類神經網路訓練不起來怎麼辦 (二)： 批次 (batch) 與動量 (momentum)

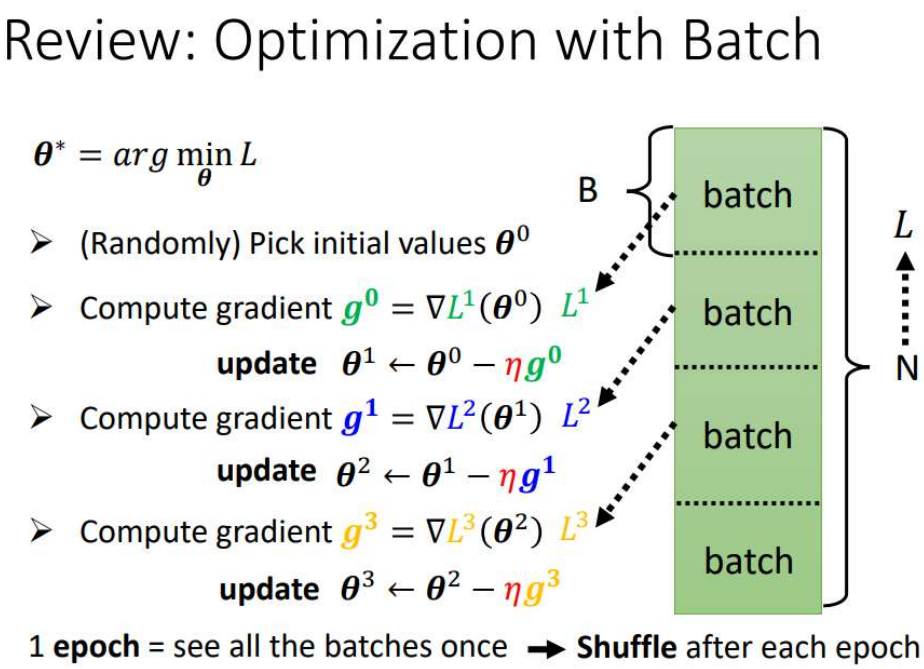
Create at 2022/06/08

- 上課資源：
  1. 類神經網路訓練不起來怎麼辦 (二)： 批次 (batch) 與動量 (momentum) (https://www.youtube.com/watch?v=zzbr1h9sF54)

批次 (batch) 與動量 (momentum) 訓練技巧

Batch

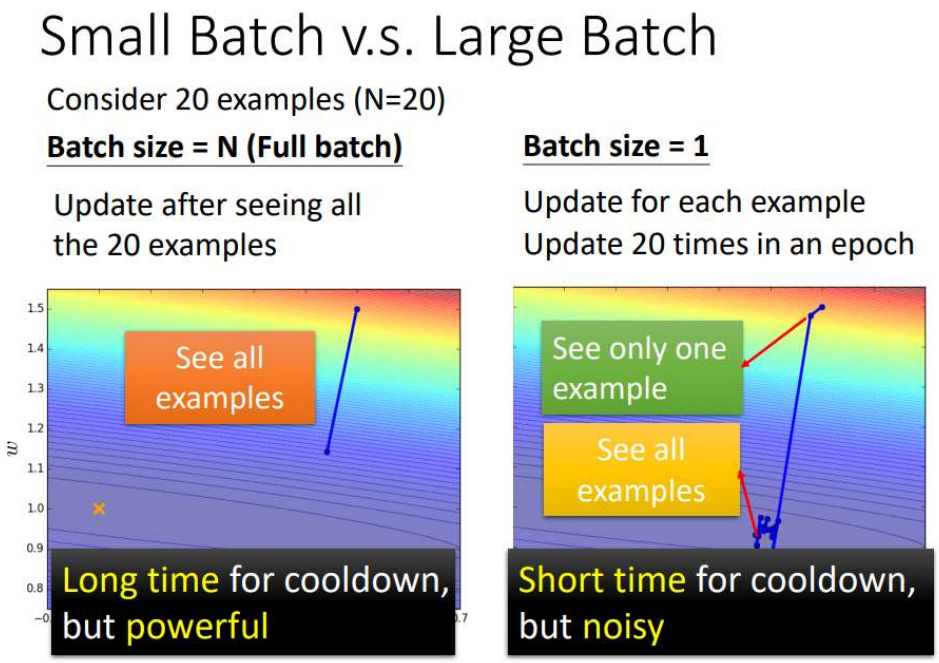
為甚麼要用 Batch ?



