite = All eigen values are positive. ⬆

For all $\boldsymbol{v}$

$\boldsymbol{v}^T H \boldsymbol{v} < 0$ ➡ Around $\boldsymbol{\theta}'$: $L(\boldsymbol{\theta}) < L(\boldsymbol{\theta}')$ ➡ **Local maxima**

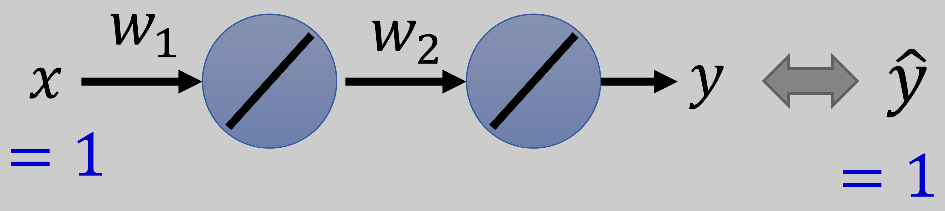
= H is negative definite = All eigen values are negative. ⬆

Sometimes $\boldsymbol{v}^T H \boldsymbol{v} > 0$, sometimes $\boldsymbol{v}^T H \boldsymbol{v} < 0$ ➡ **Saddle point**

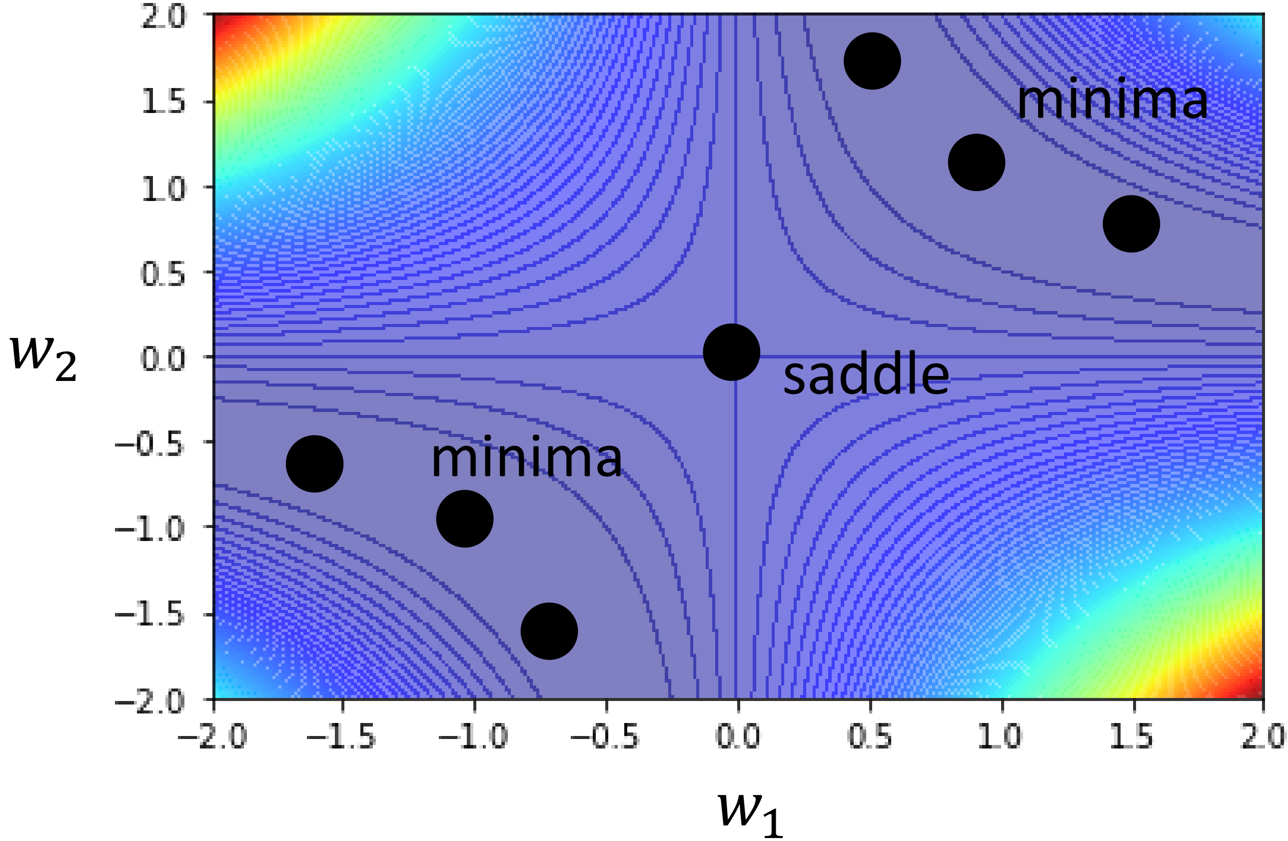
Some eigen values are positive, and some are negative. ⬆

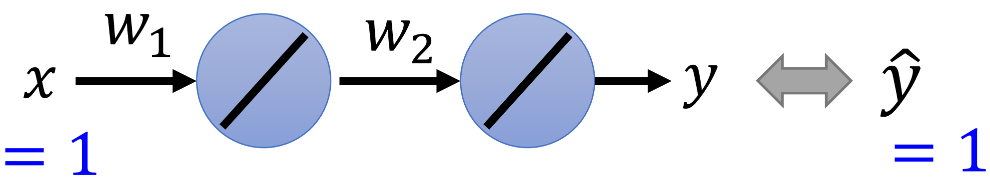
Example

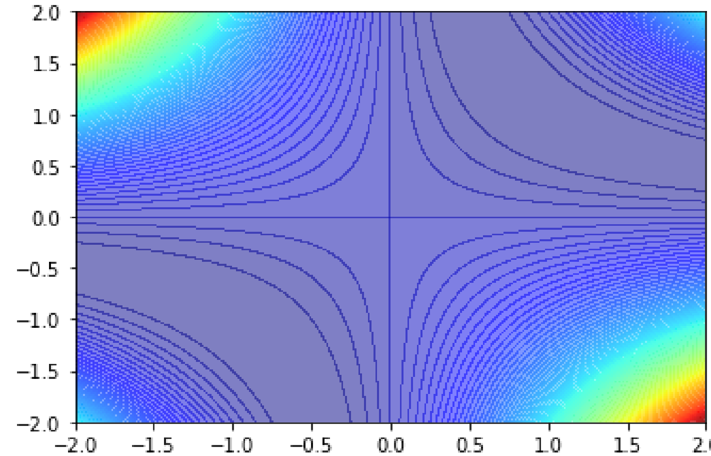
$$y = w_1 w_2 x$$



Error Surface







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

$$\boxed{\frac{\partial L}{\partial w_1}} = 2(1 - w_1 w_2)(-w_2) \quad = 0$$

$$\boxed{\frac{\partial L}{\partial w_2}} = 2(1 - w_1 w_2)(-w_1) \quad = 0$$

g

Critical point: $w_1 = 0, w_2 = 0$

$$H = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix} \quad \lambda_1 = 2, \lambda_2 = -2$$

Saddle point

H

$$\boxed{\frac{\partial^2 L}{\partial {w_1}^2}} = 2(-w_2)(-w_2) \quad = 0$$

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

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

$$\boxed{\frac{\partial^2 L}{\partial {w_2}^2}} = 2(-w_1)(-w_1) \quad = 0$$

Don't afraid of saddle point?

At critical point: $$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \overset{\boldsymbol{v}^T H \boldsymbol{v}}{\boxed{\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T H (\boldsymbol{\theta} - \boldsymbol{\theta}')}}$$

Sometimes $\boldsymbol{v}^T H \boldsymbol{v} > 0$, sometimes $\boldsymbol{v}^T H \boldsymbol{v} < 0$ ➡ Saddle point

H may tell us parameter update direction!

