深度學習基礎概論

0318

目錄

- 名詞和理論解釋與理解

- Code : simple NN model

機器學習和傳統程式的不同
以分類問題為例

傳統程式設計

建立分類器

1

定義一組分類規則

2

將這些規則編寫入電腦

3

列舉範例，然後讓程式利用規則來分類

機器學習

建立分類器

1

將範例附上正確分類方式，並展示給模型

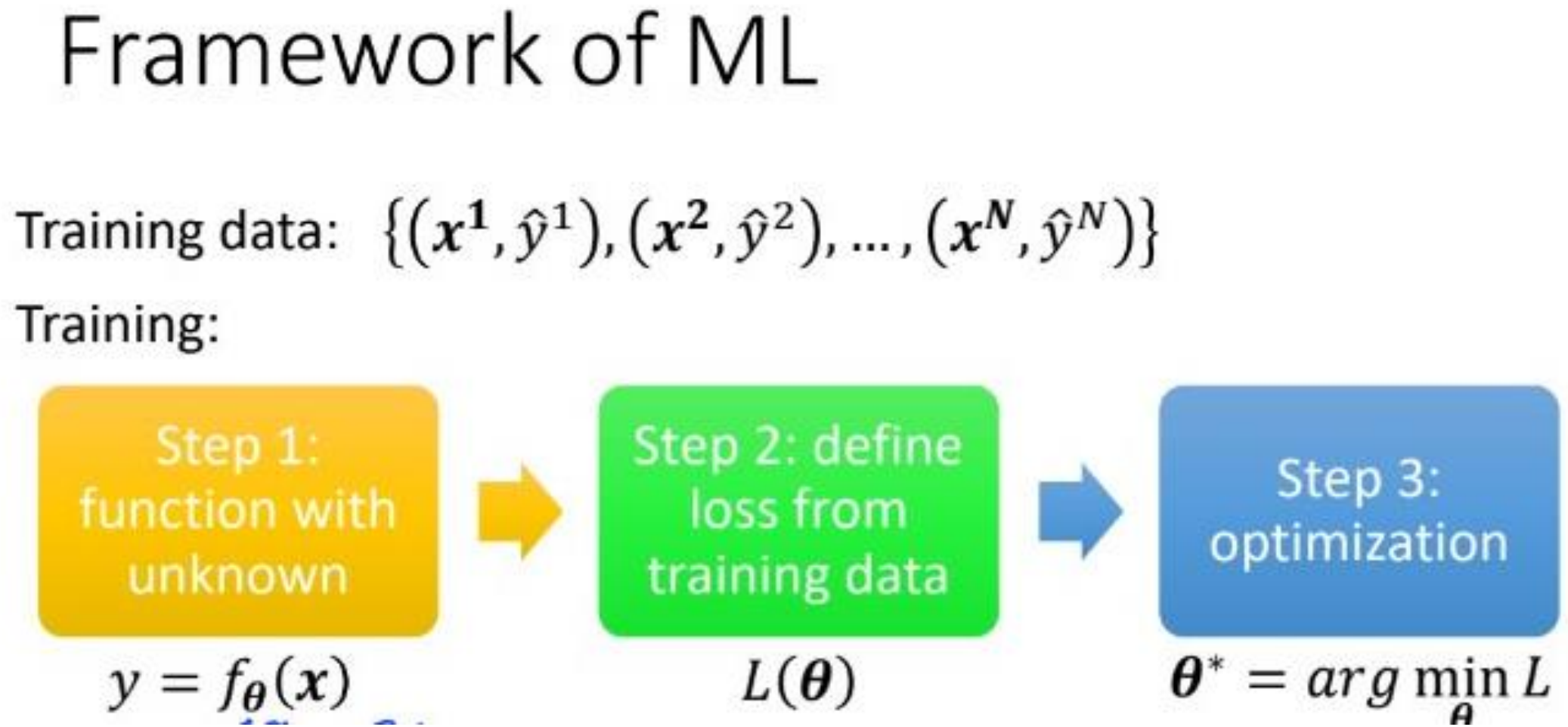
2

模型會進行猜測，我們告知它猜測是否正確

3

模型可以在訓練中學會如何正確分類。系統自行學會規則

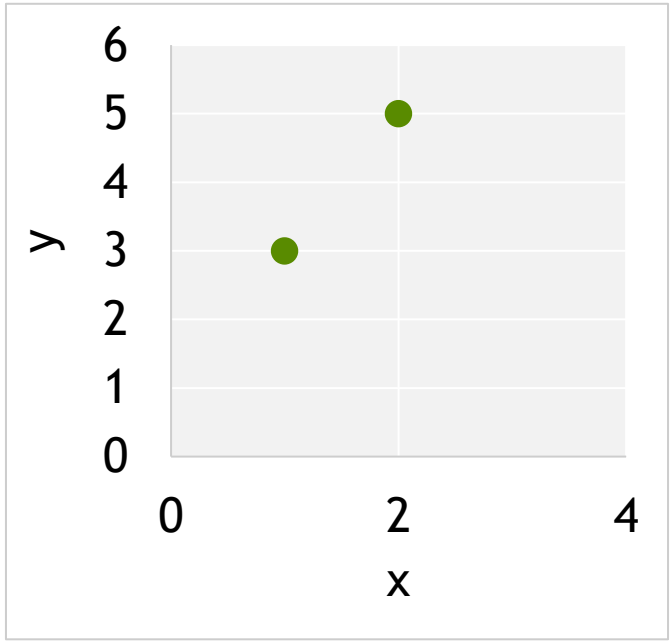
Machine Learning 步驟

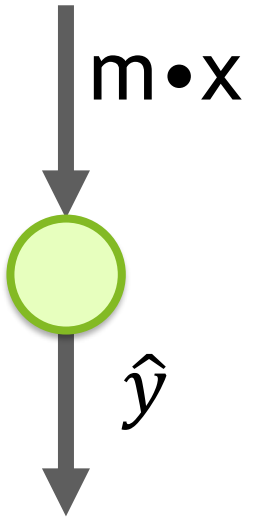


簡化的模型

$$y = mx + b$$

x	y
1	3
2	5





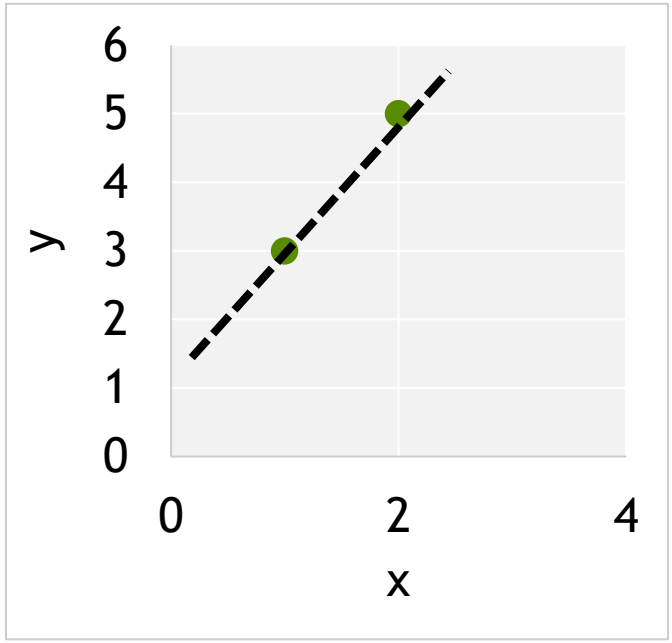
$m = ?$

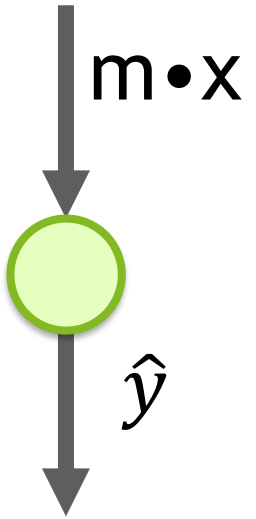
$b = ?$

簡化的模型

$$y = mx + b$$

x	y
1	3
2	5





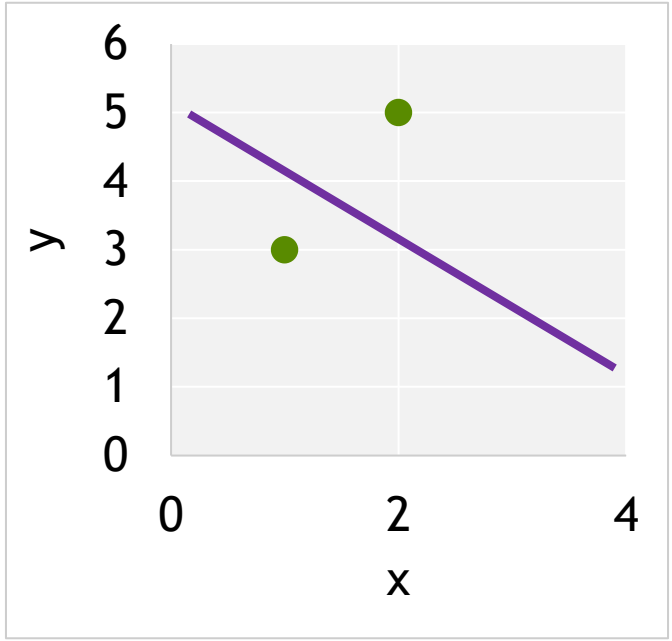
$m = ?$

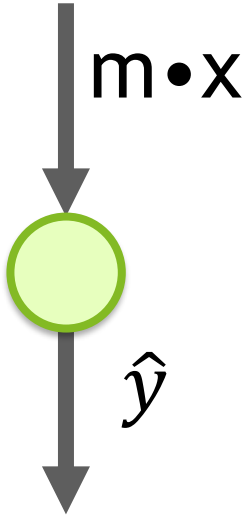
$b = ?$

簡化的模型

$$y = mx + b$$

x	y	$\hat{y}$
1	3	4
2	5	3





Start Random

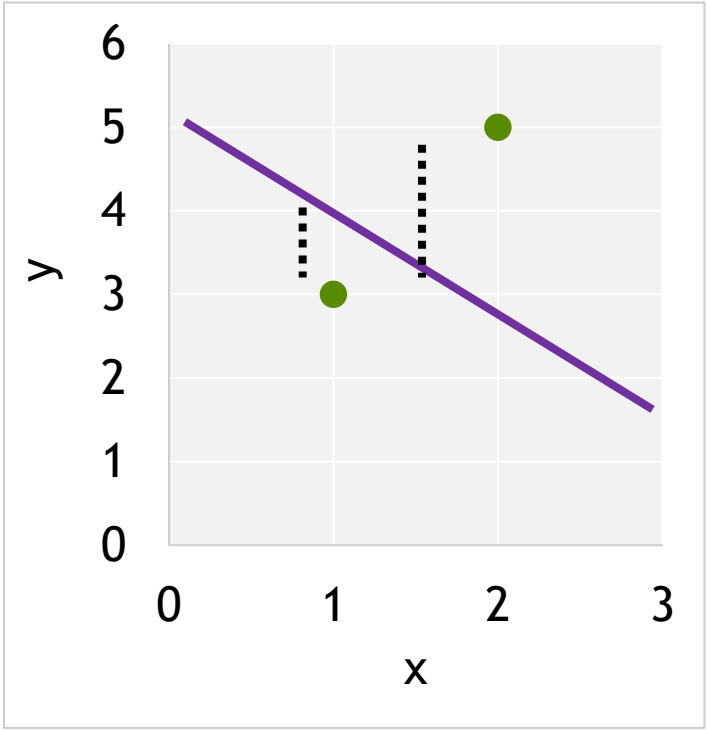
$$m = -1$$

$$b = 5$$

簡化的模型

$$y = mx + b$$

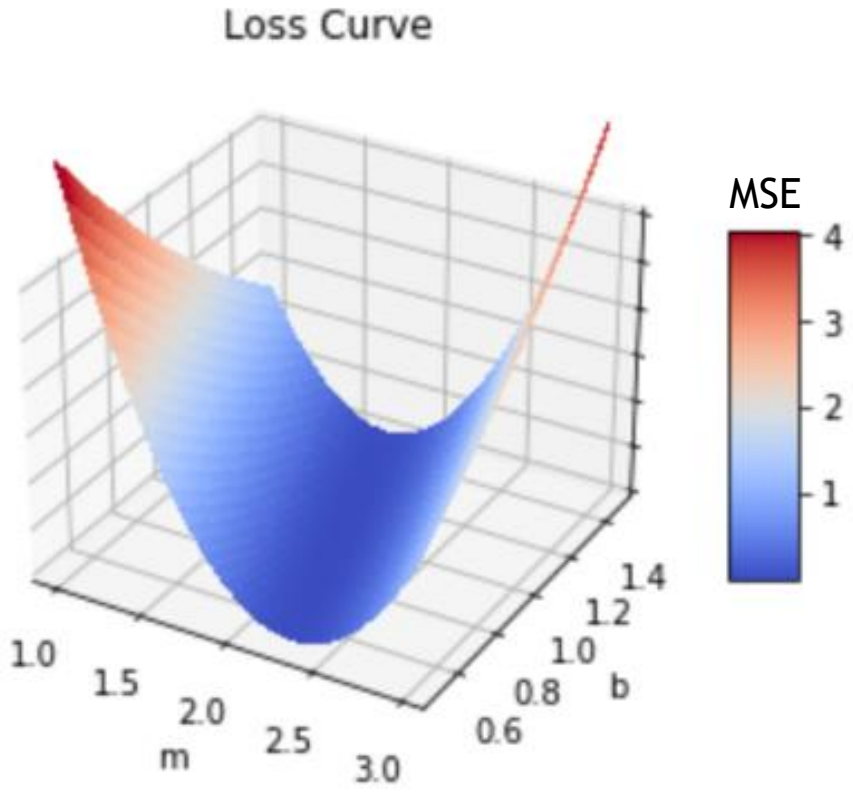
x	y	$\hat{y}$	err^2
1	3	4	1
2	5	3	4
		MSE =	2.5
		RMSE =	1.6

