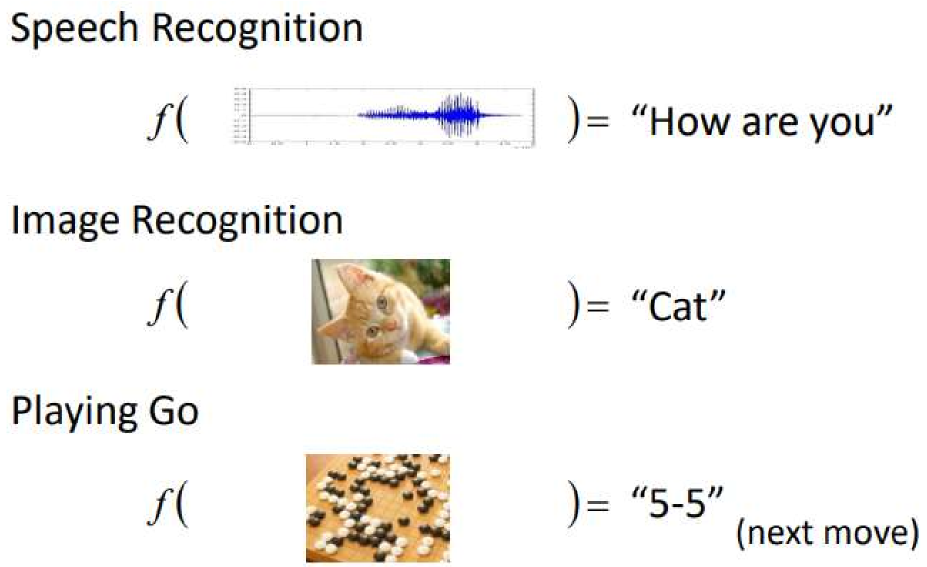
機器學習基本概念簡介

Create at 2022/05/30

- 上課資源
  1. 機器學習基本概念簡介 (上) (https://www.youtube.com/watch?v=Ye018rCVvOo)
  2. 機器學習基本概念簡介 (下) (https://www.youtube.com/watch?v=bHcJCp2Fyxs)

介紹

- Machine Learning ≈ Looking for Function



- 隨著要找的函式不同，機器學習有不同的類別
  - **Regression** : The function outputs a scalar (**數值**).
  - **Classification** : Given options (classes **類別**), the function outputs the correct one.
  - **Structured Learning** : create (**創造**) something with structure (image, document)

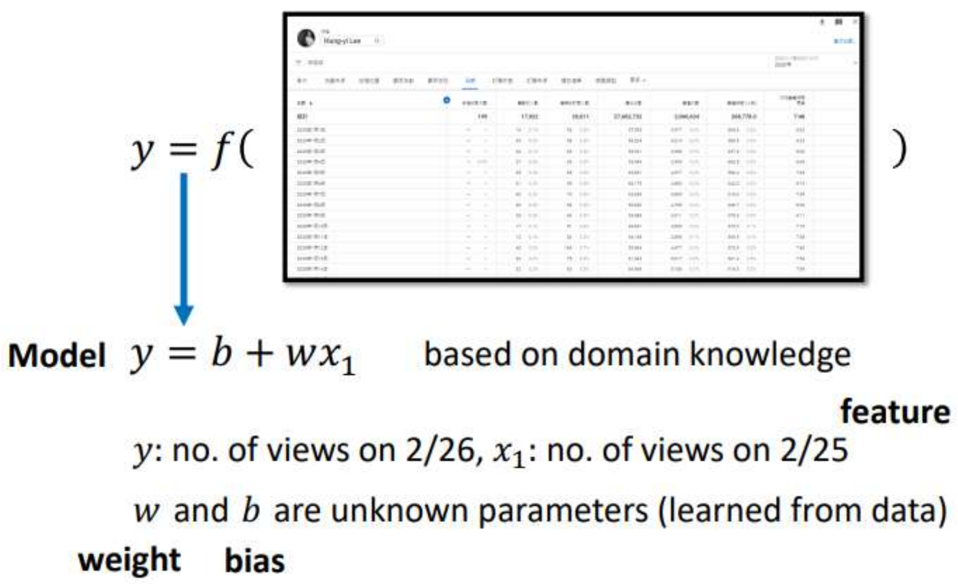
舉例

找一個函式去預測 youtube 的流量

- 輸入 : youtube 後台的資料
- 輸出 : 隔天總點閱率

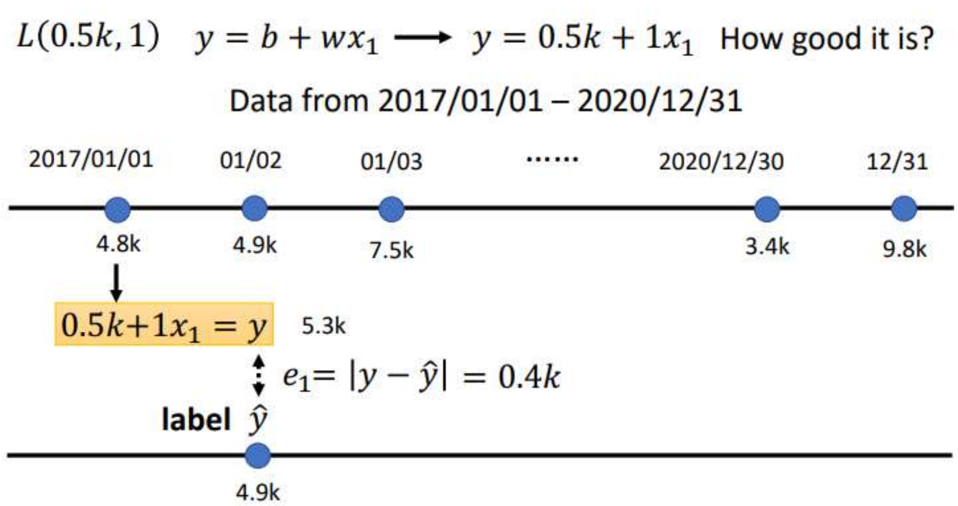
找函式的過程分成三個步驟

Step 1. Function with Unknown Parameters

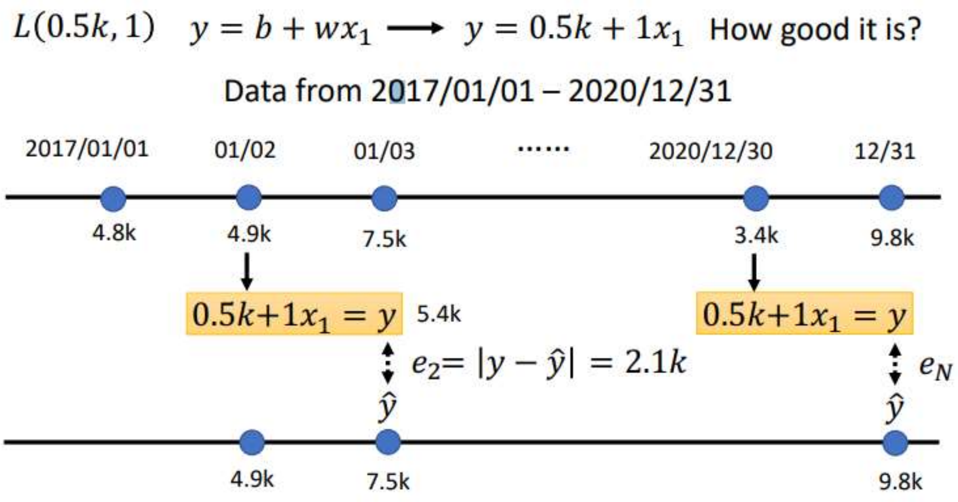


Step 2. Define Loss from Training Data

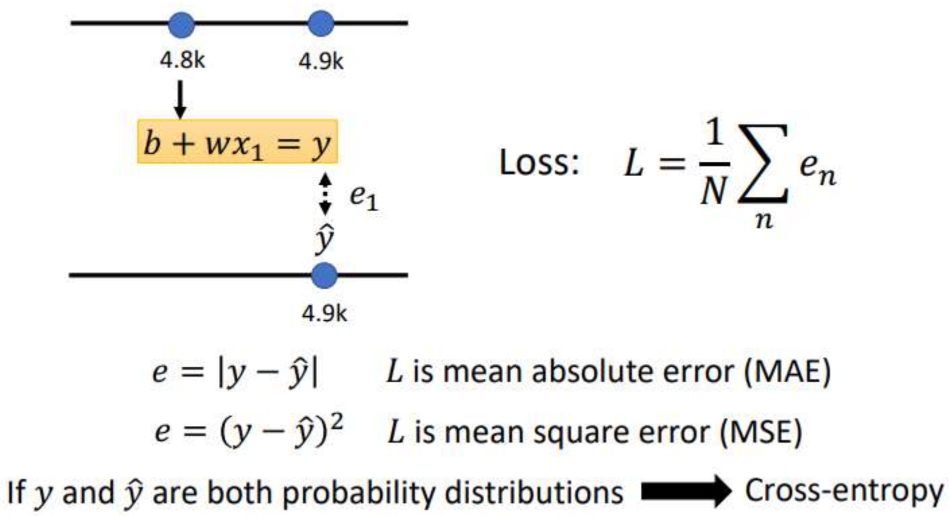
- **Loss** is a function of parameters $L(b, w)$
- Loss : how good a set of values is.



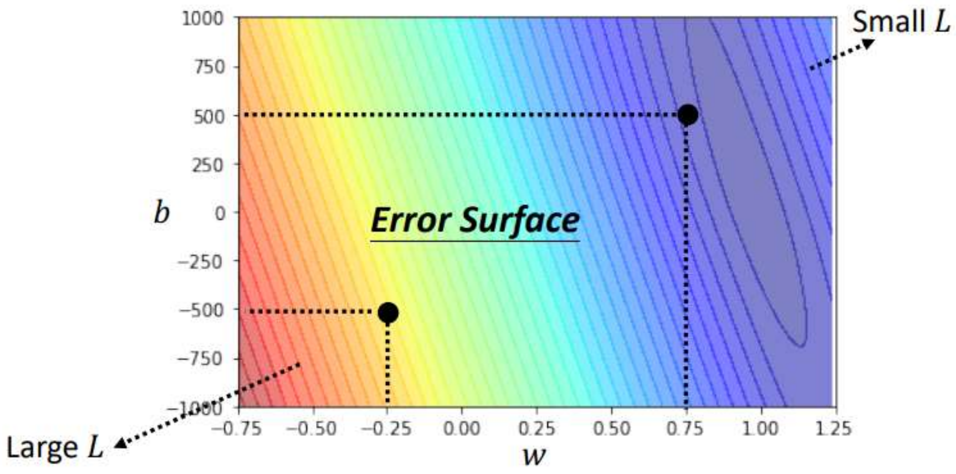
利用 01/01 預測的結果 5.3k，與 01/02 真實的結果 (**label**) 4.9k
去計算預測的差距 e_1 = 0.4k



利用 01/02 預測的結果 5.4k，與 01/03 真實的結果 7.5k
去計算預測的差距 e_2 = 2.1k
以此類推...



