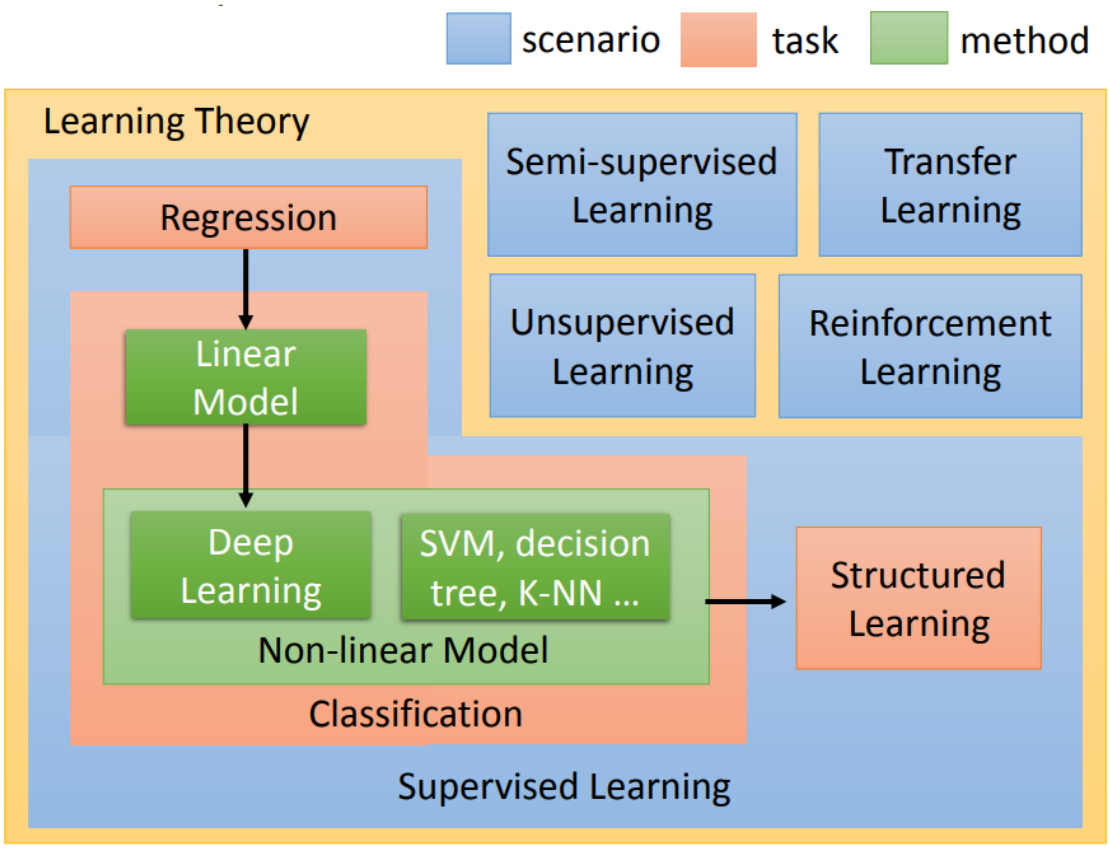
小贴士



大家注意一下这个不同的方块，我是用不同的颜色来表示。同样的颜色不同的方块是同一个类型的，这边的蓝色的方块，指的是学习的情景，通常学习的情景是你没有办法控制的。比如，因为我们没有data做监督学习，所以我们才做reinforcement learning。现在因为Alpha Go比较火，所以Alpha Go中用到的reinforcement learning会被认为比较潮。所以说有学生去面试，说明自己是做监督学习的，就会被质疑为什么不做reinforcement learning。那这个时候你就应该和他说，如果我今天可以监督学习，其实就不应该做reinforcement learning。reinforcement learning就是我们没有办法做监督学习的时候，我们才做reinforcement learning。红色的是指你的task，你要解的问题，你要解的这个问题随着你用的方程的不同，有regression、有classification、有structured。所以在不同的情境下，都有可能要解这个task。最后，在这些不同task里面有不同的model，用绿色的方块表示。