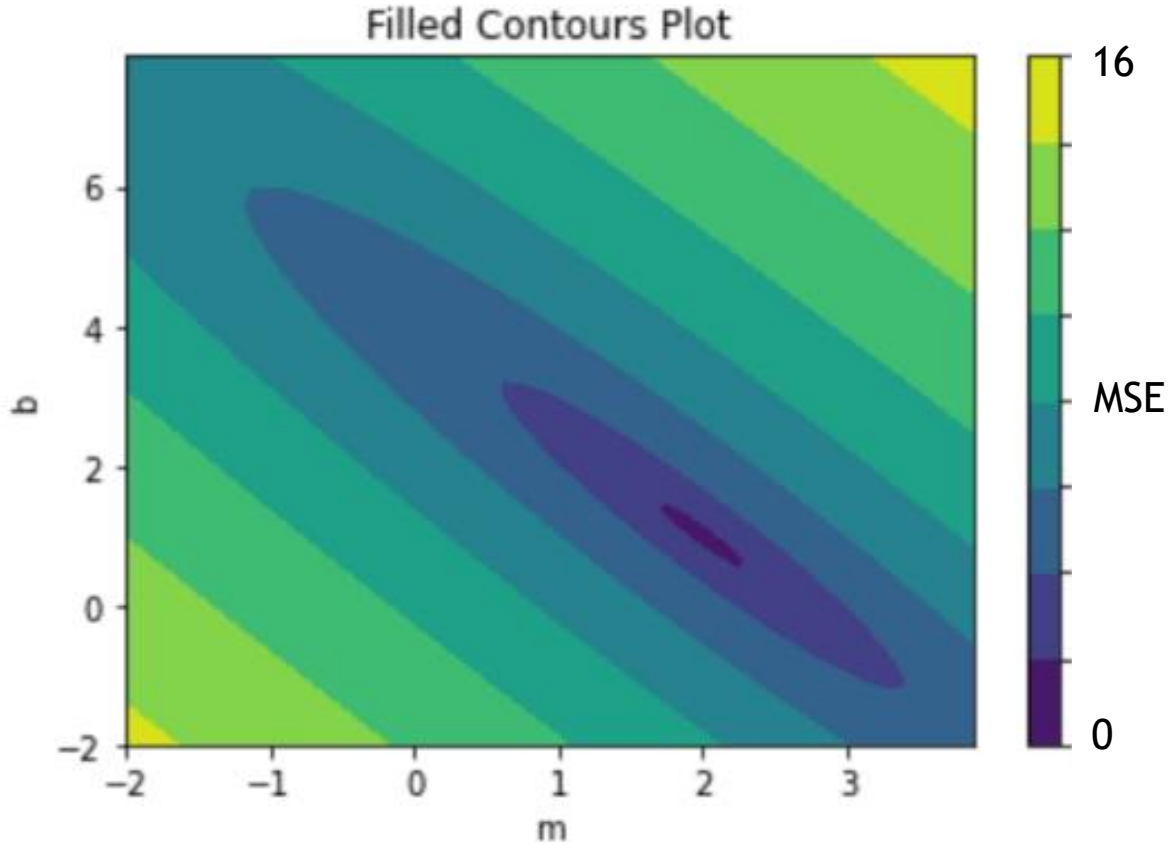


$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$$

$$RMSE = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(y - \hat{y})^2}$$

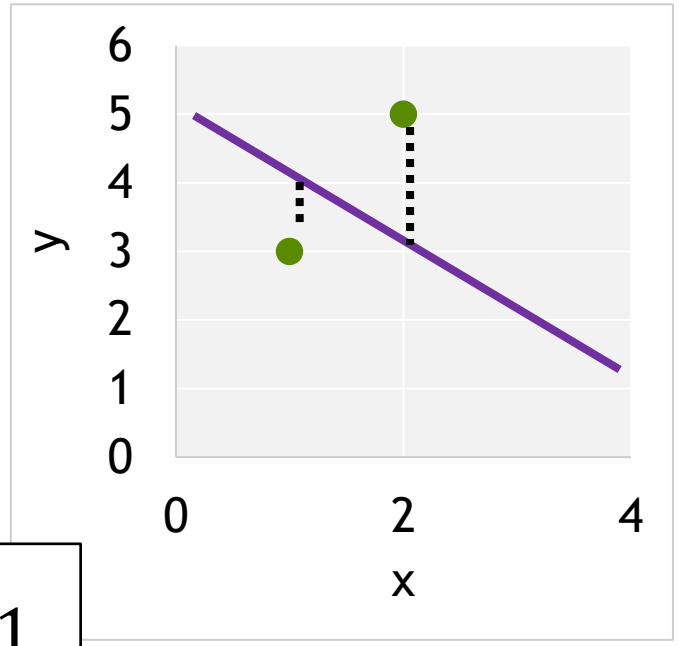
損失曲線





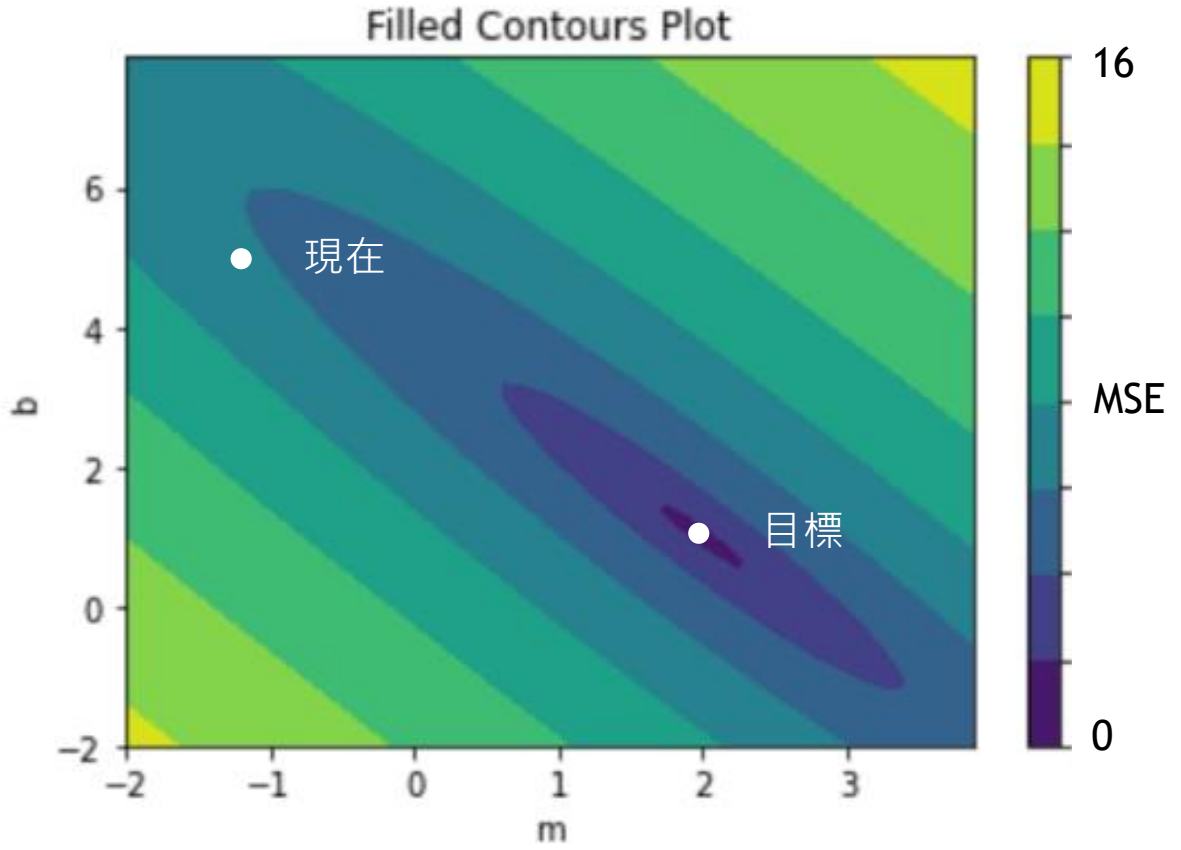
Error surface

損失曲線

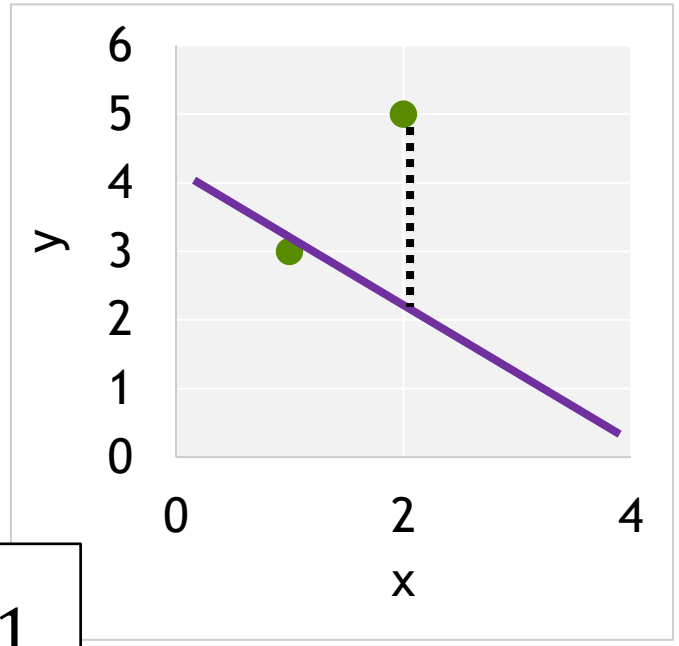


$m = -1$

$b = 5$

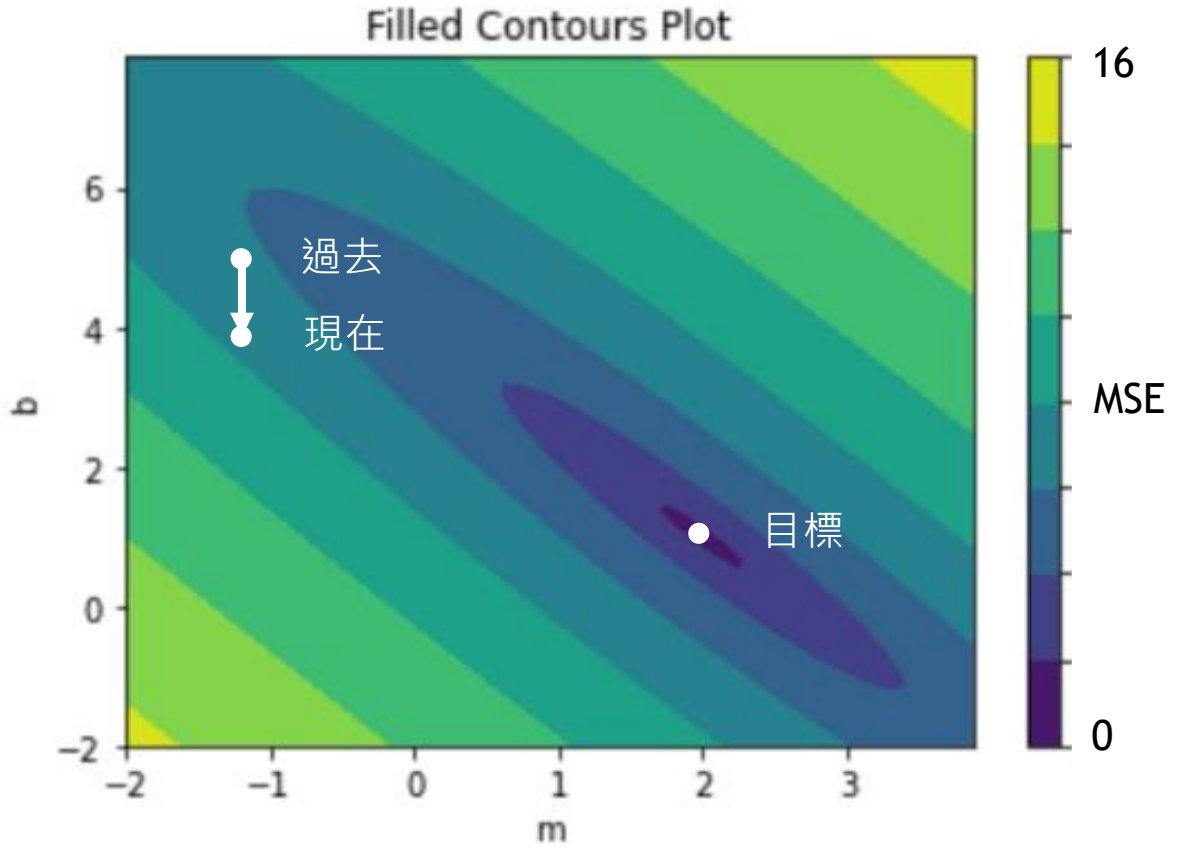


損失曲線

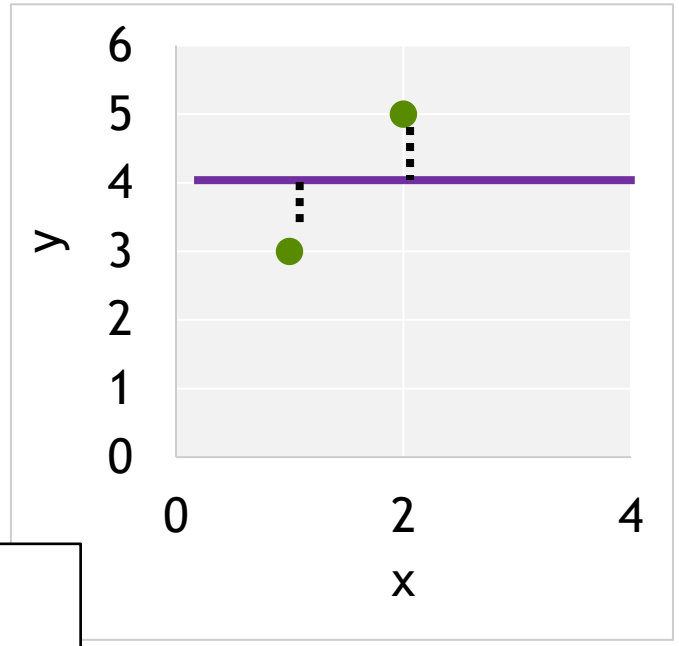


$m = -1$

$b = 4$

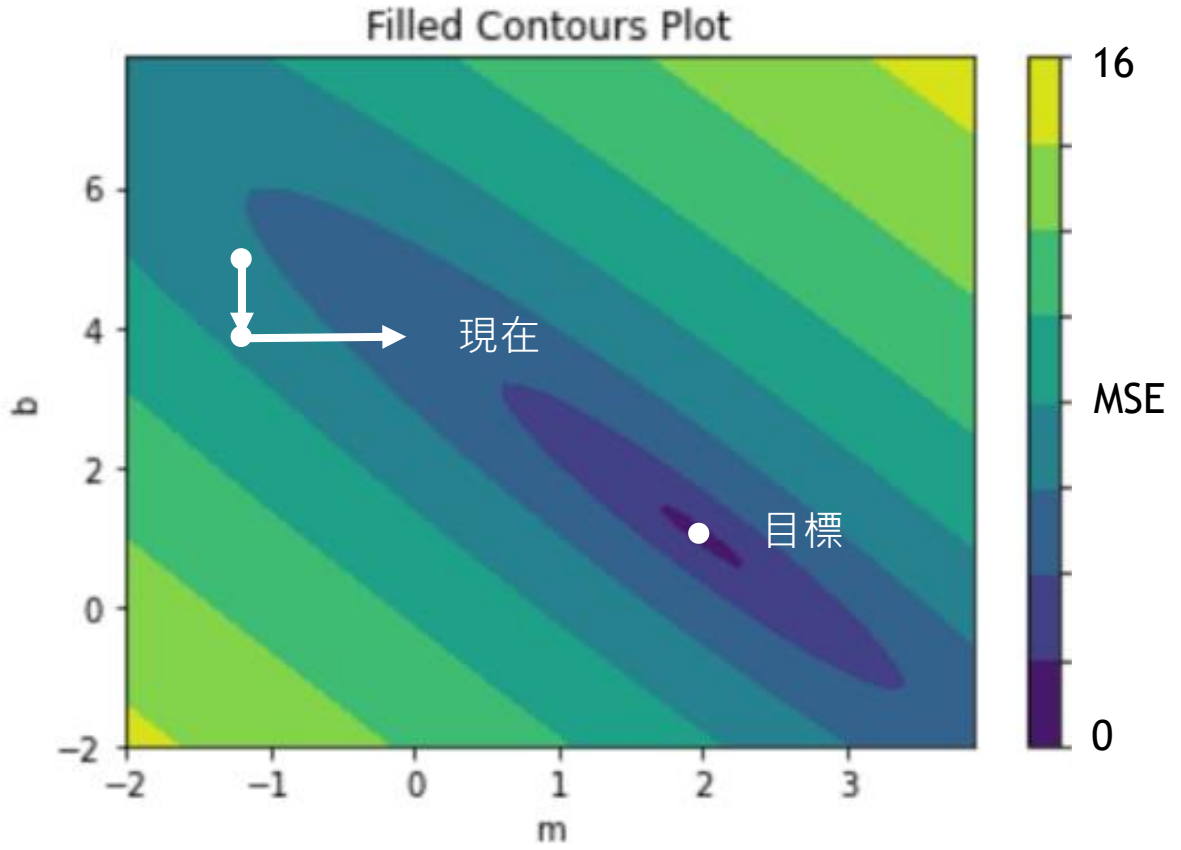


損失曲線