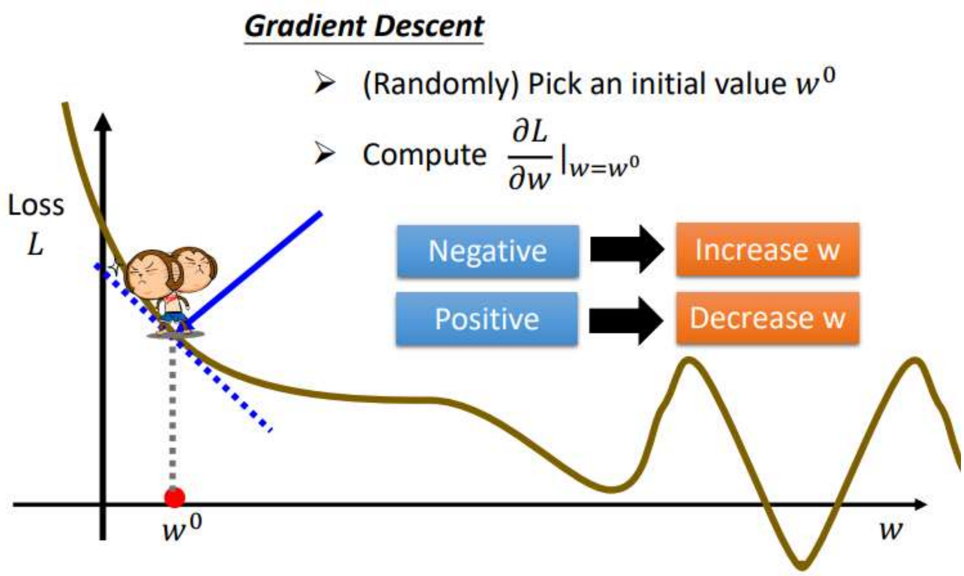
把每天的誤差都加起來，去做平均得到 Loss : L
選擇 MAE 或是 MSE 看任務的需求
如果 y 或是 $\hat{y}$ 都是機率的話，可能會選擇 Cross-entropy



窮舉 w, b

Step 3. Optimization

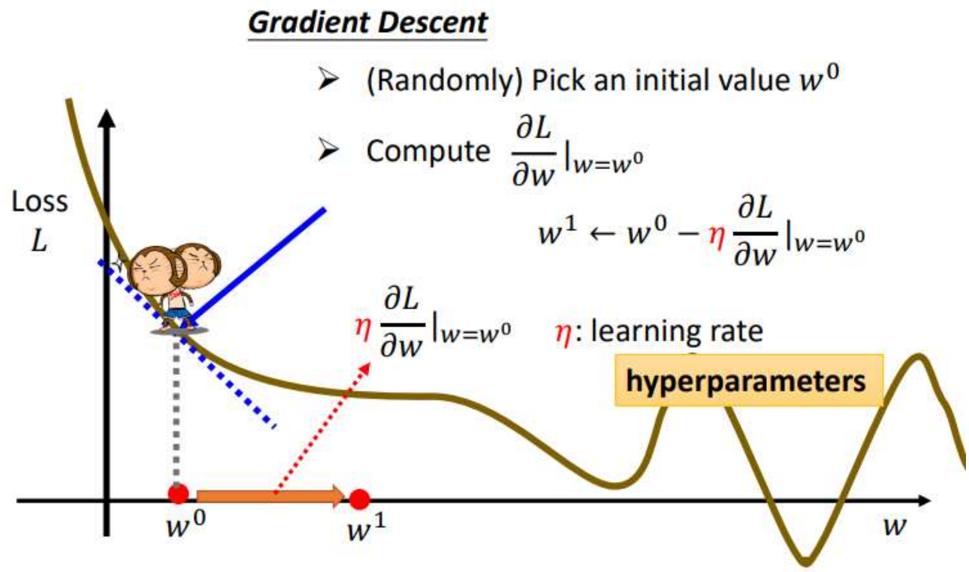
w^* [redacted] $= arg \min_{w} L$ [redacted]



Gradient Descent 方法是假設未知的參數只有一個
當 w 帶不同的數值時，得到不同的 Loss

如何找一個 w 讓 Loss 值最小？

- Step 1. 隨機找一個初始點 w^0
- Step 2. 計算 w 跟 $Loss$ 的微分是多少 (切線斜率)？
  - 若斜率為負 -> 左高右低 -> 把 w 值變大
  - 若斜率為正 -> 左低右高 -> 把 w 值變小
- Step 3. Update w iteratively



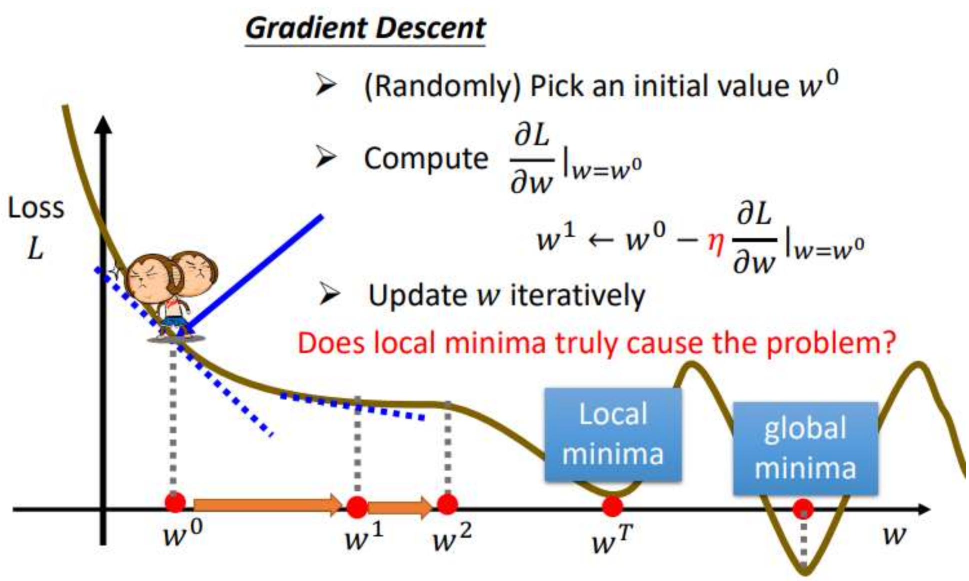
移動的步伐有多大取決於：

- 斜率的絕對值有多大
  - 斜率大，移動大一點
  - 斜率小，移動小一點
- η : learning rate，也會影響步伐的大小
  - η 設定大，每次參數 update 的量大，學習比較快
  - η 設定小，每次參數 update 的量小，學習比較慢

在機器學習裡，需要自己設定的參數稱為 **hyperparameters**

w^0 往右移一步的位置叫做 w^1 = w^0 - $\eta *$ **微分值**

Loss function 如果定義是絕對值就不會有負值
但是 Loss function 如果是其他自己決定的就可能有負值



Update w iteratively

什麼時候會停下來？

- 失去耐心，設定最多計算微分幾次 (參數更新幾次)
- 微分是 0 的時候

Gradient Descent 有個巨大的問題

- 沒有找到真正最好的解，沒有找到讓 Loss 最小的 w，可能走到 w^T (Local minima) iterative 就停住了，沒有找到 global minima
- 但是真正要面對的問題不是 Local minima

$$w^*, b^* = arg \min_{w,b} L$$

➢ (Randomly) Pick initial values w^0, b^0

➢ Compute

$$\frac{\partial L}{\partial w}|_{w=w^0,b=b^0} \qquad w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0}$$

$$\frac{\partial L}{\partial b}|_{w=w^0,b=b^0} \qquad b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$$

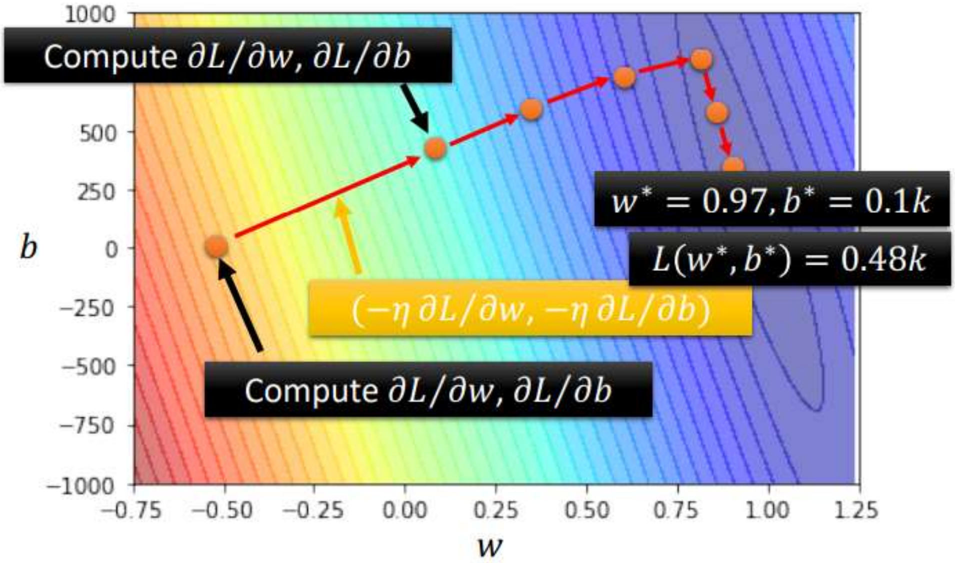
Can be done in one line in most deep learning frameworks

➢ Update w and b interatively

- Step 1. 給定兩個初始參數 w^0, b^0
- Step 2. 計算 w 對 Loss 的微分，計算 b 對 Loss 的微分，接著更新 w, b
- Step 3. Update w, b iteratively

Model $y = b + wx_1$

$$w^*, b^* = arg \min_{w,b} L$$



iteratively 找出 w, b

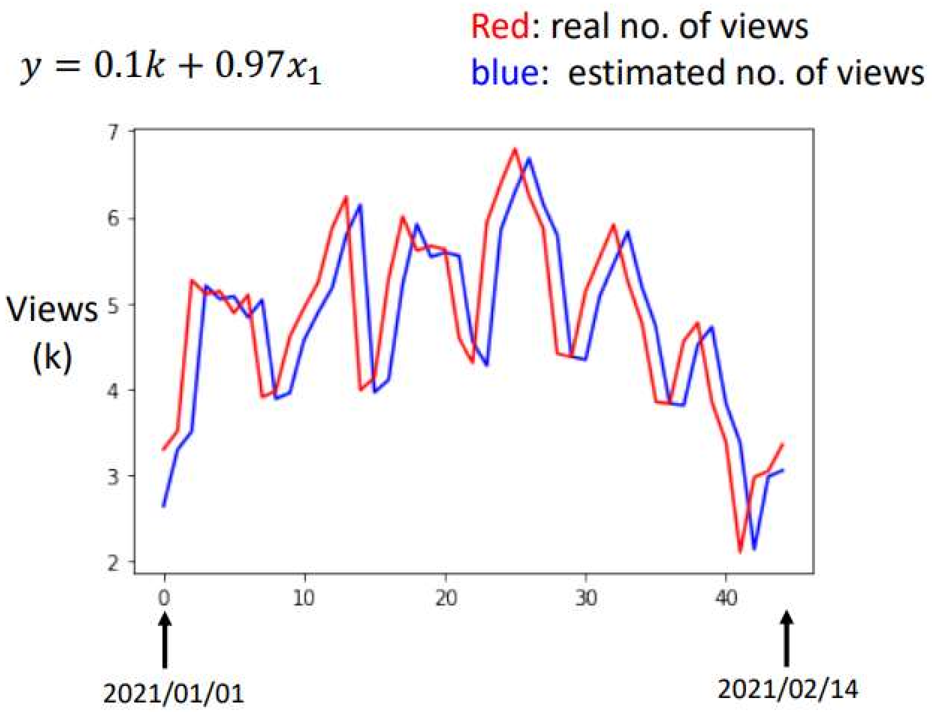
機器學習步驟

$y = 0.1k + 0.97x_1$ achieves the smallest loss $L = 0.48k$ on data of 2017 – 2020 (**training data**)

How about data of 2021 (**unseen during training**)?

$$L' = 0.58k$$

- 這三個步驟，利用已知的資料 training，去計算 Loss function
- 用 2020/12/31 去預測 2021/01/01，以此類推
- 在有看過的資料上，誤差值是 0.48k
  在沒看過的資料上，誤差值比較大，是0.58k



- 發現真實資料與實際資料，有週期性
- 每隔七天會有兩天的觀看人數特別少
- 但 $y = 0.1k + 0.97x_1$ 這個模型只參考前一天
  若知道資料是七天一個循環，應該要看七天
  需要修改模型去參考前七天的資料

