

Introduction of Machine / Deep Learning

Hung-yi Lee 李宏毅

Machine Learning ≈ Looking for Function

- Speech Recognition

$$f(\text{[Speech Waveform]}) = \text{"How are you"}$$

- Image Recognition

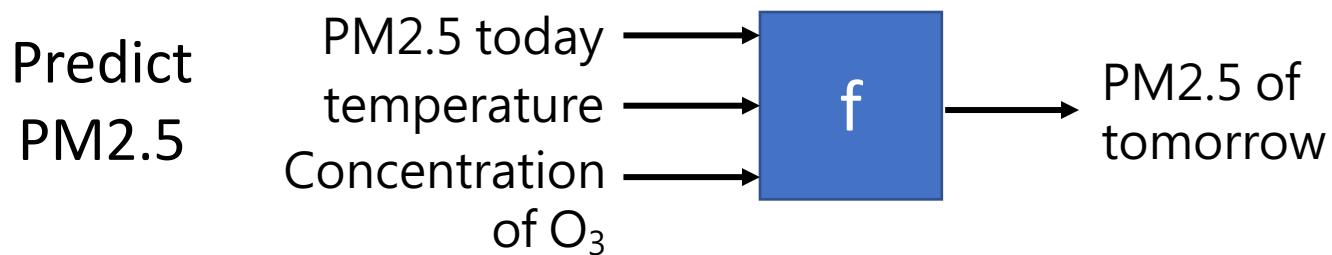
$$f(\text{[Orange Kitten Image]}) = \text{"Cat"}$$

- Playing Go

$$f(\text{[Go Board State]}) = \text{"5-5" (next move)}$$

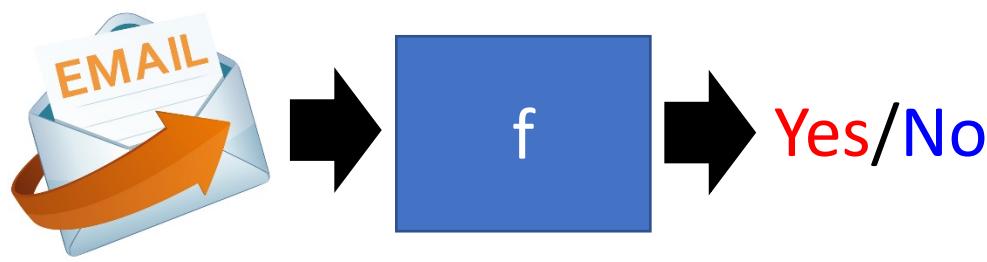
Different types of Functions

Regression: The function outputs a scalar.



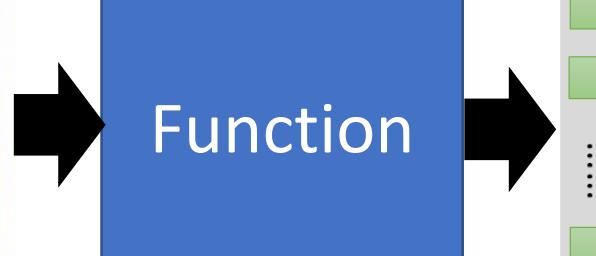
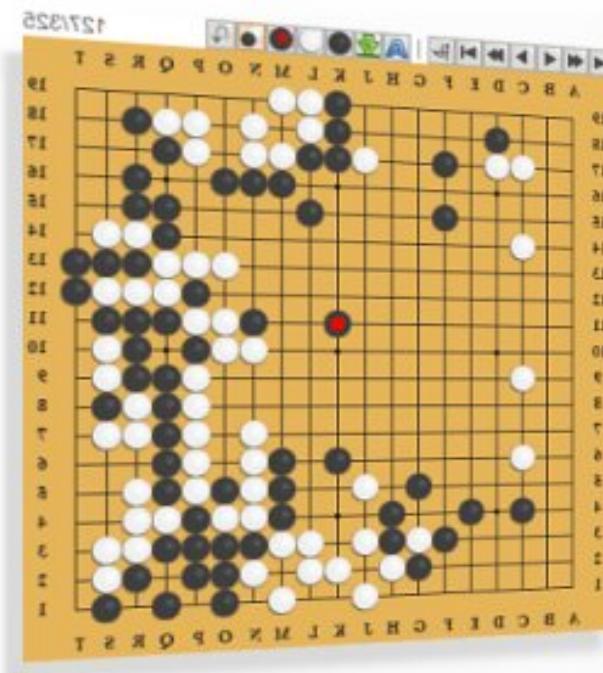
Classification: Given options (**classes**), the function outputs the correct one.

Spam filtering



Different types of Functions

Classification: Given options (**classes**), the function outputs the correct one.



Each position
is a class
(19 x 19 classes)

a position on
the board

Next move

Playing GO

共分三類

1. regression
2. classify
3. structured learning

Structured Learning

create something with
structure (image, document)



Regression,
Classification



YouTube



Kotlin



How to find a function?
A Case Study

YouTube Channel

The screenshot shows the YouTube channel page for 'NTU SPEECH LAB HUNG-YI LEE'. The header features two large, abstract black ink paintings on a light background. To the right, the channel name is displayed in a white box with a black border. Below the header, there's a profile picture of Hung-yi Lee, her name, and the fact that she has 7.24 million subscribers. There are also 'Subscribe' and 'Manage Videos' buttons. The main content area shows a video thumbnail for 'The Next Step for Machine Learning' with the Chinese subtitle '機器學習的下一步' and the English subtitle 'The Next Step for Machine Learning'. The video has 131,510 views and was uploaded 2 years ago. Below the video, there's a link to the full video and a note about past content.

NTU SPEECH LAB
HUNG-YI LEE

Hung-yi Lee
7.24 萬 位訂閱者

首頁 影片 播放清單 社群 頻道 簡介 搜尋

The Next Step for Machine Learning

機器學習的下一步

在真實的應用中還少了甚麼

觀看次數：131,510次 · 2 年前

這是「機器學習」2019 年春季班的上課錄影，只有之前在同一門課沒有講過的新內容會被上傳到 YouTube 頻道上。請見以下播放清單：<https://www.youtube.com/watch?v=XnyM3...>

這門課過去的內容請見以下播放清單：
<https://www.youtube.com/watch?v=CXgbe...>

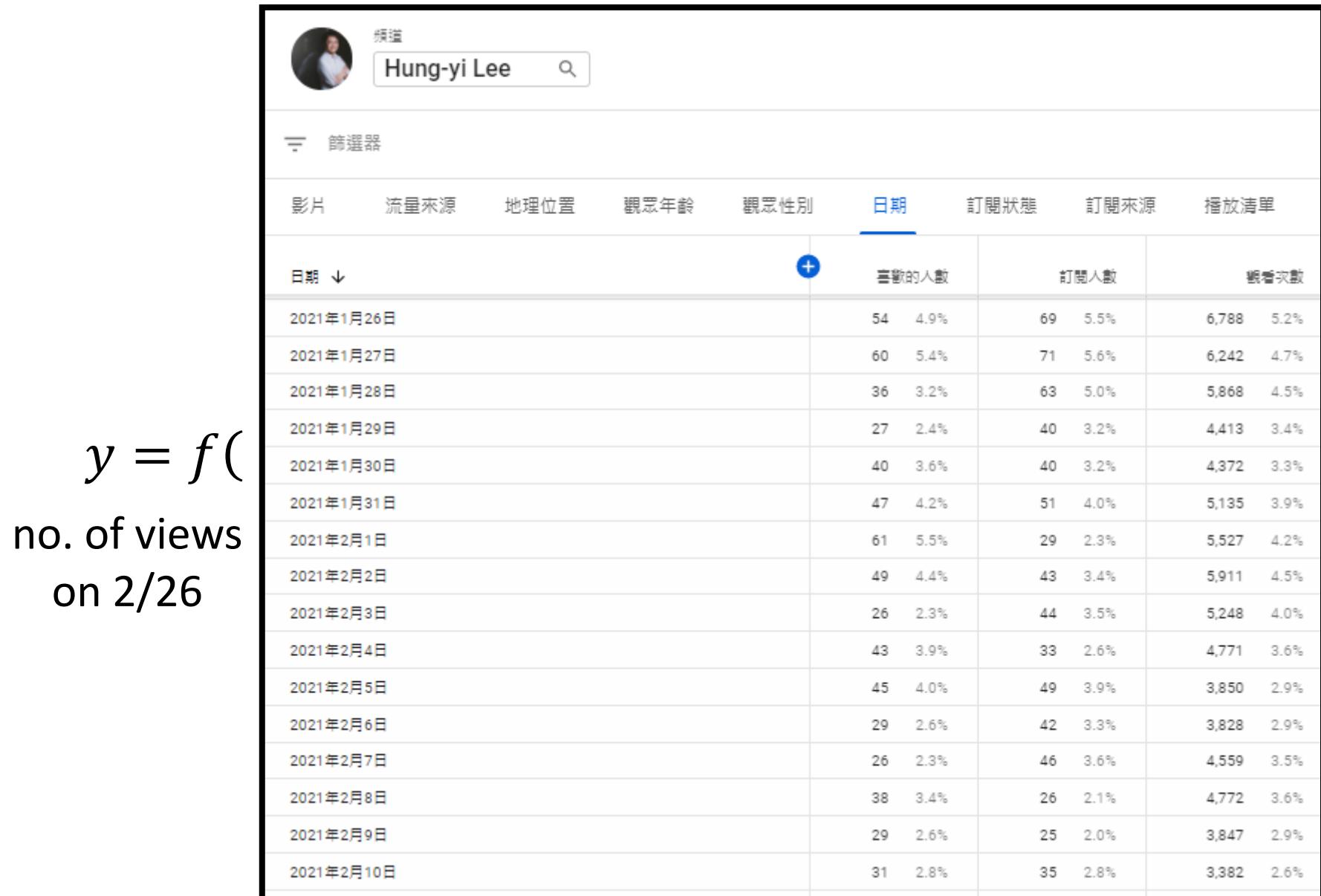
連結網址：<http://research.csie.ntu.edu.tw/~thekkk/> | RSS | 布景

顯示較多內容

<https://www.youtube.com/c/HungyiLeeNTU>

The function we want to find ...

$y = f($
no. of views
on 2/26



A screenshot of a YouTube analytics dashboard for a channel named 'Hung-yi Lee'. The dashboard shows a table of data with the following columns: 日期 (Date), 喜歡的人數 (Likes), 訂閱人數 (Subscribers), and 觀看次數 (Views). The data spans from January 26, 2021, to February 10, 2021. The '日期' column is sorted by date. The 'Likes' column includes percentages. The 'Subscribers' column includes percentages. The 'Views' column includes percentages.

日期	喜歡的人數	訂閱人數	觀看次數
2021年1月26日	54 4.9%	69 5.5%	6,788 5.2%
2021年1月27日	60 5.4%	71 5.6%	6,242 4.7%
2021年1月28日	36 3.2%	63 5.0%	5,868 4.5%
2021年1月29日	27 2.4%	40 3.2%	4,413 3.4%
2021年1月30日	40 3.6%	40 3.2%	4,372 3.3%
2021年1月31日	47 4.2%	51 4.0%	5,135 3.9%
2021年2月1日	61 5.5%	29 2.3%	5,527 4.2%
2021年2月2日	49 4.4%	43 3.4%	5,911 4.5%
2021年2月3日	26 2.3%	44 3.5%	5,248 4.0%
2021年2月4日	43 3.9%	33 2.6%	4,771 3.6%
2021年2月5日	45 4.0%	49 3.9%	3,850 2.9%
2021年2月6日	29 2.6%	42 3.3%	3,828 2.9%
2021年2月7日	26 2.3%	46 3.6%	4,559 3.5%
2021年2月8日	38 3.4%	26 2.1%	4,772 3.6%
2021年2月9日	29 2.6%	25 2.0%	3,847 2.9%
2021年2月10日	31 2.8%	35 2.8%	3,382 2.6%

1. Function with Unknown Parameters

$$y = f($$



日期	新增的影片數	喜歡的人數	使用的訂閱人數	總共次數	被喜歡次數	被喜歡小時 (小時)	平均被喜歡程度
總計	199	17,022	26,011	27,602,732	2,066,634	268,778.0	7:48
2020年1月1日	-	16 0.1%	52 0.2%	57,093	3,977 0.2%	565.6 0.2%	8:32
2020年1月2日	-	33 0.2%	58 0.2%	56,204	4,214 0.2%	589.8 0.2%	8:23
2020年1月3日	-	24 0.1%	89 0.3%	53,321	3,284 0.2%	457.4 0.2%	8:20
2020年1月4日	1 0.5%	27 0.2%	66 0.3%	51,599	3,559 0.2%	483.5 0.2%	8:09
2020年1月5日	-	35 0.2%	85 0.3%	63,001	4,677 0.2%	596.4 0.2%	7:39
2020年1月6日	-	31 0.2%	69 0.3%	60,175	4,682 0.2%	642.0 0.2%	8:13
2020年1月7日	-	40 0.2%	70 0.3%	61,858	4,695 0.2%	618.4 0.2%	7:54
2020年1月8日	-	39 0.2%	59 0.2%	59,900	4,785 0.2%	646.7 0.2%	8:06
2020年1月9日	-	28 0.2%	64 0.3%	51,988	4,911 0.2%	670.9 0.3%	8:11
2020年1月10日	-	17 0.1%	51 0.2%	49,631	3,066 0.2%	372.0 0.1%	7:16
2020年1月11日	-	12 0.1%	54 0.2%	31,168	2,898 0.1%	369.5 0.1%	7:38
2020年1月12日	-	40 0.2%	169 0.7%	53,964	4,477 0.2%	572.9 0.2%	7:40
2020年1月13日	-	29 0.2%	75 0.3%	61,043	5,017 0.2%	661.4 0.3%	7:54
2020年1月14日	-	32 0.2%	83 0.3%	64,968	5,186 0.3%	618.3 0.3%	7:09

Model $y = b + wx_1$ based on domain knowledge

feature

y : no. of views on 2/26, x_1 : no. of views on 2/25

w and b are unknown parameters (learned from data)

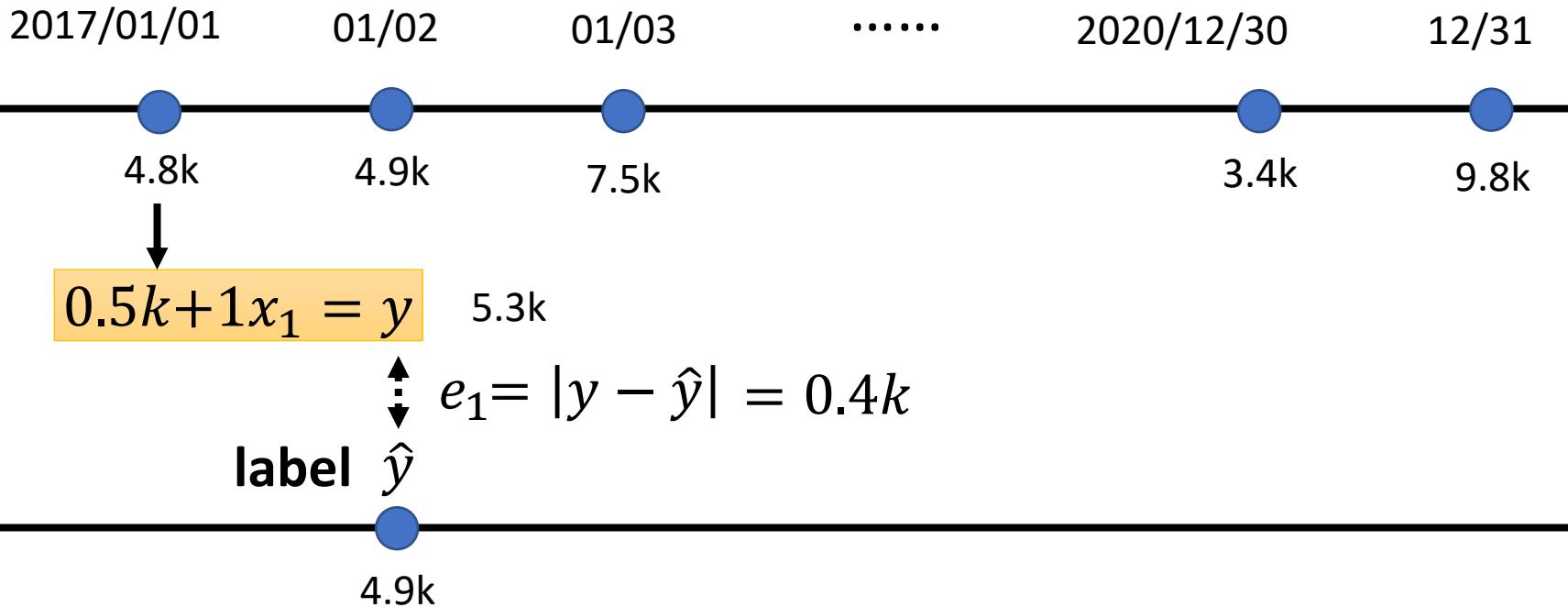
weight bias

2. Define Loss from Training Data

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

$$L(0.5k, 1) \quad y = b + wx_1 \rightarrow y = 0.5k + 1x_1 \quad \text{How good it is?}$$