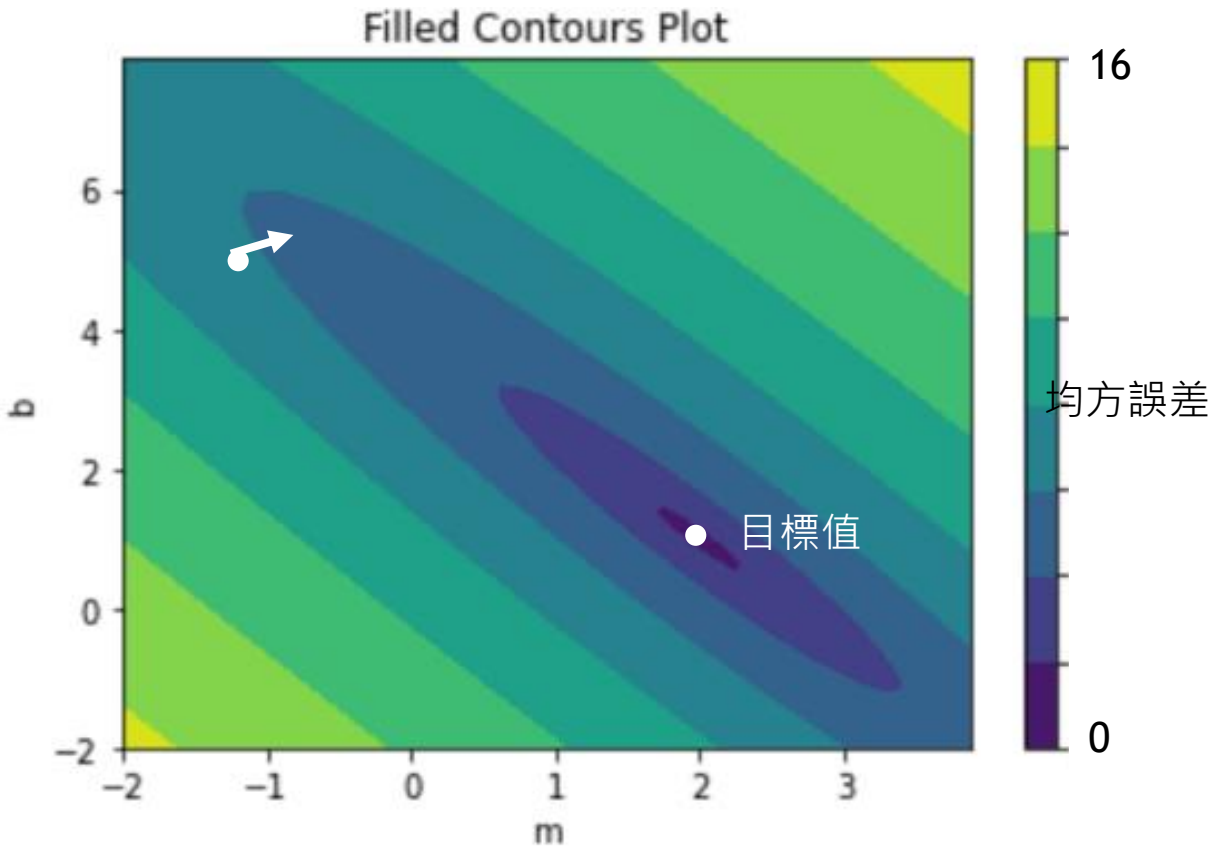
$m = 0$

$b = 4$



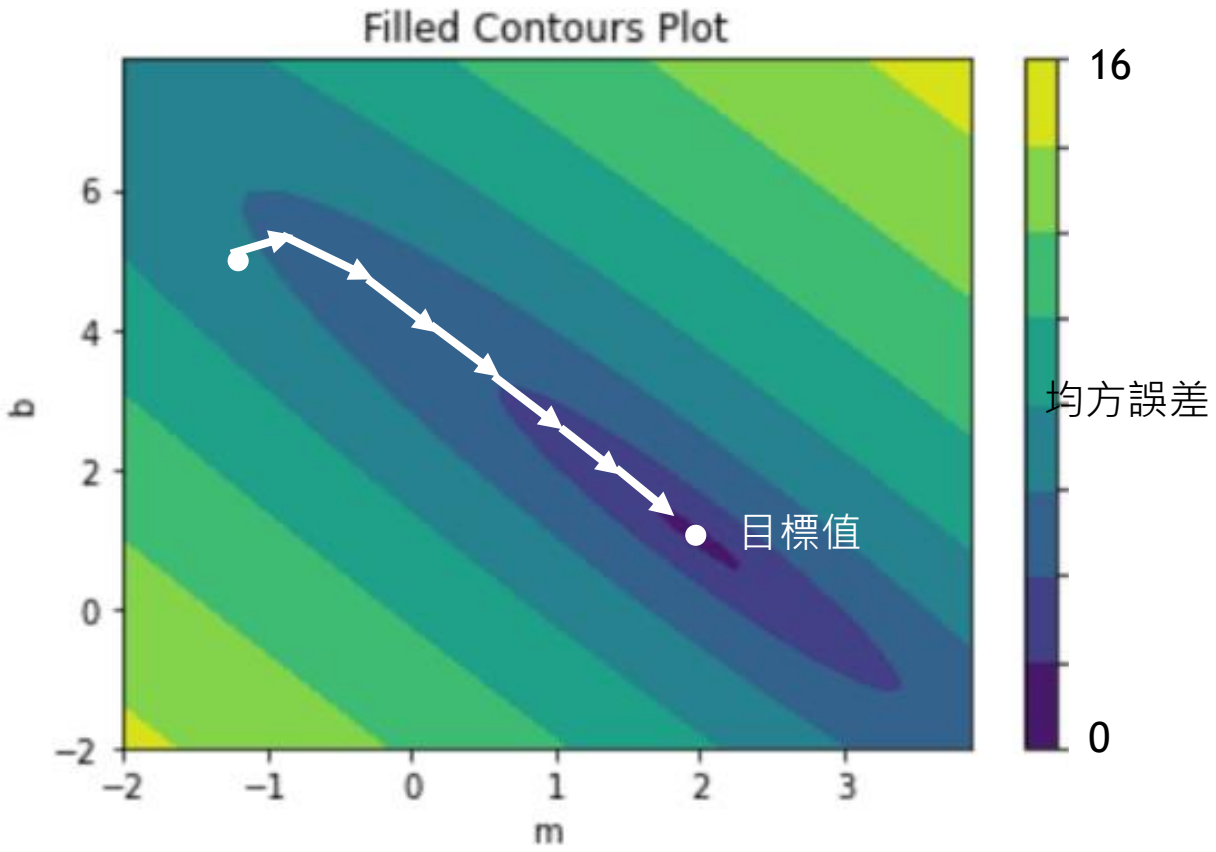
損失曲線

梯度	哪個方向的損失率降低最多
λ：學習率	移動距離
Epoch	以完整資料集進行模型更新



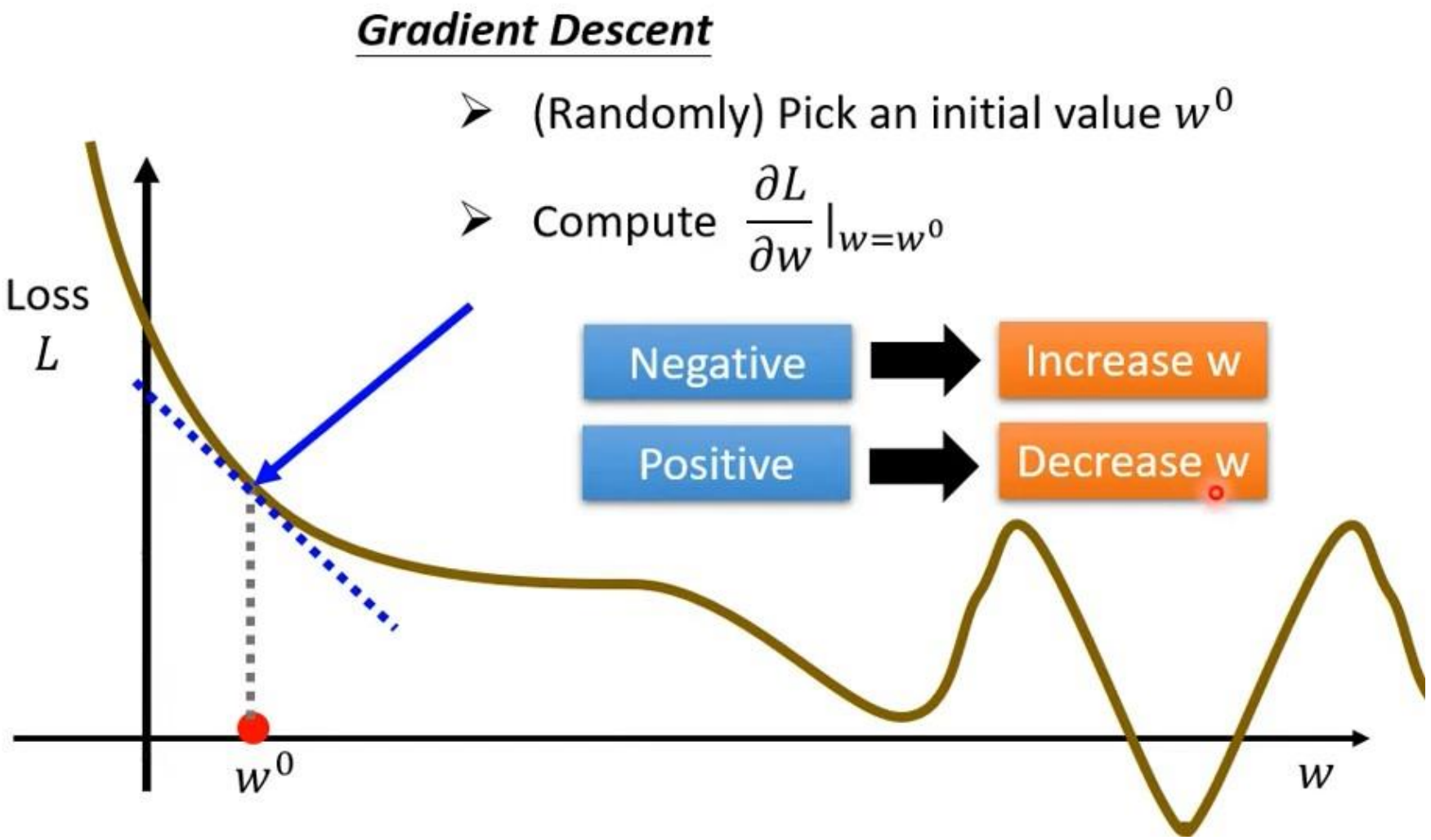
損失曲線

梯度	哪個方向的損失率降低最多
λ：學習率	移動距離
Epoch	以完整資料集進行模型更新



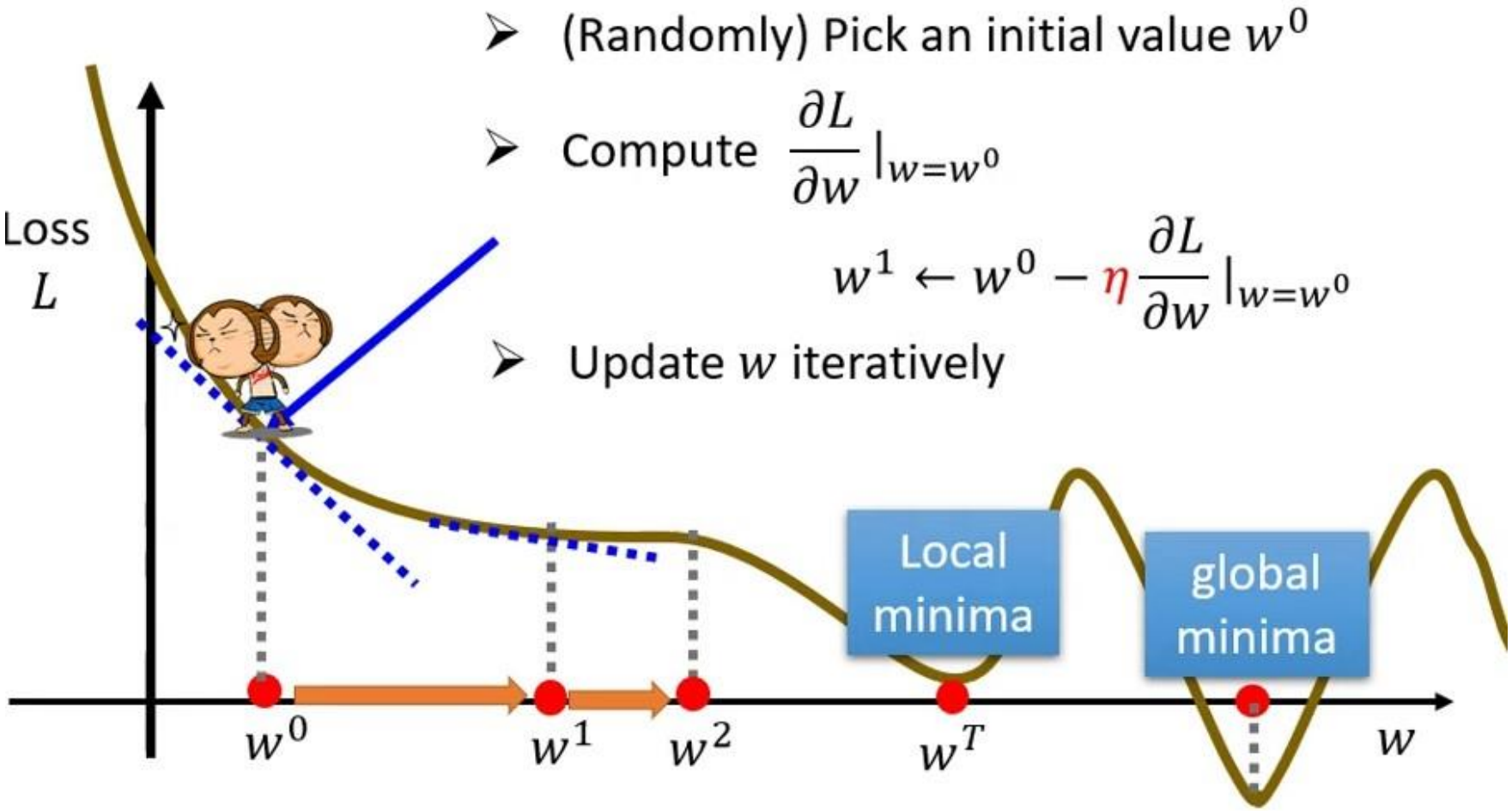
Optimization

$$\Theta = \arg\min_{\{\theta\}} L$$



- Gradient 目標: $\frac{\partial \mathrm{L}}{\partial w} = 0$

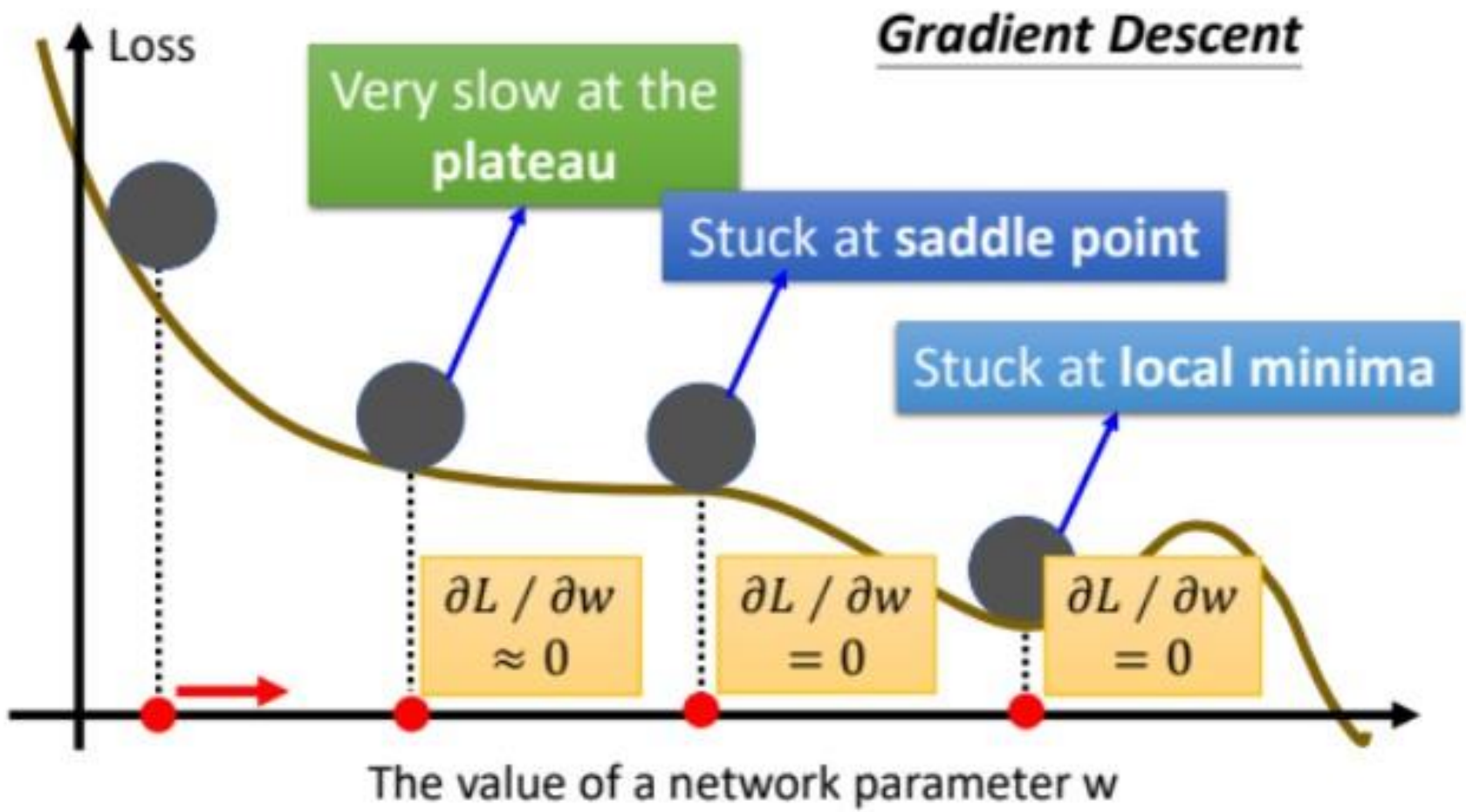
- η: learning rate，每次參數更新的大小



Local Minima and Saddle point