👍 **之前筆記** (https://hackmd.io/Vw2JGubVTnSWwWiXDIfMPw?view#Optimization-of-New-Model)

- 不是真的對所有 Data 算出來的 Loss 做微分，是把所有的 Data 分成一個一個的 Batch (Mini batch)
- 只拿一個 Batch 的資料出來算
- **所有的 Batch 都看過一遍，叫做一個 Epoch**
- 在把所有資料分成一個一個 Batch 時，會做一件事情叫做 shuffle

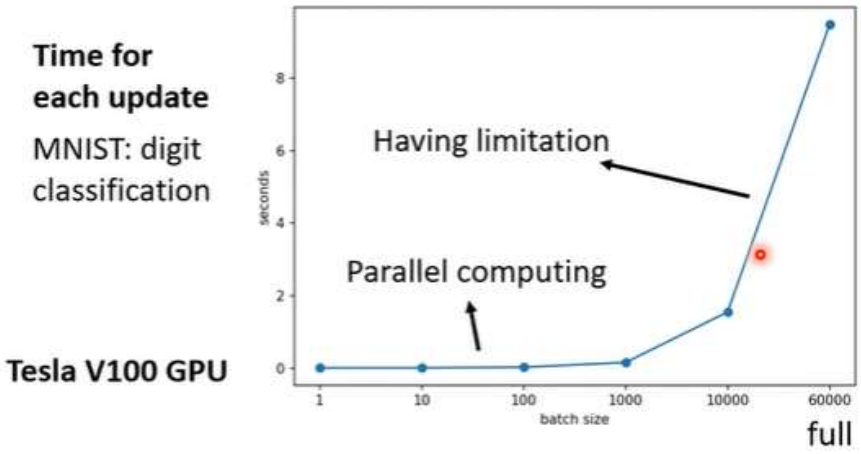
- **shuffle 有很多不同的做法**
  - 常見：在每一個 epoch 開始之前會分一次 Batch，每一個 epoch 的 Batch 都不一樣



- Full batch：蓄力時間長，威力比較大
- Batch size = 1：技能冷卻時間短，比較不準 (亂槍打鳥型)
- **但如果考慮平行運算的話，Full batch 不一定時間比較長**

Small Batch v.s. Large Batch

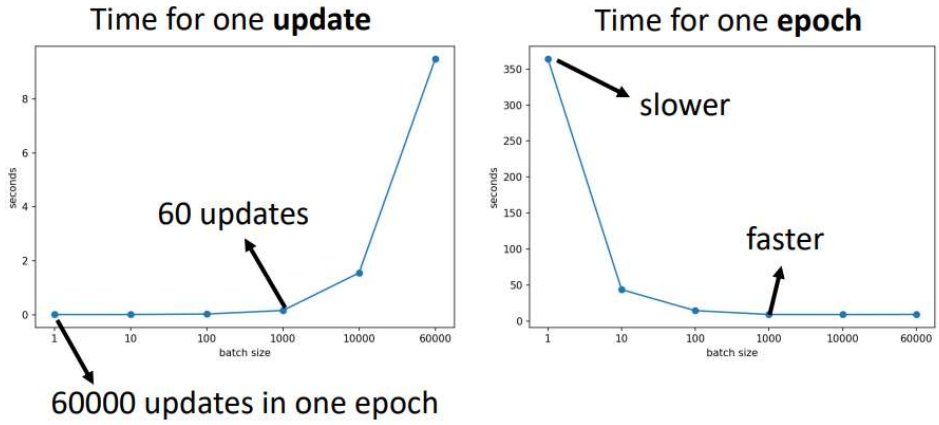
- Larger batch size does **not** require longer time to compute gradient (unless batch size is too large)