Data from 2017/01/01 – 2020/12/31



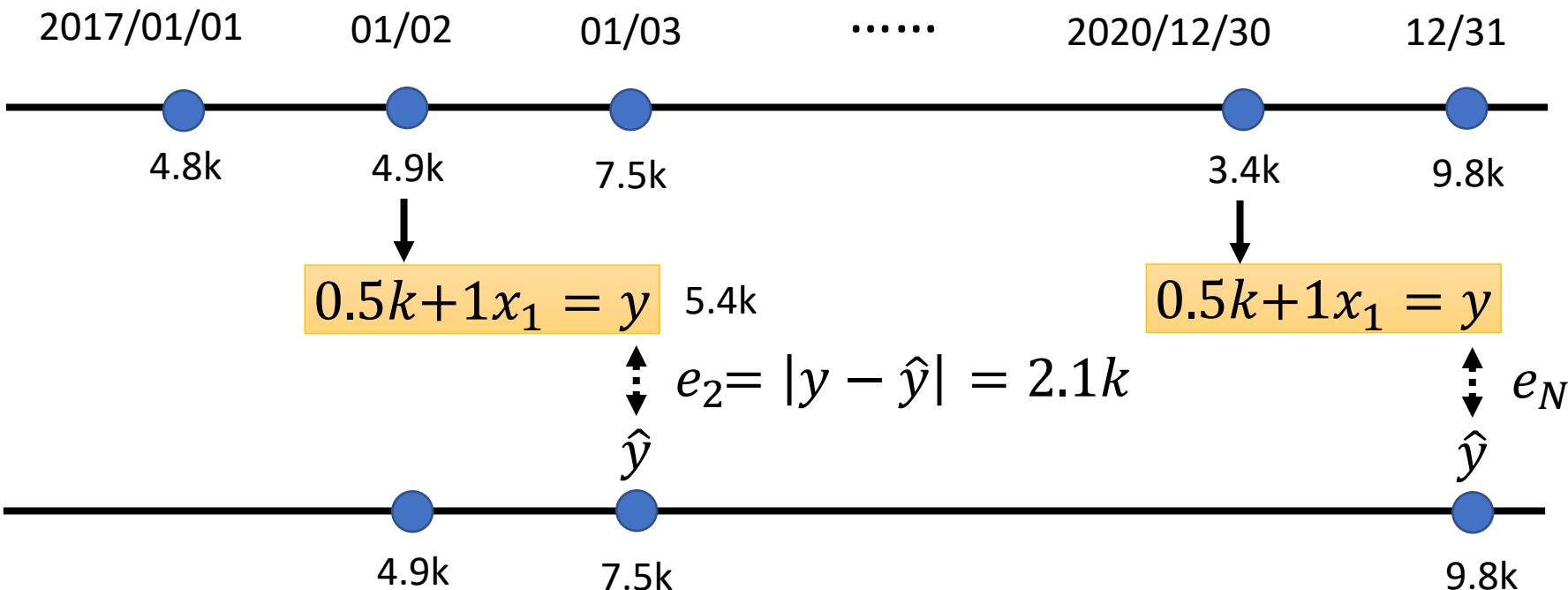
2. Define **Loss**

from Training Data

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

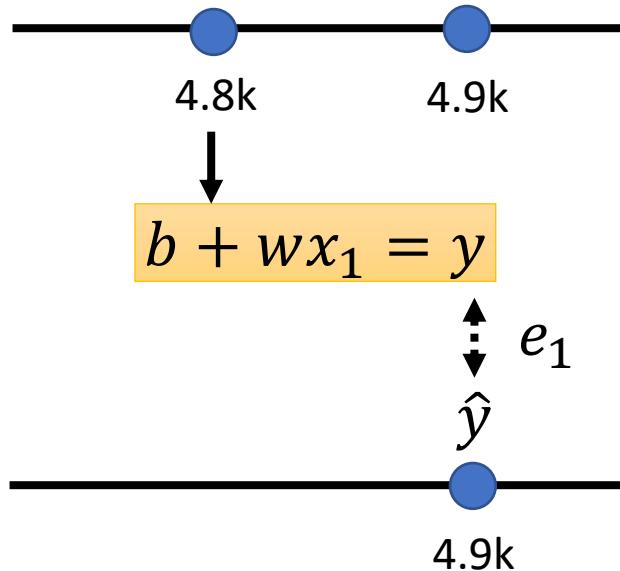
$$L(0.5k, 1) \quad y = b + wx_1 \rightarrow y = 0.5k + 1x_1 \quad \text{How good it is?}$$

Data from 2017/01/01 – 2020/12/31



2. Define Loss from Training Data

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



$$\text{Loss: } L = \frac{1}{N} \sum_n e_n$$

$$e = |y - \hat{y}| \quad L \text{ is mean absolute error (MAE)}$$

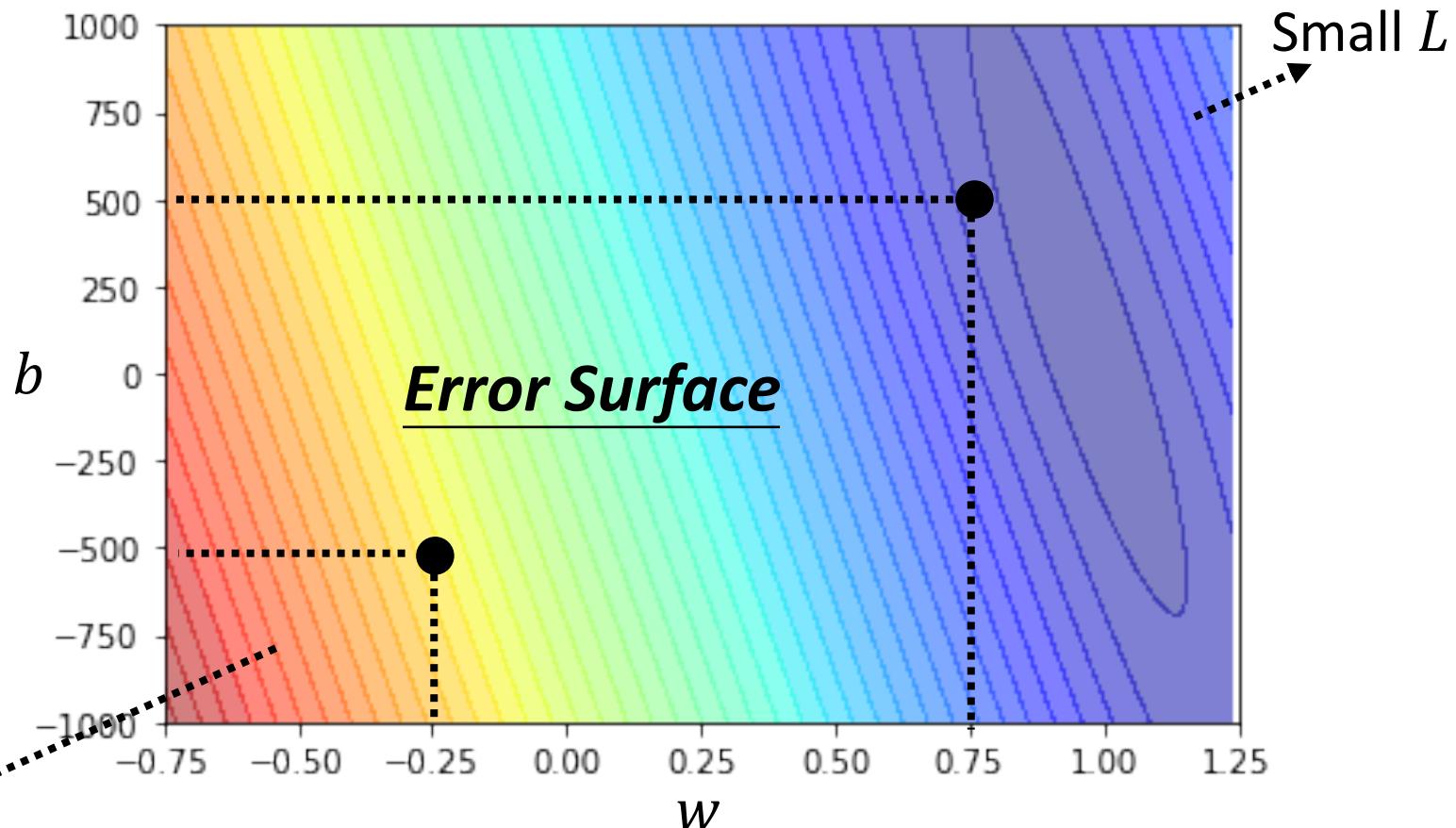
$$e = (y - \hat{y})^2 \quad L \text{ is mean square error (MSE)}$$

If y and \hat{y} are both probability distributions \rightarrow Cross-entropy

2. Define Loss from Training Data

Model $y = b + wx_1$

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

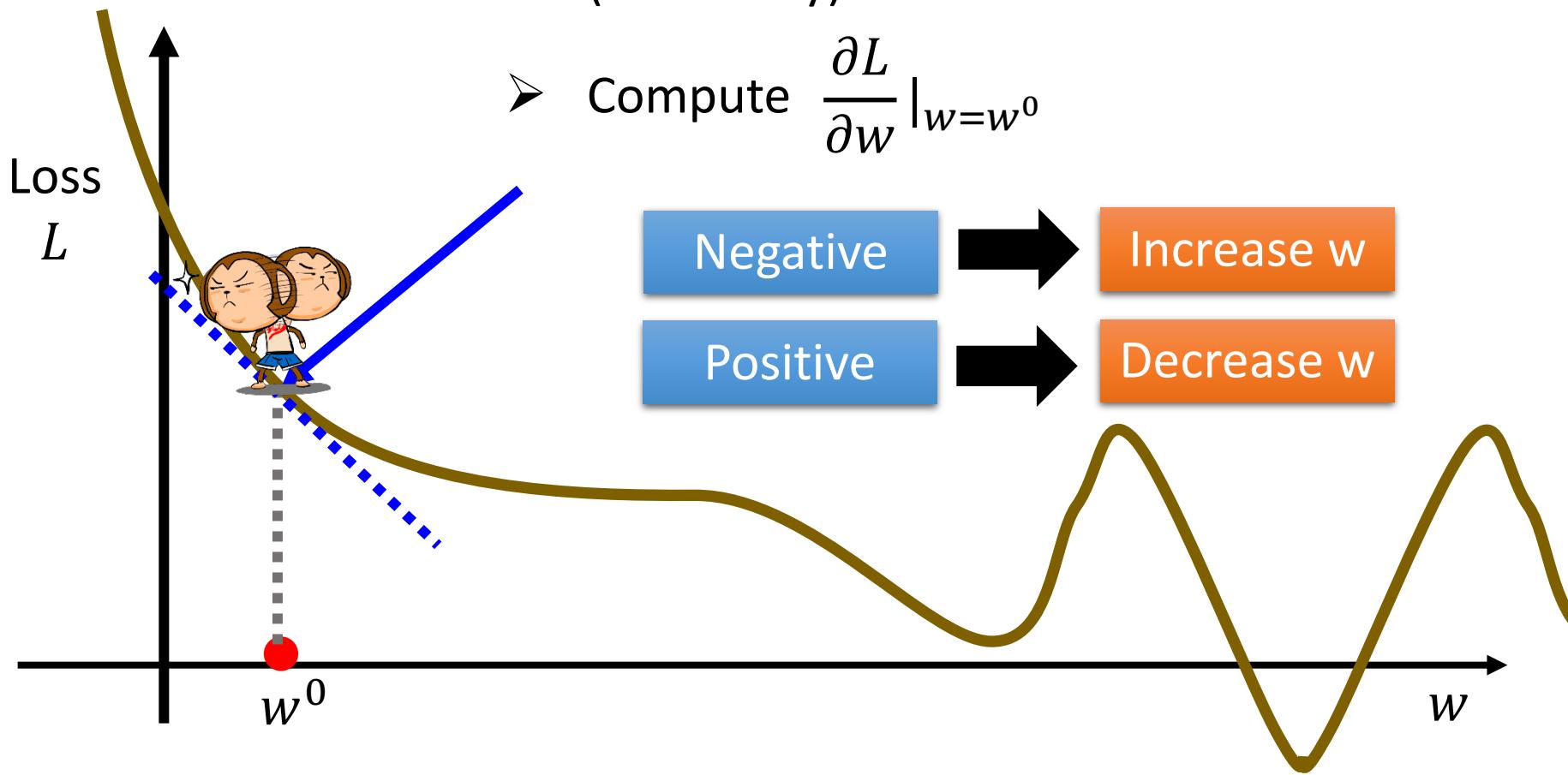


3. Optimization

$$w^* = \arg \min_w L$$

Gradient Descent

- (Randomly) Pick an initial value w^0
- Compute $\frac{\partial L}{\partial w} |_{w=w^0}$



3. Optimization

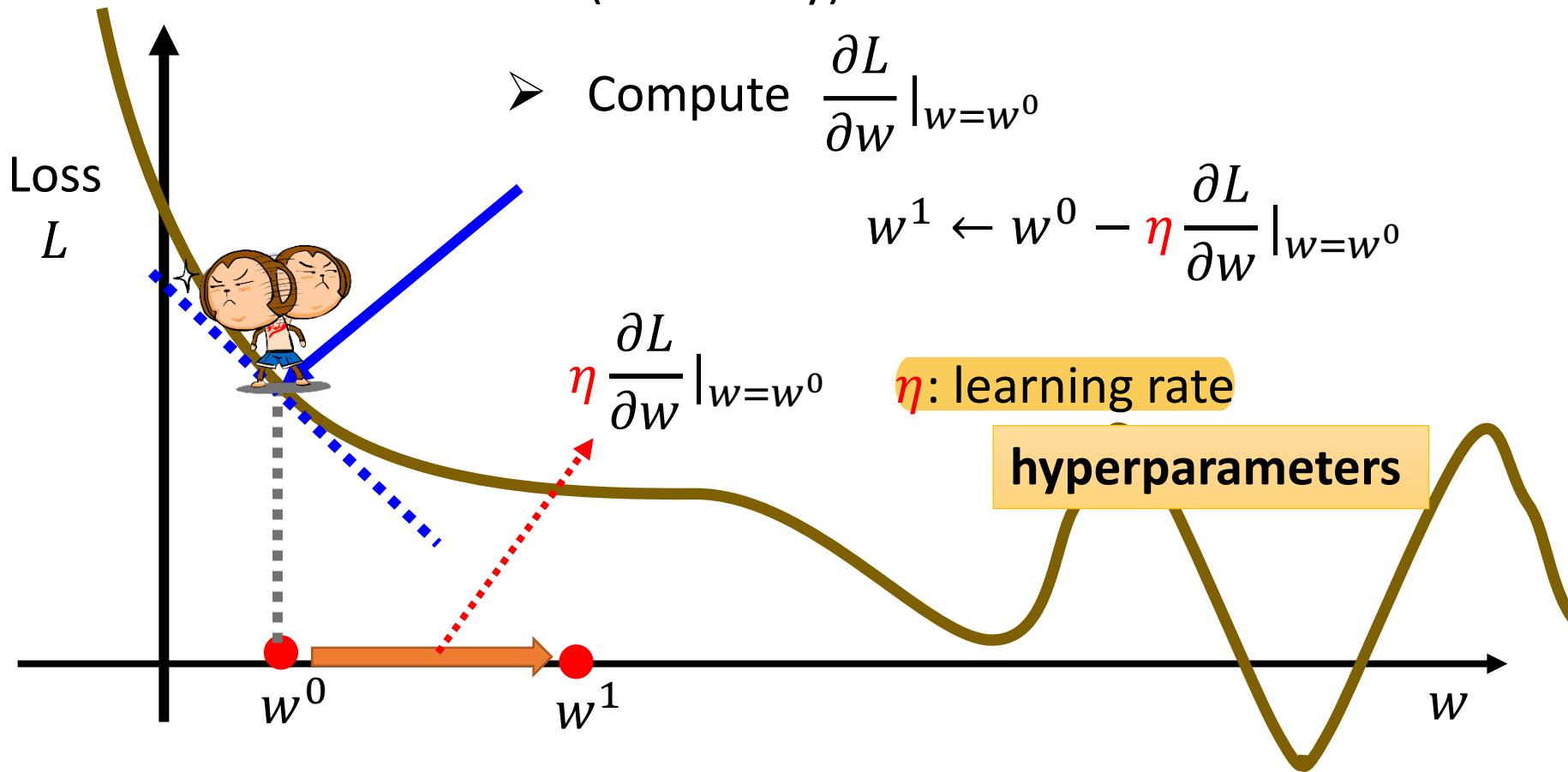
$$w^* = \arg \min_w L$$

Gradient Descent

➤ (Randomly) Pick an initial value w^0

➤ Compute $\frac{\partial L}{\partial w} |_{w=w^0}$

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



3. Optimization

$$w^* = \arg \min_w L$$

Gradient Descent

➤ (Randomly) Pick an initial value w^0

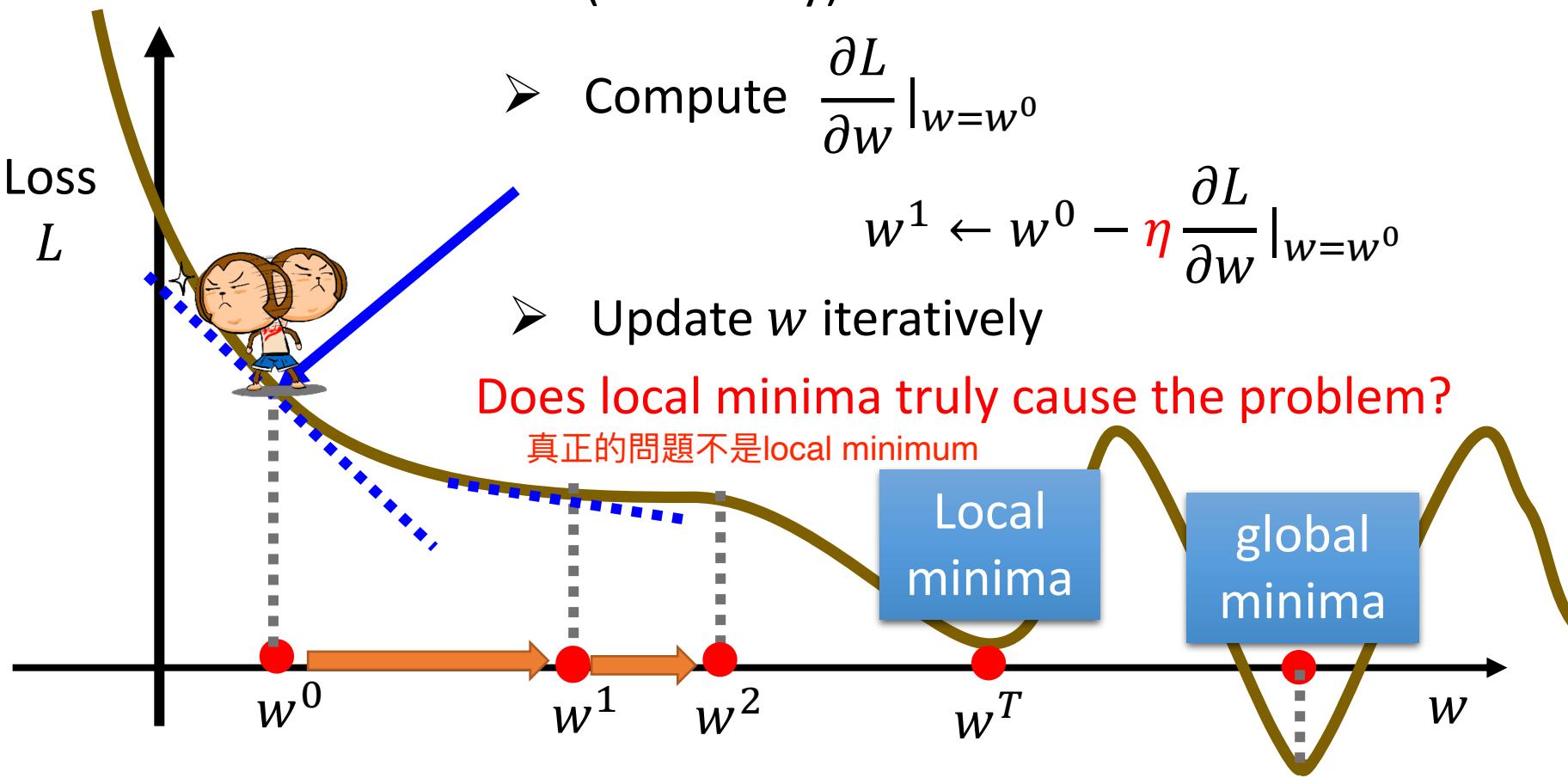
➤ Compute $\frac{\partial L}{\partial w} |_{w=w^0}$

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

➤ Update w iteratively

Does local minima truly cause the problem?