$\boldsymbol{u}$ is an eigen vector of H

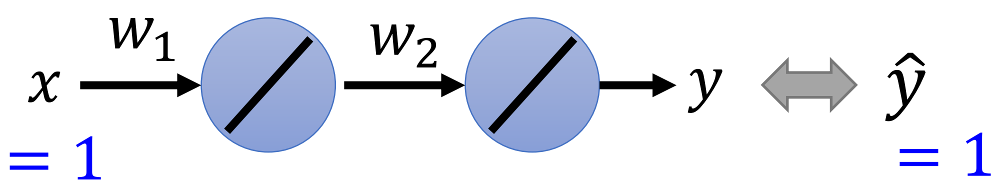
λ is the eigen value of $\boldsymbol{u}$

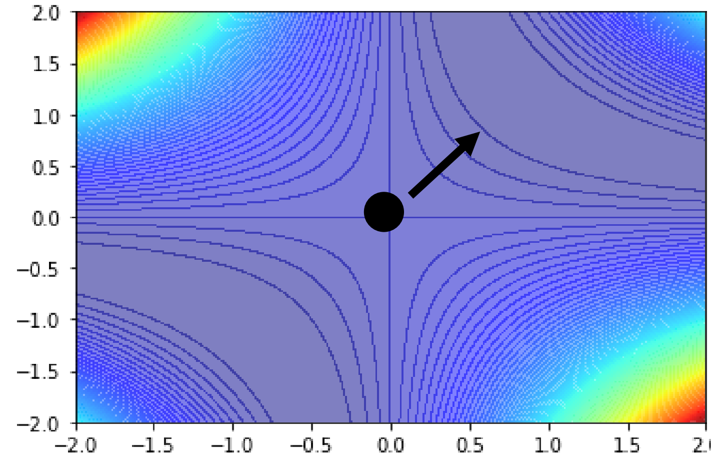
$\lambda < 0$

➡ $$\underset{<0}{\boldsymbol{u}^T H \boldsymbol{u}} = \boldsymbol{u}^T(\lambda \boldsymbol{u}) = \underset{<0}{\lambda \|\boldsymbol{u}\|^2}$$

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}') + \frac{1}{2}(\overset{\boldsymbol{u}}{\boldsymbol{\theta} - \boldsymbol{\theta}'})^T H (\overset{\boldsymbol{u}}{\boldsymbol{\theta} - \boldsymbol{\theta}'})$$ ➡ $L(\boldsymbol{\theta}) < L(\boldsymbol{\theta}')$

$\boldsymbol{\theta} - \boldsymbol{\theta}' = \boldsymbol{u}$ $\quad$ $\boldsymbol{\theta} = \boldsymbol{\theta}' + \boldsymbol{u}$ $\quad$ Decrease L





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

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

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

Critical point: $w_1 = 0, w_2 = 0$

$$H = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix} \quad \lambda_1 = 2, \lambda_2 = -2$$

Saddle point

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

Update the parameter along the direction of $\boldsymbol{u}$

You can escape the saddle point and decrease the loss.

(this method is seldom used in practice)

End of Warning

Saddle Point v.s. Local Minima

- A.D. 1543

來源《三體Ⅲ·死神永生》

Saddle Point v.s. Local Minima

- The Magician Diorena（魔法師狄奧倫娜）

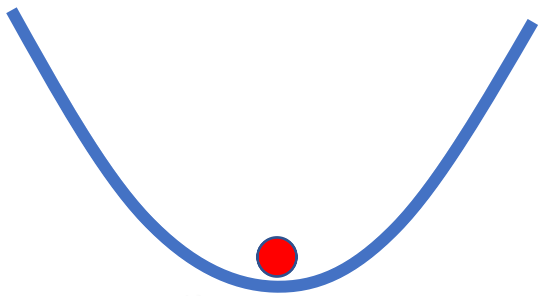
From 3 dimensional space, it is sealed.

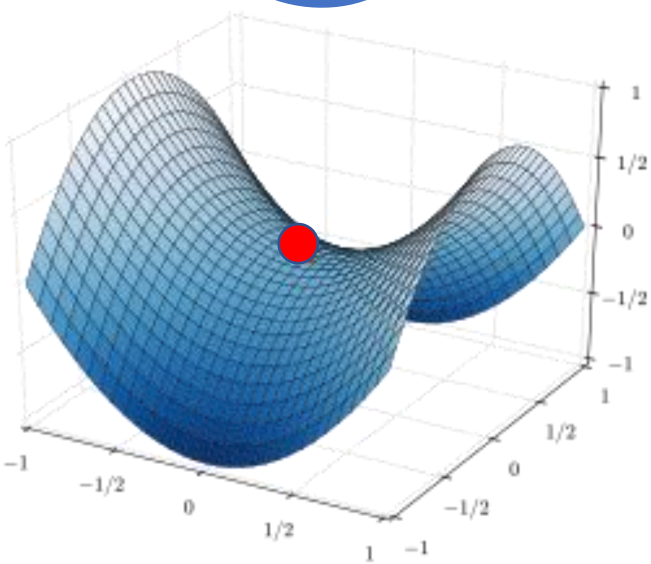
It is not in higher dimensions.

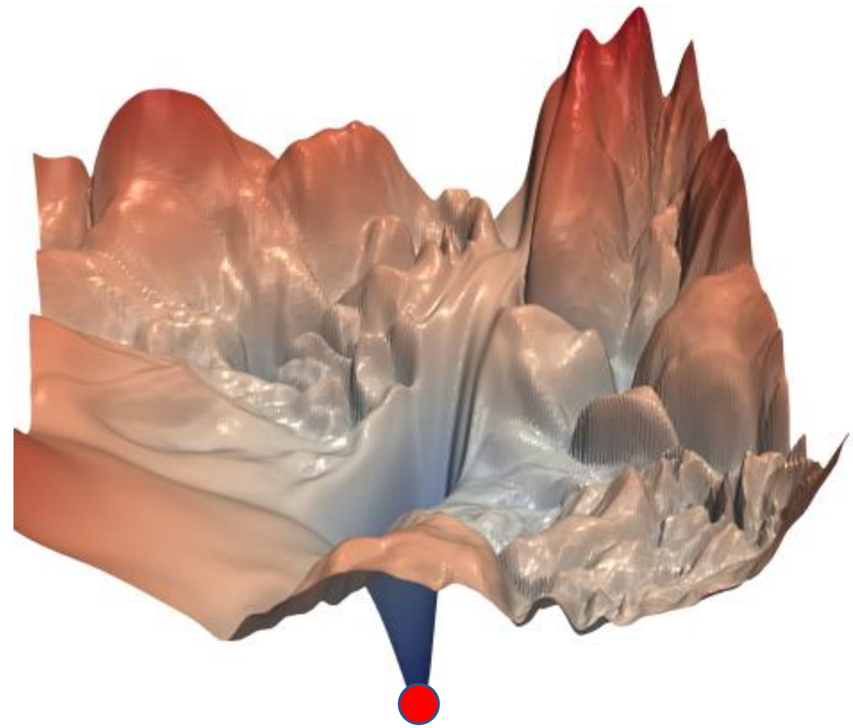
Source of image: https://read01.com/mz2DBPE.html#.YECz22gzbIU