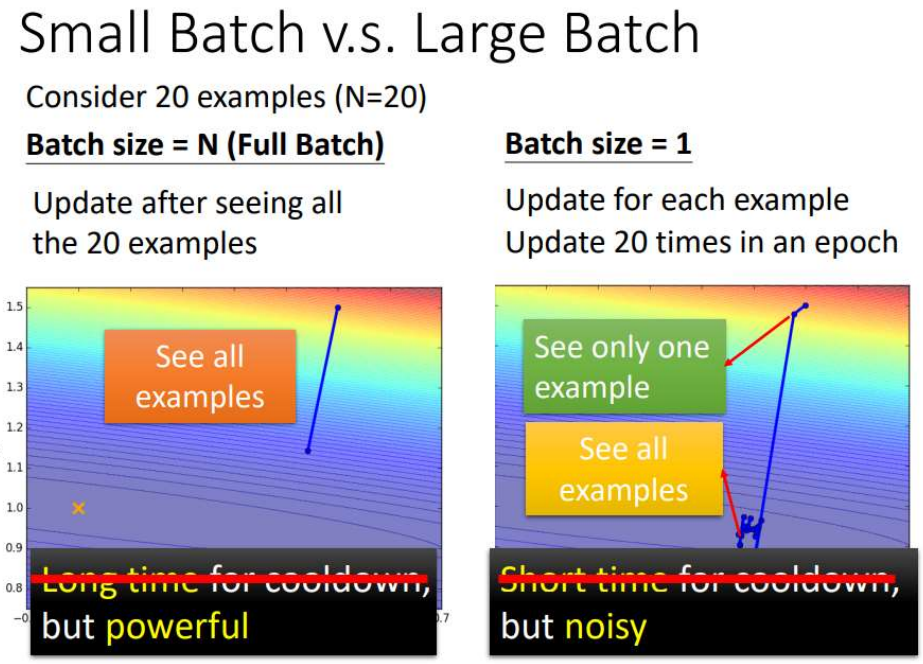
- 比較大的 batch size，要算 loss 進而算 gradient 所需要的時間，不一定比小的 batch size 要花的時間長
- 給機器一個 batch，計算出 gradient 進而 update 參數要花多少時間？
  - 有 GPU 可以做平行運算，所以 1000 筆資料是平行處理的 (但 GPU 的計算能力還是有侷限，當 Batch size 真的很大時，GPU 跑完一個 Batch 計算出 gradient 所花費的時間還是會隨著 Batch size 增加而增長)

Small Batch v.s. Large Batch

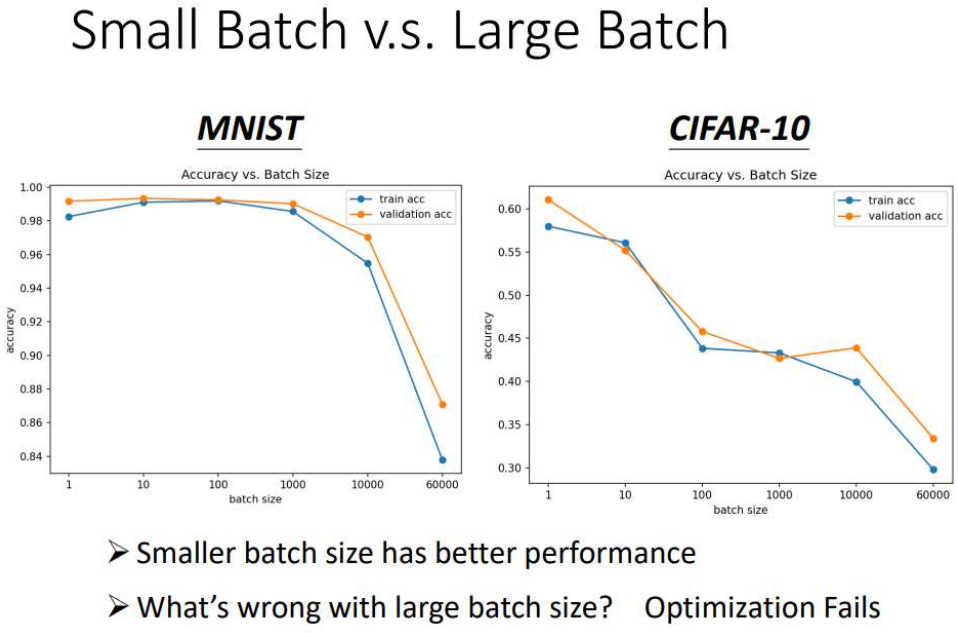
- Smaller batch requires longer time for one epoch (longer time for seeing all data once)