真正的問題不是local minimum



3. Optimization

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

- (Randomly) Pick initial values w^0, b^0
- Compute

$$\begin{array}{|c|}\hline \frac{\partial L}{\partial w} \Big|_{w=w^0, b=b^0} \\ \hline \frac{\partial L}{\partial b} \Big|_{w=w^0, b=b^0} \\ \hline \end{array}$$

↓

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

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

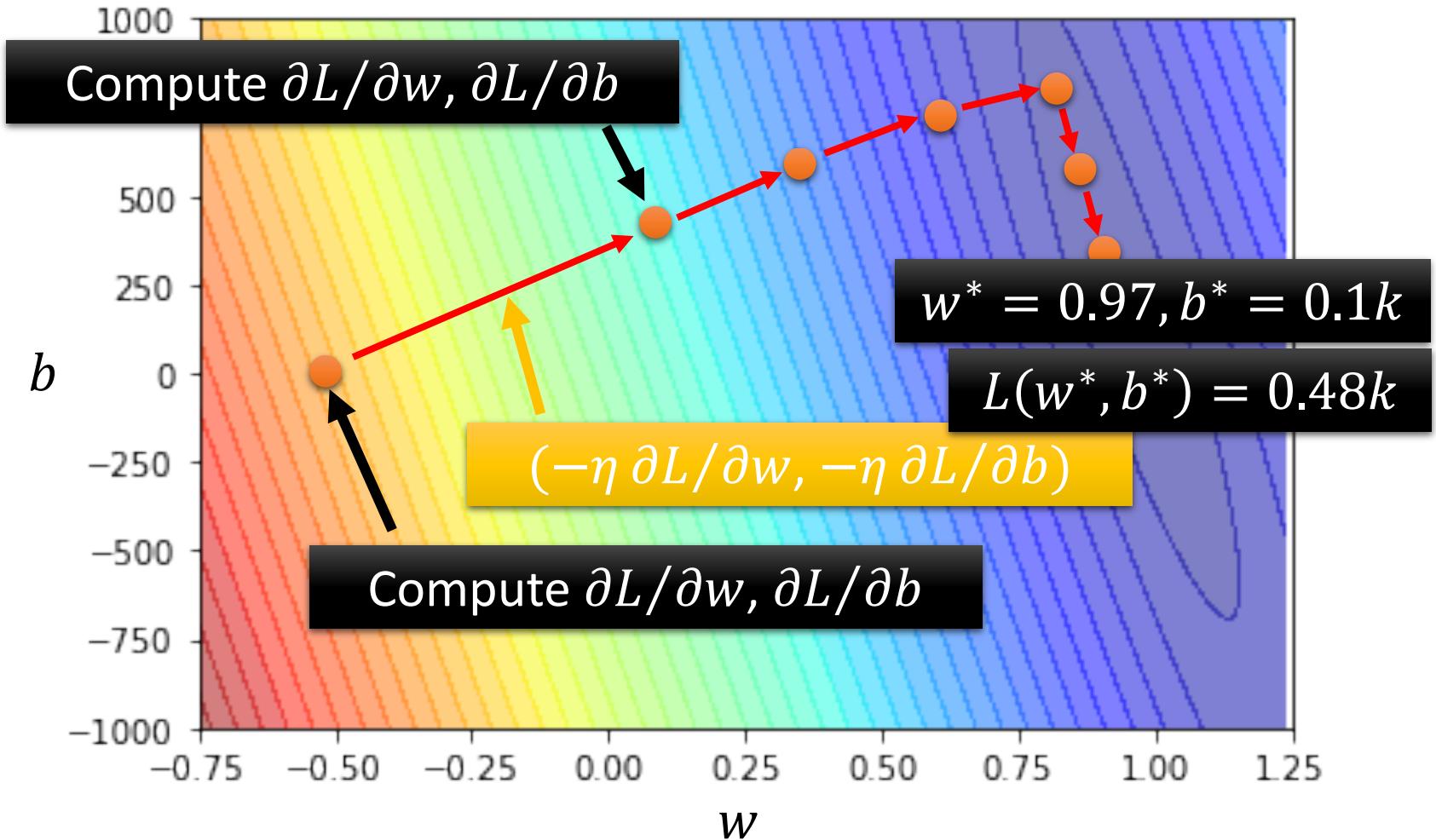
Can be done in one line in most deep learning frameworks

- Update w and b interatively

$$\text{Model } y = b + wx_1$$

3. Optimization

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



Machine Learning is so simple

$$w^* = 0.97, b^* = 0.1k$$

$$y = b + wx_1$$

$$L(w^*, b^*) = 0.48k$$

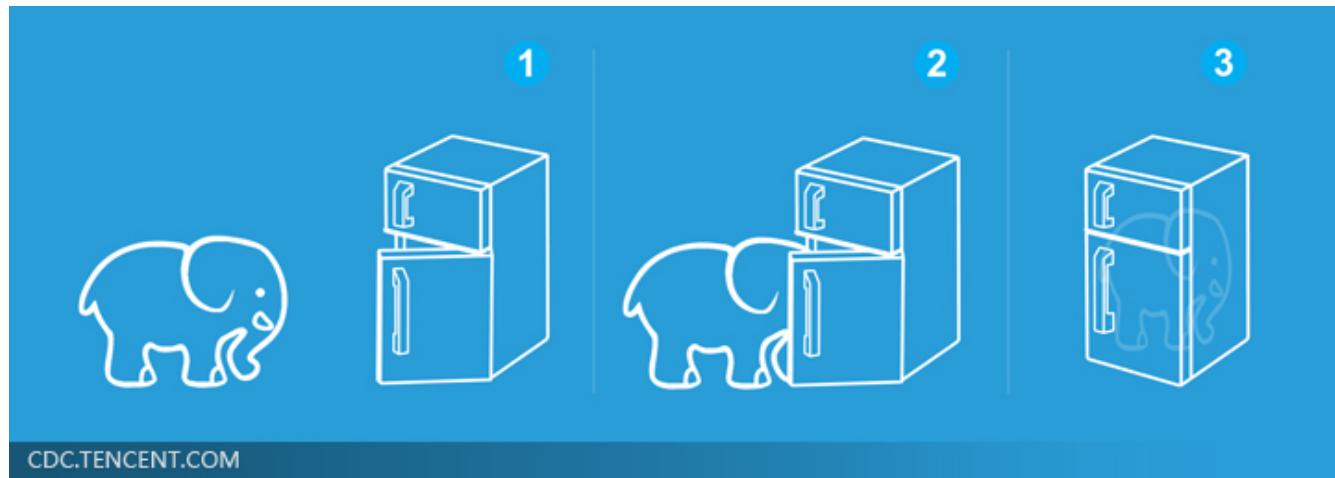
Step 1:
function with
unknown



Step 2: define
loss from
training data



Step 3:
optimization



Machine Learning is so simple

$$y = b + wx_1$$

$$w^* = 0.97, b^* = 0.1k$$

$$L(w^*, b^*) = 0.48k$$



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

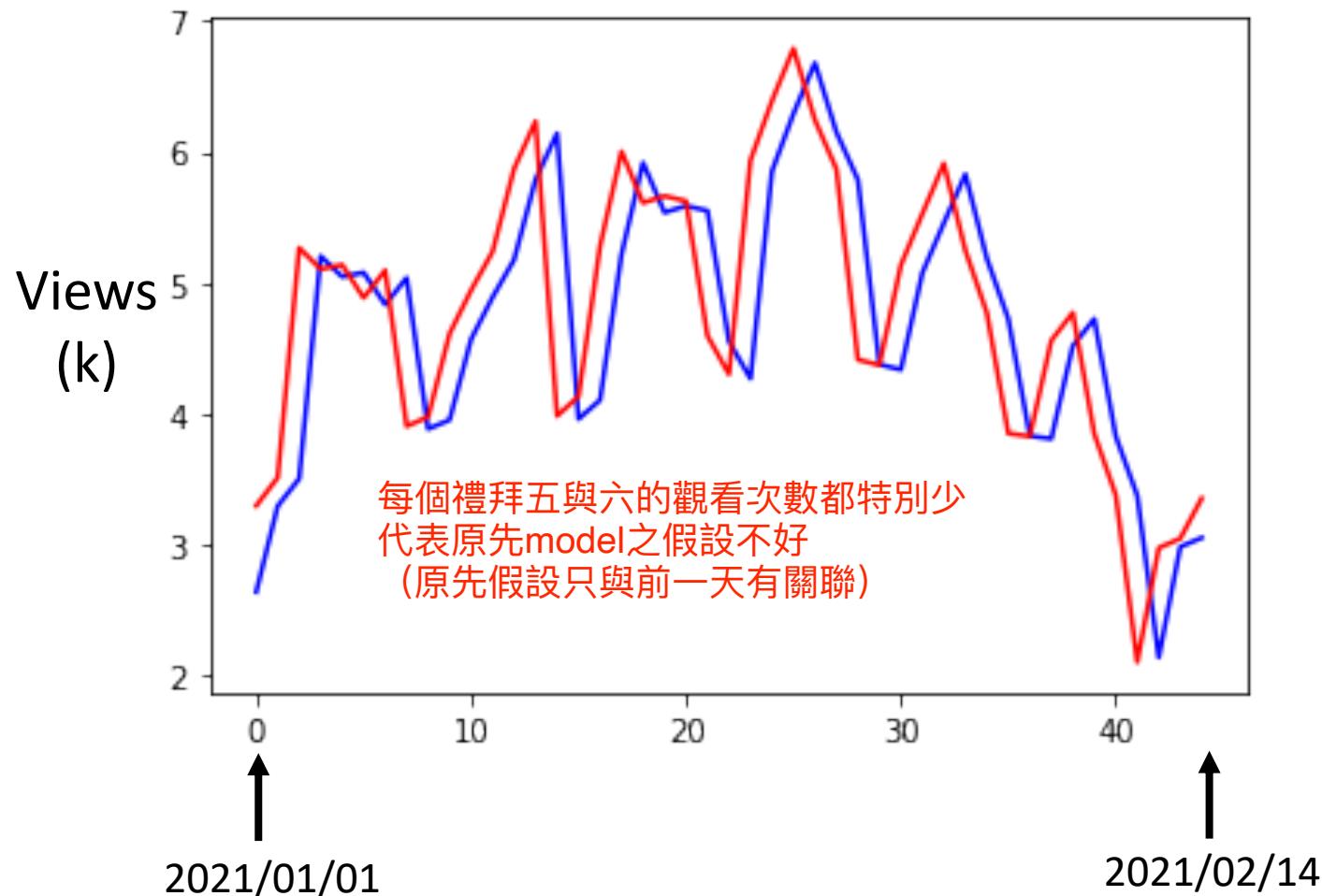
How about data of 2021 (**unseen during training**)?

在testing data上預測出來的loss為0.58k
比train data上之0.48k還大

$$L' = 0.58k$$

$$y = 0.1k + 0.97x_1$$

Red: real no. of views
blue: estimated no. of views



因此需對模型進行修改
而通常要怎麼修改，要取決於你對問題的了解
這就是domain knowledge

$$y = b + w x_1$$

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

$$L = 0.32k$$

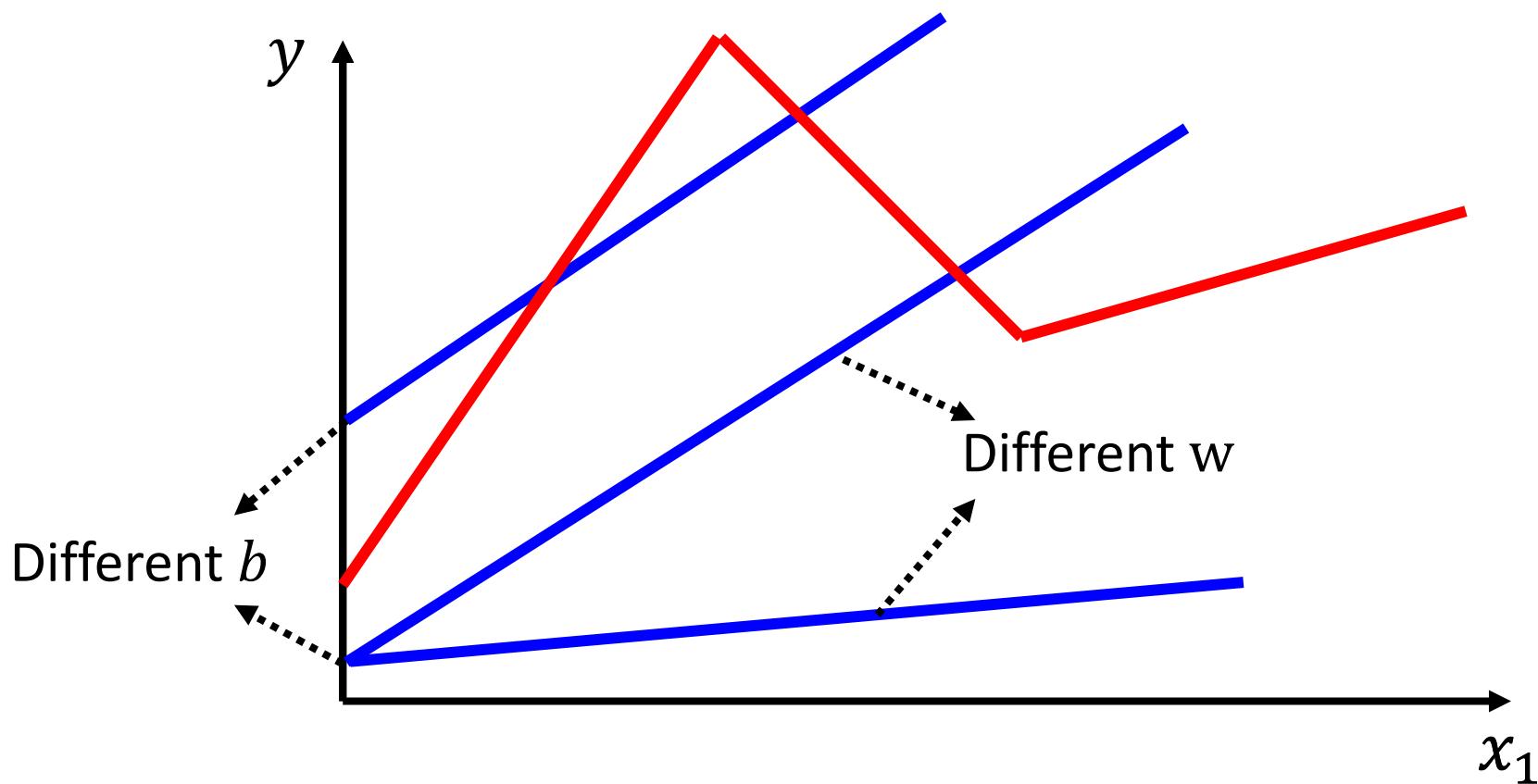
2021

$$L' = 0.46k$$

$y = b + \text{sum}(x_j * w_j)$ 的 model 稱為 linear model

Linear models

Linear models are too simple ... we need more sophisticated modes.

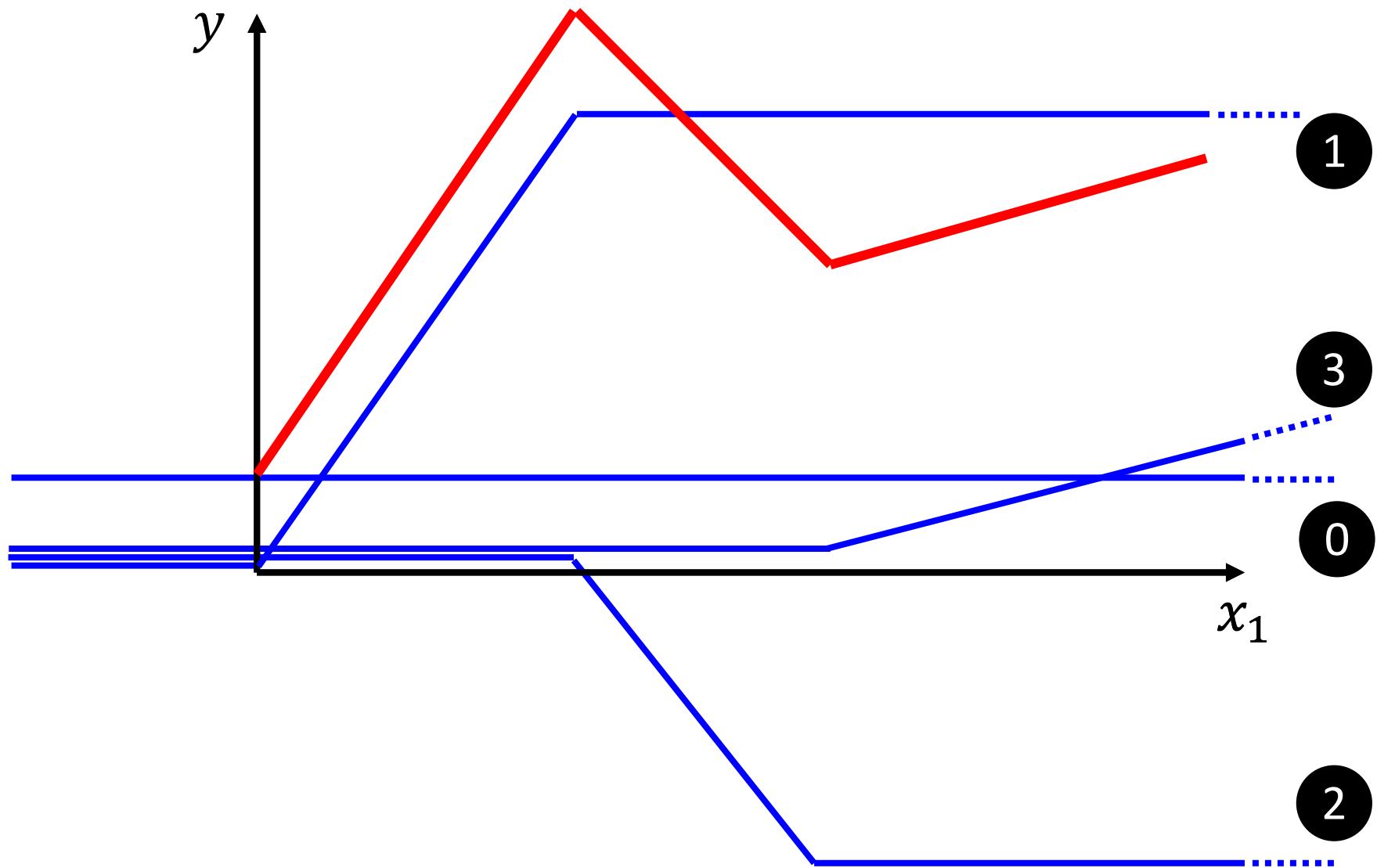
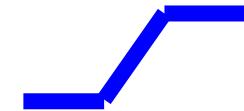