Source of image: https://arxiv.org/abs/1712.09913

Saddle Point v.s. Local Minima





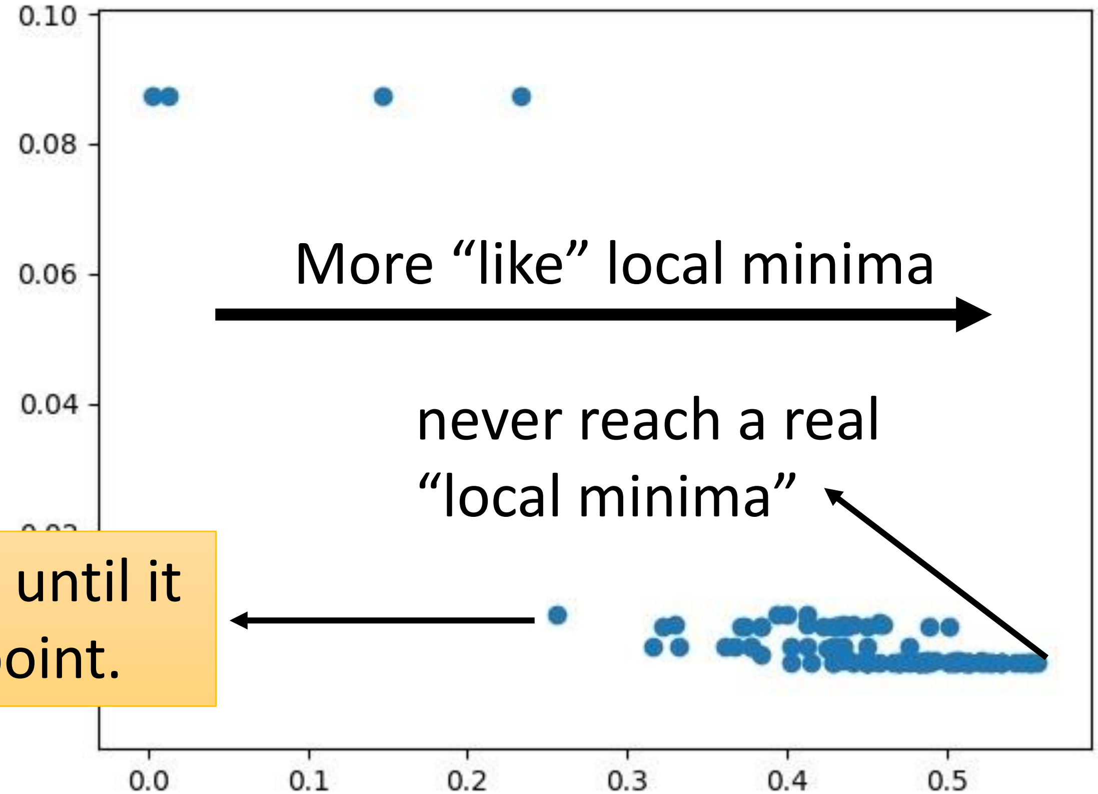


Saddle point in higher dimension?

When you have lots of parameters, perhaps local minima is rare?

Empirical Study

Training Loss



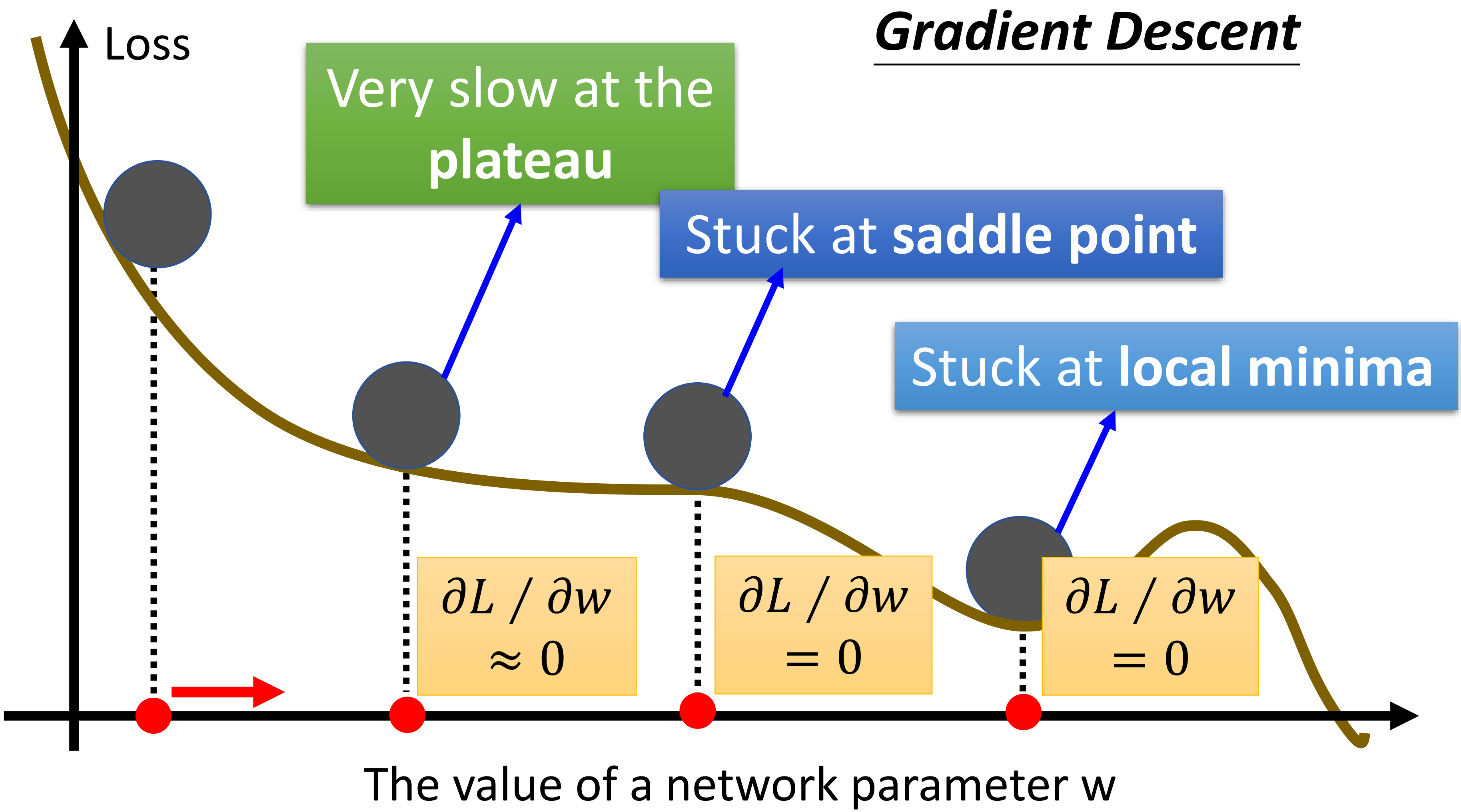
Minimum Ratio

Train a network once, until it converges to critical point.

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

Source:https://docs.google.com/presentation/d/1siUFXARYRpNiMeSRwgFbt7mZVjkMPhR5od09w0Z8xaU/edit#slide=id.g31470fd33a_0_33

Small Gradient ...



Tips for training:
Batch and Momentum

Batch

Review: Optimization with Batch

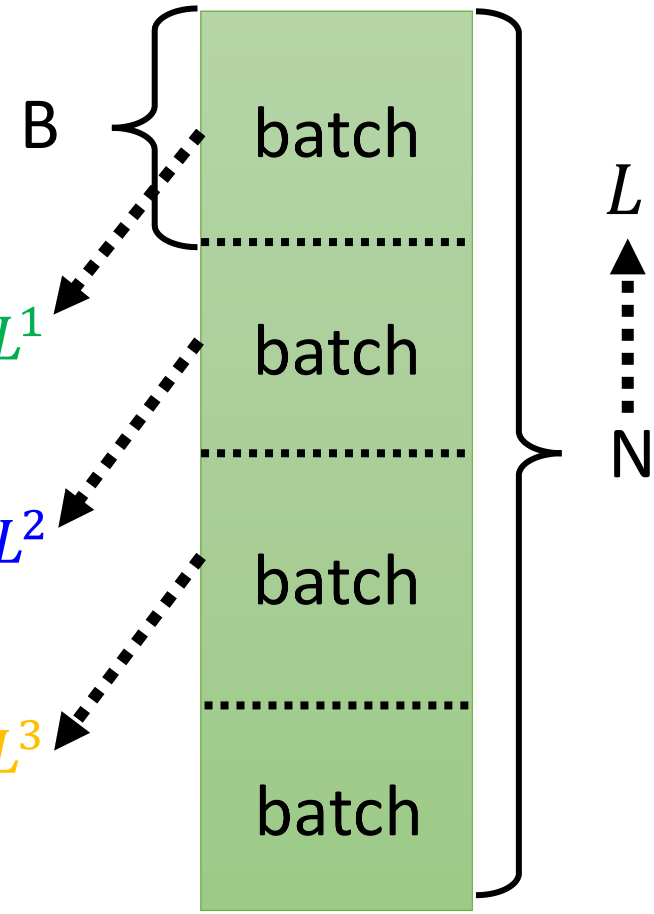
$$\boldsymbol{\theta}^* = arg\,\min_{\boldsymbol{\theta}} L$$

- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- Compute gradient $\boldsymbol{g}^{\mathbf{0}} = \nabla L^1(\boldsymbol{\theta}^0)$ L^1

  update $\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}^{\mathbf{0}}$
- Compute gradient $\boldsymbol{g}^{\mathbf{1}} = \nabla L^2(\boldsymbol{\theta}^1)$ L^2

  update $\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}^{\mathbf{1}}$
- Compute gradient $\boldsymbol{g}^{\mathbf{3}} = \nabla L^3(\boldsymbol{\theta}^2)$ L^3

  update $\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}^{\mathbf{3}}$



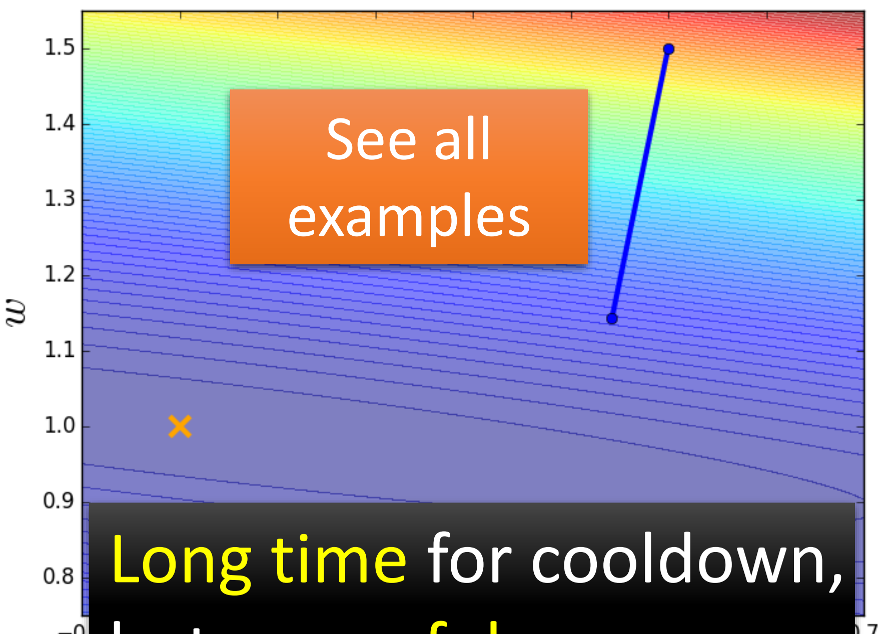
1 **epoch** = see all the batches once → **Shuffle** after each epoch

Small Batch v.s. Large Batch

Consider 20 examples (N=20)

Batch size = N (Full batch)

Update after seeing all the 20 examples

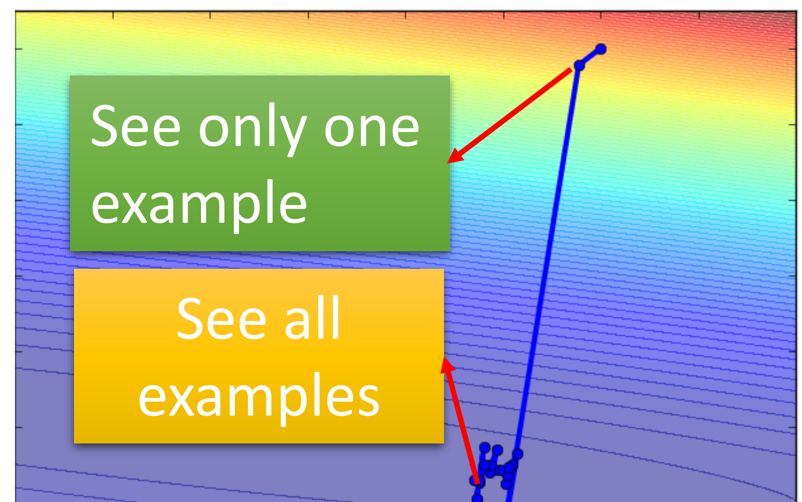


Long time for cooldown, but powerful

Batch size = 1

Update for each example

Update 20 times in an epoch



Short time for cooldown, but noisy

oldest slides: http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20(v4).pdf
old slides: http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/Keras.pdf

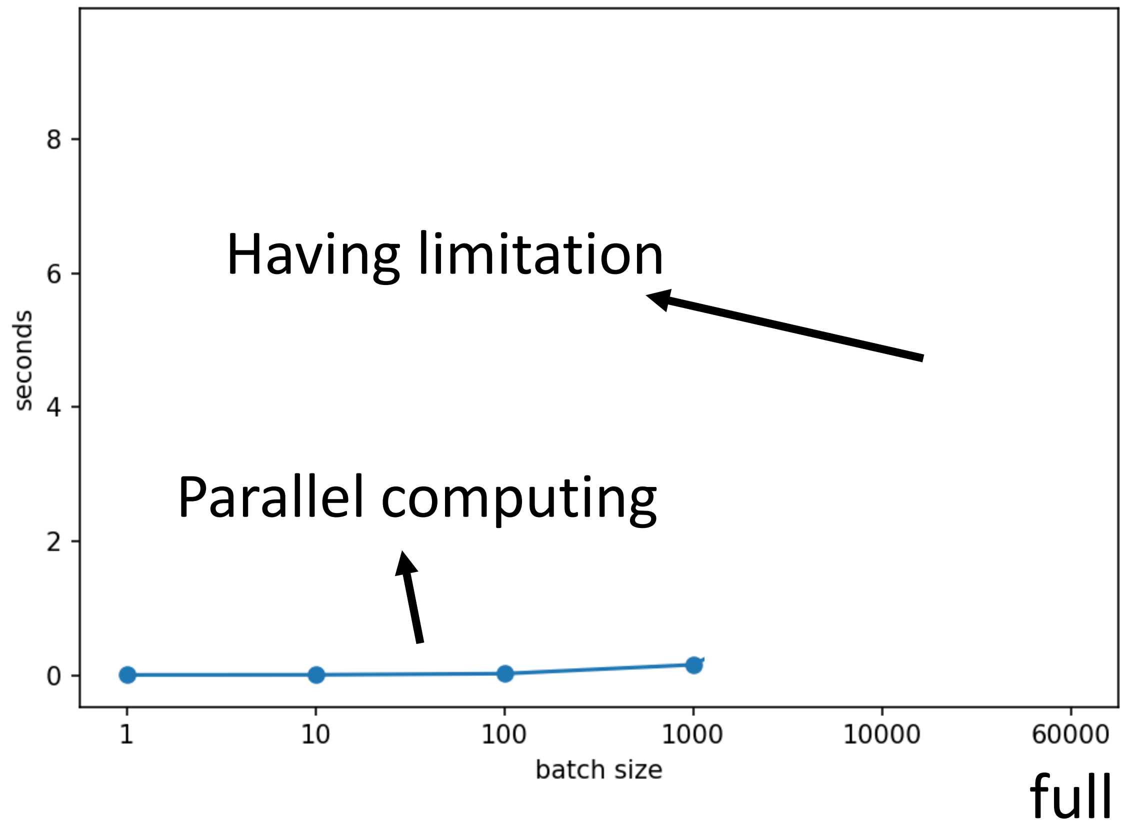
Small Batch v.s. Large Batch

- Larger batch size does **not** require longer time to compute gradient (unless batch size is too large)