$y = b + wx_1$ 2017 - 2020 $L = 0.48k$ 2021 $L' = 0.58k$

$y = b + \sum_{j=1}^{7} w_j x_j$ 2017 - 2020 $L = 0.38k$ 2021 $L' = 0.49k$

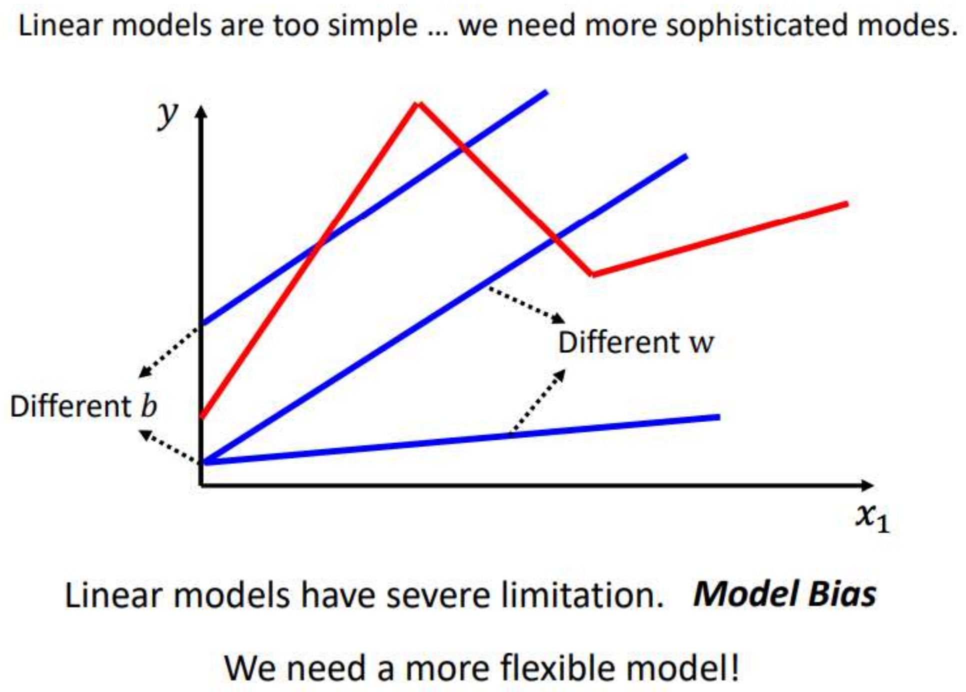
b	w_1^*	w_2^*	w_3^*	w_4^*	w_5^*	w_6^*	w_7^*
0.05k	0.79	-0.31	0.12	-0.01	-0.10	0.30	0.18

$y = b + \sum_{j=1}^{28} w_j x_j$ 2017 - 2020 $L = 0.33k$ 2021 $L' = 0.46k$

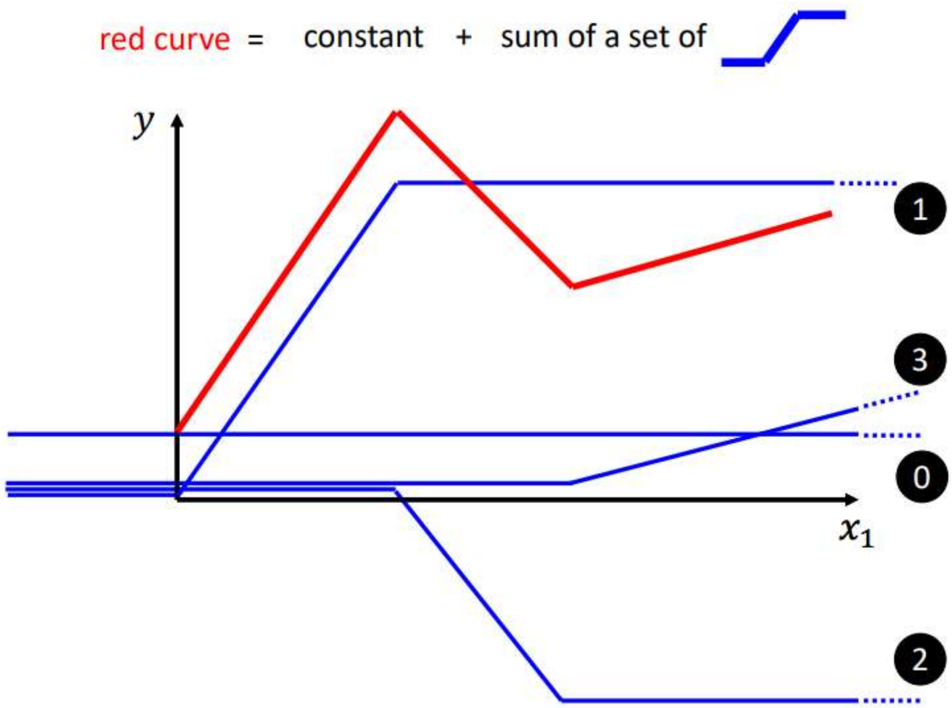
$y = b + \sum_{j=1}^{56} w_j x_j$ 2017 - 2020 $L = 0.32k$ 2021 $L' = 0.46k$

Linear models

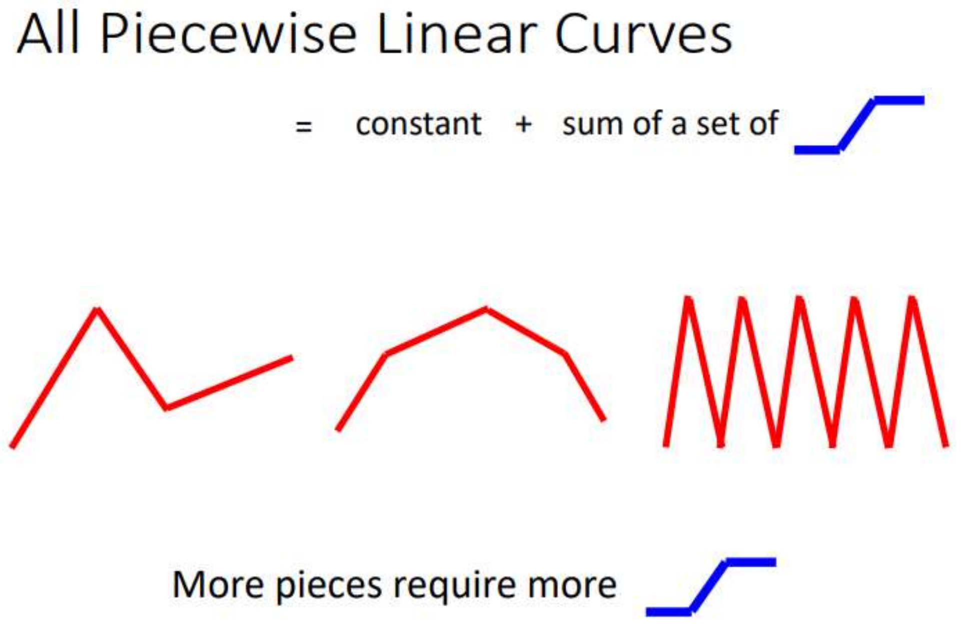
參考比較多天的資料，Loss 有得到較好的結果
但最後，考慮天數這件事，可能到達極限，無法讓 Loss 持續得到更好的結果



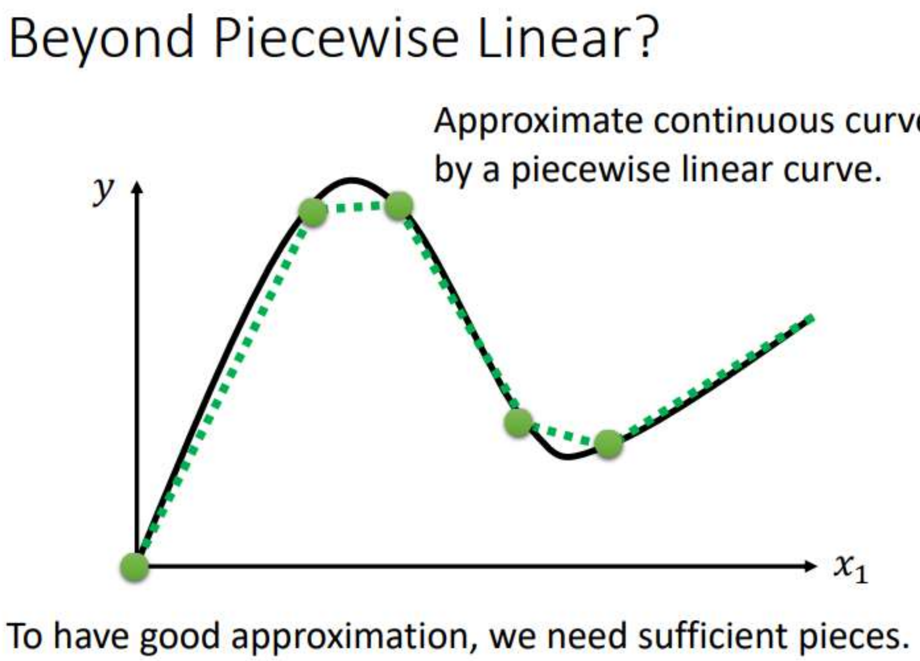
- Linear 太簡單，只能解決簡單的問題，永遠無法造出紅色那條線
- 根據 model 的限制，無法模擬真實的狀況，稱為 **Model Bias**
- 需要寫一個更複雜未知參數的 model