Linear models have severe limitation. **Model Bias**

model可涵蓋的函數範圍不足

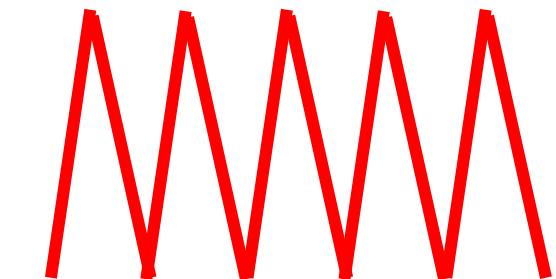
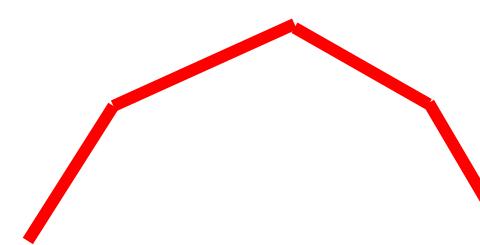
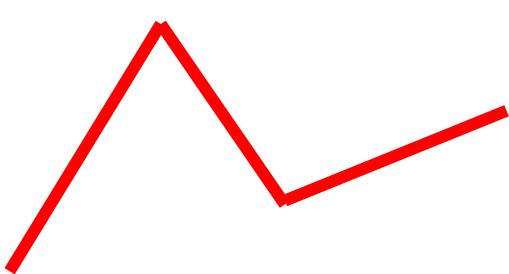
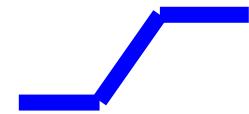
We need a more flexible model!

red curve = constant + sum of a set of

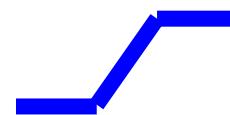


All Piecewise Linear Curves

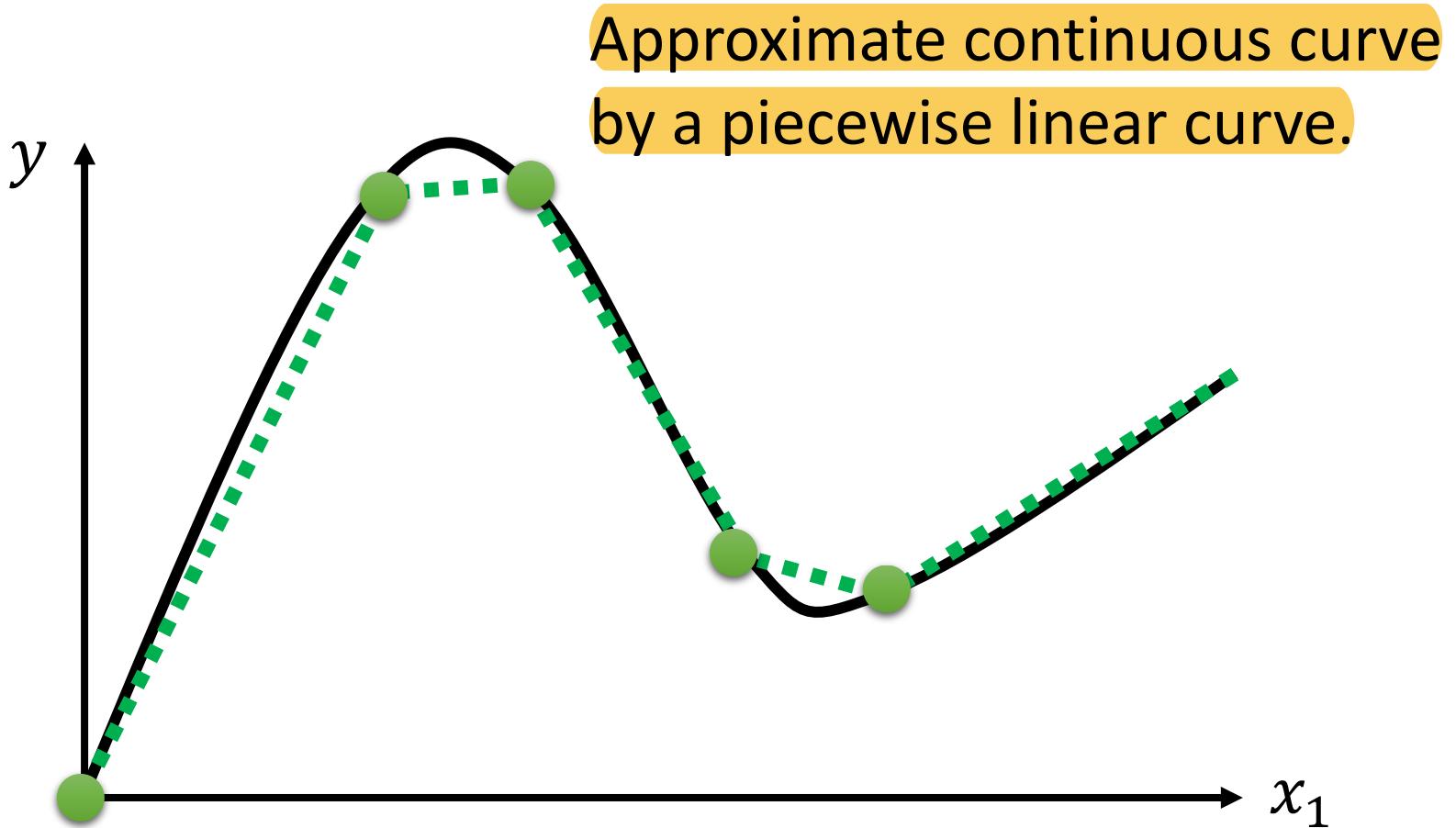
= constant + sum of a set of



More pieces require more

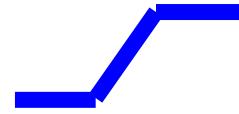


Beyond Piecewise Linear?



To have good approximation, we need sufficient pieces.

red curve = constant + sum of a set of



How to represent
this function?

Hard Sigmoid



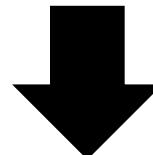
Sigmoid Function

$$y = c \frac{1}{1 + e^{-(b+wx_1)}}$$

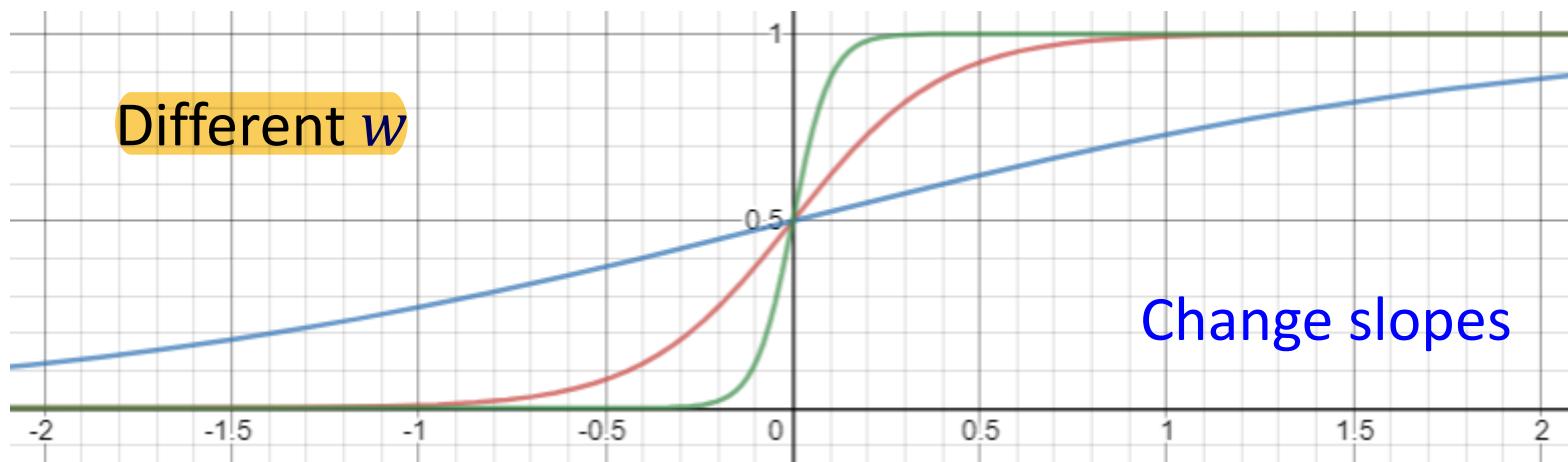
$$= c \text{ sigmoid}(b + wx_1)$$

when $x \rightarrow \infty, y \rightarrow c$

when $x \rightarrow -\infty, y \rightarrow 0$

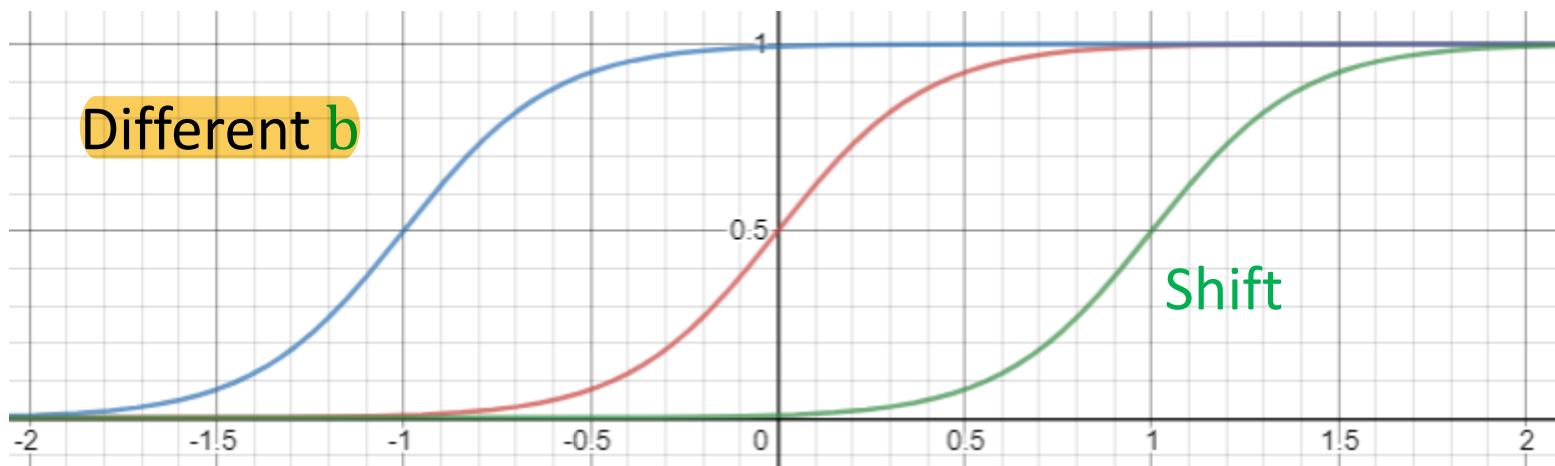


Different w



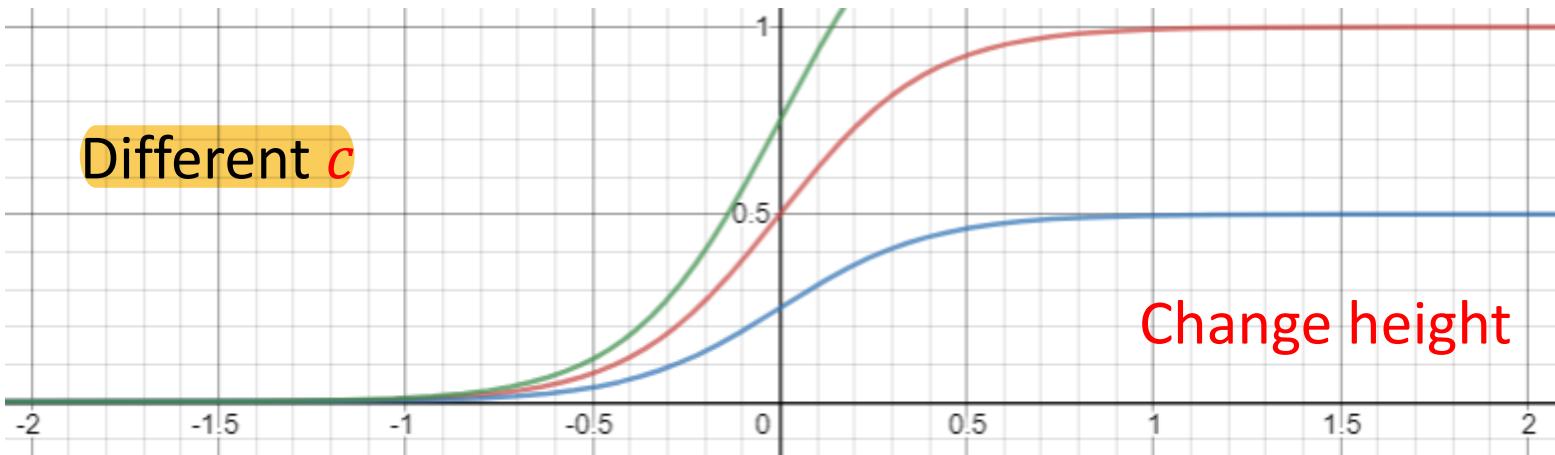
Change slopes

Different b



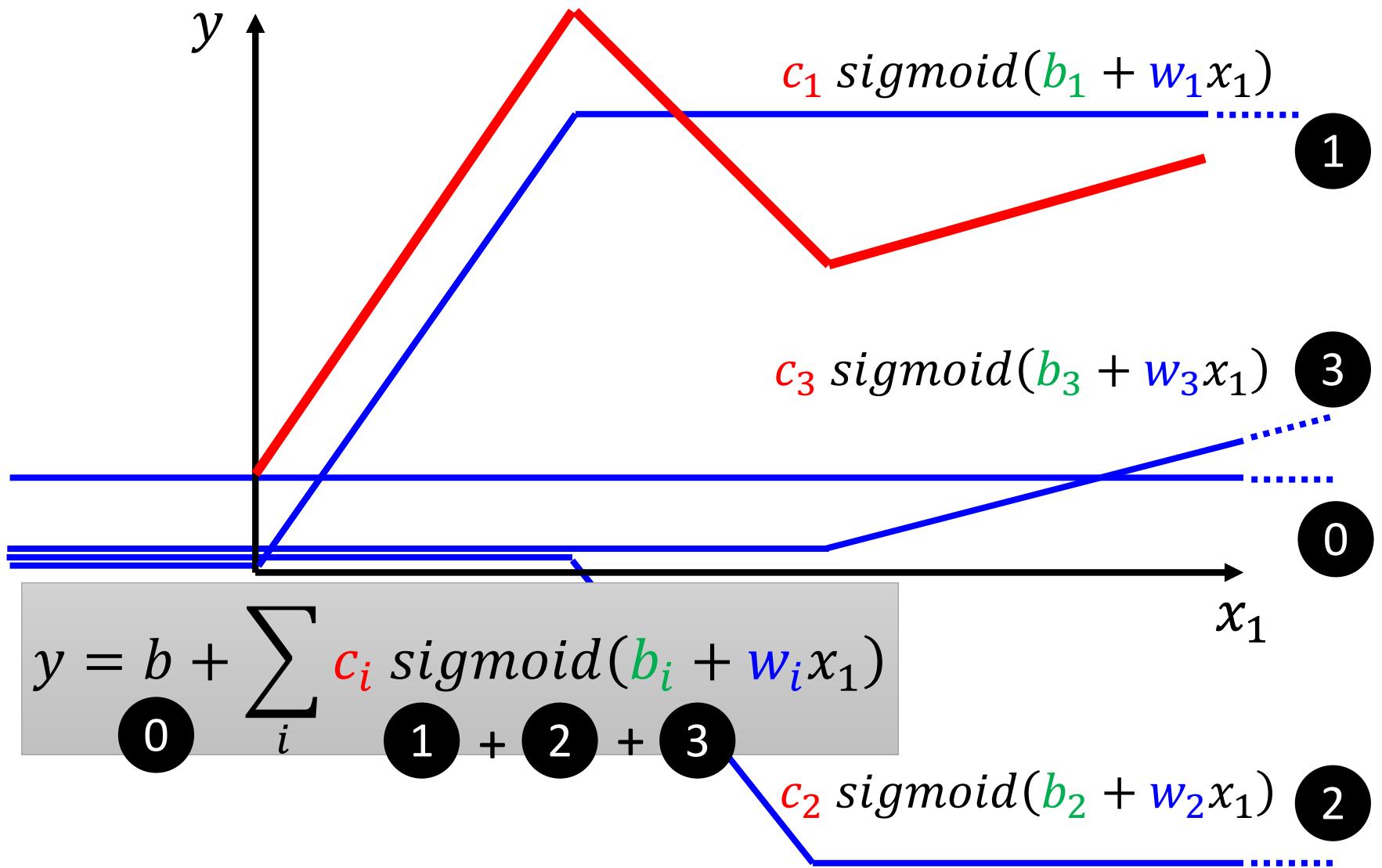
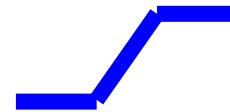
Shift