Time for each update

MNIST: digit classification

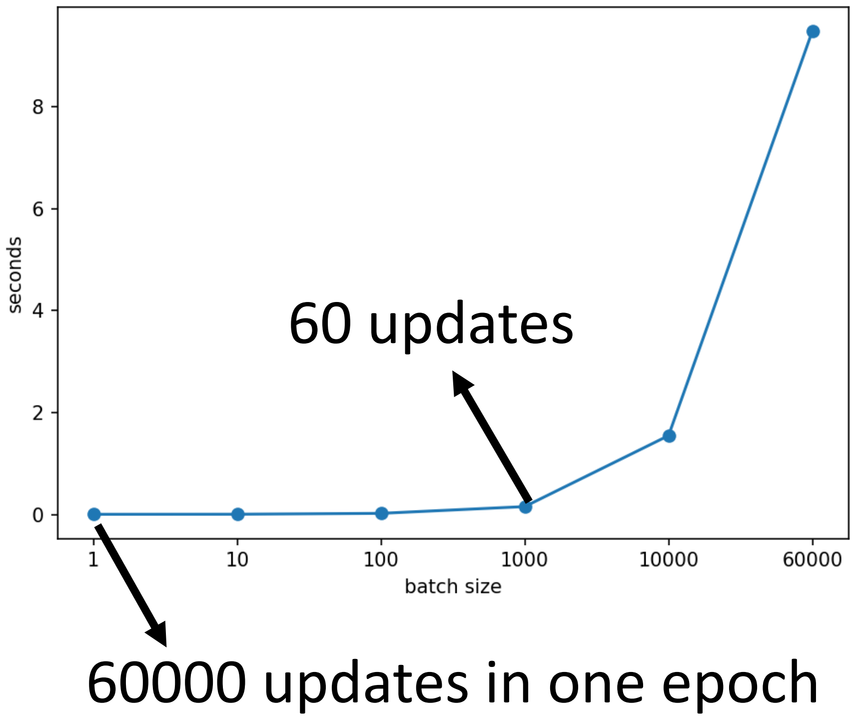
Tesla V100 GPU



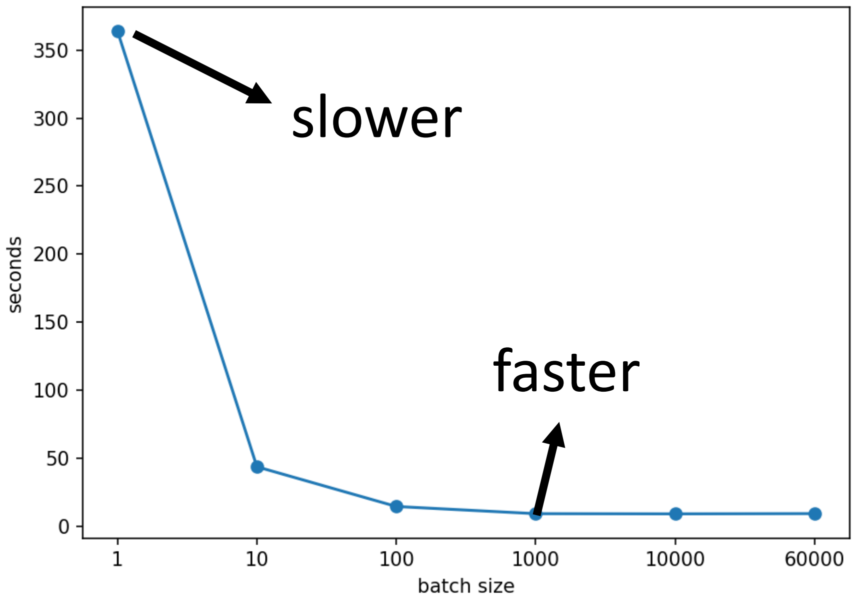
Small Batch v.s. Large Batch

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

Time for one **update**



Time for one **epoch**

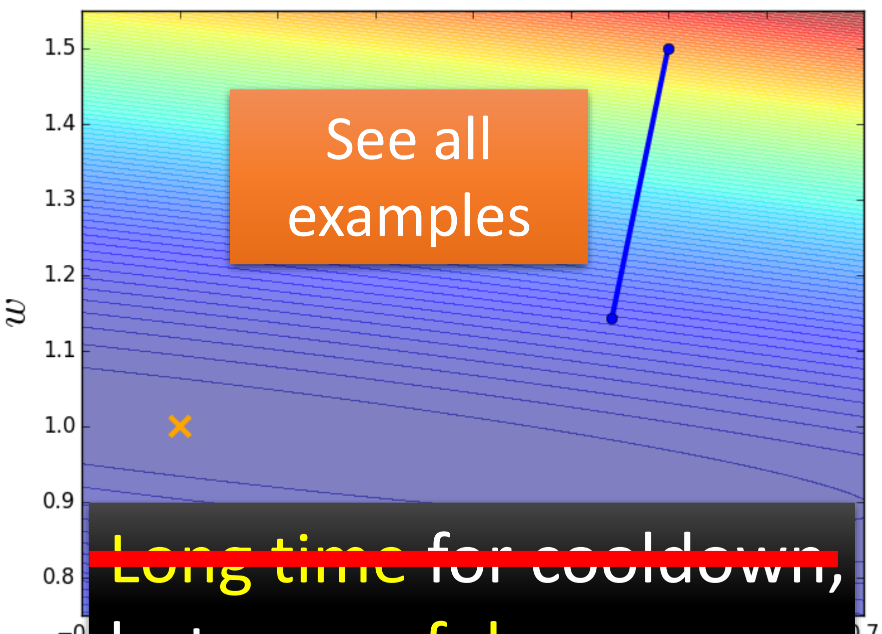


Small Batch v.s. Large Batch

Consider 20 examples (N=20)

Batch size = N (Full Batch)