Gradient descent = 0 ，一階微分為 0，可能是進入到 saddle point 或 local minima

可以透過 Hessian matrix 去檢查是否進入到 saddle point 或是 local minima 並提供離開的方向

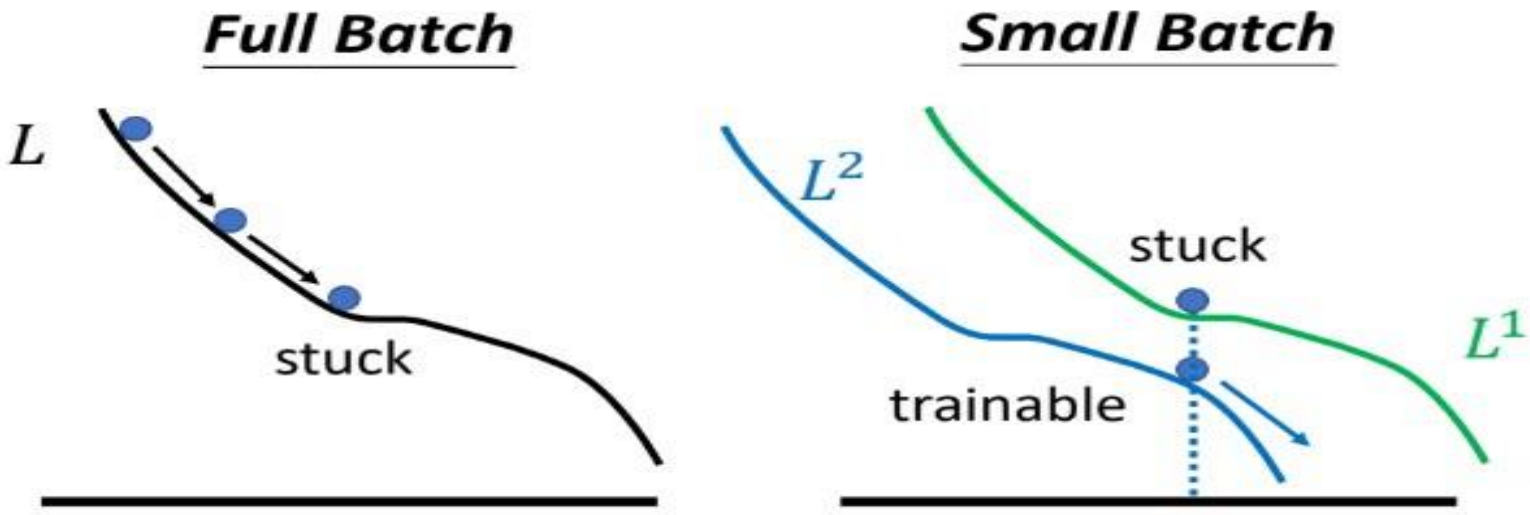

Gradient Descent
Loss
Very slow at the plateau
Stuck at saddle point
Stuck at local minima
$\partial L / \partial w \approx 0$
$\partial L / \partial w = 0$
$\partial L / \partial w = 0$
The value of a network parameter w

To avoid fall into local minima

- 事實上掉入 local minima 的機率並沒有想像中的高
- Smaller batch size and momentum help us not to fall into local minima