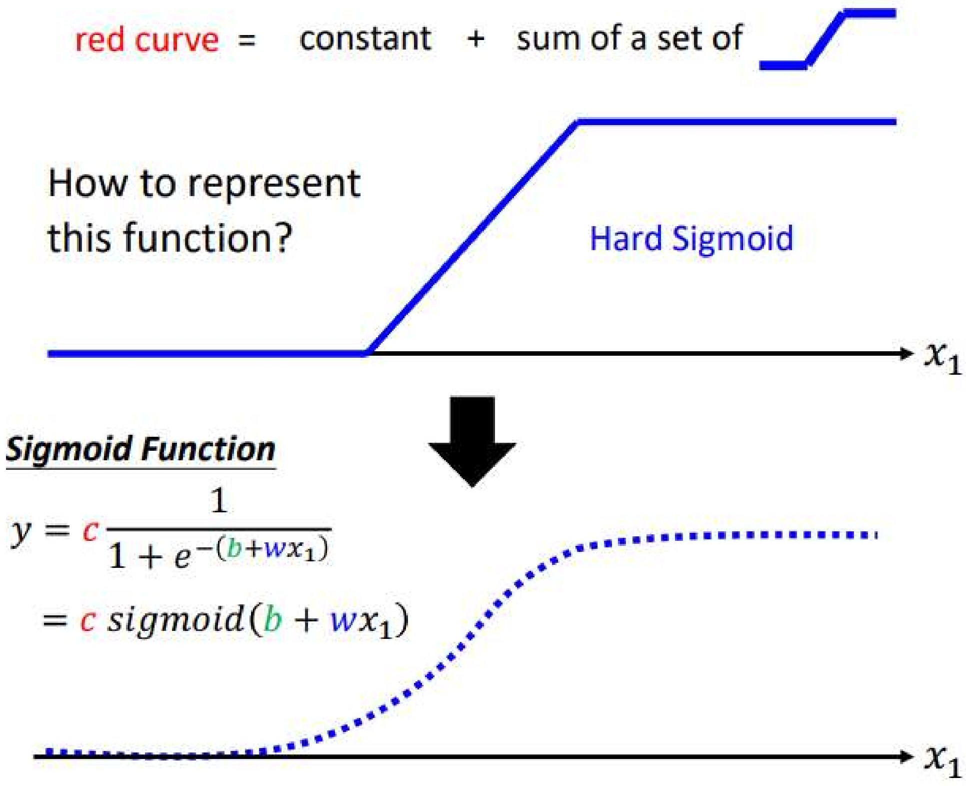
看轉折點，利用 constant + 一堆的藍色 function



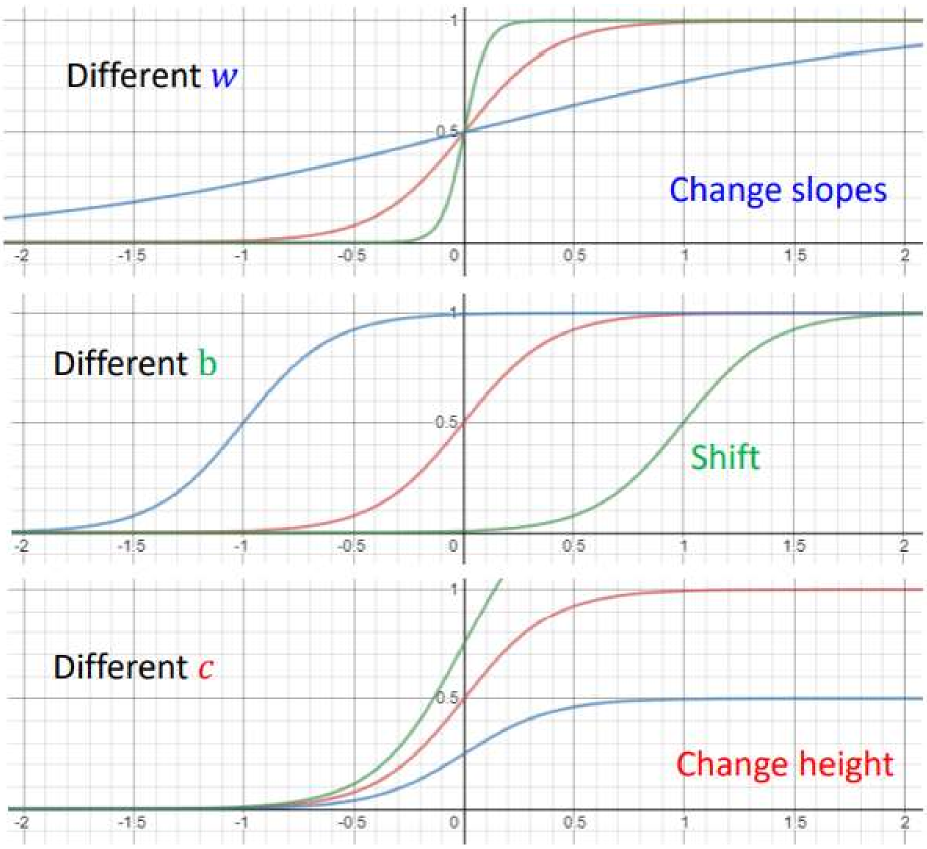
- Piecewise Linear Curves : Curve 是由很多線段所組成
- 當 piecewise linear curve 越複雜，轉折點越多，需要的藍色 function 就越多



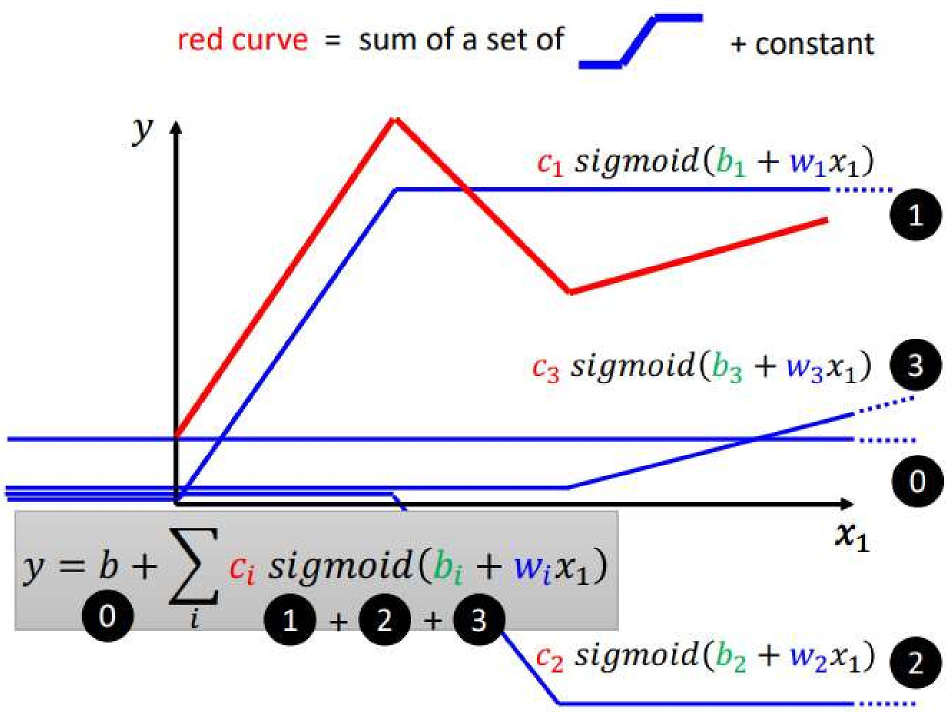
- 先取得一些點，再把這些點連起來，變成一個 piecewise linear
- 若點取得的夠好、夠多，用夠多的藍色 function，就可以變成一個任何連續的曲線
- **用曲線去逼近藍色的 function**



- 用 **Sigmoid Function** 去逼近藍色的 function (通常稱為 Hard Sigmoid)
- sigmoid function : 當輸入非常大的時候，sigmoid 就會趨近於 1，所以結果就會停在常數 c 的地方 ; 當輸入負數非常大的時候，分母就會非常大，y 值就會趨近於 0
- sigmoid function 可以理解成是 S 型的 function
- 調整 b, w, c，可以製造各種不同形狀的 sigmoid function



- 若改變 w，可以改變曲線的斜率
- 若改變 b，可以讓曲線左右移動
- 若改變 c，可以改變曲線的高度



- 假設 b, w, c 是未知的參數，可以製造各種不同的藍色 function
- 疊起來之後，就可以製造不同紅色的 curve，就可以製造不同的 piecewise linear curve，就可以去逼近各式各樣不同的 continue curve function

New Model: More Features

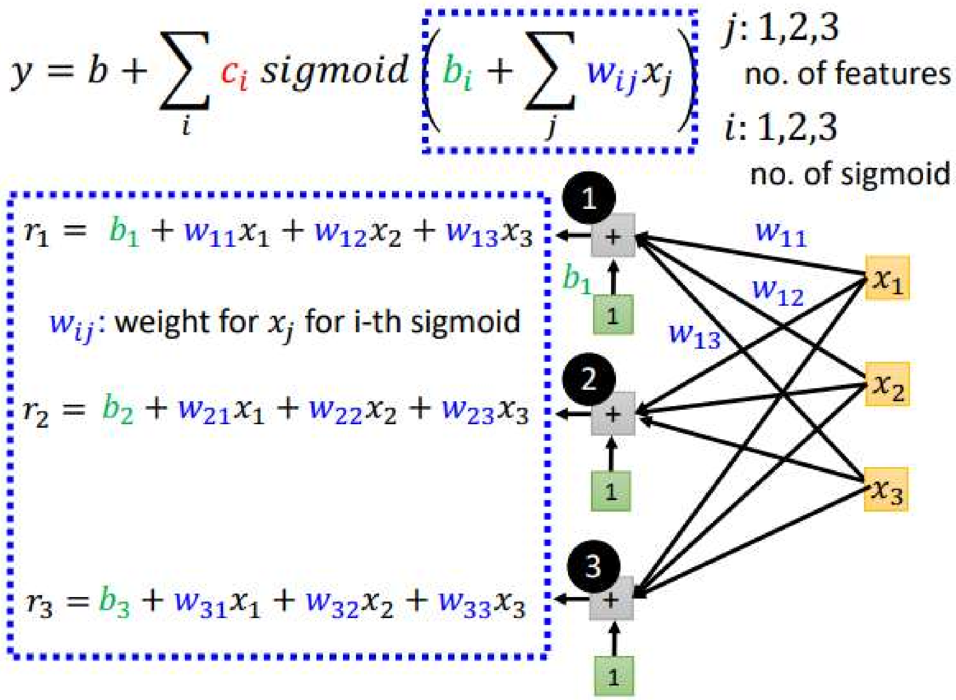
$$y = \underline{b + wx_1}$$

$$y = b + \sum_i c_i\ sigmoid(\underline{b_i + w_ix_1})$$

$$y = \underline{b + \sum_j w_jx_j}$$

$$\boxed{y = b + \sum_i c_i\ sigmoid\left(\underline{b_i + \sum_j w_{ij}x_j}\right)}$$

- 減少 model 的 bias



x_1 : 一天前的觀看人數
x_2 : 兩天前的觀看人數
x_3 : 三天前的觀看人數

i : 代表藍色的 function (目前每個藍色 function 都用 sigmoid 去近似)

- 第一個 sigmoid (i = 1) : $b_1 + w_{11}x_1 + w_{12}x_2 + w_{13}x_3 = r_1$
- 第二個 sigmoid (i = 2) : $b_2 + w_{21}x_1 + w_{22}x_2 + w_{23}x_3 = r_2$
- 第三個 sigmoid (i = 1) : $b_3 + w_{31}x_1 + w_{32}x_2 + w_{33}x_3 = r_3$

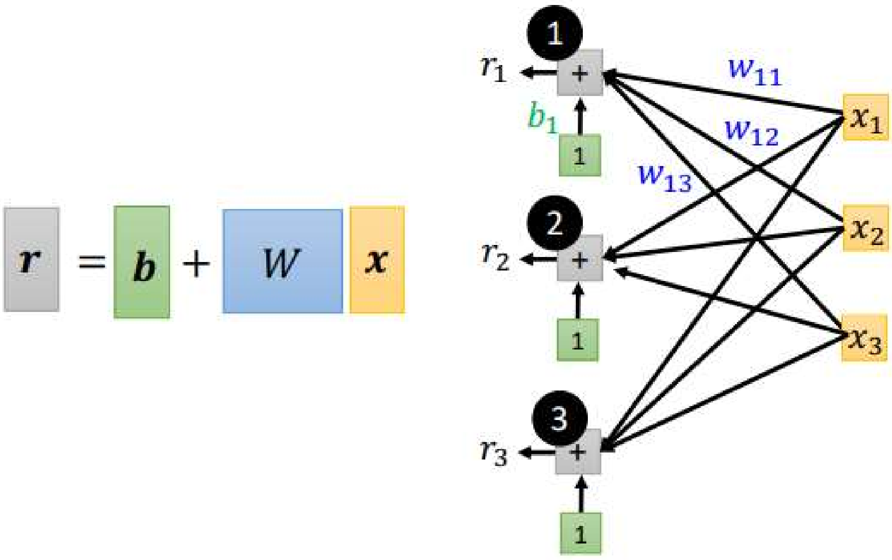
$$y = b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right) \quad \begin{matrix} i: 1,2,3 \\ j: 1,2,3 \end{matrix}$$

$$\boxed{\begin{aligned} r_1 &= b_1 + w_{11}x_1 + w_{12}x_2 + w_{13}x_3 \\ r_2 &= b_2 + w_{21}x_1 + w_{22}x_2 + w_{23}x_3 \\ r_3 &= b_3 + w_{31}x_1 + w_{32}x_2 + w_{33}x_3 \end{aligned}}$$

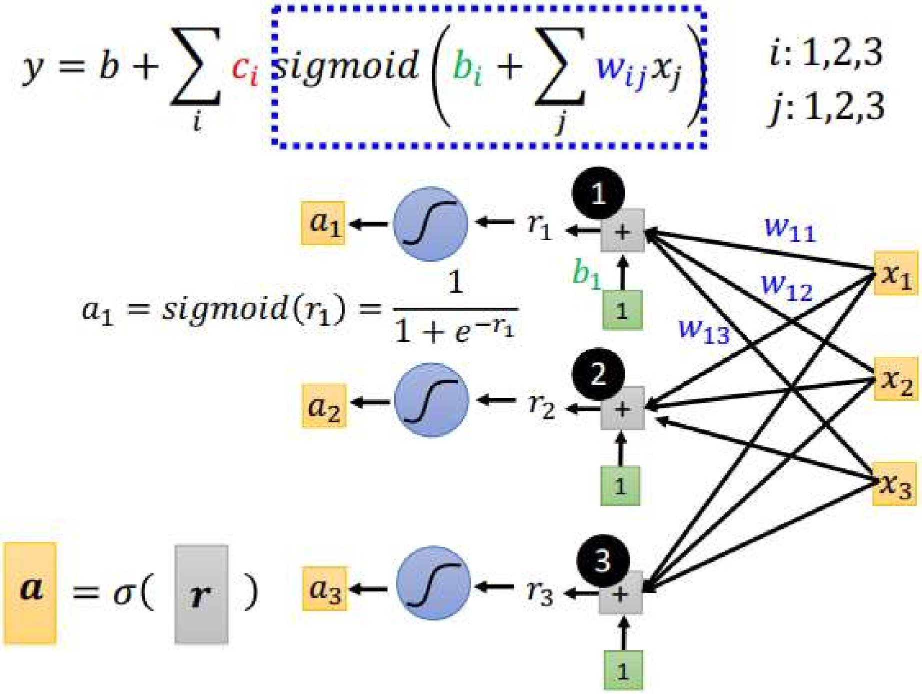
$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} + \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