Update after seeing all the 20 examples

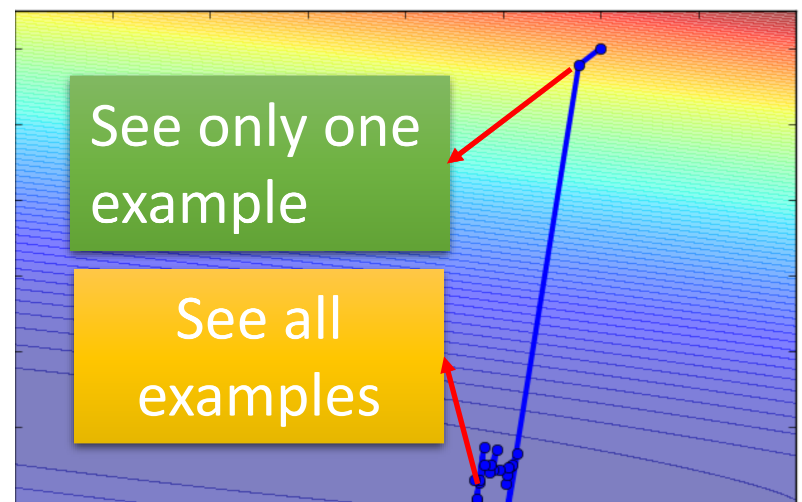


~~Long time for cooldown,~~ but powerful

Batch size = 1

Update for each example

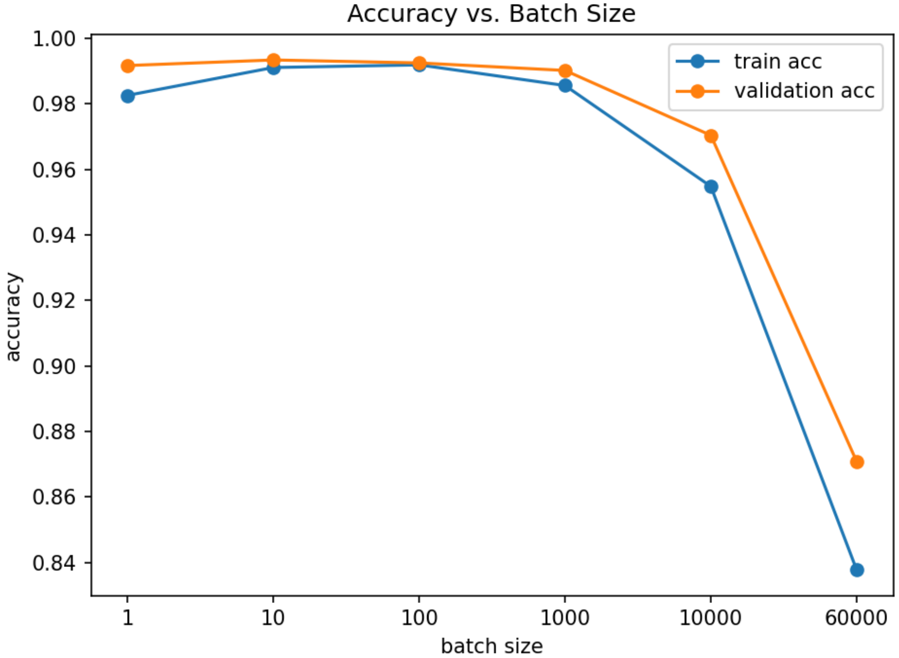
Update 20 times in an epoch



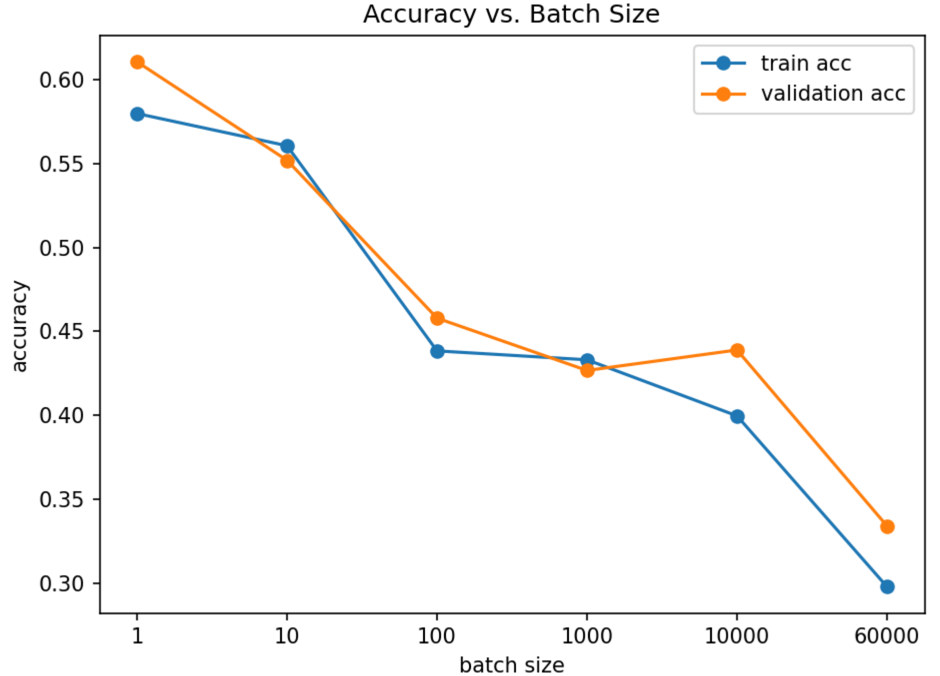
~~Short time for cooldown,~~ but noisy

Small Batch v.s. Large Batch

MNIST



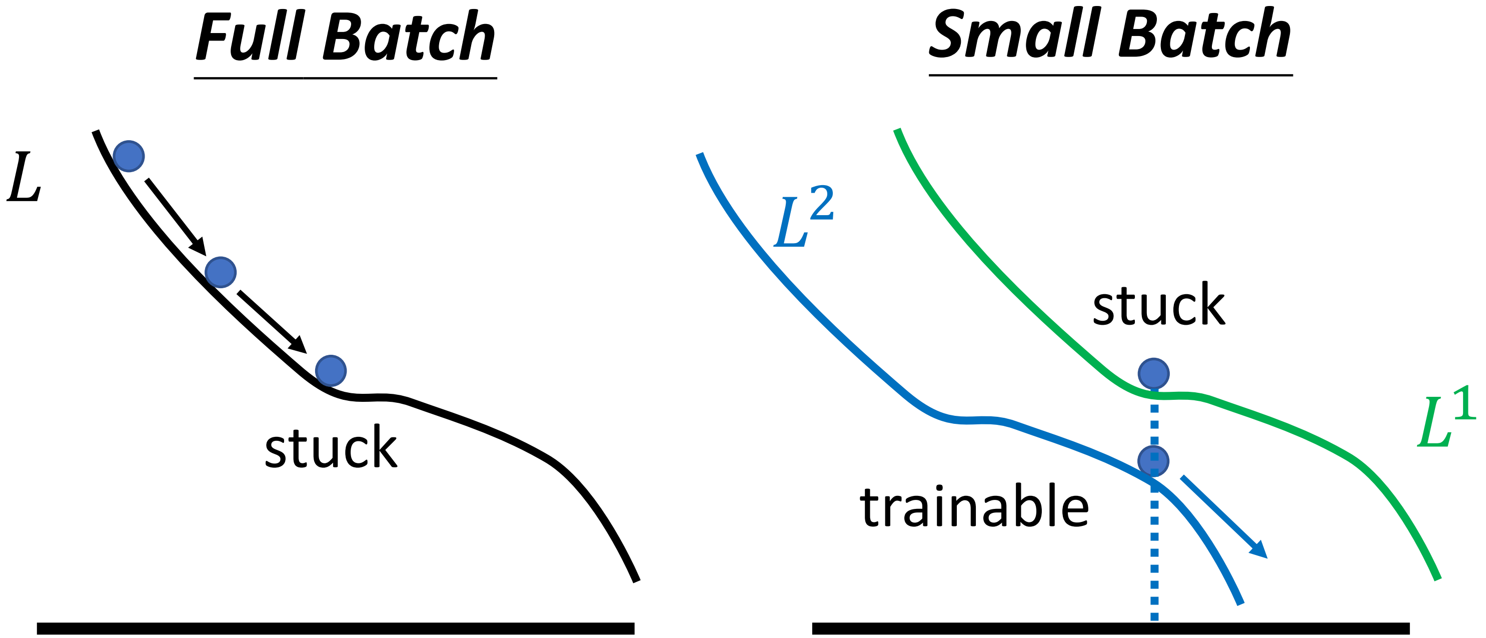
CIFAR-10



- ➢ 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



Small Batch v.s. Large Batch

- Small batch is better on testing data?

SB = 256

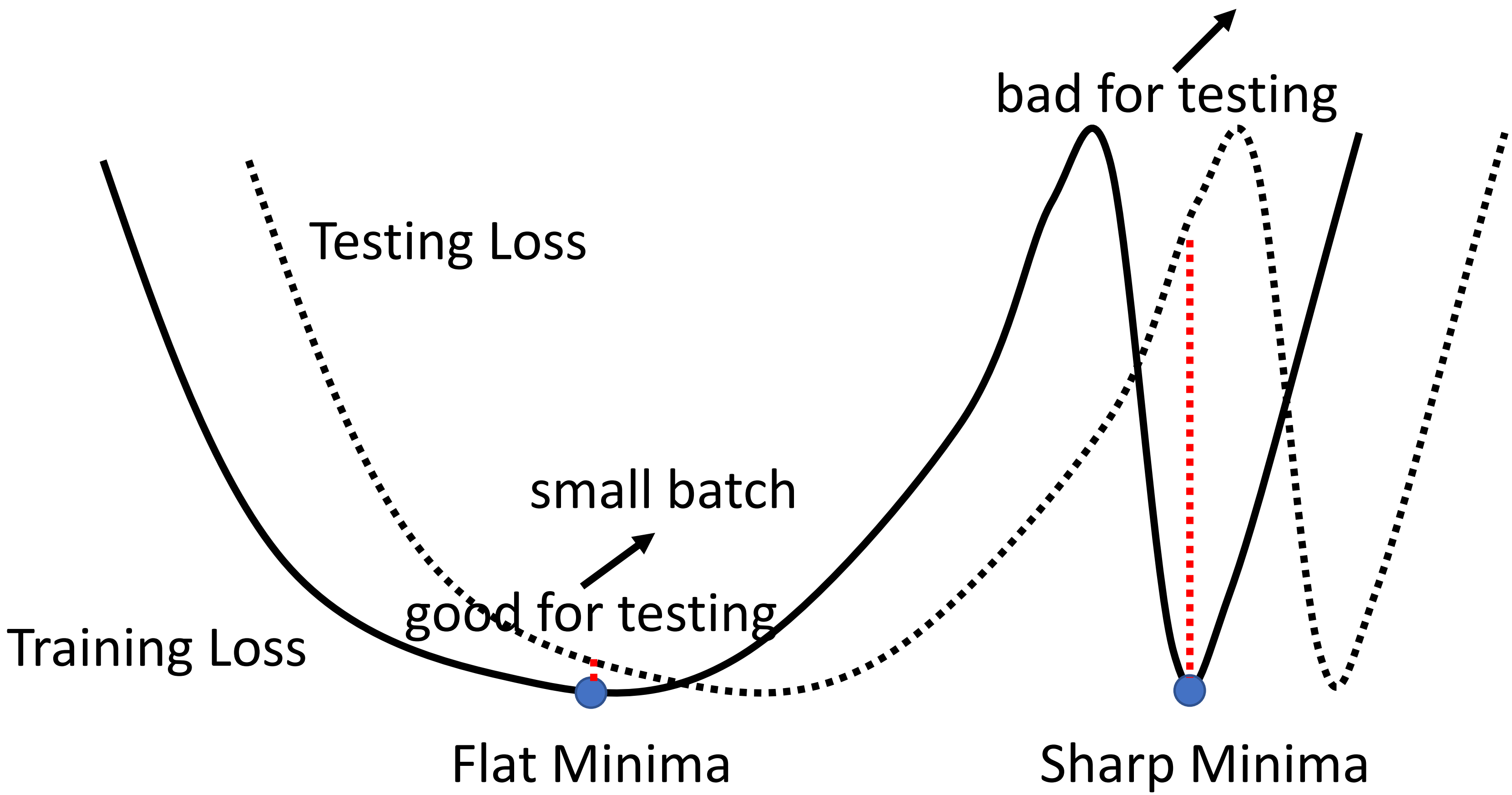
LB = 0.1 x data set

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	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

Small Batch v.s. Large Batch

- Small batch is better on testing data?