Batch size

- 不是一次將所有 training data 拿去 train
- 而是每次 train 都拿一個 batch 去計算 loss 並更新參數
- Batch size is a hyperparameter
- 看完一次 training set 就稱為一個 epoch



Small Batch v.s. Large Batch

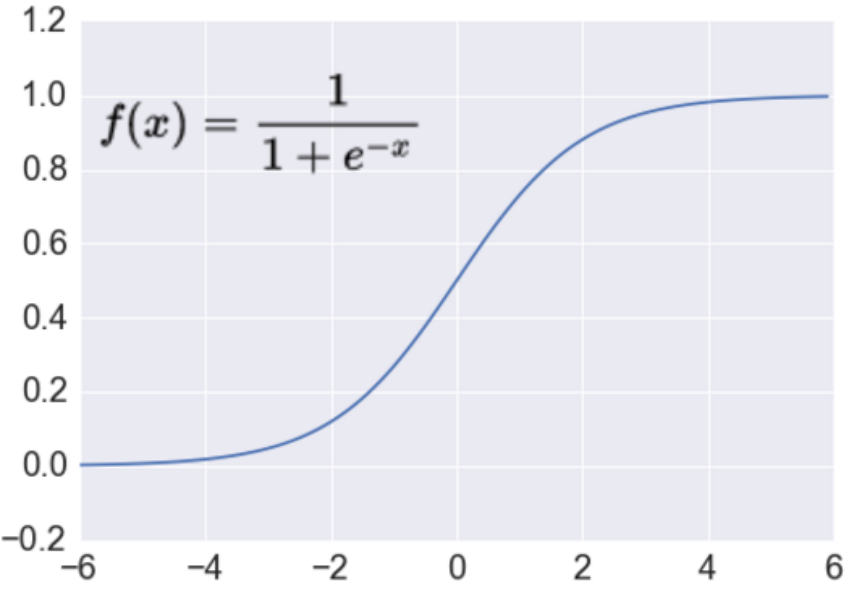
	Small	Large
Speed for one update (no parallel)	Faster	Slower
Speed for one update (with parallel)	Same	Same (not too large)
Time for one epoch	Slower	Faster WIN
Gradient	Noisy	Stable
Optimization	Better WIN	Worse
Generalization	Better WIN	Worse

Batch size is a hyperparameter you have to decide.