- 當 batch size 小的時候，跑完一個 epoch 花的時間比大的 batch size 還多
- 左圖：update 一次參數所需要的時間
- 右圖：跑完一個完整 Epoch 所需的時間
- 沒有平行運算的時候，覺得大的 batch 跑得比較慢
- 有平行運算的時候，大的 batch 反而花的時間比較少



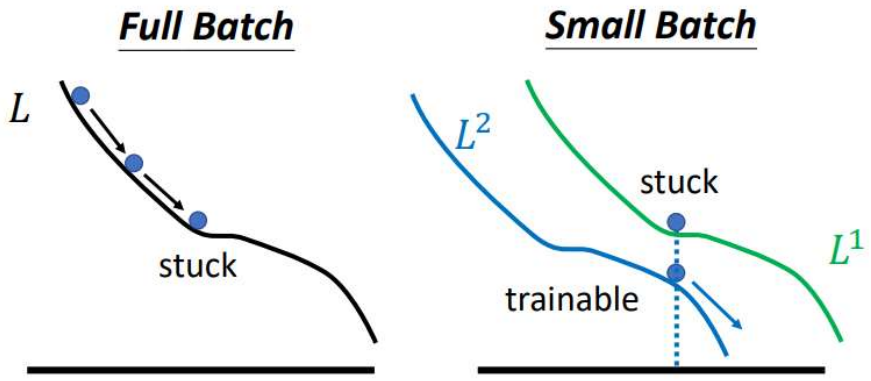
直接用技能冷卻時間長短不是一個精確的描述
大的 batch 並沒有比較吃虧



- 橫軸 : batch size
- 縱軸 : accuracy
- 看 validation set 上的結果，batch size 越大 accuracy 結果越差
- 不是 overfitting，因為看 training set 的結果，也是 batch size 越大 training 的結果也是越差，是 optimization 的問題

Small Batch v.s. Large Batch

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



- 假設是 Full Batch，今天在 update 參數時是沿著一個 loss function 來 update 參數，當 update 參數走到一個 local minima / saddle point 就停下來了，用 gradient descent 的方法就沒辦法再更新參數
- 假如是 small Batch，每次挑一個 batch 出來算它的 loss，等於是每次 update 參數的時候，用的 loss function 都是有差異的
  - 選到第一個 batch 時，是用 L1 來算 Gradient
  - 選到第二個 batch 時，是用 L2 來算 Gradient
  - 假設用 L1 算 gradient 時發現 gradient 是 0，但 L2 的 function 跟 L1 不一樣，所以 L1 卡住 L2 不一定會卡住
  - 所以 L1 卡住沒關係，換下一個 batch 來，L2 再算 gradient，還是有辦法 training model 讓 loss 變小

On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima
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

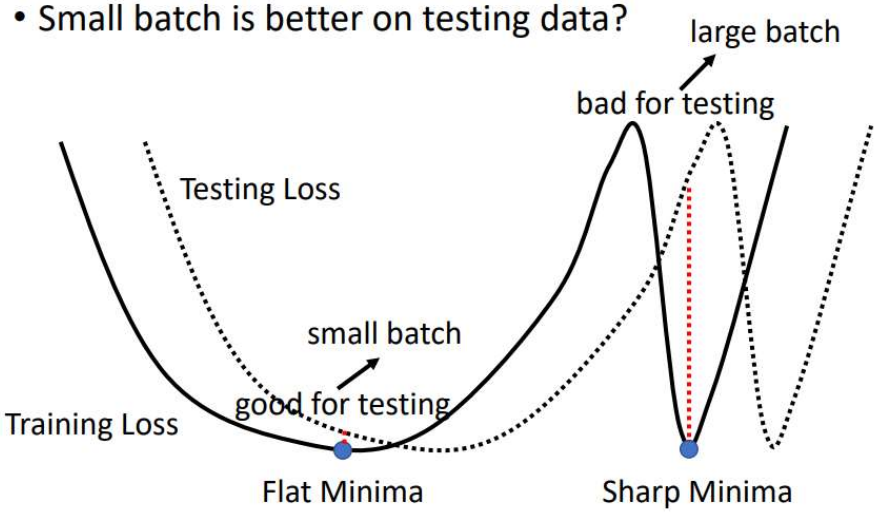
	Training Accuracy		Testing Accuracy	
Name	SB	LB	SB	LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
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	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