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
Gradient	Noisy	Stable
Optimization	Better	Worse
Generalization	Better	Worse

Batch size is a hyperparameter you have to decide.

Have both fish and bear's paws?

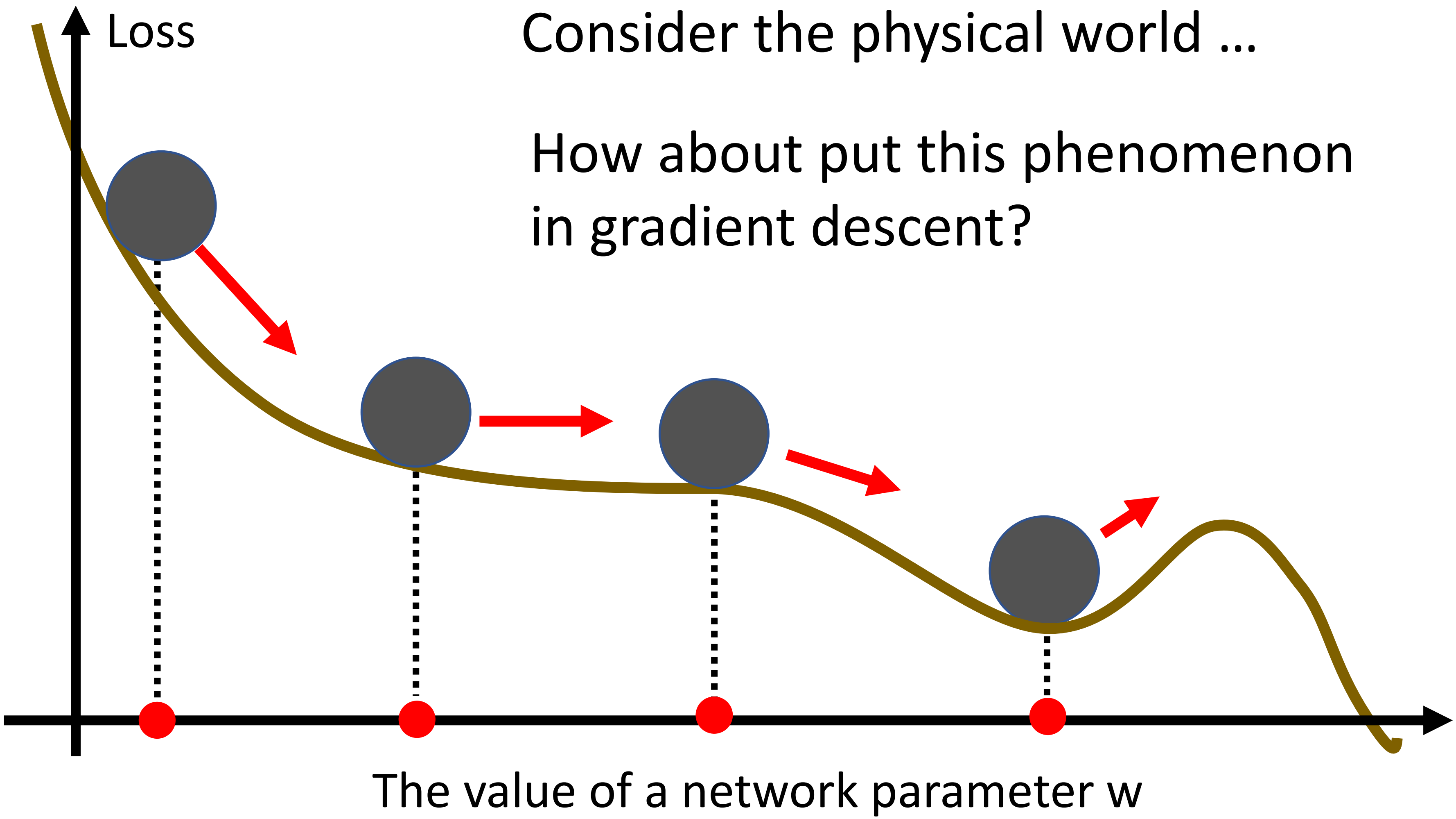
- Large Batch Optimization for Deep Learning: Training BERT in 76 minutes (https://arxiv.org/abs/1904.00962)
- Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes (https://arxiv.org/abs/1711.04325)
- Stochastic Weight Averaging in Parallel: Large-Batch Training That Generalizes Well (https://arxiv.org/abs/2001.02312)
- Large Batch Training of Convolutional Networks (https://arxiv.org/abs/1708.03888)
- Accurate, large minibatch sgd: Training imagenet in 1 hour (https://arxiv.org/abs/1706.02677)

Momentum

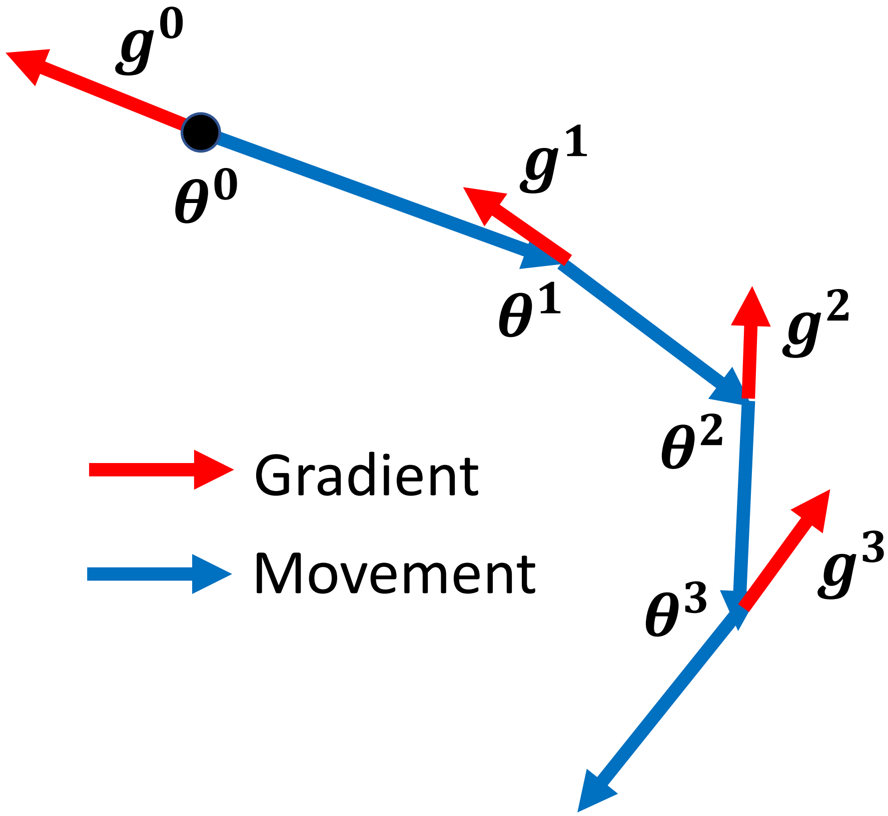
Small Gradient …

Consider the physical world …

How about put this phenomenon in gradient descent?