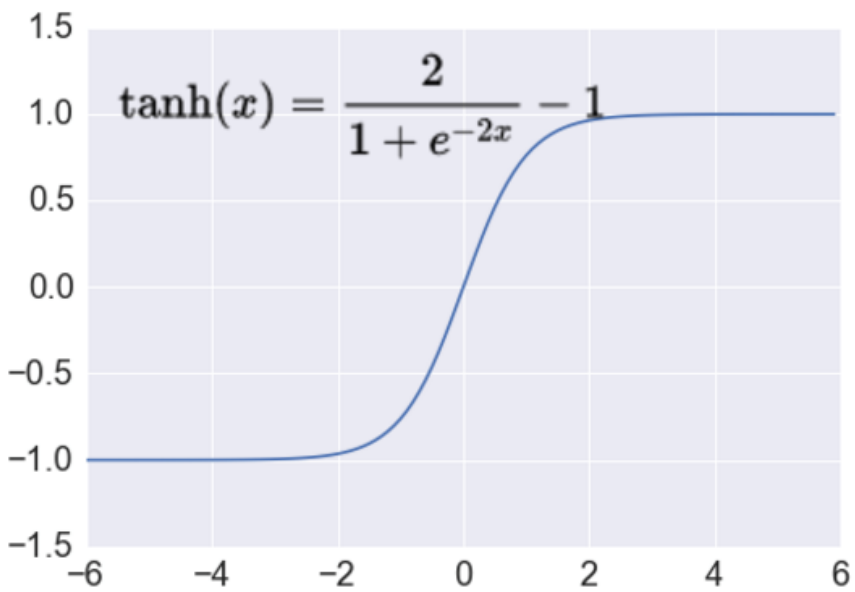
Activation function

- 目的:*引入非線性*
- 若沒有 activation function 則輸入跟輸出為線性組合，無法有效的去 fit 目標函數

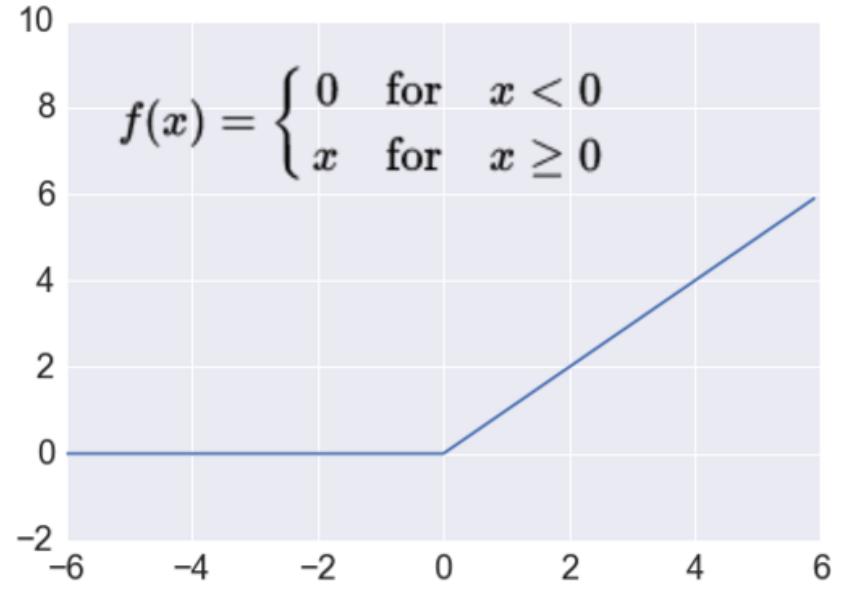
Sigmoid



TanH

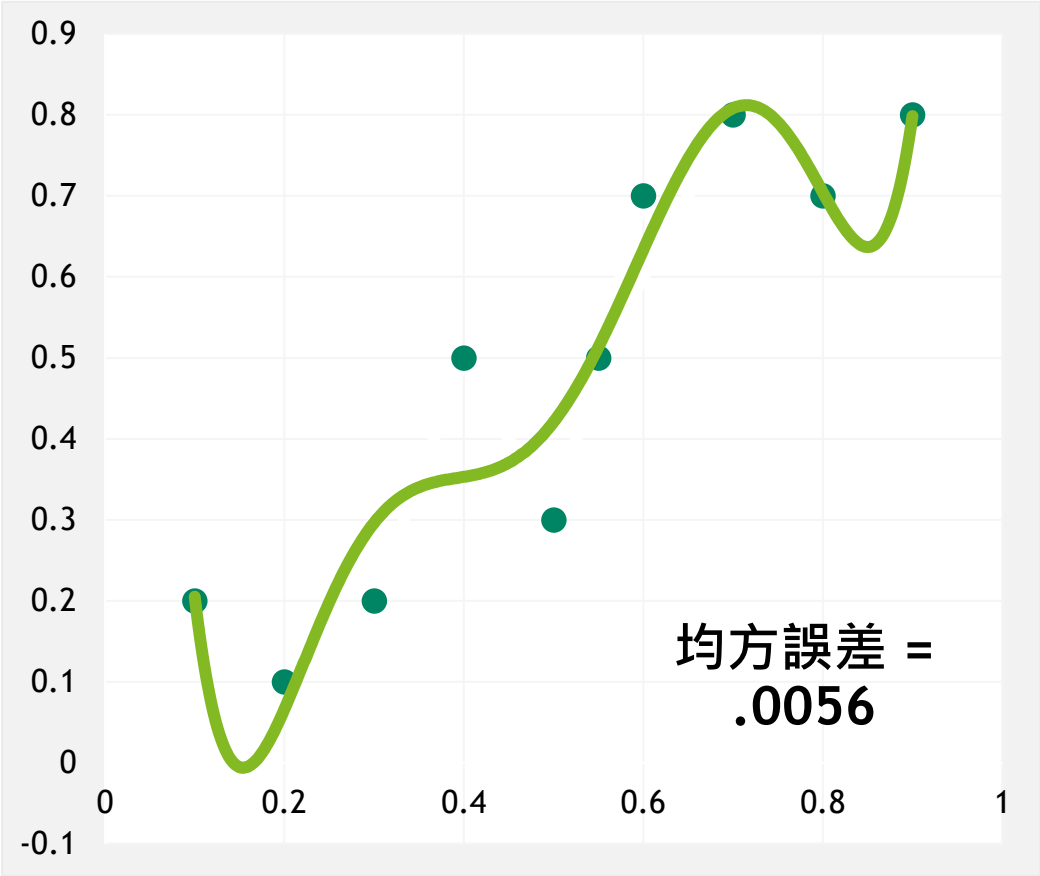


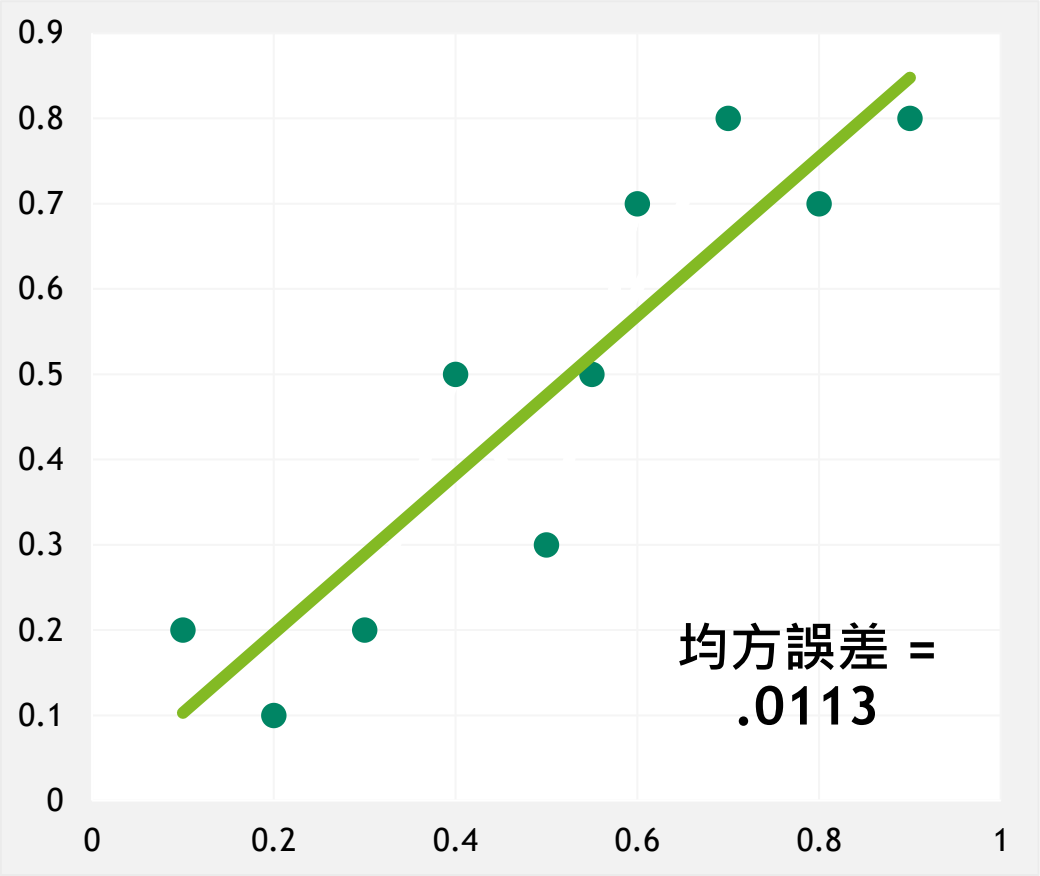
ReLU



過度擬合

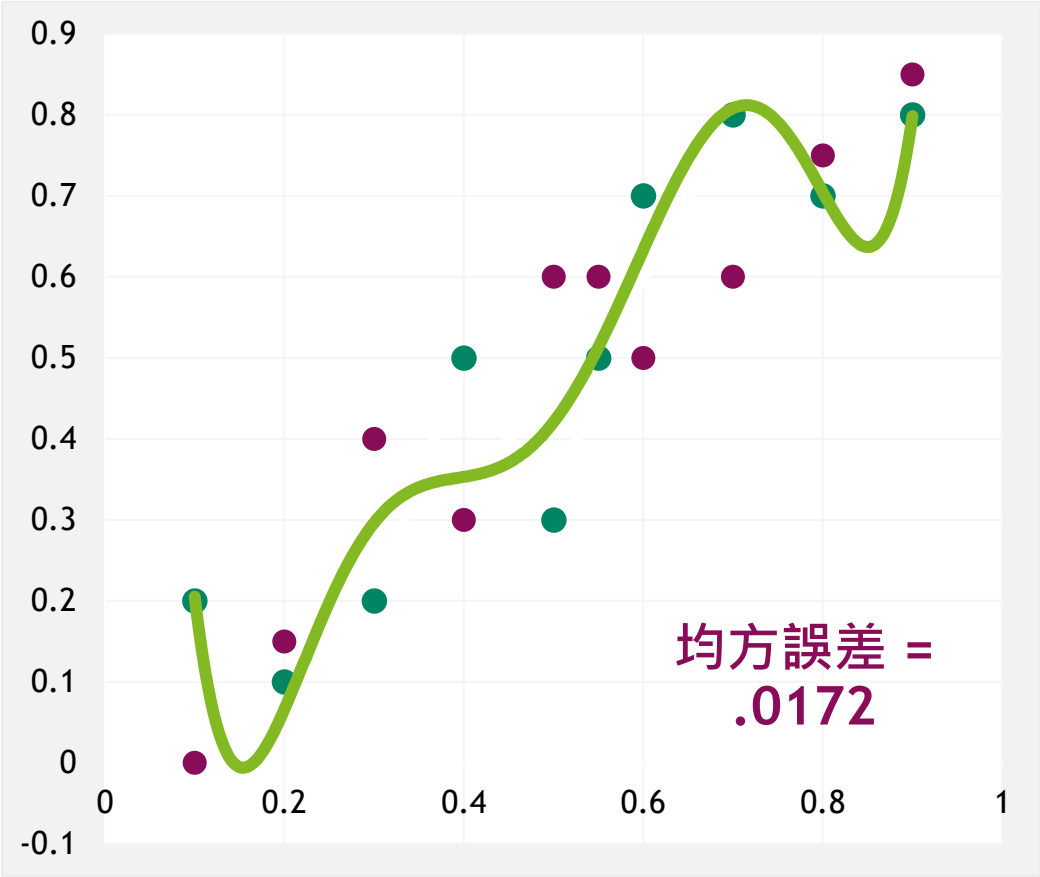
哪條趨勢線更理想？

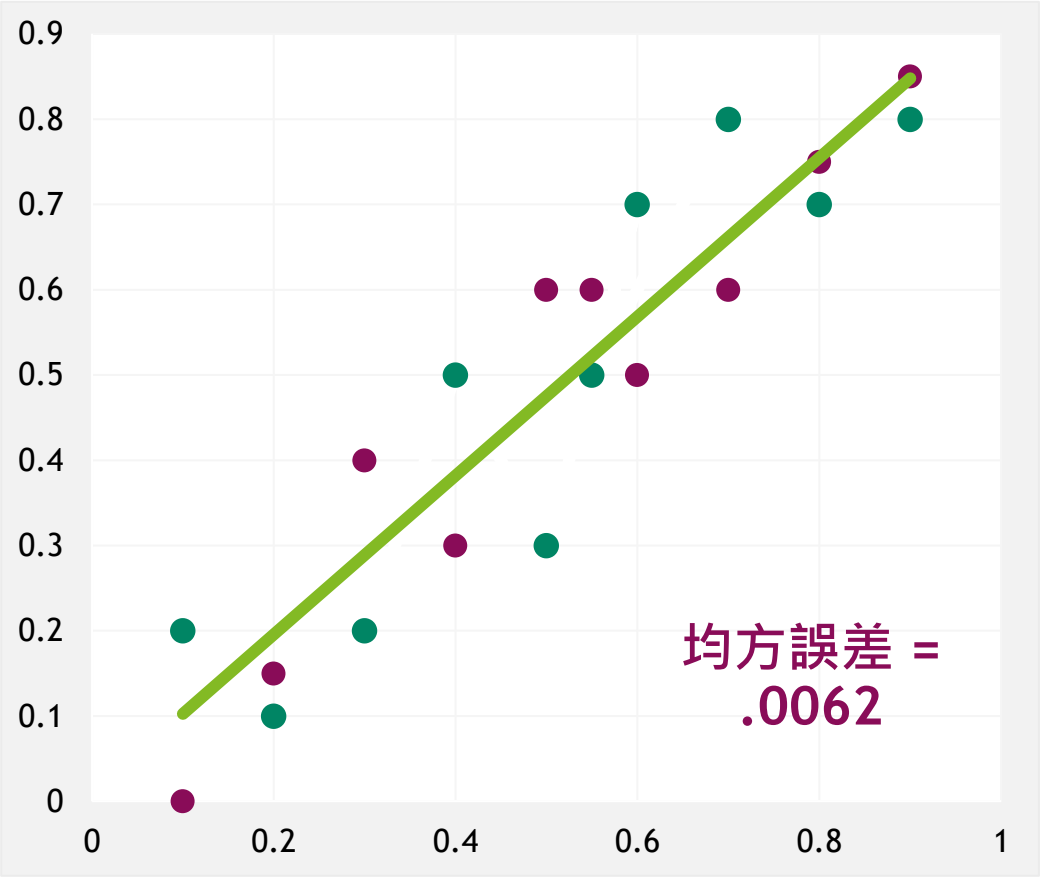




過度擬合

哪條趨勢線更理想？





訓練 VS 驗證資料

避免背誦

訓練資料

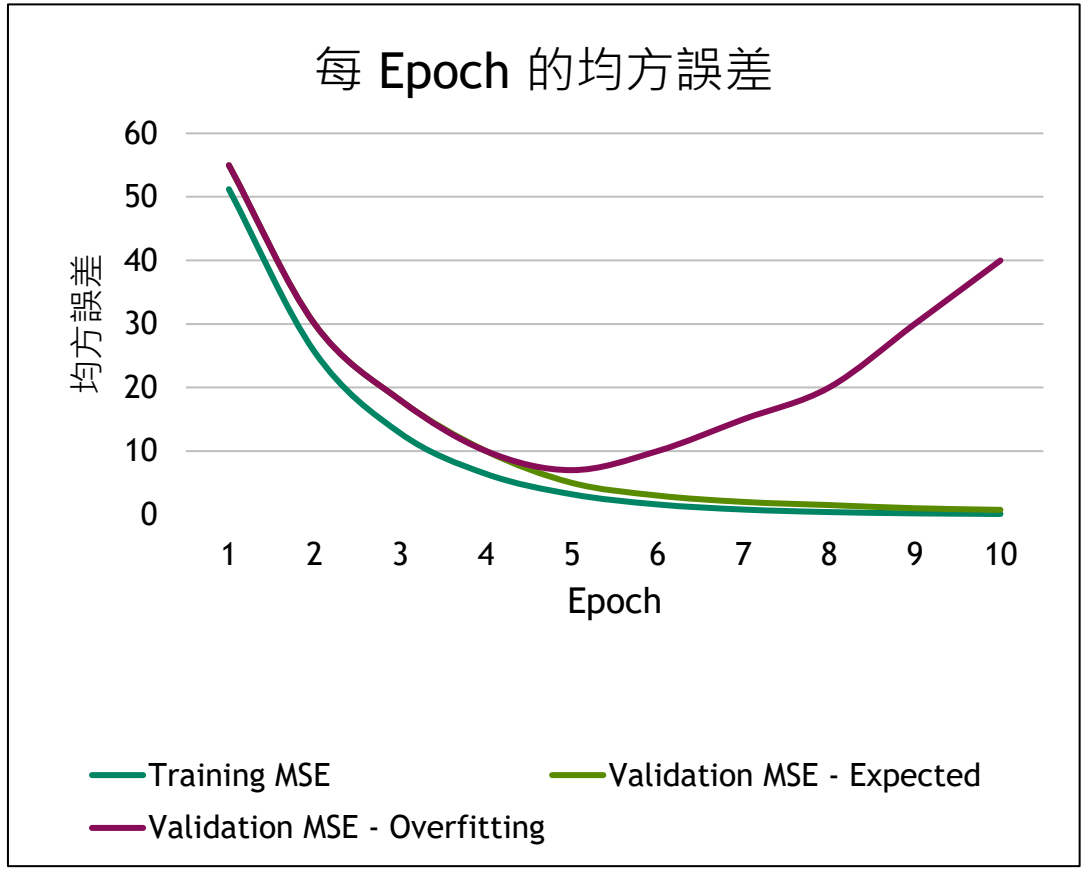
- 讓模型進行學習的核心資料集