Different c



Change height

red curve = sum of a set of + constant



New Model: More Features

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

$$y = b + \sum_i c_i \text{sigmoid}(\frac{b_i + w_i x_1}{})$$

$$y = b + \sum_j w_j x_j$$

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

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

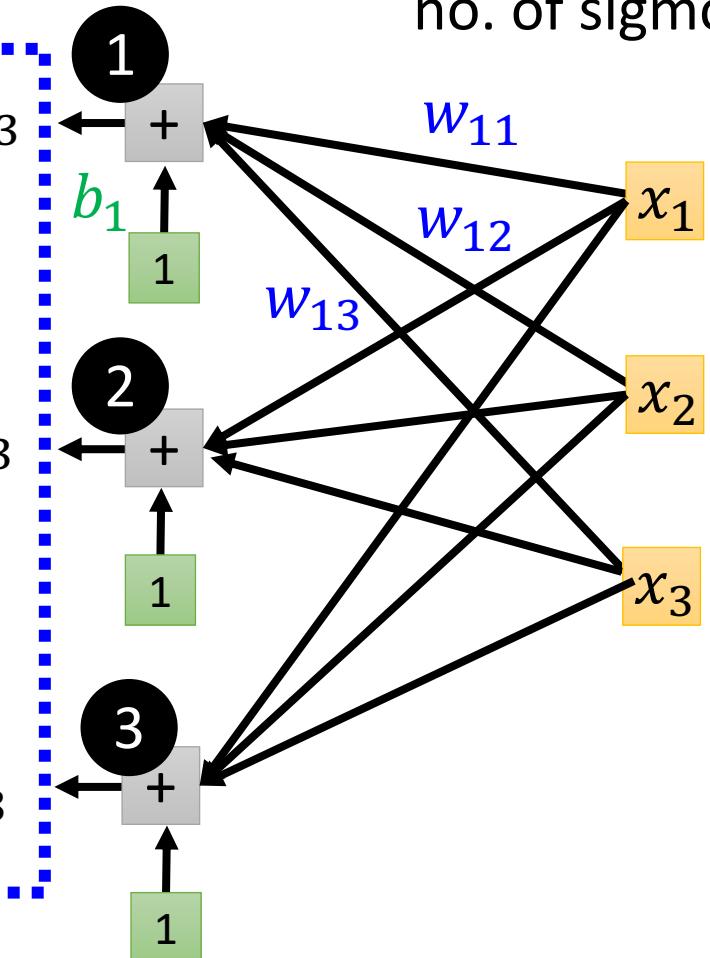
$j: 1, 2, 3$
 no. of features
 $i: 1, 2, 3$
 no. of sigmoid

$$r_1 = b_1 + w_{11}x_1 + w_{12}x_2 + w_{13}x_3$$

w_{ij} : weight for x_j for i-th sigmoid

$$r_2 = b_2 + w_{21}x_1 + w_{22}x_2 + w_{23}x_3$$

$$r_3 = b_3 + w_{31}x_1 + w_{32}x_2 + w_{33}x_3$$



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

$$r_1 = \textcolor{green}{b}_1 + \textcolor{blue}{w}_{11}x_1 + \textcolor{blue}{w}_{12}x_2 + \textcolor{blue}{w}_{13}x_3$$

$$r_2 = \textcolor{green}{b}_2 + \textcolor{blue}{w}_{21}x_1 + \textcolor{blue}{w}_{22}x_2 + \textcolor{blue}{w}_{23}x_3$$

$$r_3 = \textcolor{green}{b}_3 + \textcolor{blue}{w}_{31}x_1 + \textcolor{blue}{w}_{32}x_2 + \textcolor{blue}{w}_{33}x_3$$

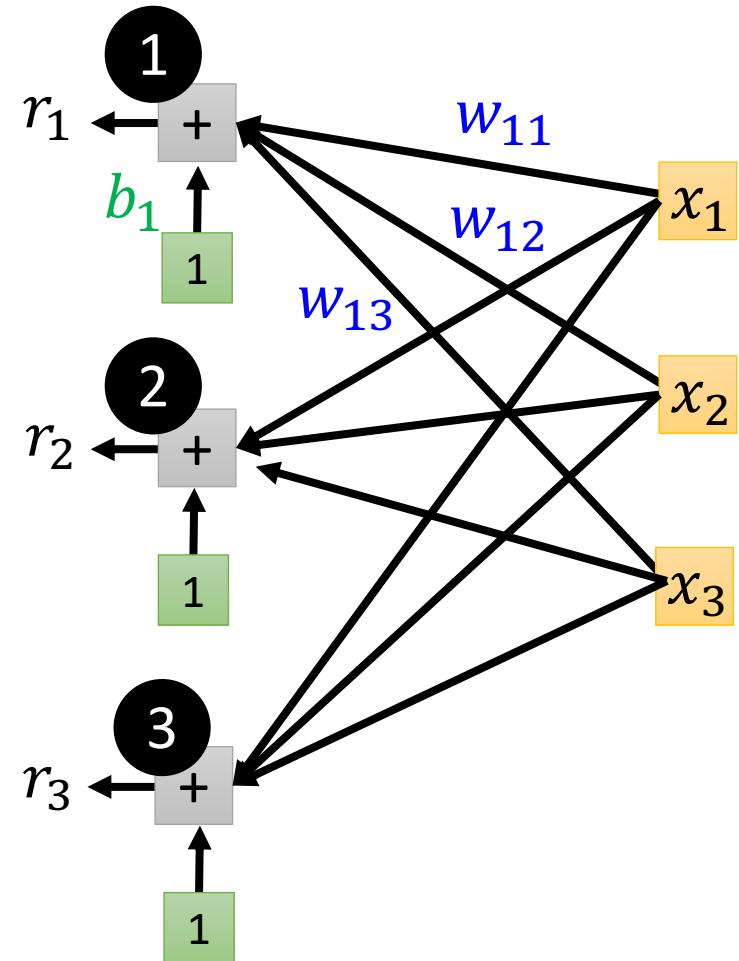
$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} \textcolor{green}{b}_1 \\ \textcolor{green}{b}_2 \\ \textcolor{green}{b}_3 \end{bmatrix} + \begin{bmatrix} \textcolor{blue}{w}_{11} & \textcolor{blue}{w}_{12} & \textcolor{blue}{w}_{13} \\ \textcolor{blue}{w}_{21} & \textcolor{blue}{w}_{22} & \textcolor{blue}{w}_{23} \\ \textcolor{blue}{w}_{31} & \textcolor{blue}{w}_{32} & \textcolor{blue}{w}_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boxed{\mathbf{r}} = \boxed{\mathbf{b}} + \boxed{W} \boxed{\mathbf{x}}$$

$$y = b + \sum_i \textcolor{red}{c_i} \text{ sigmoid} \left(\textcolor{green}{b_i} + \sum_j \textcolor{blue}{w_{ij}} x_j \right)$$

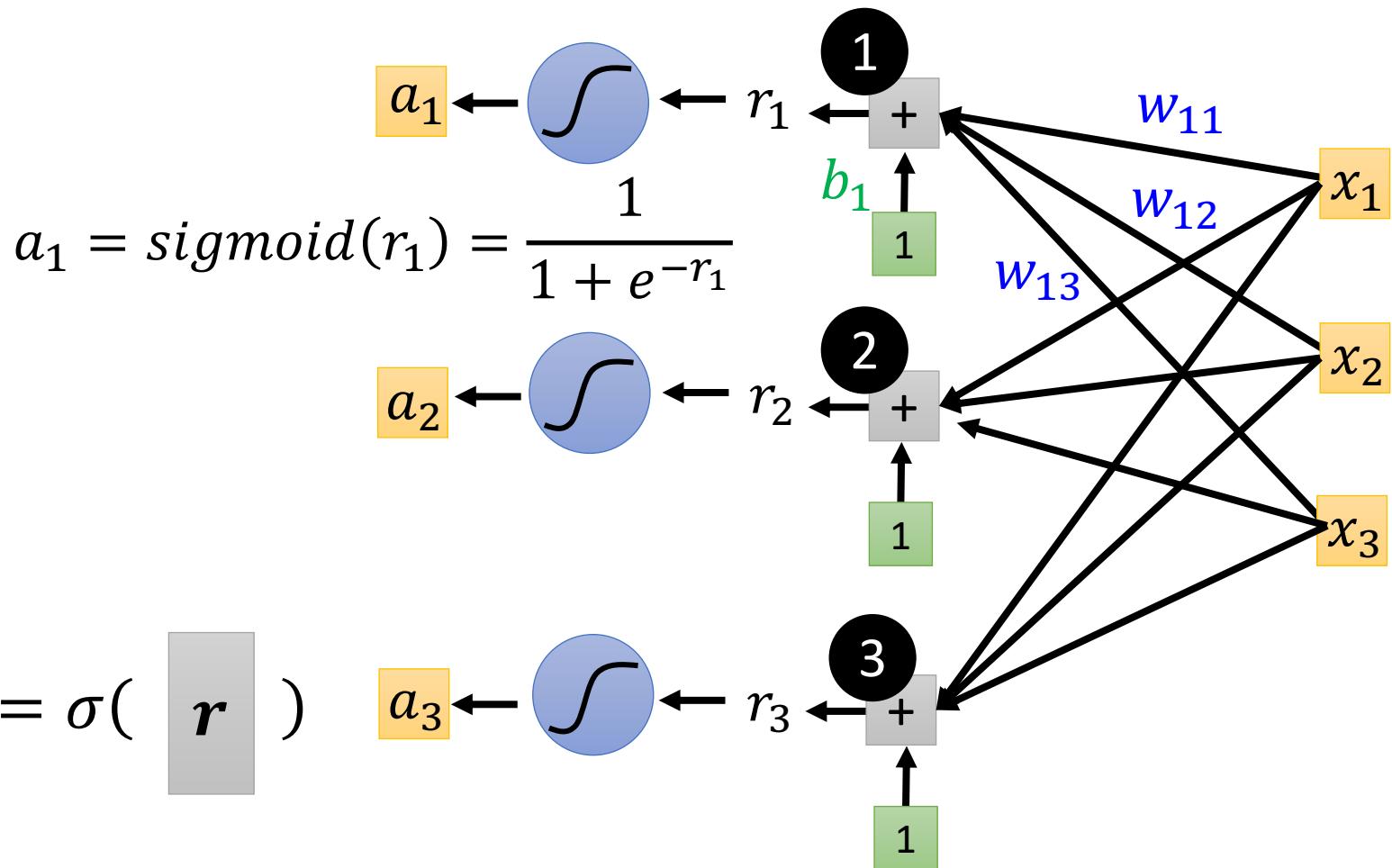
$i: 1, 2, 3$
 $j: 1, 2, 3$

$$\begin{matrix} r \\ \hline \end{matrix} = \begin{matrix} b \\ \hline \end{matrix} + \begin{matrix} W \\ \hline \end{matrix} \begin{matrix} x \\ \hline \end{matrix}$$



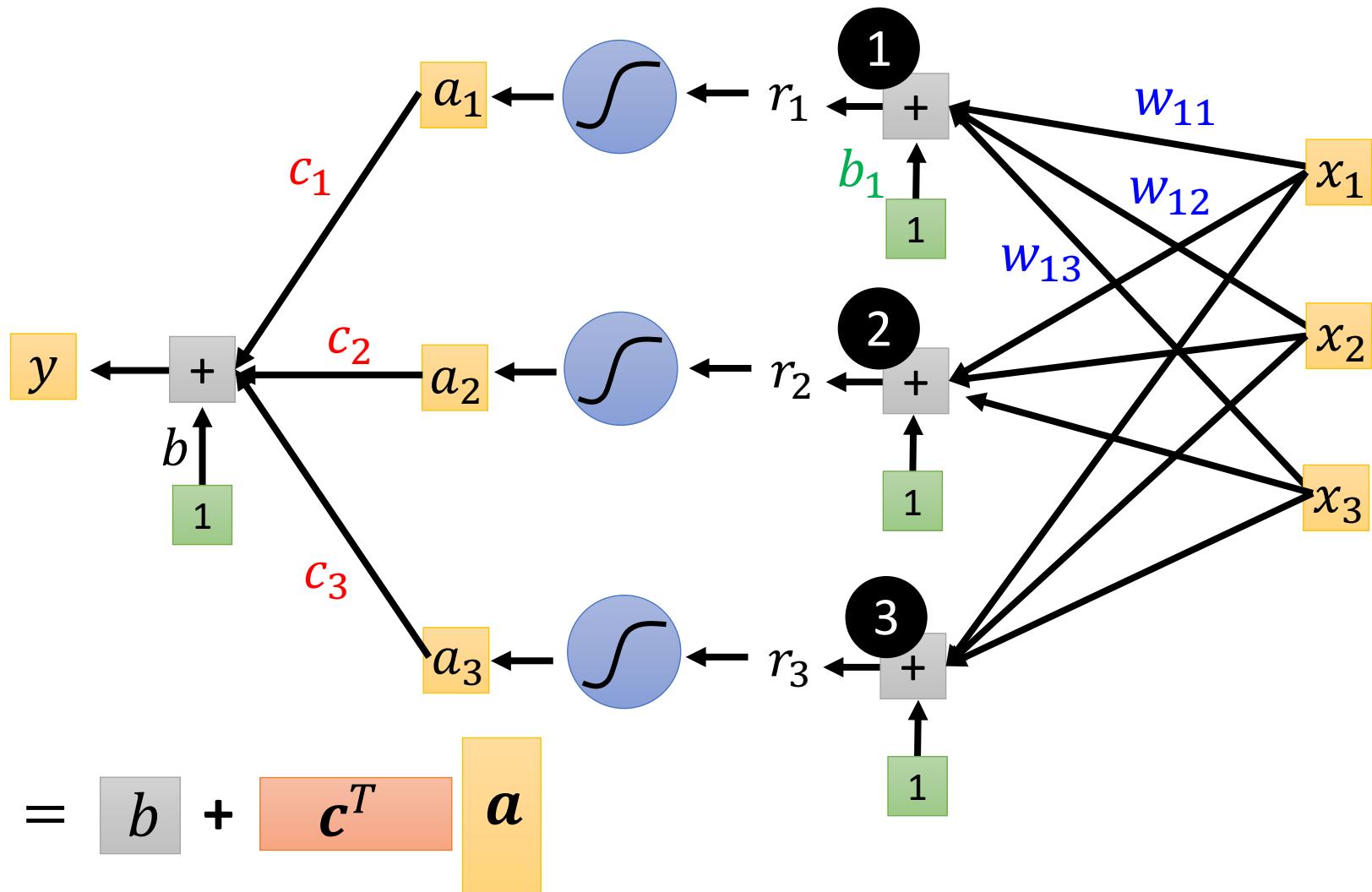
$$y = b + \sum_i c_i \text{sigmoid} \left(b_i + \sum_j w_{ij} x_j \right)$$