找函数

1. Function with Unknown Parameters

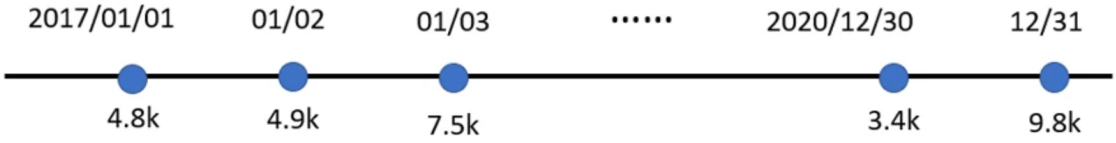
$y = f($  $)$

2. Define Loss from Training Data

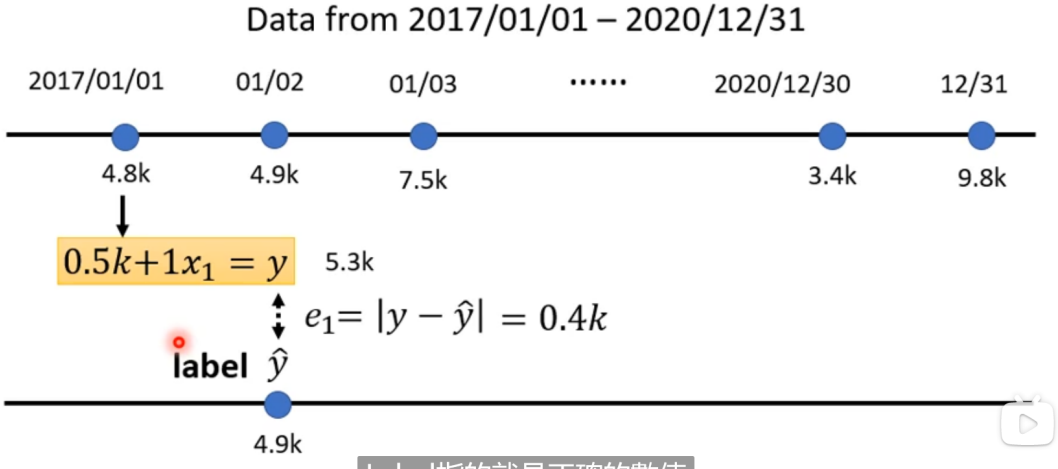
- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.

$L(0.5k, 1)$ $y = b + wx_1 \longrightarrow y = 0.5k + 1x_1$ How good it is?

Data from 2017/01/01 – 2020/12/31



预测明天的youtube订阅人数：

1. gess f :
   - **Model**: $y = b + wx_1$ base on domain(field) knowledge
   - y: predict number, x_1: past number(**feature**)
   - w, b are unknown parameters (learn from data)(**weight,bias**)
   - 

   Label指的就是正確的數值

2. def $Loss(b, w)$:
   - **label** : $\hat{y}$ ， $e_1 = |y - \hat{y}|$ (MAE) ， $e_1' = (y - \hat{y})^2$(MSE)

- $\mathbf{L} = \frac{1}{n}\sum_1^n e_i$

3. Optimization:**Gradient Decent**
   - find: $w^*, b^* = \arg\min_{w,b} L$
   - Pick an initial value w^0
   - Compute $\frac{\partial L}{\partial w}\Big|_{w=w^0}$
     - **hyperparameters** : η : learning rate
   - Update w itertively