(Vanilla) Gradient Descent



Starting at $\boldsymbol{\theta}^0$

Compute gradient $\boldsymbol{g}^0$

Move to $\boldsymbol{\theta}^1 = \boldsymbol{\theta}^0 - \eta \boldsymbol{g}^0$

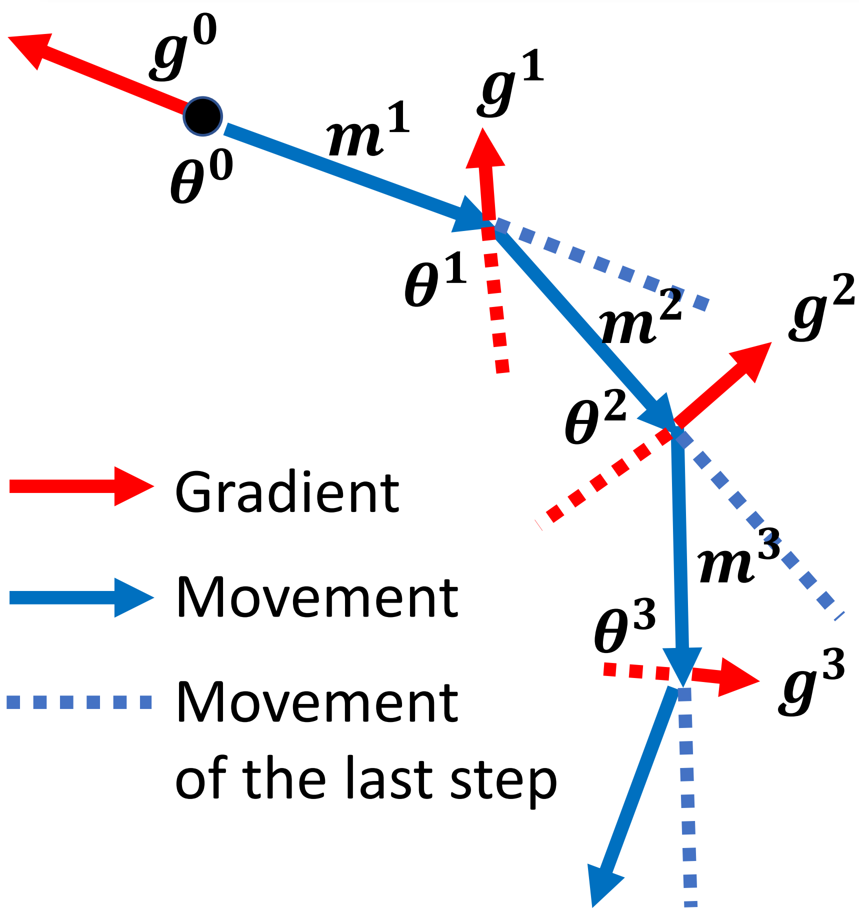
Compute gradient $\boldsymbol{g}^1$

Move to $\boldsymbol{\theta}^2 = \boldsymbol{\theta}^1 - \eta \boldsymbol{g}^1$

⋮

Gradient Descent + Momentum

Movement: **movement of last step** minus **gradient** at present



Starting at $\boldsymbol{\theta}^0$

Movement $\boldsymbol{m}^0 = \boldsymbol{0}$

Compute gradient $\boldsymbol{g}^0$

Movement $\boldsymbol{m}^1 = \lambda\boldsymbol{m}^0 - \eta\boldsymbol{g}^0$

Move to $\boldsymbol{\theta}^1 = \boldsymbol{\theta}^0 + \boldsymbol{m}^1$

Compute gradient $\boldsymbol{g}^1$

Movement $\boldsymbol{m}^2 = \lambda\boldsymbol{m}^1 - \eta\boldsymbol{g}^1$

Move to $\boldsymbol{\theta}^2 = \boldsymbol{\theta}^1 + \boldsymbol{m}^2$

Movement not just based on gradient, but previous movement.

Gradient Descent + Momentum

Movement: **movement of last step** minus **gradient** at present

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

$$\boldsymbol{m}^0 = \boldsymbol{0}$$

$$\boldsymbol{m}^1 = -\eta \boldsymbol{g}^0$$

$$\boldsymbol{m}^2 = -\lambda\eta \boldsymbol{g}^0 - \eta \boldsymbol{g}^1$$

$$\vdots$$

Starting at $\boldsymbol{\theta}^0$

Movement $\boldsymbol{m}^0 = \boldsymbol{0}$

Compute gradient $\boldsymbol{g}^0$

Movement $\boldsymbol{m}^1 = \lambda \boldsymbol{m}^0 - \eta \boldsymbol{g}^0$

Move to $\boldsymbol{\theta}^1 = \boldsymbol{\theta}^0 + \boldsymbol{m}^1$

Compute gradient $\boldsymbol{g}^1$

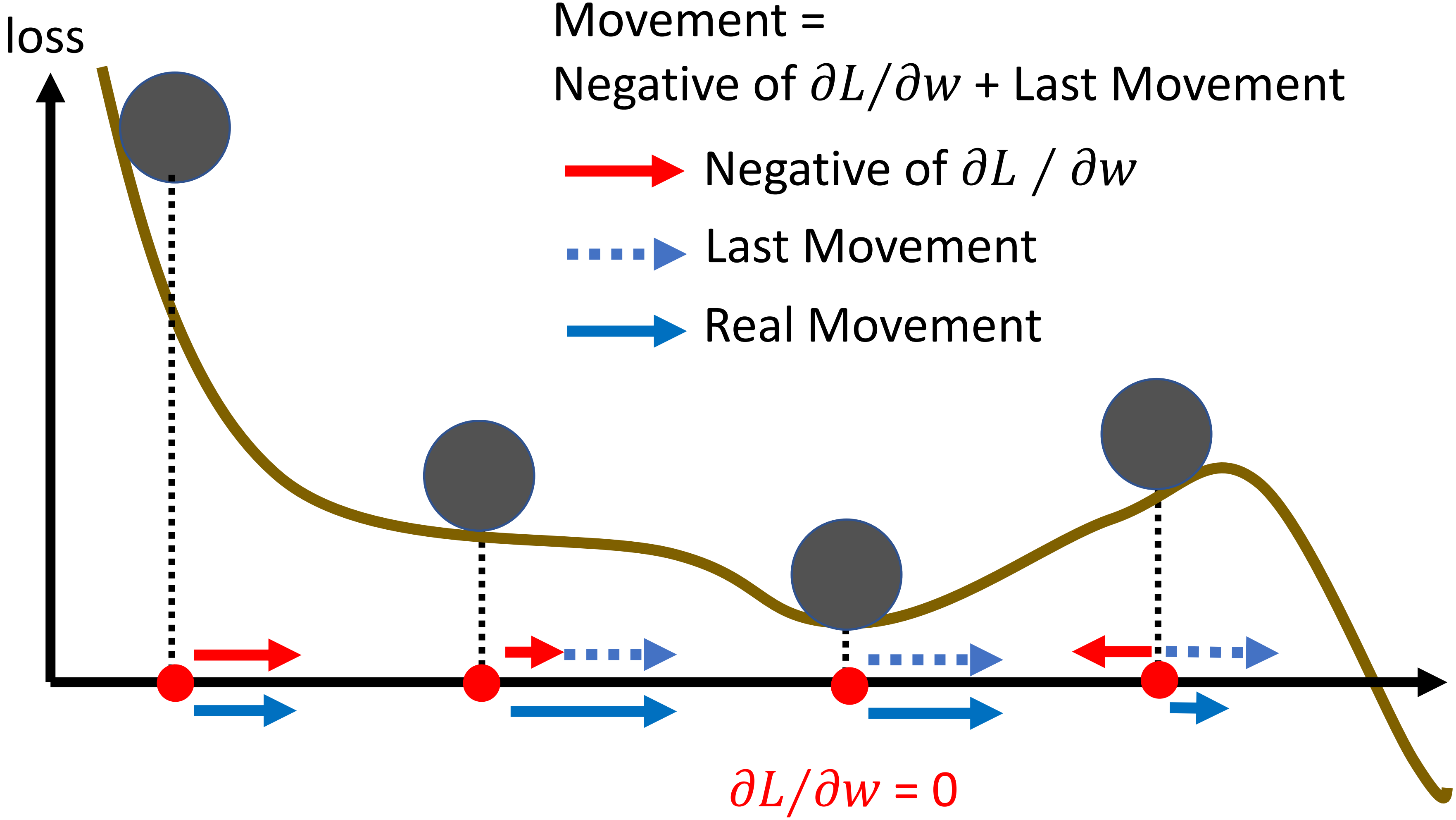
Movement $\boldsymbol{m}^2 = \lambda \boldsymbol{m}^1 - \eta \boldsymbol{g}^1$

Move to $\boldsymbol{\theta}^2 = \boldsymbol{\theta}^1 + \boldsymbol{m}^2$

Movement not just based on gradient, but previous movement.

Gradient Descent + Momentum

Movement =
Negative of $\partial L/\partial w$ + Last Movement



Concluding Remarks

- Critical points have zero gradients.
- Critical points can be either saddle points or local minima.
  - Can be determined by the Hessian matrix.
  - It is 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.

Acknowledgement

- 感謝作業二助教團隊(陳宣叡、施貽仁、孟妍李威緒)幫忙跑實驗以及蒐集資料