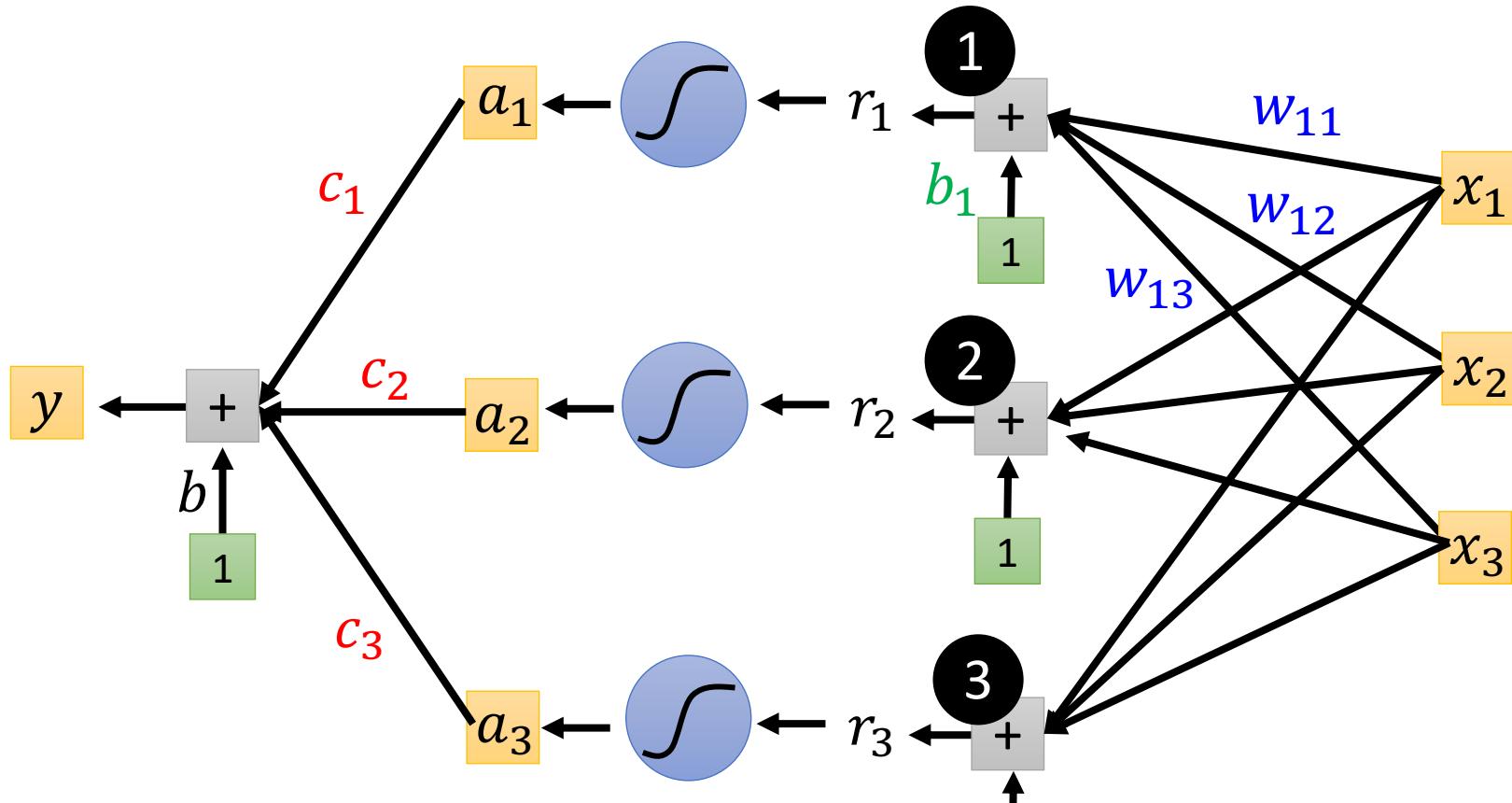
$i: 1, 2, 3$
 $j: 1, 2, 3$



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

$i: 1, 2, 3$
 $j: 1, 2, 3$

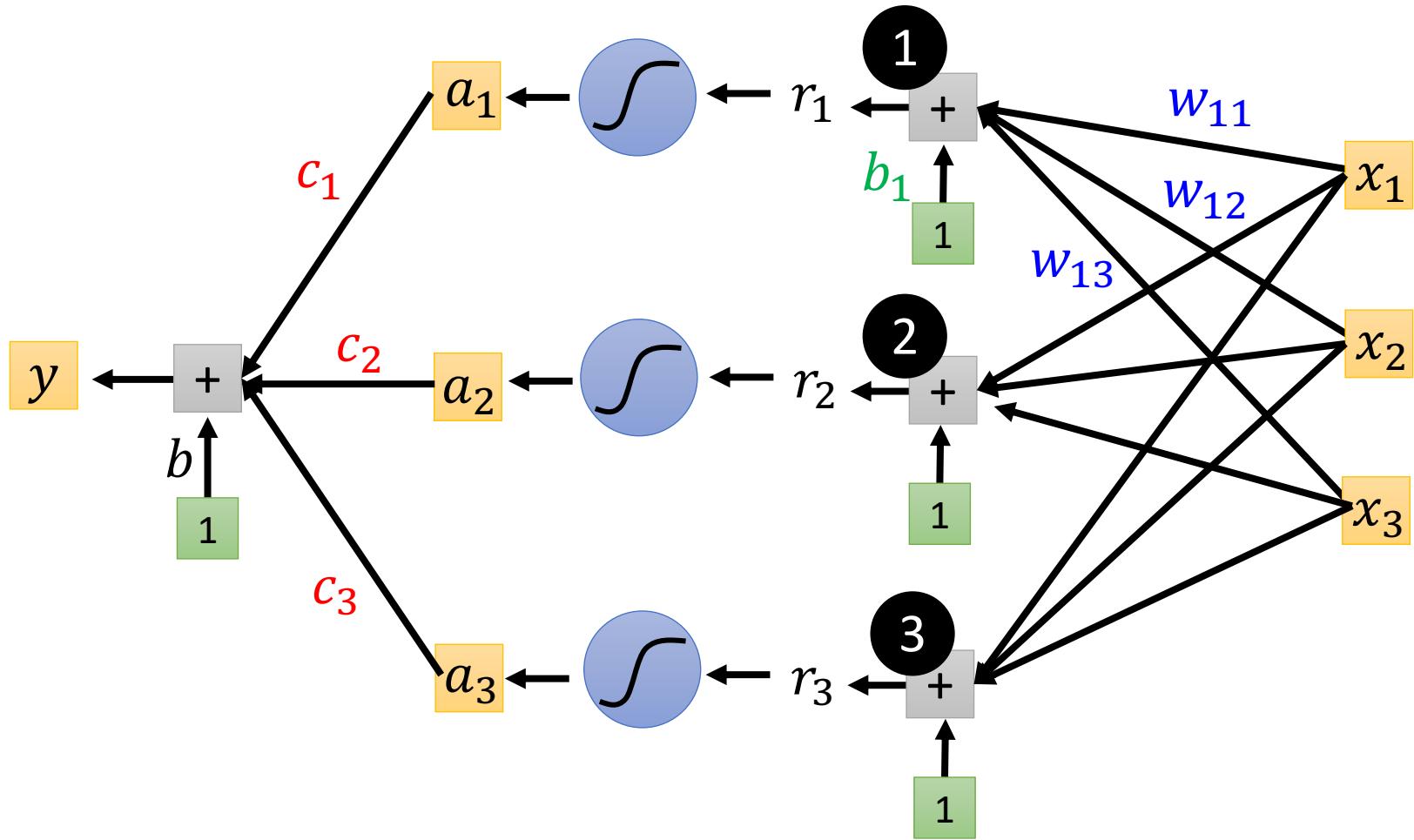




$$y = b + c^T a$$

$$a = \sigma(r)$$

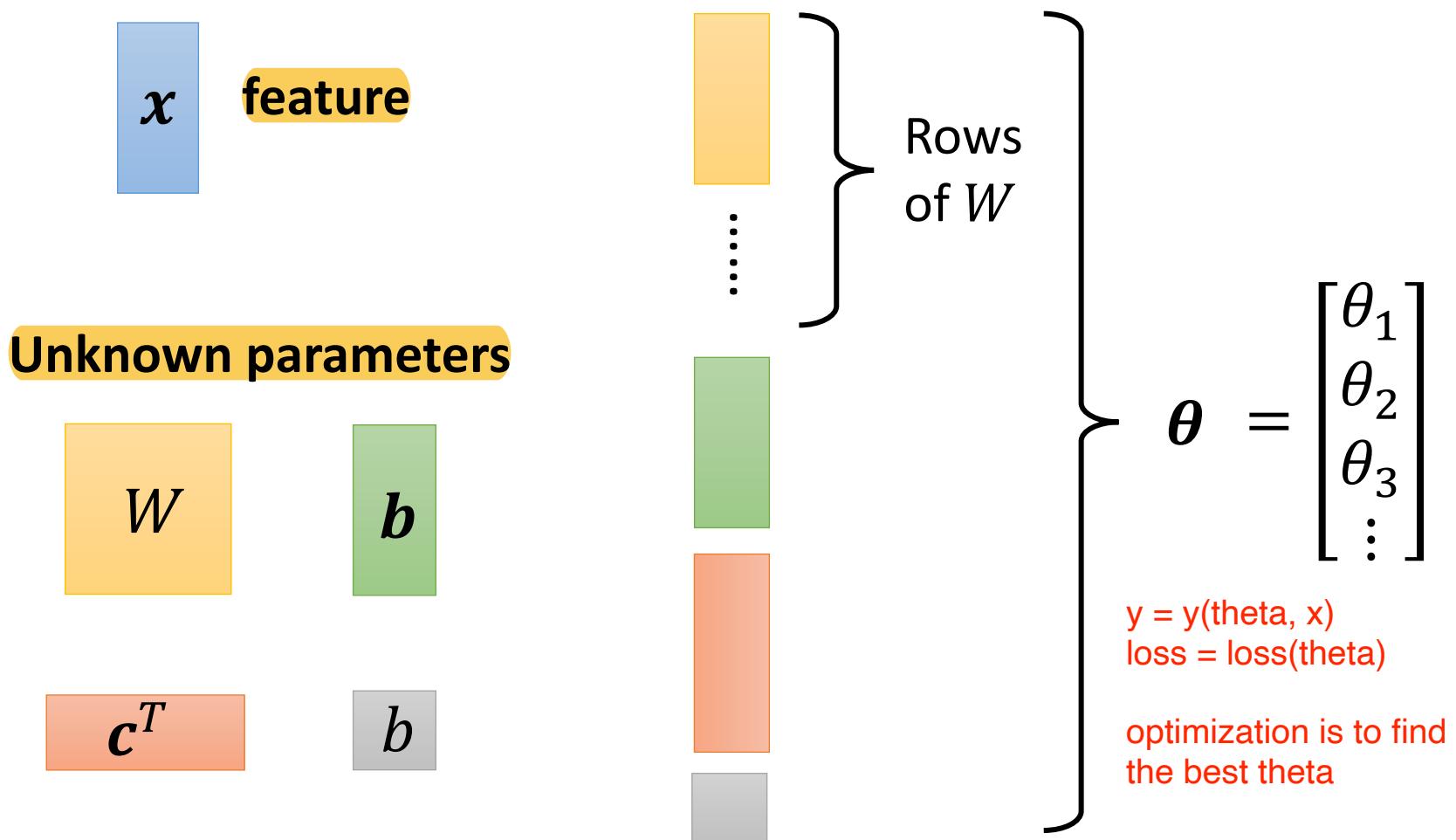
$$r = b + Wx$$



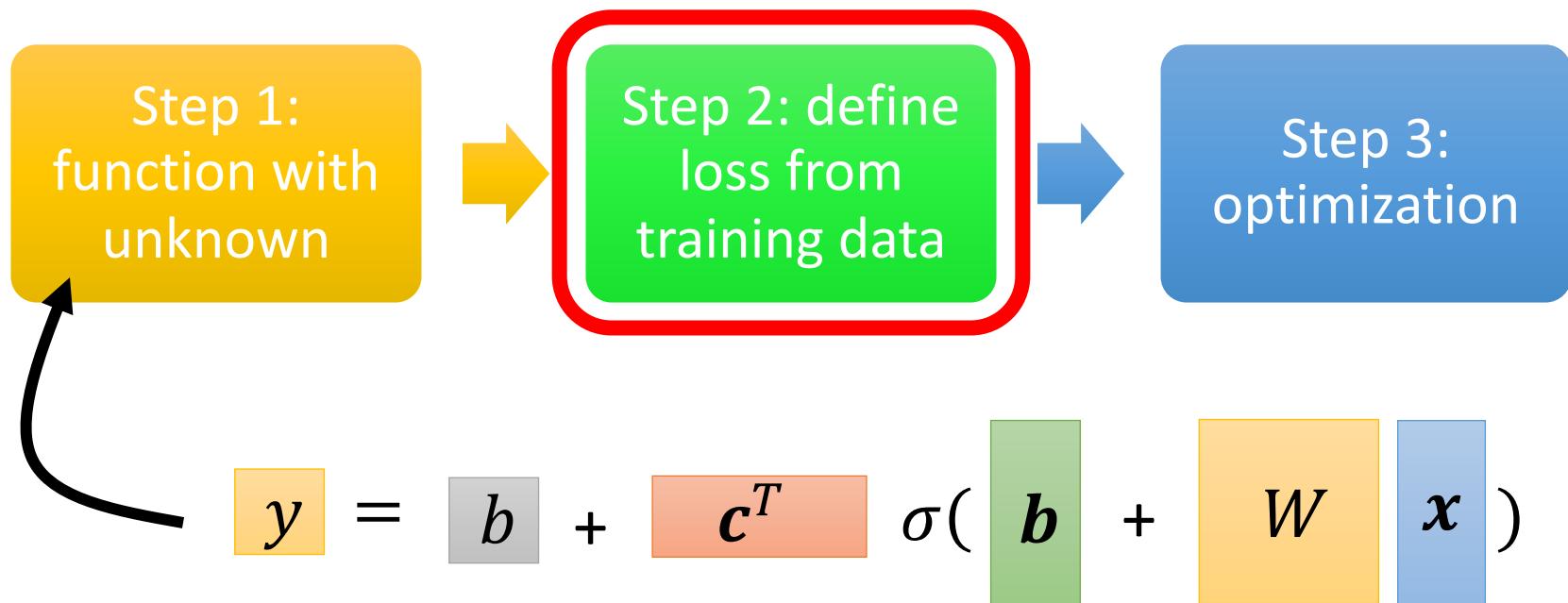
$$y = b + \mathbf{c}^T \sigma(\mathbf{b} + \mathbf{W} \mathbf{x})$$

Function with unknown parameters

$$y = b + c^T \sigma(b) + Wx$$

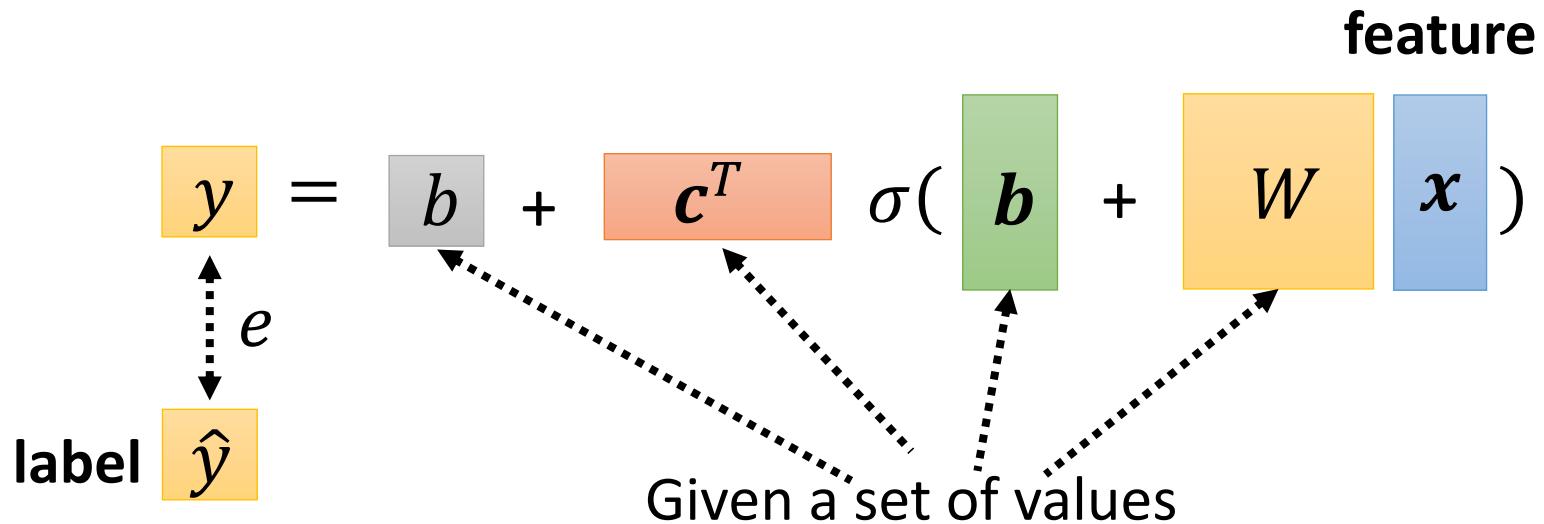


Back to ML Framework



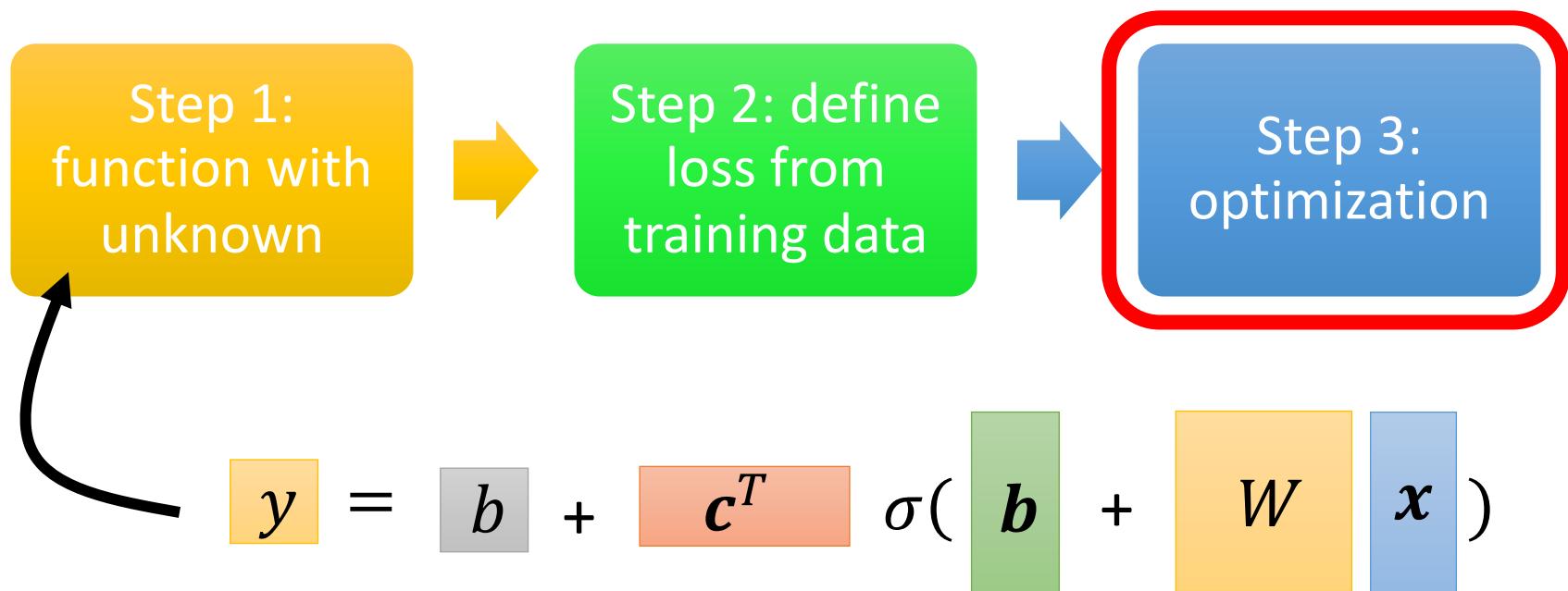
LOSS

- Loss is a function of parameters $L(\theta)$
- Loss means how good a set of values is.



$$\text{Loss: } L = \frac{1}{N} \sum_n e_n$$

Back to ML Framework



Optimization of New Model

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

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

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

gradient $\mathbf{g} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} |_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \frac{\partial L}{\partial \theta_2} |_{\boldsymbol{\theta}=\boldsymbol{\theta}^0} \\ \vdots \end{bmatrix}$

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

$$\mathbf{g} = \nabla L(\boldsymbol{\theta}^0)$$

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

Optimization of New Model

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

- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- Compute gradient $\mathbf{g} = \nabla L(\boldsymbol{\theta}^0)$
$$\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \mathbf{g}$$
- Compute gradient $\mathbf{g} = \nabla L(\boldsymbol{\theta}^1)$
$$\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \mathbf{g}$$
- Compute gradient $\mathbf{g} = \nabla L(\boldsymbol{\theta}^2)$
$$\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \mathbf{g}$$

Optimization of New Model

將training data分成n個batch

一次拿一個batch中的所有data來update參數

將每一個batch都update過一次，則稱為完成一個epoch

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

batch size也是一個hyper parameter

➤ (Randomly) Pick initial values θ^0

➤ Compute gradient $\mathbf{g} = \nabla L^1(\theta^0)$

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

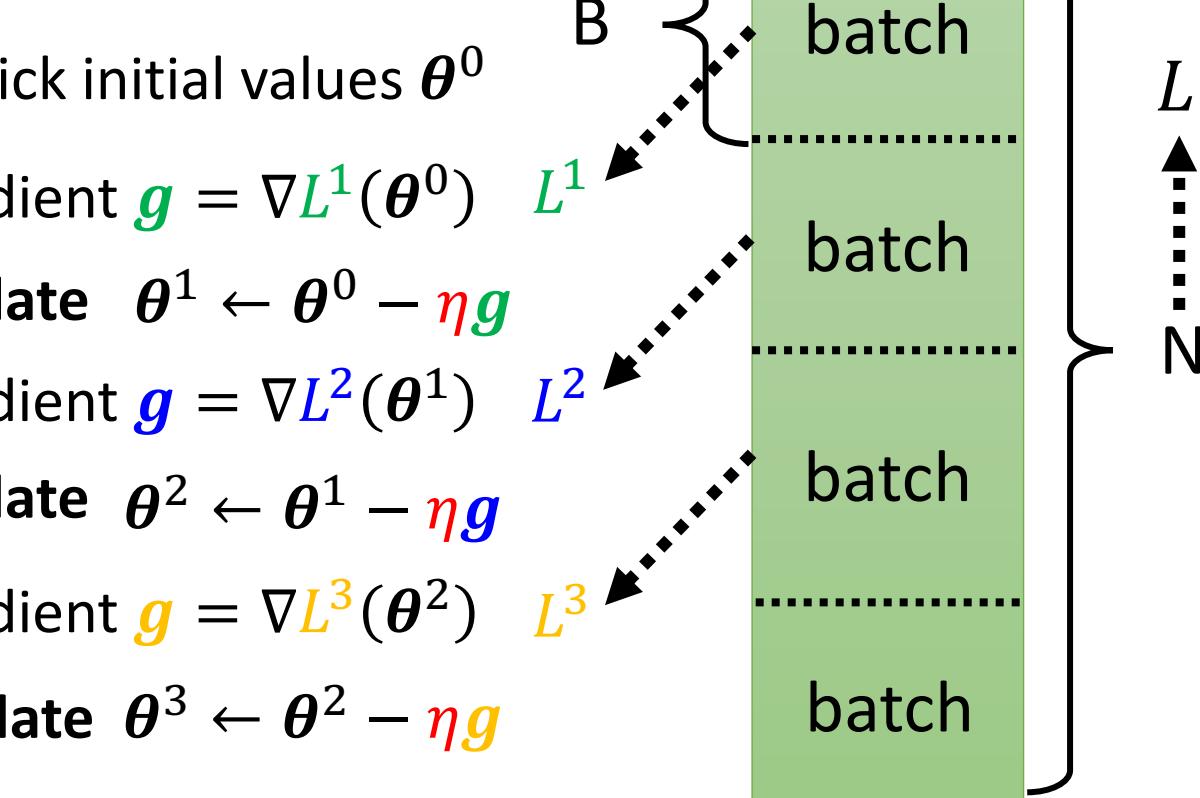
➤ Compute gradient $\mathbf{g} = \nabla L^2(\theta^1)$

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

➤ Compute gradient $\mathbf{g} = \nabla L^3(\theta^2)$

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

1 epoch = see all the batches once



Optimization of New Model

Example 1

- 10,000 examples ($N = 10,000$)
- Batch size is 10 ($B = 10$)

How many update in **1 epoch**?