利用矩陣寫法簡化

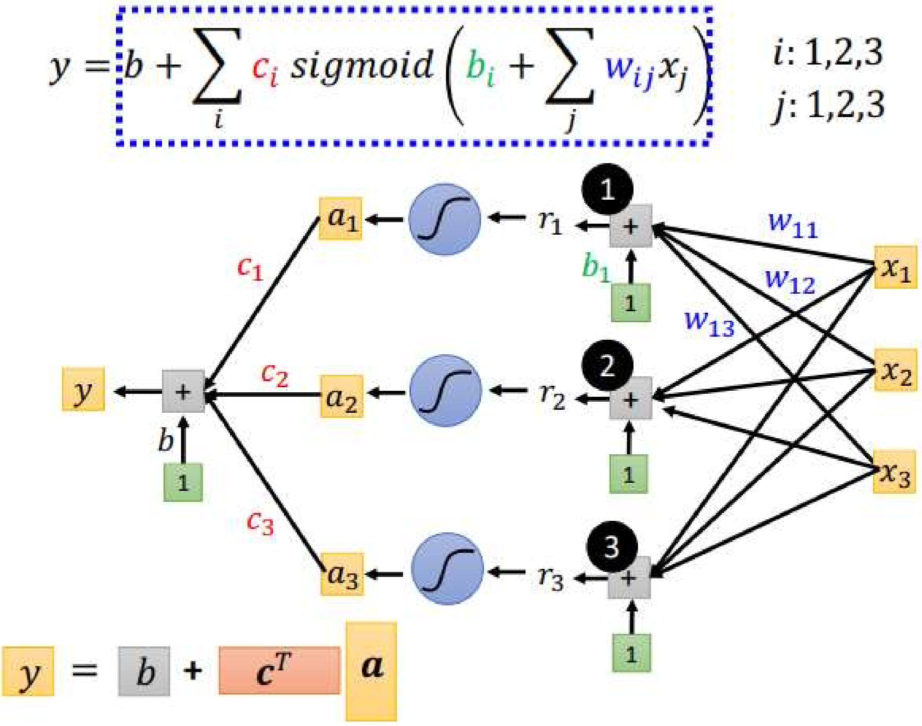
$$y = b + \sum_i c_i \, sigmoid\boxed{\left(b_i + \sum_j w_{ij} x_j\right)} \quad \begin{matrix} i: 1,2,3 \\ j: 1,2,3 \end{matrix}$$



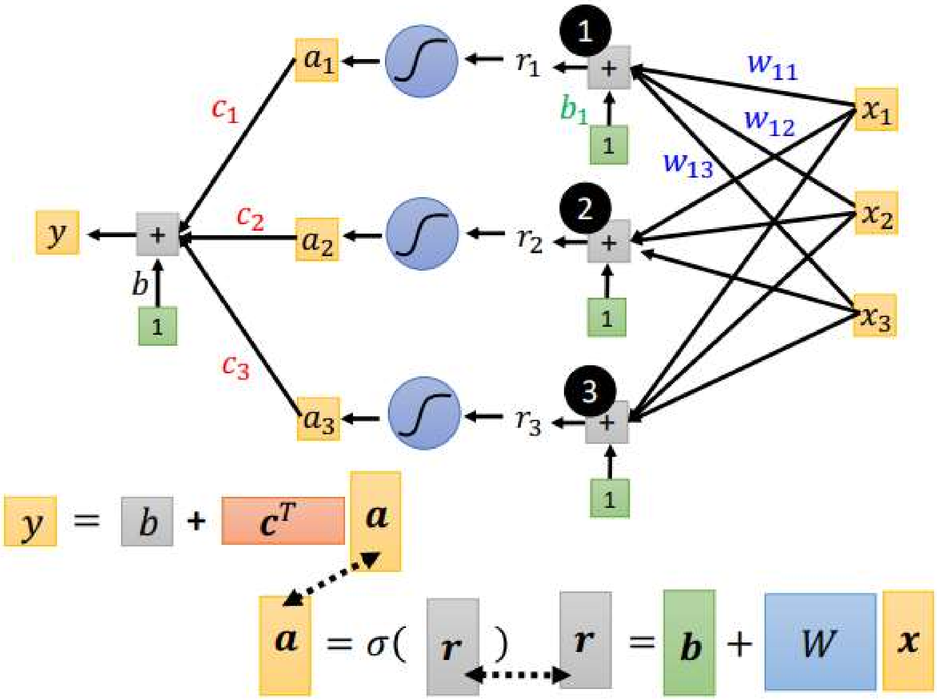
括號內的寫法簡化成 $r = b + Wx$



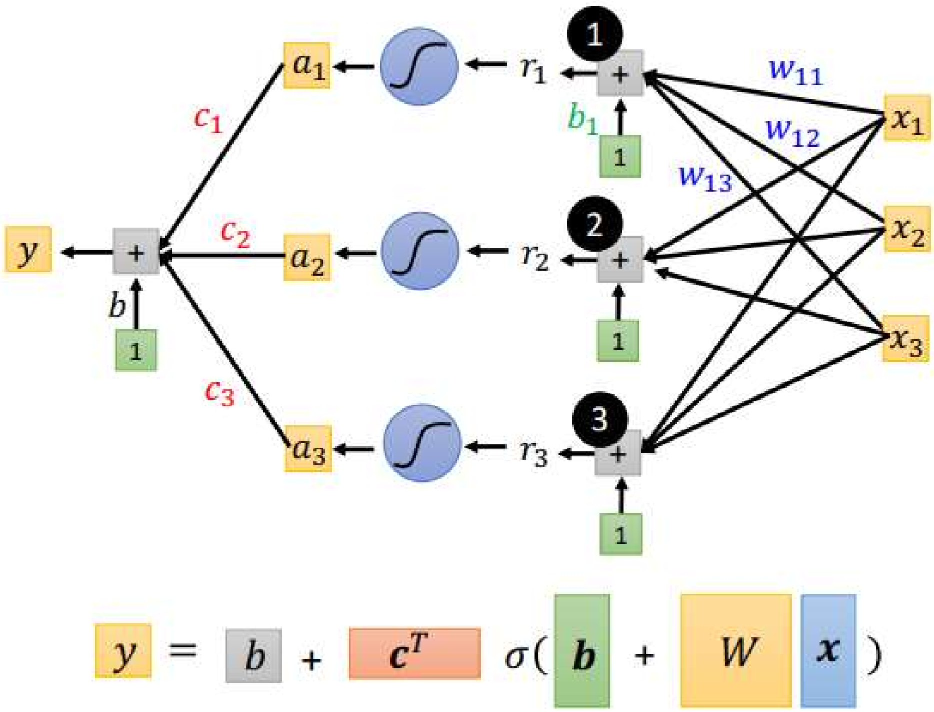
r_1, r_2, r_3 去分別通過 sigmoid function，得到 a_1, a_2, a_3
式子簡寫為 $a = \sigma(r)$



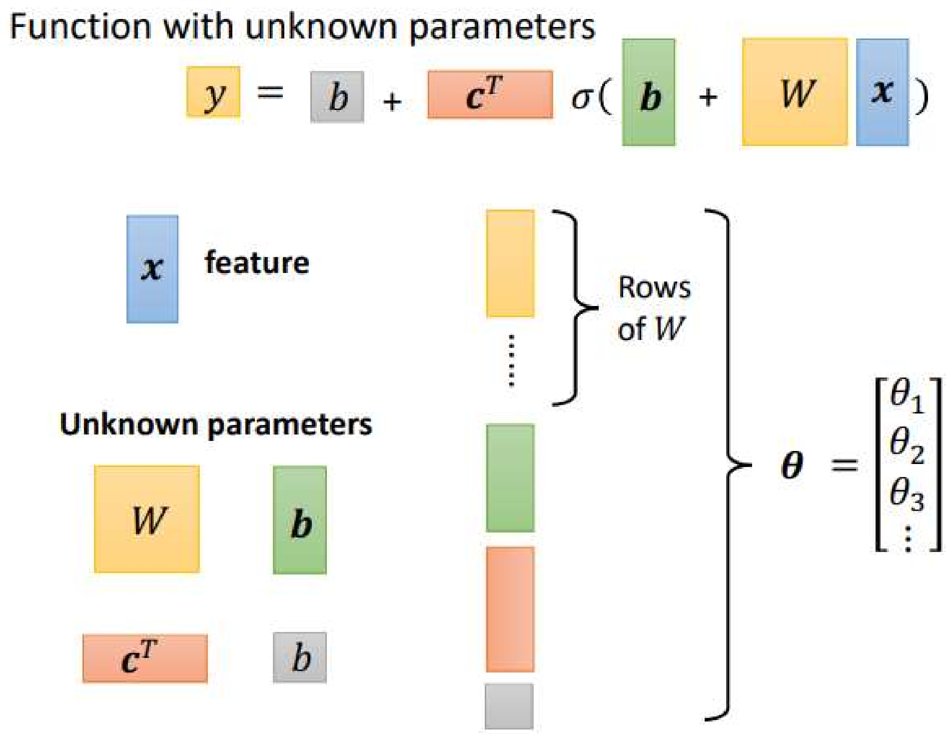
sigmoid 輸出乘上 c_i 加上 b，$a_1c_1 + a_2c_2 + a_3c_3 + b = y$
式子簡寫為 $y = b + c^T a$