- 小的 batch 也對 **testing** 有幫助
- 在左邊 training 結果差不多的時候，在右邊 testing 時，大的 batch 差代表 overfitting

On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima
https://arxiv.org/abs/1609.04836

Small Batch v.s. Large Batch

- Small batch is better on testing data?



- 如果 local minima 在峽谷裡面，是壞的 local minima
- 如果 local minima 在盆地上，是好的 local minima
- 大的 batch size 會傾向於走到峽谷裡
- 小的 batch size 會傾向於走到盆地

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

Batch size is a hyperparameter you have to decide.

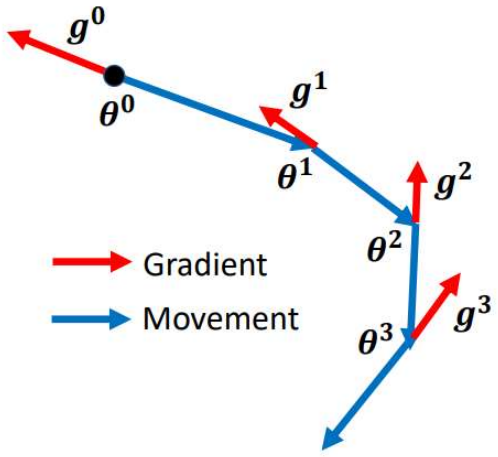
比較小的 batch 跟大的 batch

Momentum

有可能可以對抗 saddle point 或 local minima 的技術

一般的 Gradient Descent

(Vanilla) Gradient Descent



Starting at θ^0

Compute gradient g^0

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

Compute gradient g^1

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

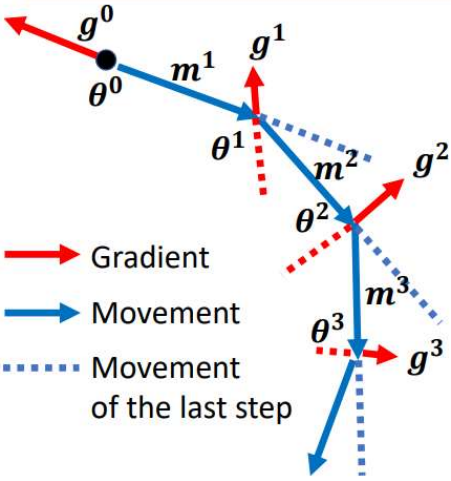
⋮

一般的 Gradient Descent

一般的 Gradient Descent + Momentum

Gradient Descent + Momentum

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



Starting at θ^0

Movement $m^0 = 0$

Compute gradient g^0

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

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

Compute gradient g^1

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

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

Movement not just based on gradient, but previous movement.

- 每一次在移動參數時，不是只往 gradient descent 的反方向來移動參數
- **往 gradient descent 的反方向加上前一步移動的方向，去移動參數**