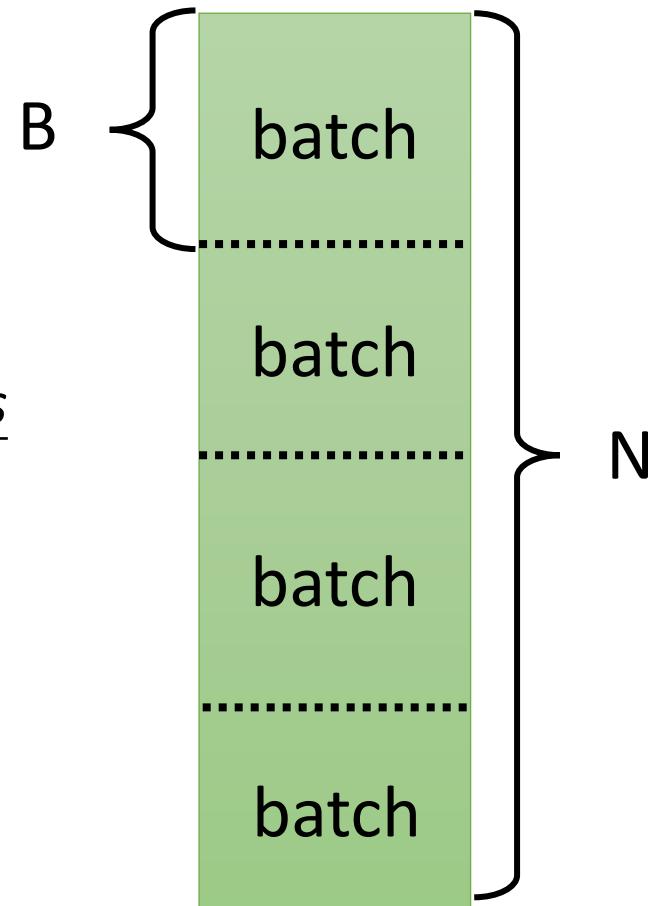
1,000 updates

Example 2

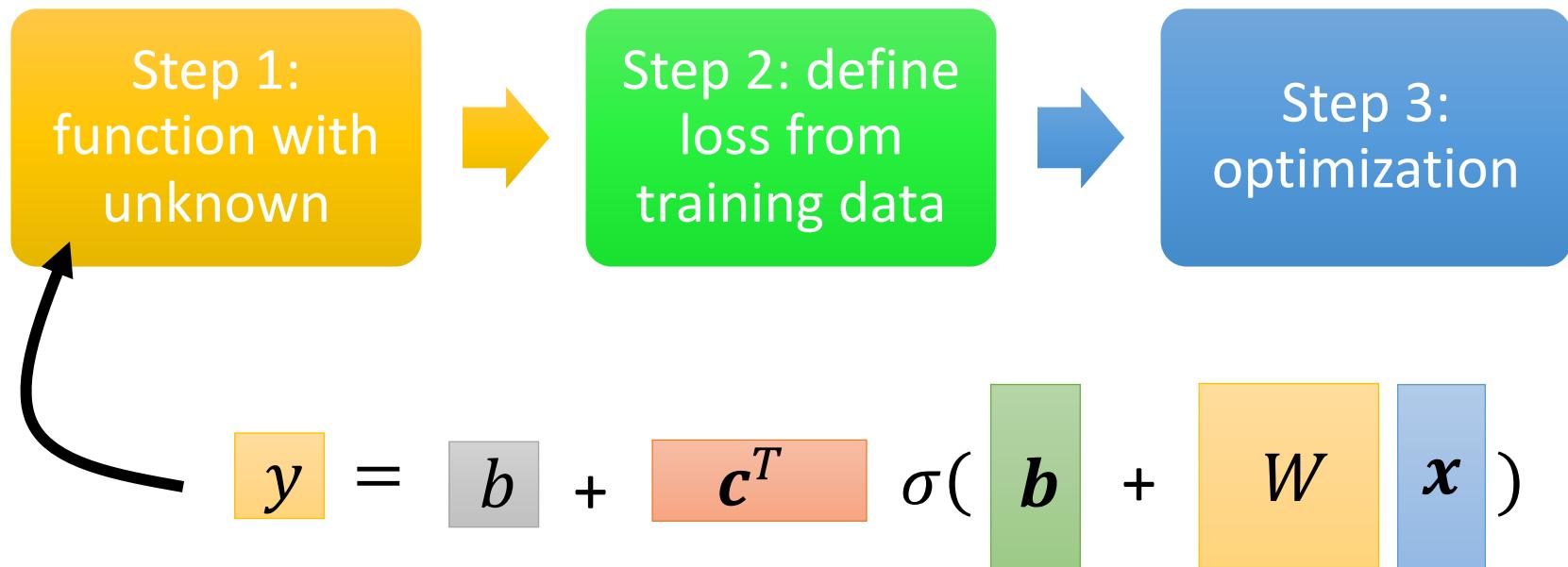
- 1,000 examples ($N = 1,000$)
- Batch size is 100 ($B = 100$)

How many update in **1 epoch**?

10 updates



Back to ML Framework

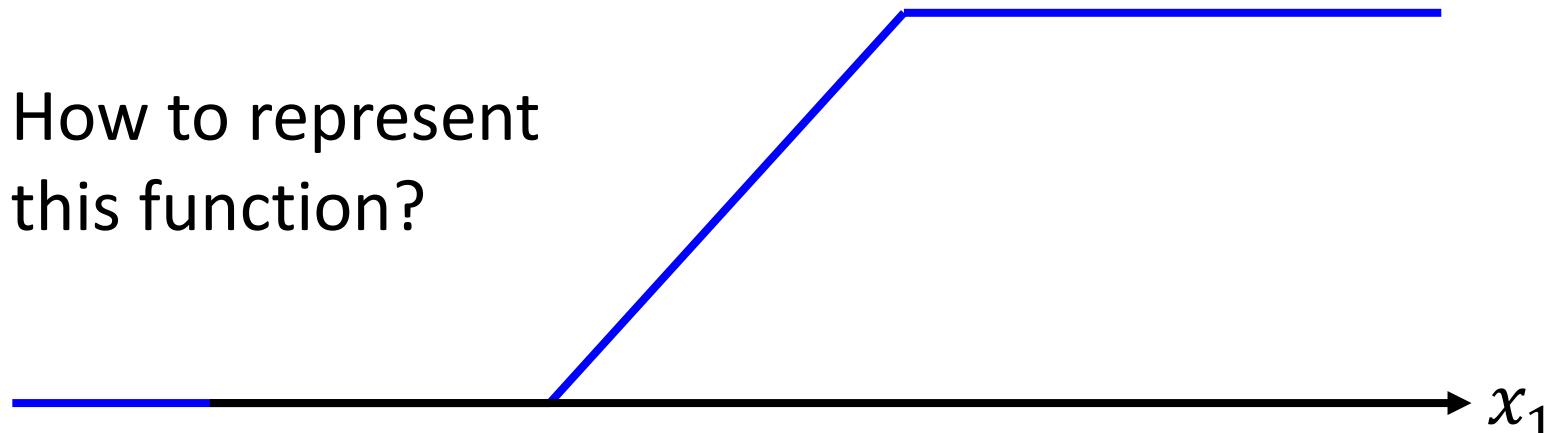


More variety of models ...

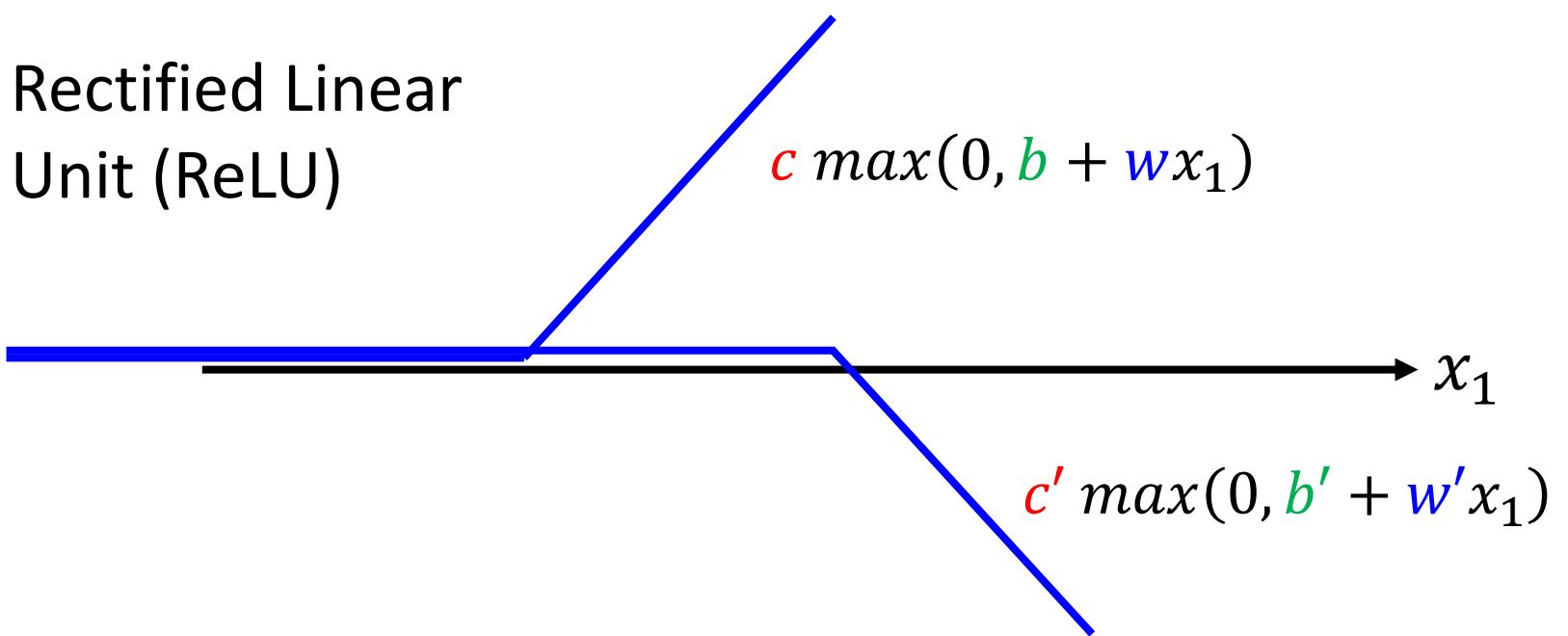
Sigmoid → ReLU

ReLU可以由兩個 $c * \max(0, b+wx)$ 之function合成

How to represent
this function?



Rectified Linear
Unit (ReLU)



Sigmoid → ReLU

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

Activation function

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

Which one is better?

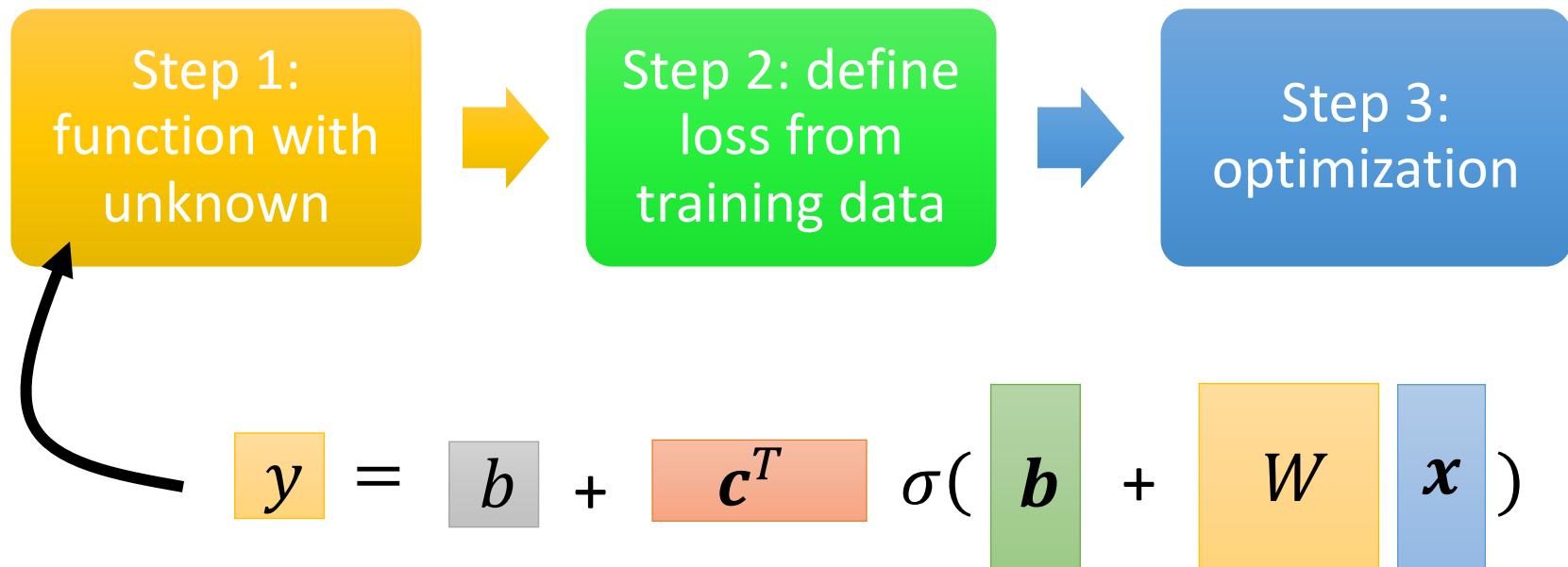
Experimental Results

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

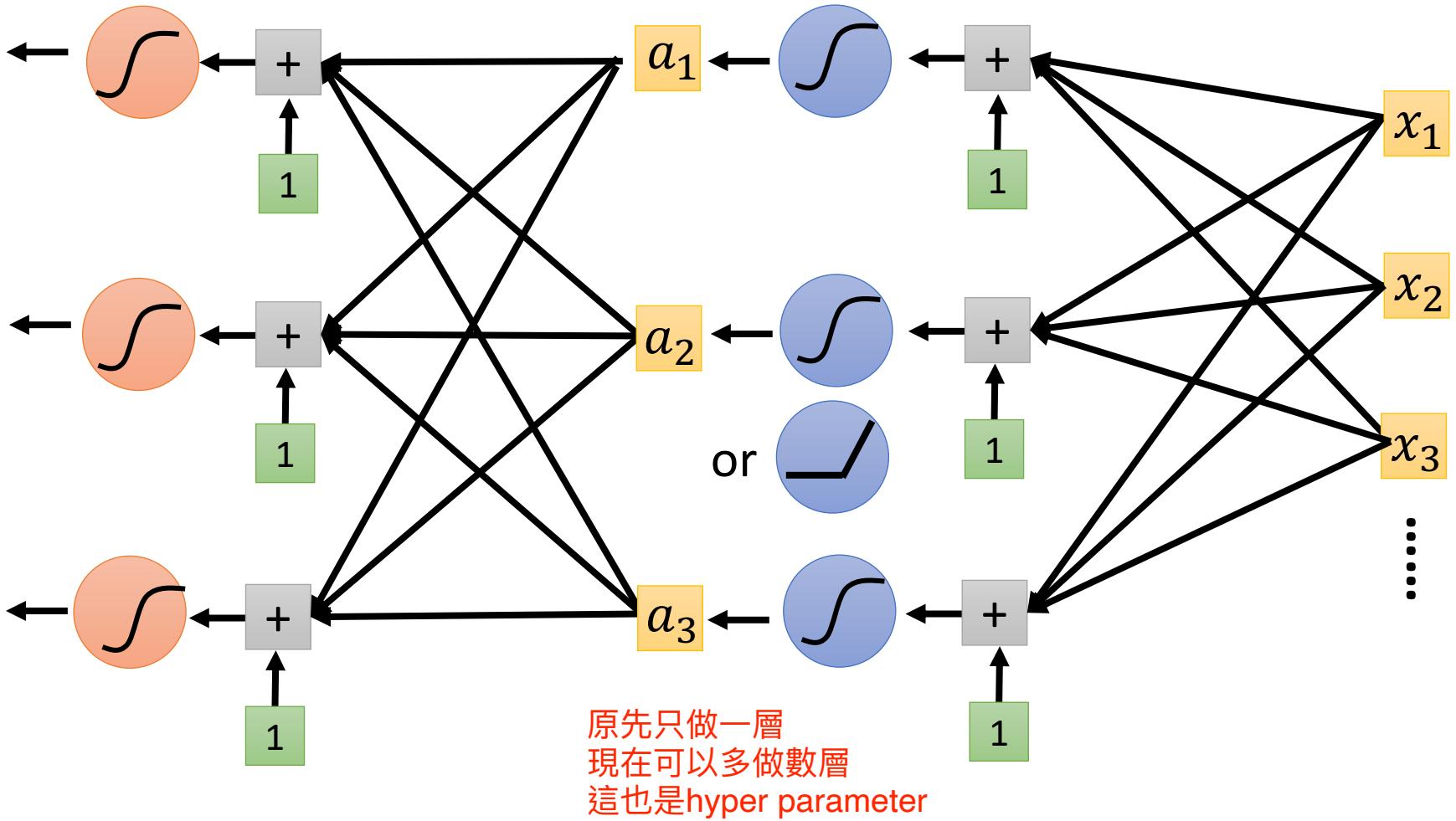
	linear
2017 – 2020	0.32k
2021	0.46k

若改用10個relu
training loss = 0.32K
testing loss = 0.45K

Back to ML Framework



Even more variety of models ...