整個簡式連接起來

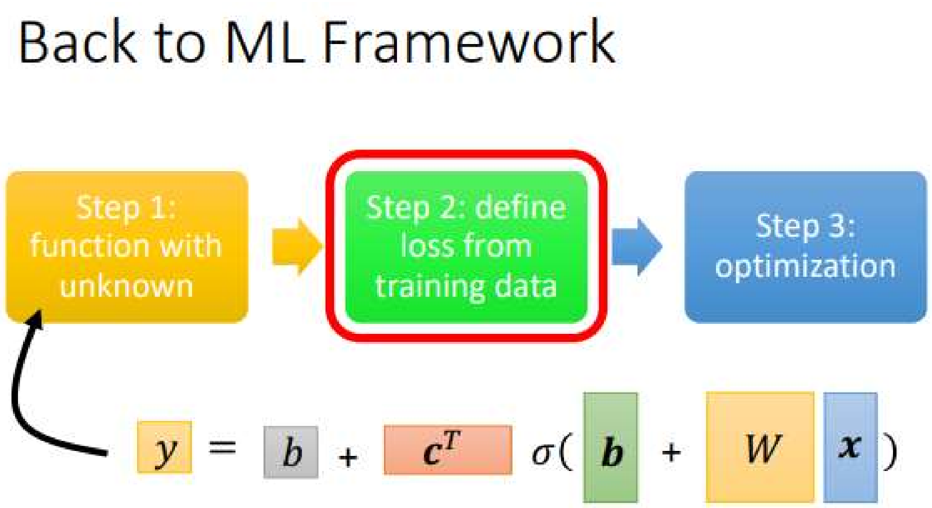


完整簡式 : $y = b + c^T\sigma(b + Wx)$

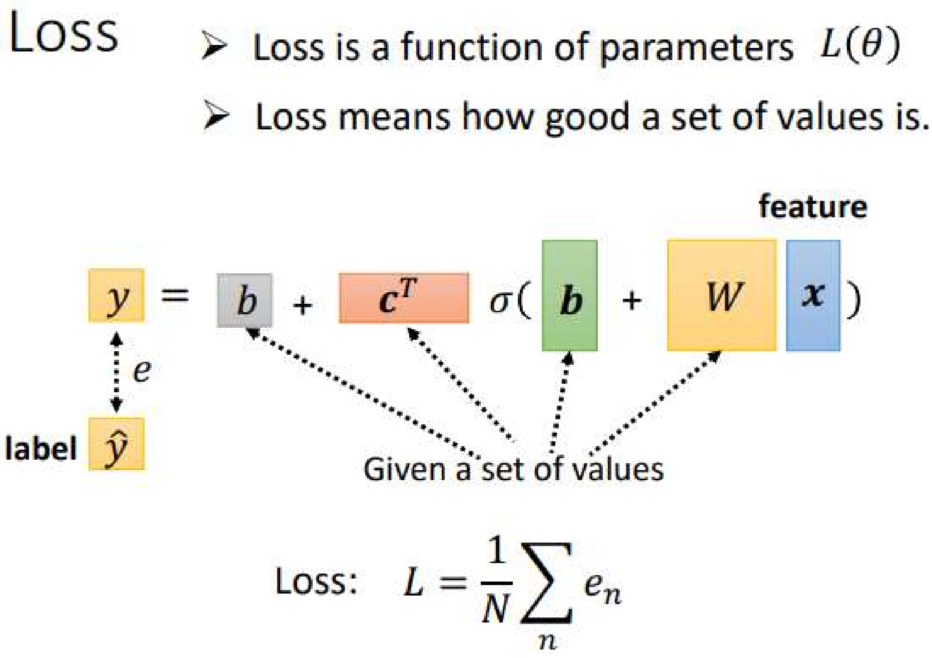


把 W, b, c^T, b 拉直，當成一個長的向量 θ

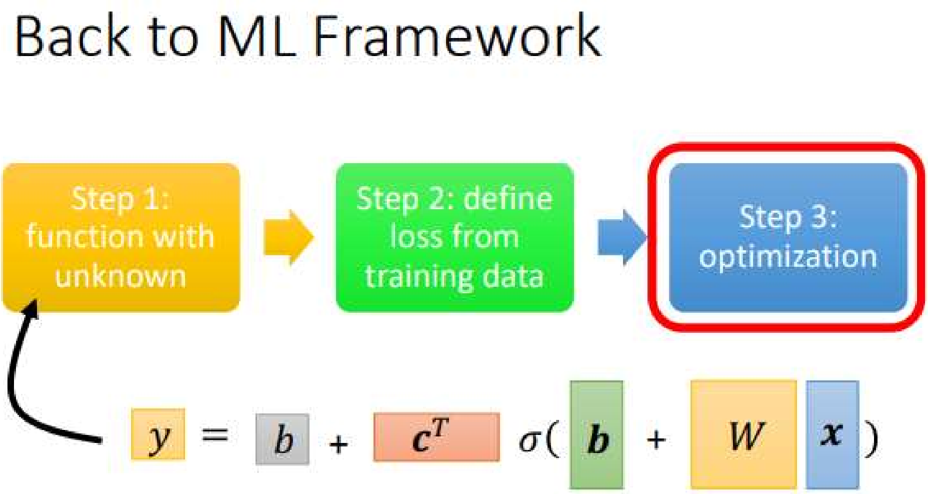
$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{bmatrix}$$



改寫了機器學習的第一步，重新定了一個有未知參數的 function



- 因為現在未知的參數很多，所以 Loss function 用 $L(\theta)$ 表示，用 θ 代表所有未知的參數
- 帶入一個 feature x，輸出 y 去與 label $\hat{y}$ 做比較，得到一個誤差值
- 將所有的誤差值加起來做平均，得到 Loss



接著，第三個 optimization 與之前也沒什麼不同，即使換了一個模型，optimization 的演算法還是 gradient descent

Optimization of New Model

Optimization of New Model

$$\boldsymbol{\theta}^* = arg \min_{\boldsymbol{\theta}} L \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{bmatrix}$$

➢ (Randomly) Pick initial values $\boldsymbol{\theta}^0$

$$\underset{\text{gradient}}{\boldsymbol{g}} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \frac{\partial L}{\partial \theta_2}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix} \qquad \begin{bmatrix} \theta_1^1 \\ \theta_2^1 \\ \vdots \end{bmatrix} \leftarrow \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \\ \vdots \end{bmatrix} - \begin{bmatrix} \eta \frac{\partial L}{\partial \theta_1}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \eta \frac{\partial L}{\partial \theta_2}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix}$$

$$\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^0) \qquad \boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$$

- 找一組 θ 可以讓 L 越小越好，得到最小 L 的 θ 稱為 θ^*
- 找一個初始的 θ^0，去對每一個參數計算對 L 微分
- 每一個微分結果集合起來，寫成一個向量，用 g 來表示，稱為 gradient
- 式子可以寫成 $g = \nabla L(\theta^0)$
- 接著去更新參數，得到 θ^1

Optimization of New Model

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

➢ (Randomly) Pick initial values $\boldsymbol{\theta}^0$

➢ Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^0)$

$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$$

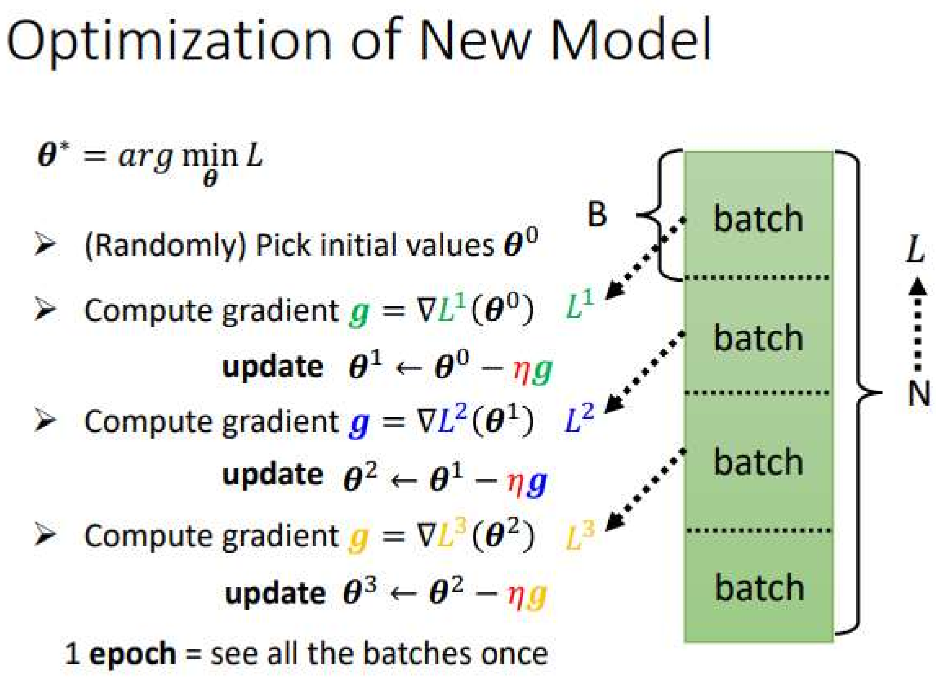
➢ Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^1)$

$$\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}$$

➢ Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^2)$

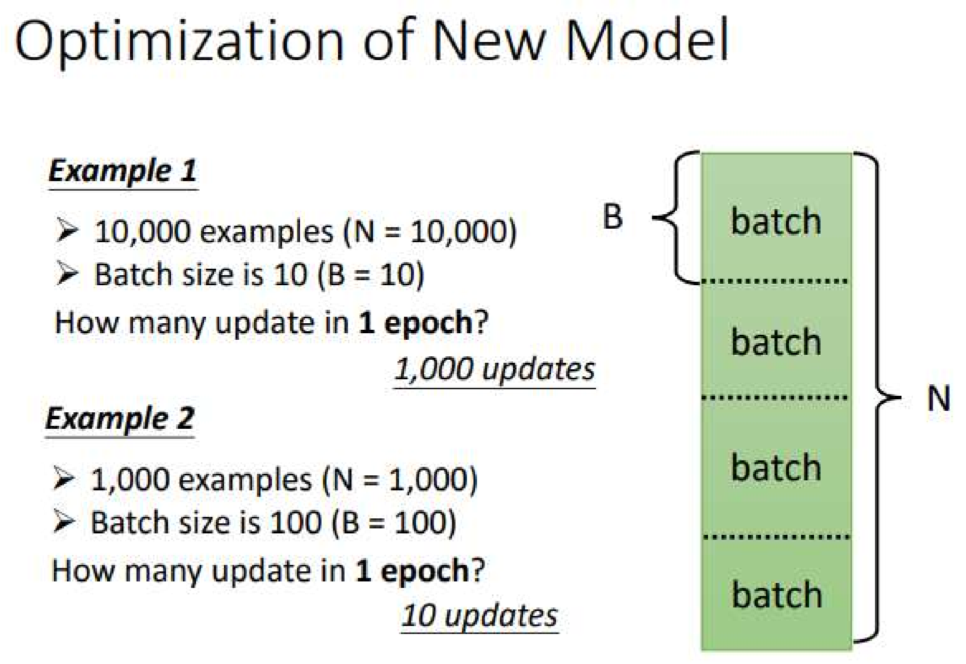
$$\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}$$

- 取一個初始 θ^0，算 gradient
- 根據 gradient，把 θ^0 更新成 θ^1
- 然後再算一次 gradient，把 θ^1 更新成 θ^2
- 然後再算一次 gradient，把 θ^2 更新成 θ^3
- 重複這個動作，直到不想做

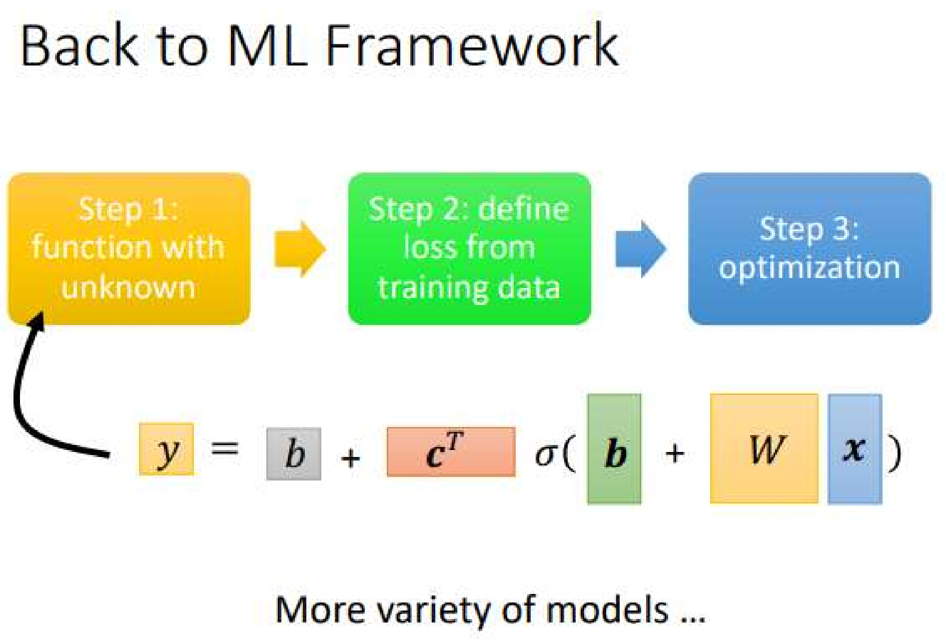


- 實際在做 gradient descent 時，會把 N 筆資料分成多個 batch，隨機分每個 batch 有 B 筆資料
- 原本是拿 N 的每筆資料算一個 Loss L，現在只拿一個 batch 的 data 出來算一個 Loss L^1