驗證資料

- 用來測試模型是否真的瞭解規則的新資料 (可歸納)

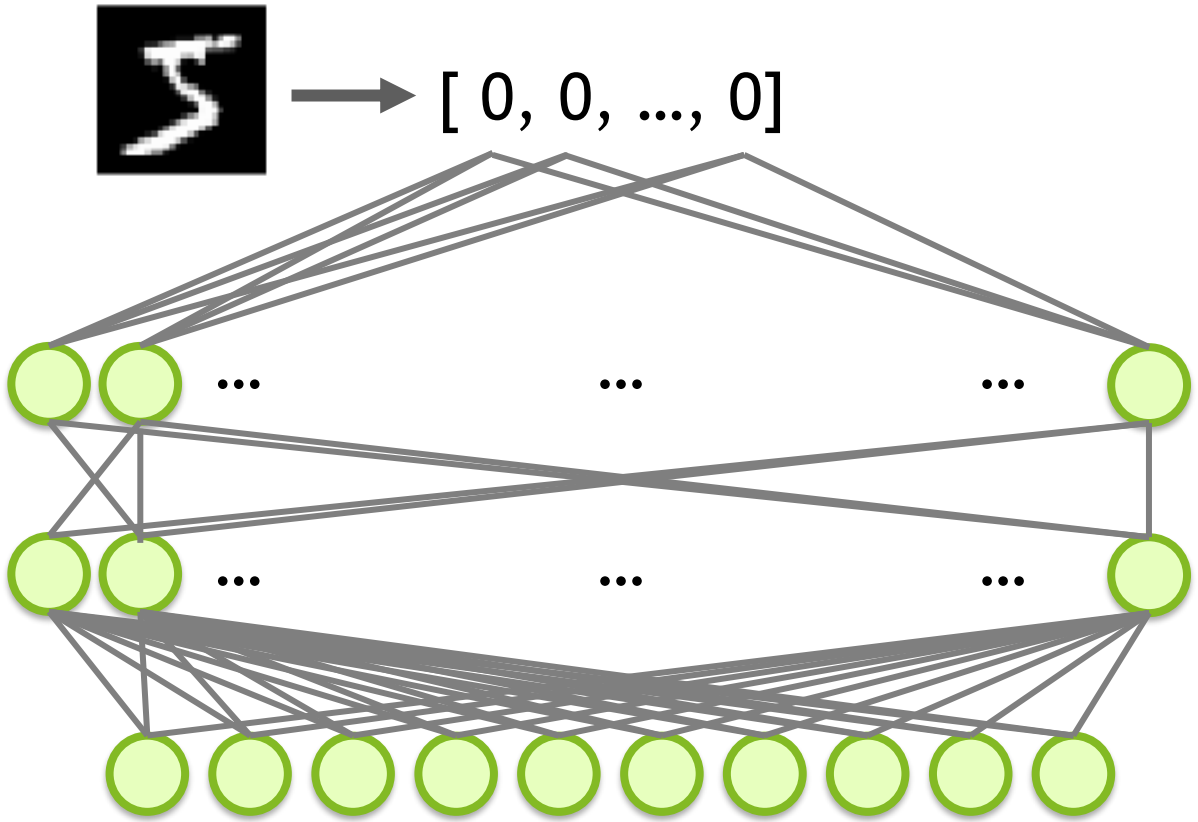
過度擬合

- 模型在訓練資料上效果很好，在驗證資料上卻效果不佳 (背誦的證據)
- 理想情況下，兩個資料集的精確度和損失率應該趨於接近

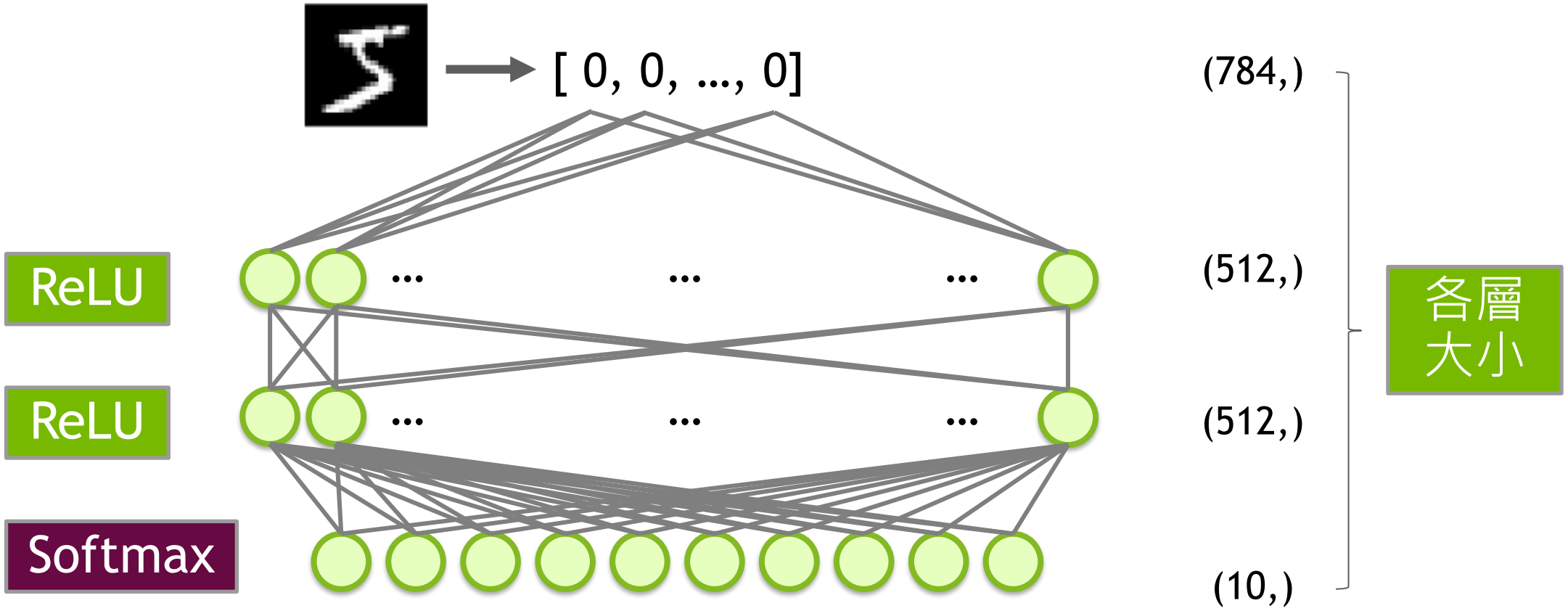


從迴歸到分類

MNIST 模型



MNIST 模型



One hot encoding

- 在分類問題中， output 是對應到一個個的類別，像是 蘋果、香蕉、橘子，但電腦看不懂中文，所以要把水果轉呈相對應的數字

某一種編碼

蘋果: 0
香蕉: 1
橘子: 2

One hot vector

蘋果 : [1, 0, 0]
香蕉 : [0, 1, 0]
橘子 : [0, 0, 1]

- 分 n 類就會有 n 維的向量
- 左邊的再進入 loss function 會出現問題

Loss Function : ategorical_crossentropy

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

y_i : ture category
$\hat{y}_i$: predicted category
只有預測完全正確下才不會有 loss : $1 * \log(1) = 1 * 0 = 0$

E.g. : 分三類

Label = 1
Pred = [0.4, 0.6 , 0.5]
$Loss = -[0 * log(0.4) + 1 * \log(0.6) + 2 * \log(0.5)]$
$= -(-1.89712)$
$= 1.90712$

但 1 跟 0 和 1 跟 2 之間的距離是無意義的，因為不是香蕉和蘋果或是香蕉跟橘子哪個比較近

E.g.: 分三類

Label = [0, 1, 0]
Pred = [0.4, 0.6 , 0.5]
$Loss = -[0 * \log(0.4) + 1 * \log(0.6) + 0 * \log(0.5)]$
$= -(-0.5108256)$
$= 0.5108256$