$$a' = \sigma(b' + W' a) \quad a = \sigma(b + W x)$$

Experimental Results

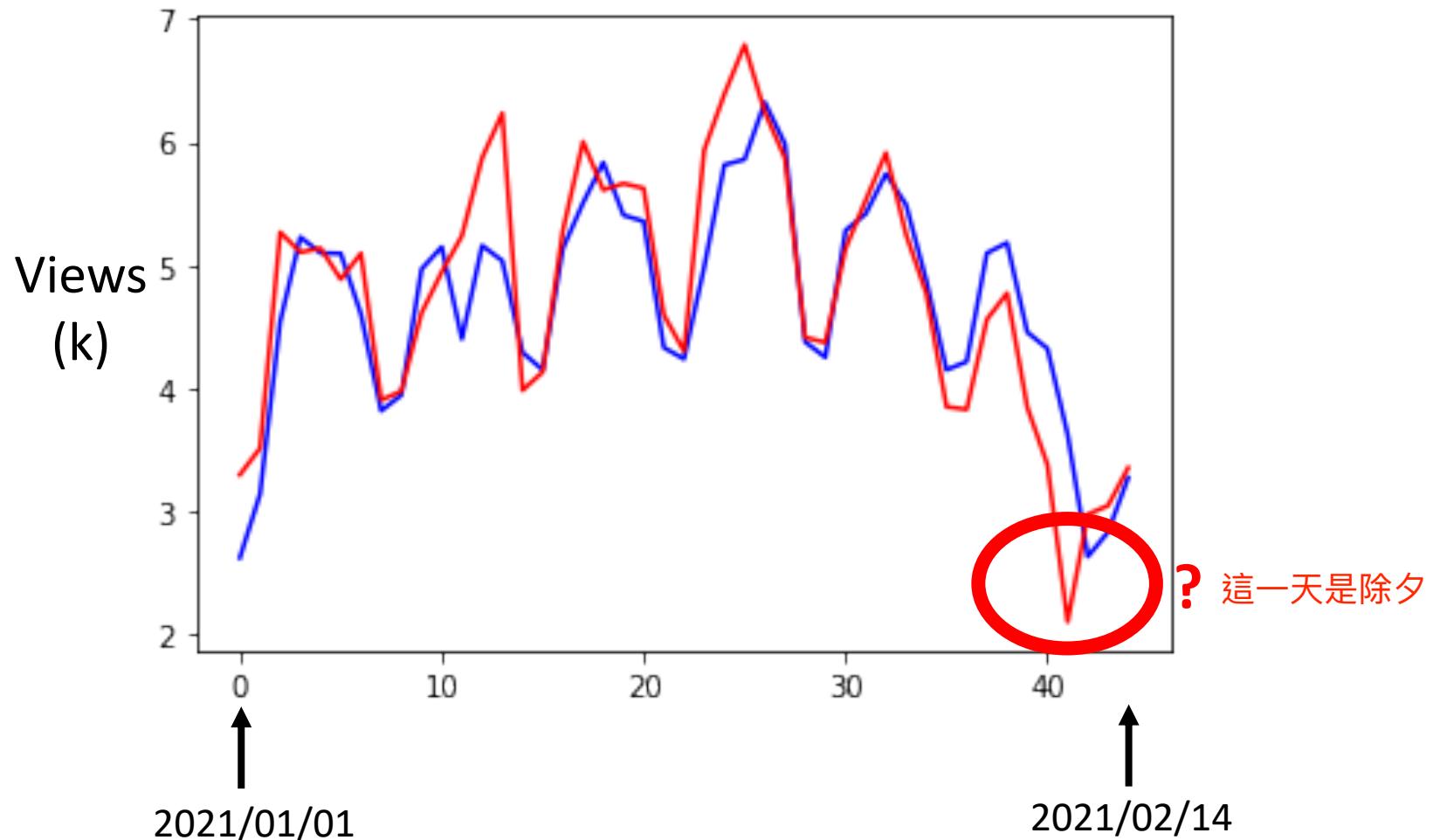
- Loss for multiple hidden layers
 - 100 ReLU for each layer
 - input features are the no. of views in the past 56 days

	1 layer
2017 – 2020	0.28k
2021	0.43k

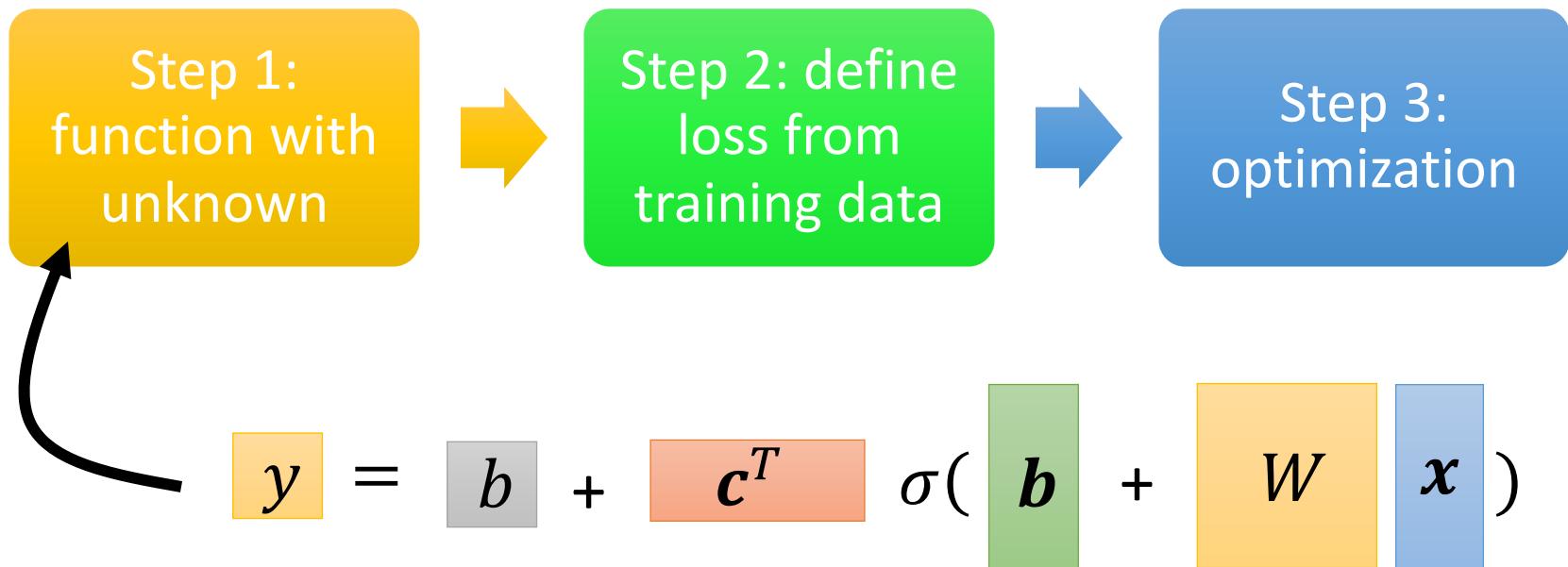
2 layer
training data: 0.18K
testing data: 0.39K

3 layers

Red: real no. of views
blue: estimated no. of views



Back to ML Framework

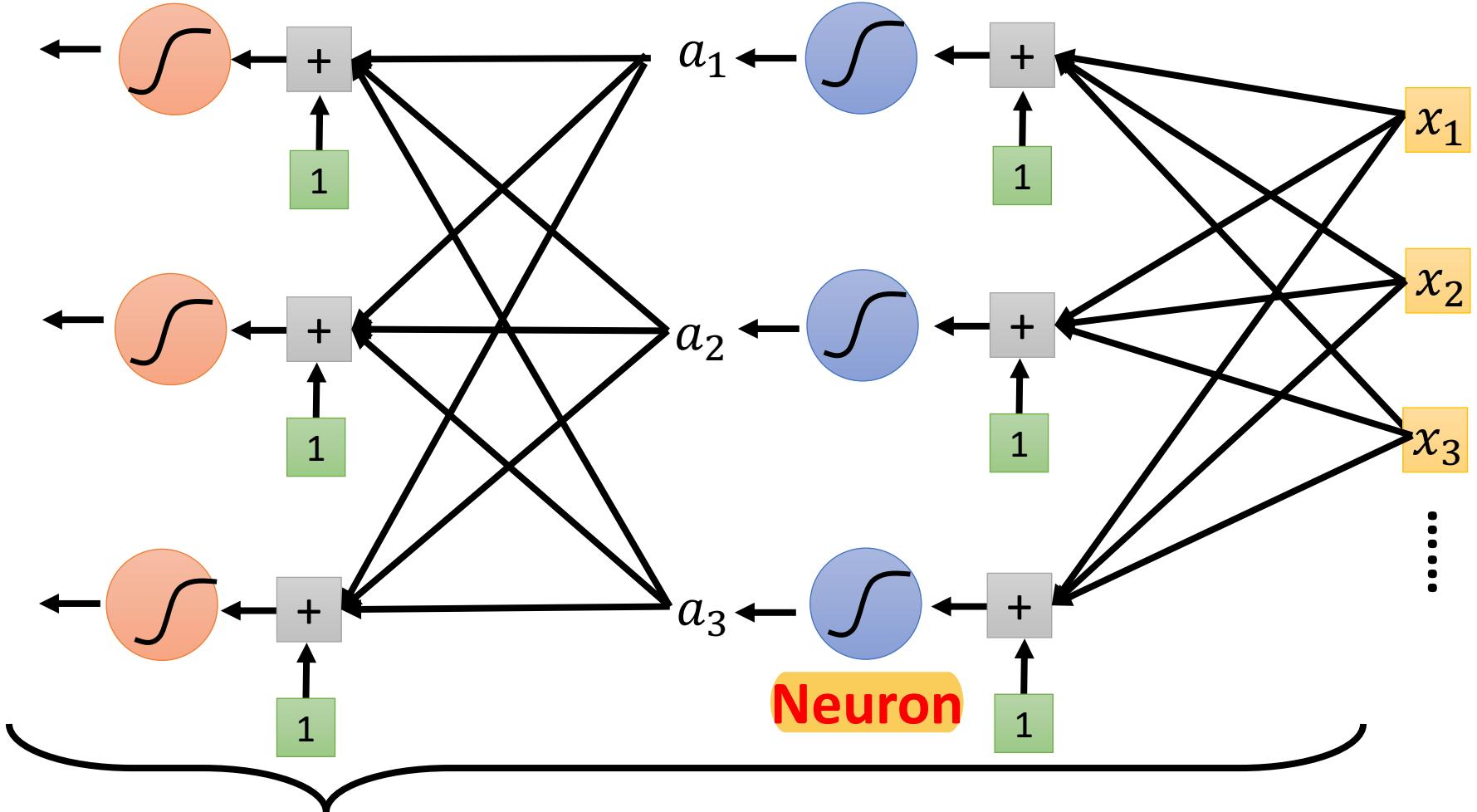


It is not *fancy* enough.

Let's give it a *fancy* name!

hidden layer

hidden layer



Neural Network

This mimics human brains ... (???)

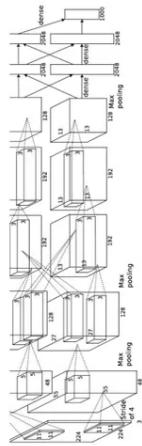
Many layers means Deep → Deep Learning

Deep = Many hidden layers

影像辨識

http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf

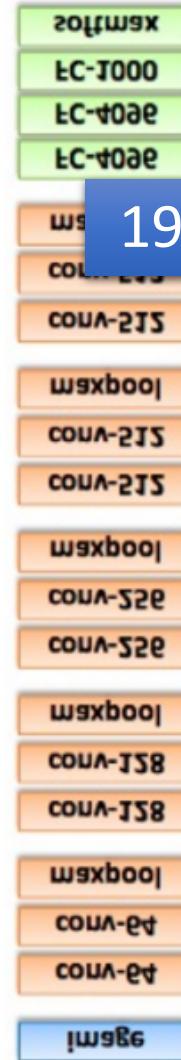
16.4%



AlexNet (2012)

8 layers

7.3%

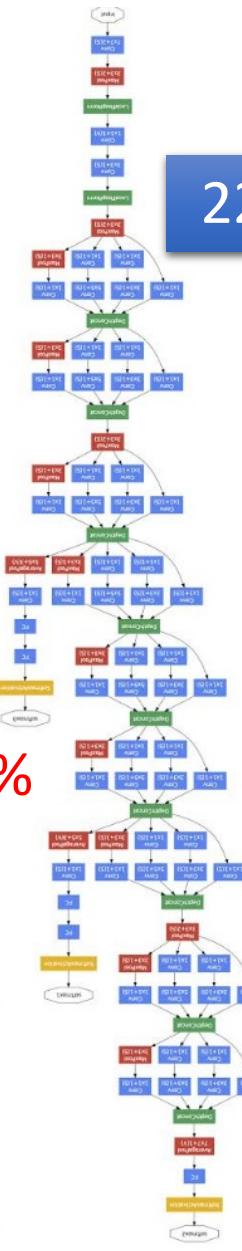


VGG (2014)

19 layers

6.7%

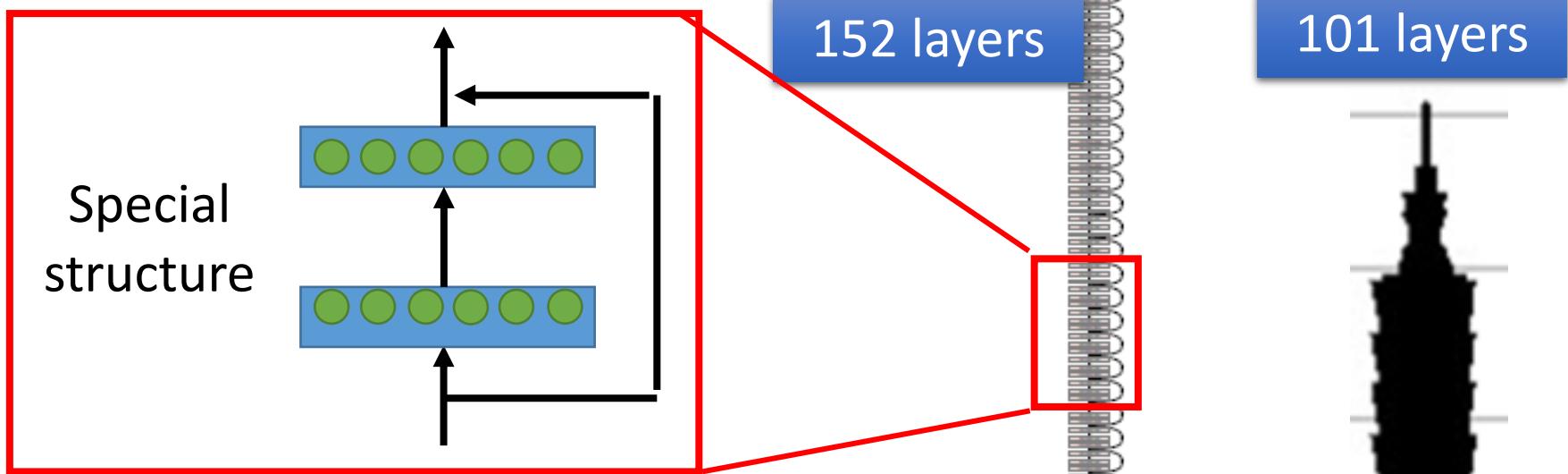
1



GoogleNet (2014)

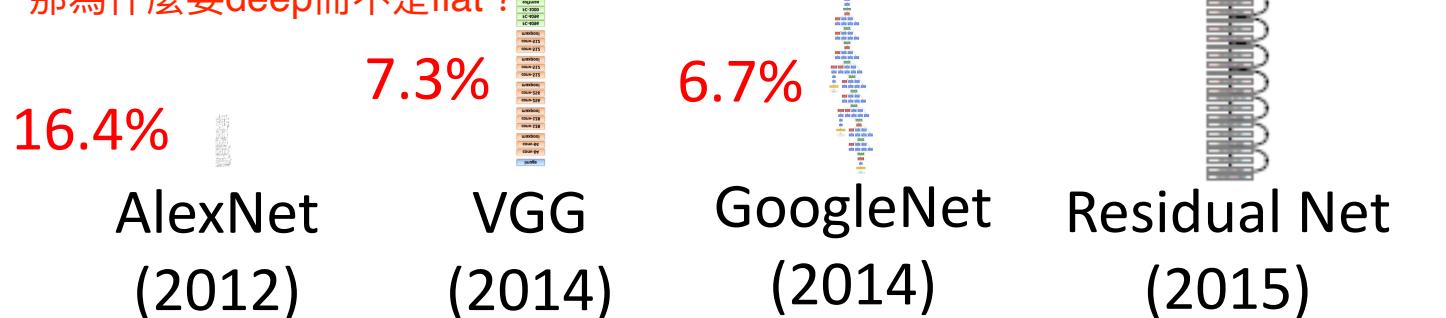
22 layers

Deep = Many hidden layers



Why we want “*Deep*” network,
not “*Fat*” network?

不是說用一排的RELU or sigmoid就可以模擬出任何函數嗎？
那為什麼要deep而不是flat？



Why don't we go deeper?

- 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

Why don't we go deeper?

- 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	4 layer
2017 – 2020	0.28k	0.18k	0.14k	0.10k
2021	0.43k	0.39k	0.38k	0.44k

Better on training data, worse on unseen data

→ Overfitting

Let's predict no. of views today!

- If we want to select a model for predicting no. of views today, which one will you use?

	1 layer	2 layer	3 layer	4 layer
2017 – 2020	0.28k	0.18k	0.14k	0.10k
2021	0.43k	0.39k	0.38k	0.44k

We will talk about model selection next time. ☺

To learn more

Basic Introduction



<https://youtu.be/Dr-WRIEFefw>

Backpropagation
Computing gradients in
an efficient way



<https://youtu.be/ibJpTrp5mcE>