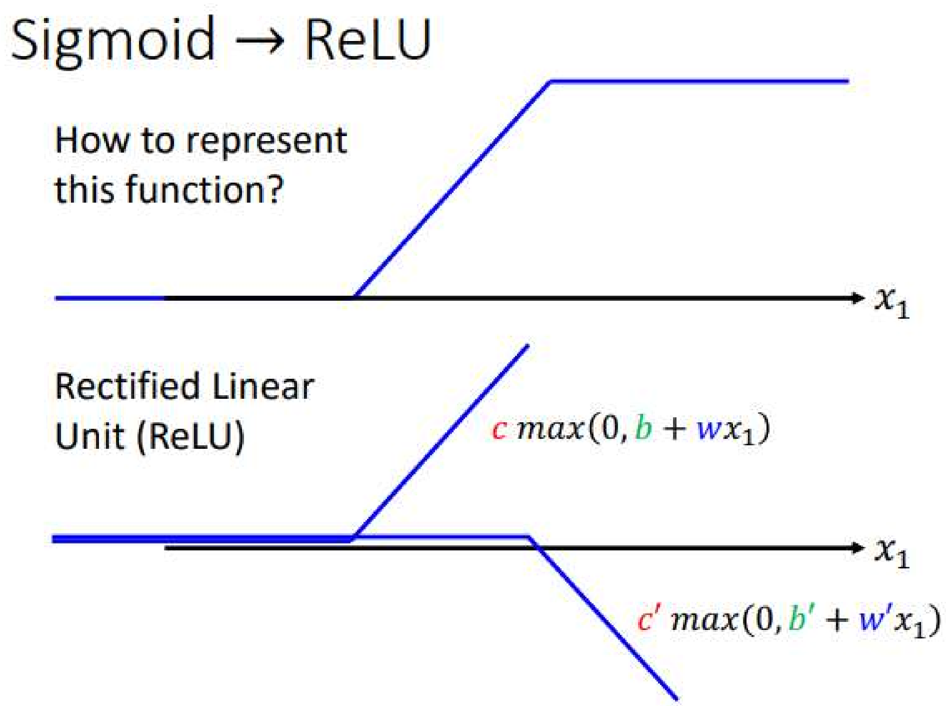
- 先選一個 batch 去算 L^1 ，根據 L^1 去算 gradient，再用 gradient 更新參數
- 再選一個 batch 去算 L^2 ，根據 L^2 去算 gradient，再用 gradient 更新參數
- 再選一個 batch 去算 L^3 ，根據 L^3 去算 gradient，再用 gradient 更新參數
- 每更新一次參數，稱為 1 update
- 把所有的 batch 都看過一次，稱為 1 epoch



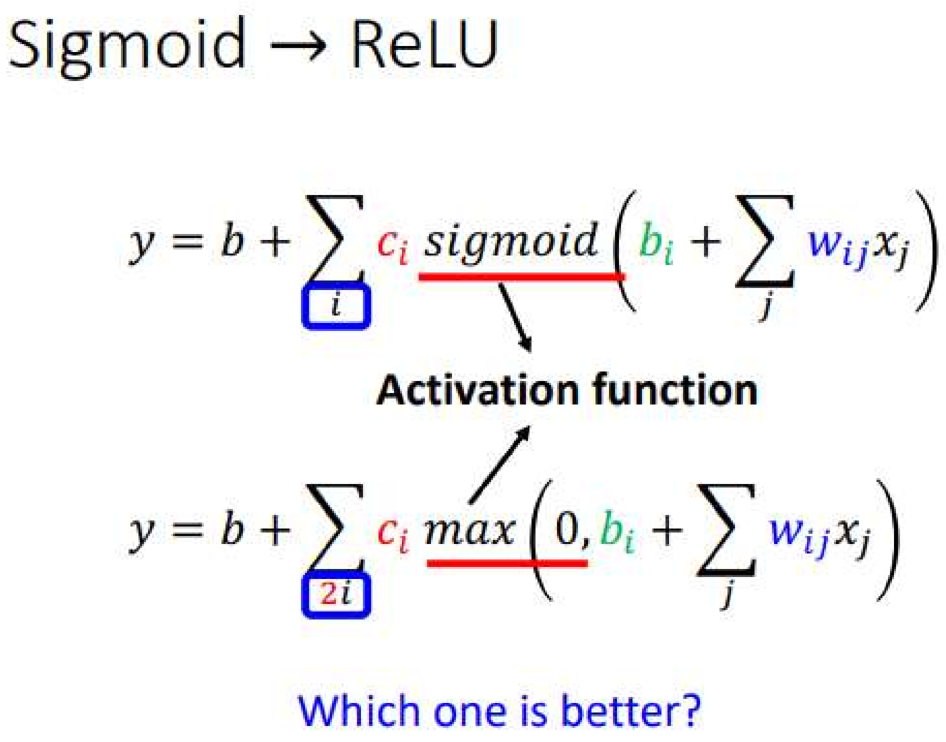
1 epoch 的 update 次數，取決於他的 batch size 有多大



還可以對模型做更多的變形



上面的 Hard Sigmoid Function 可以看成是兩個 Rectified Linear Unit (ReLU) 的加總



- 把 Sigmoid 換成 ReLU，因為 2 個 ReLU 才能組成一個 Sigmoid，所以 Sigma i 變成 $2i$
- Sigmoid 跟 ReLU 是最常見的 Activation function

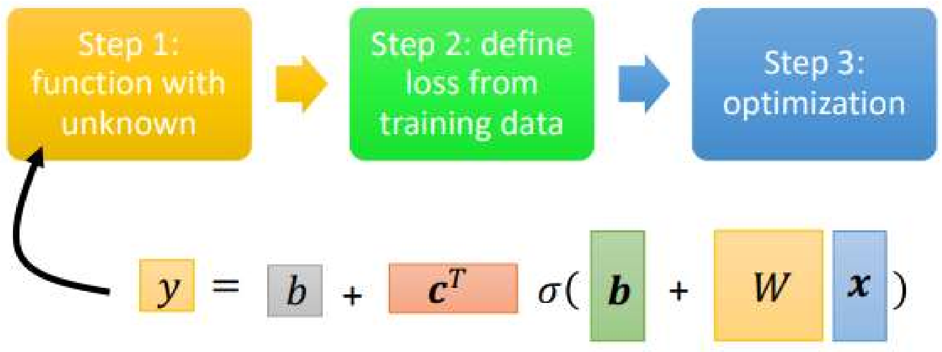
Experimental Results

$$y = b + \sum_{2i} c_i \, max\left(0, b_i + \sum_j w_{ij} x_j\right)$$

	linear	10 ReLU	100 ReLU	1000 ReLU
2017 – 2020	0.32k	0.32k	0.28k	0.27k
2021	0.46k	0.45k	0.43k	0.43k

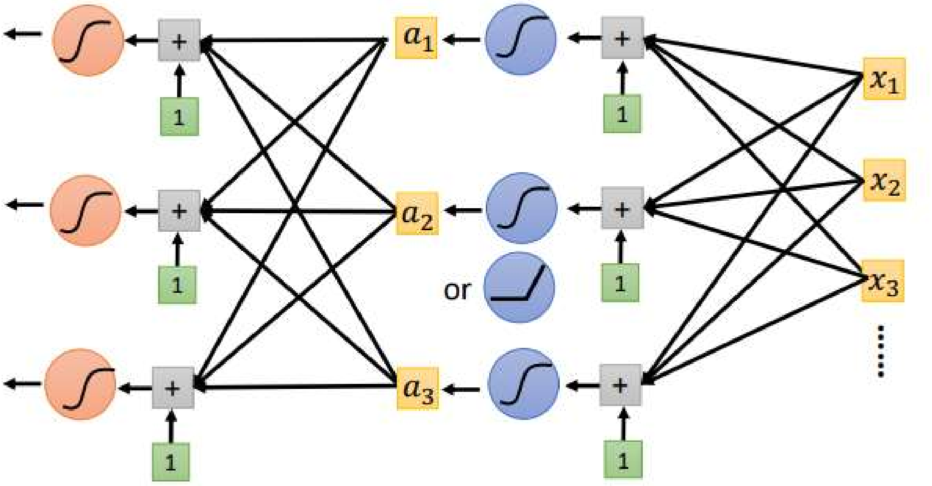
- 如果用 linear model，訓練資料上的 Loss 是 0.32k，沒看過的資料上 Loss 是 0.46k
- 如果用 100 ReLU，訓練資料上的 Loss 是 0.28k，沒看過的資料上 Loss 是 0.43k，因為 100 ReLU 可以製造比較複雜的曲線，可以製造有 100 個折線的 piecewise function

Back to ML Framework



Even more variety of models ...

還可以繼續改模型



$$\boldsymbol{a}' = \sigma(\boldsymbol{b}' + W'\boldsymbol{a}) \qquad \boldsymbol{a} = \sigma(\boldsymbol{b} + W\boldsymbol{x})$$

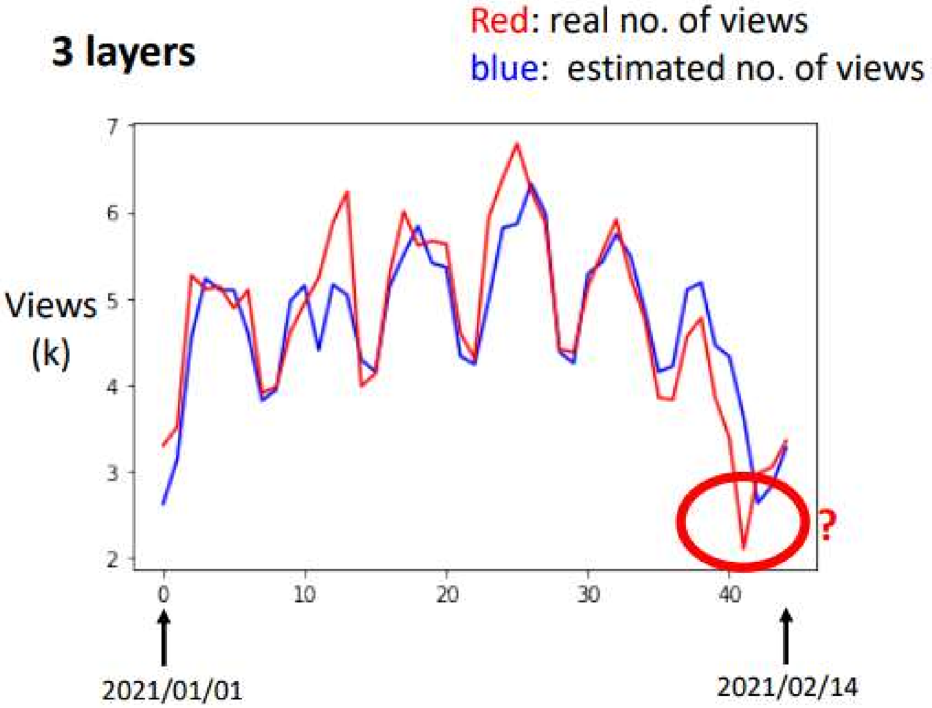
可以把同樣的事情重複多做幾次

Experimental Results

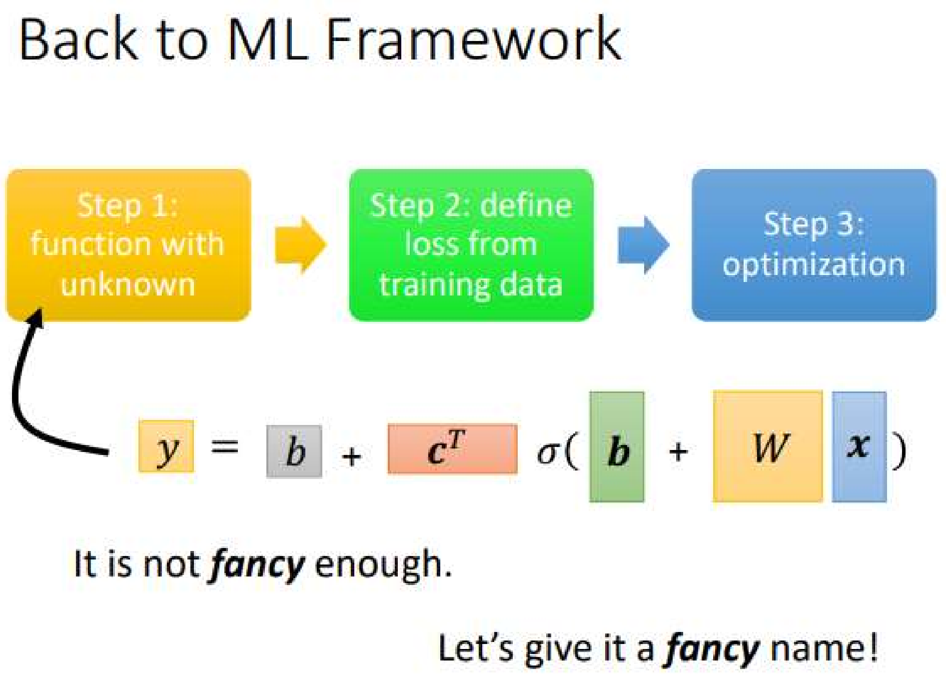
- Loss for multiple hidden layers
  - 100 ReLU for each layer
  - input features are the no. of views in the past 56 days