Gradient Descent + Momentum

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

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

$m^0 = 0$

$m^1 = -\eta g^0$

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

⋮

Starting at θ^0

Movement $m^0 = 0$

Compute gradient g^0

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

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

Compute gradient g^1

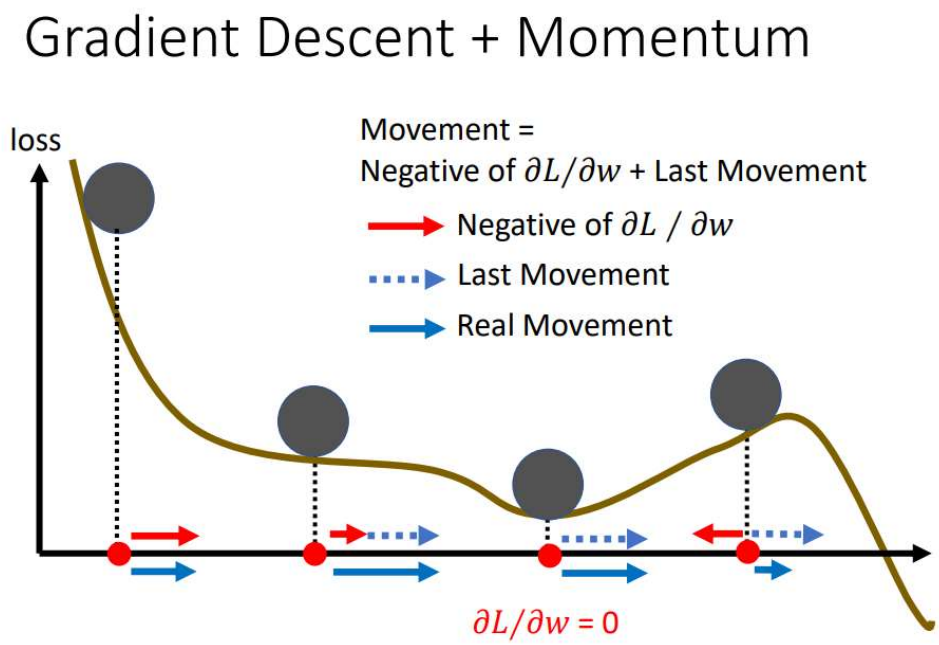
Movement $m^2 = \lambda m^1 - \eta g^1$

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

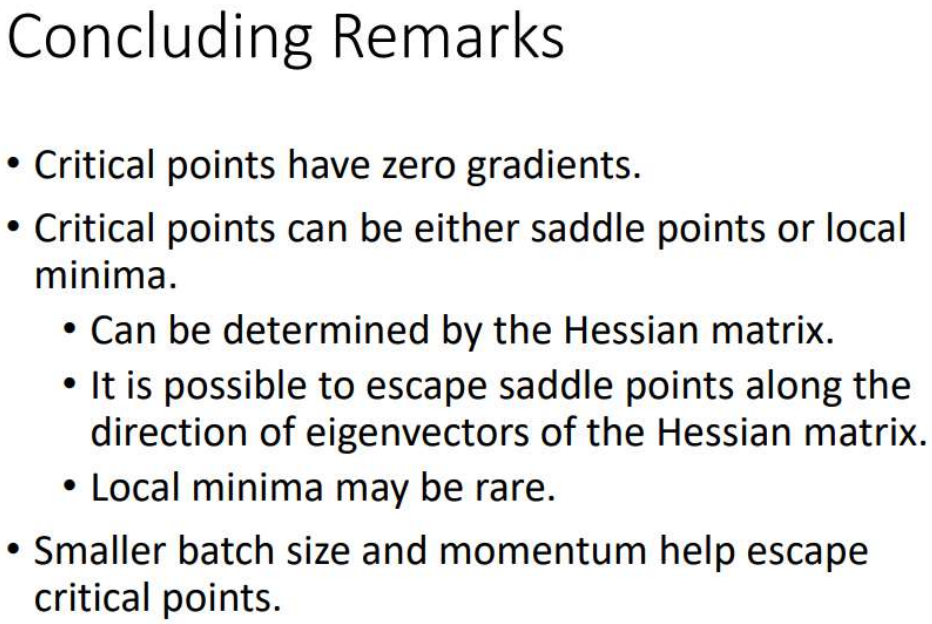
Movement not just based on gradient, but previous movement.

加上 Momentum

我們 update 的方向，不是只考慮現在的 gradient，而是考慮過去所有 gradient 的總和



- 用 Momentum 方法，就算 gradient 是 0，還是可以依照前一步的方向繼續走
- 這個是 Momentum 的好處



總結

課程網頁 (https://speech.ee.ntu.edu.tw/~hylee/ml/2022-spring.php)

tags: **2022 李宏毅_機器學習**