	1 layer	2 layer	3 layer
2017 – 2020	0.28k	0.18k	0.14k
2021	0.43k	0.39k	0.38k

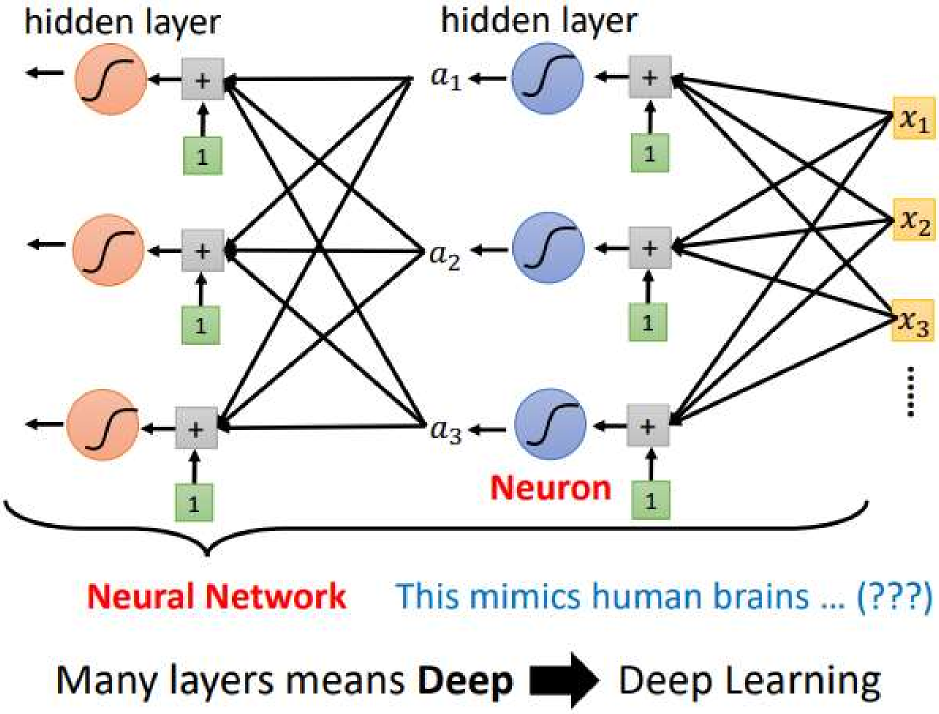
每次都做 100 個 ReLU



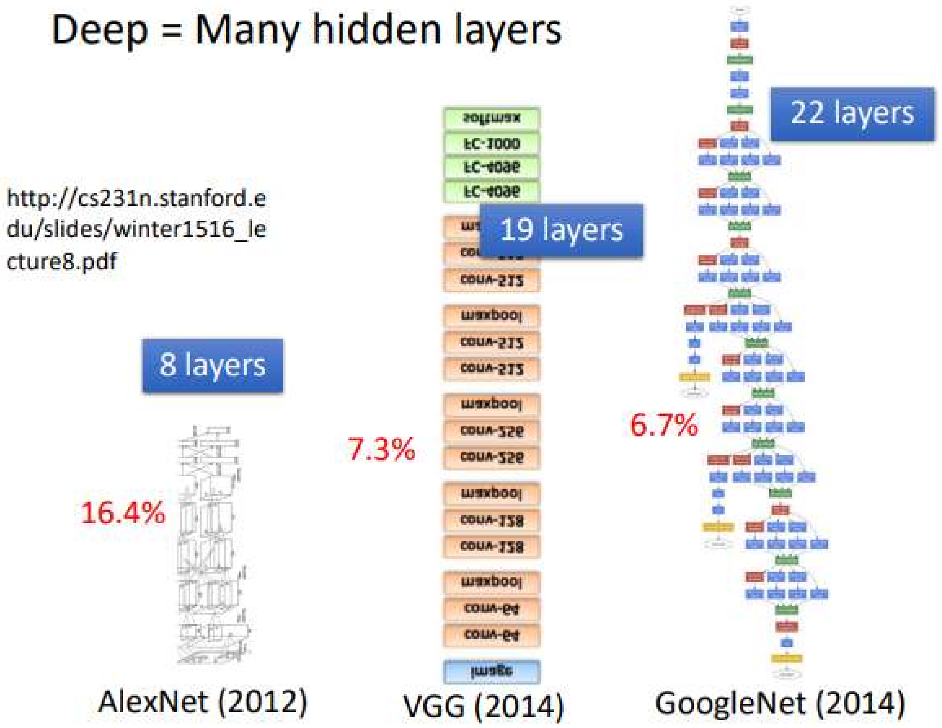
做 3 層 ReLU 的結果，最後出現高估觀看人數的結果
因為機器不知道除夕夜過年

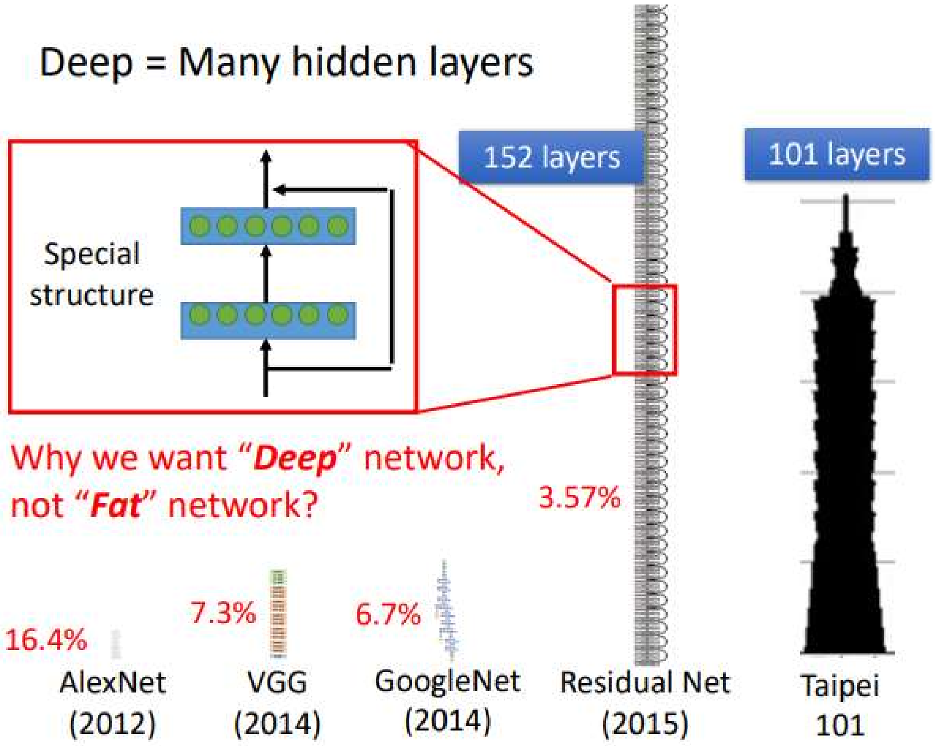


一個模型需要一個好的名字

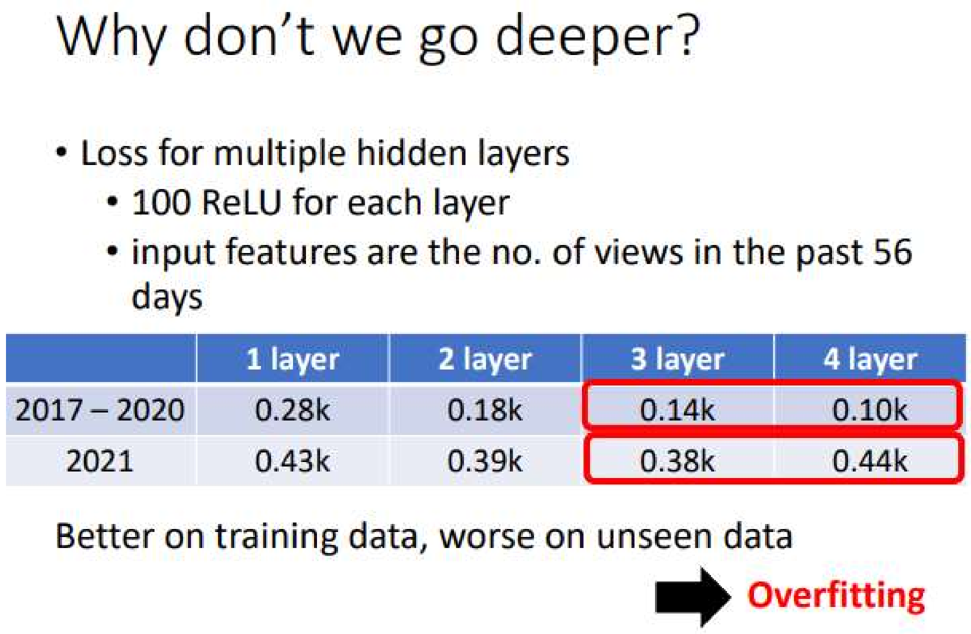


Sigmoid 或是 ReLU 叫做 Neuron
很多個 neuron 叫做 Neural Network
有很多 hidden layer 稱為 **Deep Learning**

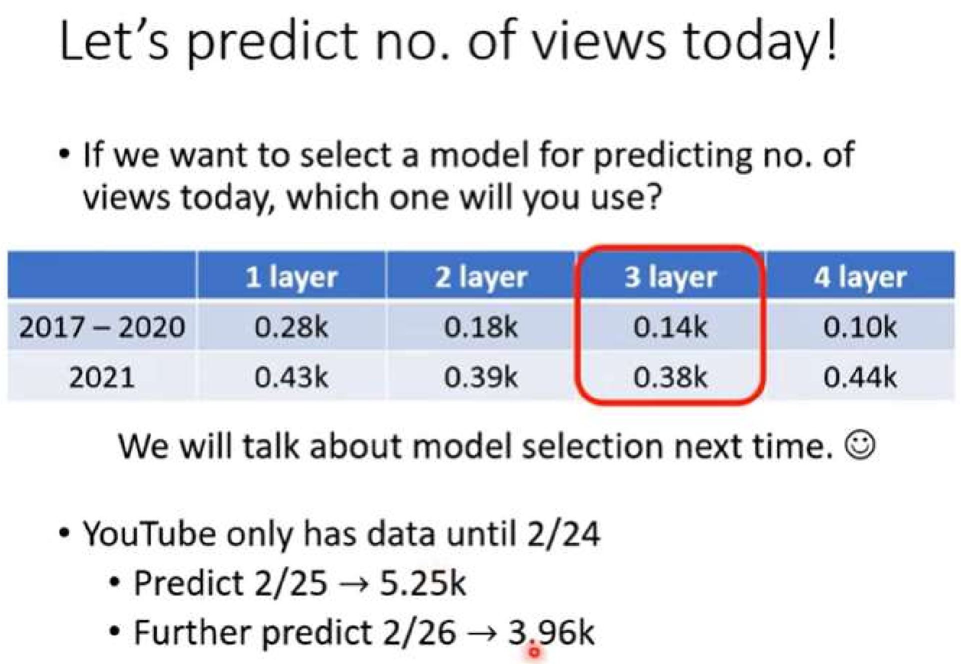




layer 越來越多
ReLU、Sigmoid 排成一排，會得到一個肥胖的 Neural Network



做 4 層 ReLU 之後，在沒看過的資料上 Loss 反而變得不好了
這種訓練資料結果改善，但測試資料變差的狀況，稱為 **Overfitting**



如果要預測今天的觀看人數，要選 3 層
我們在意的是沒有看過的資料的預測結果

延伸教材：

[1. Brief Introduction of Deep Learning (https://www.youtube.com/watch?v=Dr-WRlEFefw)](https://www.youtube.com/watch?v=Dr-WRlEFefw)
[2. Backpropagation (https://www.youtube.com/watch?v=ibJpTrp5mcE)](https://www.youtube.com/watch?v=ibJpTrp5mcE)

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

tags: `2022` `李宏毅_機器學習`