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

   - ##local minia & global minia

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

- (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

Machine Learning is so simple ……

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

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

$y = b + wx_1$

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

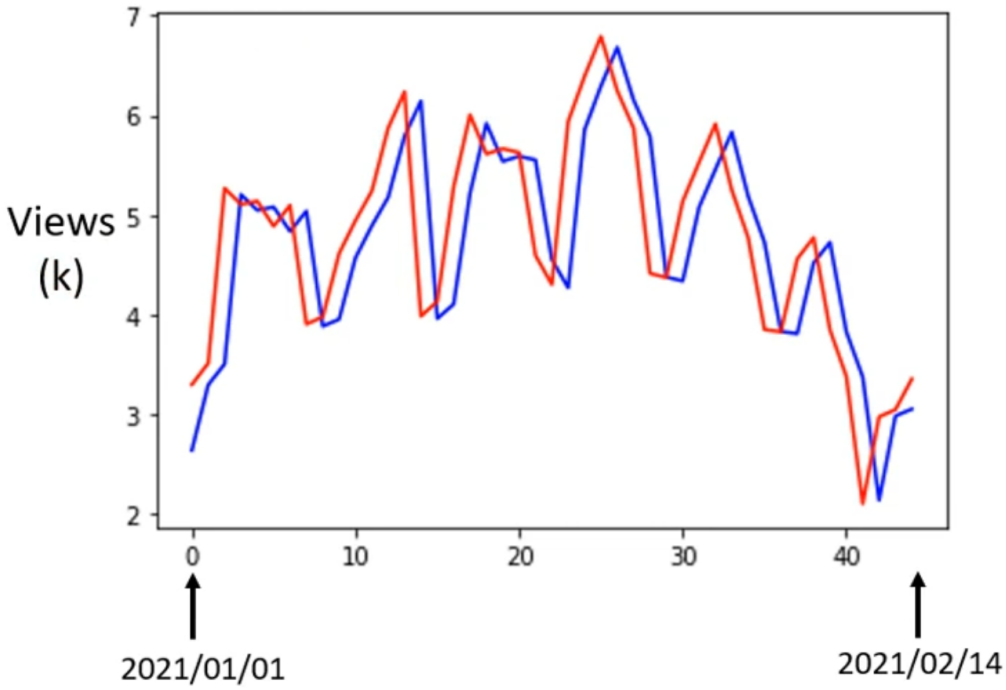
How about data of 2021 (**unseen during training**)?

$L' = 0.58k$

$y = 0.1k + 0.97x_1$

Red: real no. of views

blue: estimated no. of views



- **Nearly move the yesterday's data**

- **periodic**
- **通常一个模型的修改，来自于你对问题的理解**

$$y = b + wx_1$$

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 $L' = 0.46k$

$$y = b + \sum_{j=1}^{56} w_j x_j$$

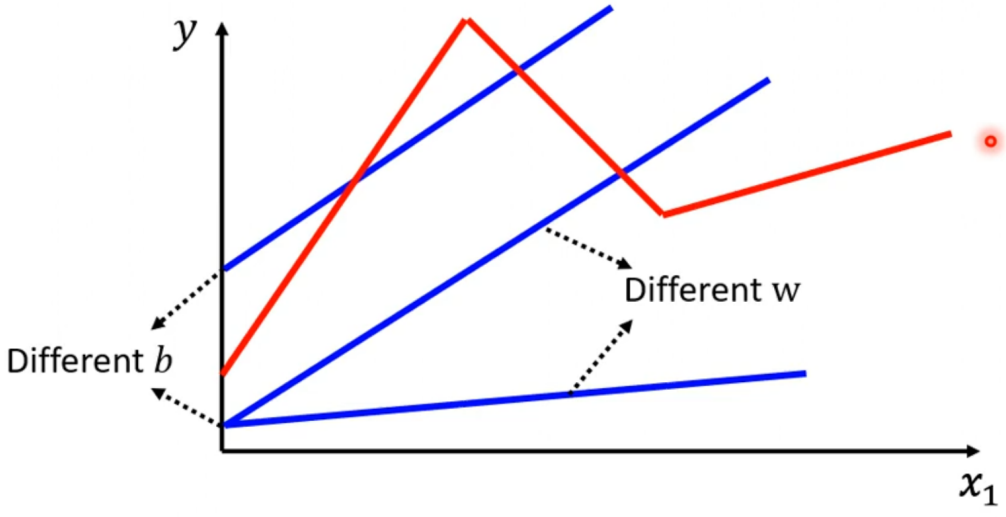
2017 - 2020 $L = 0.32k$

2021 $L' = 0.46k$

考慮更多天沒有辦法再更進步了

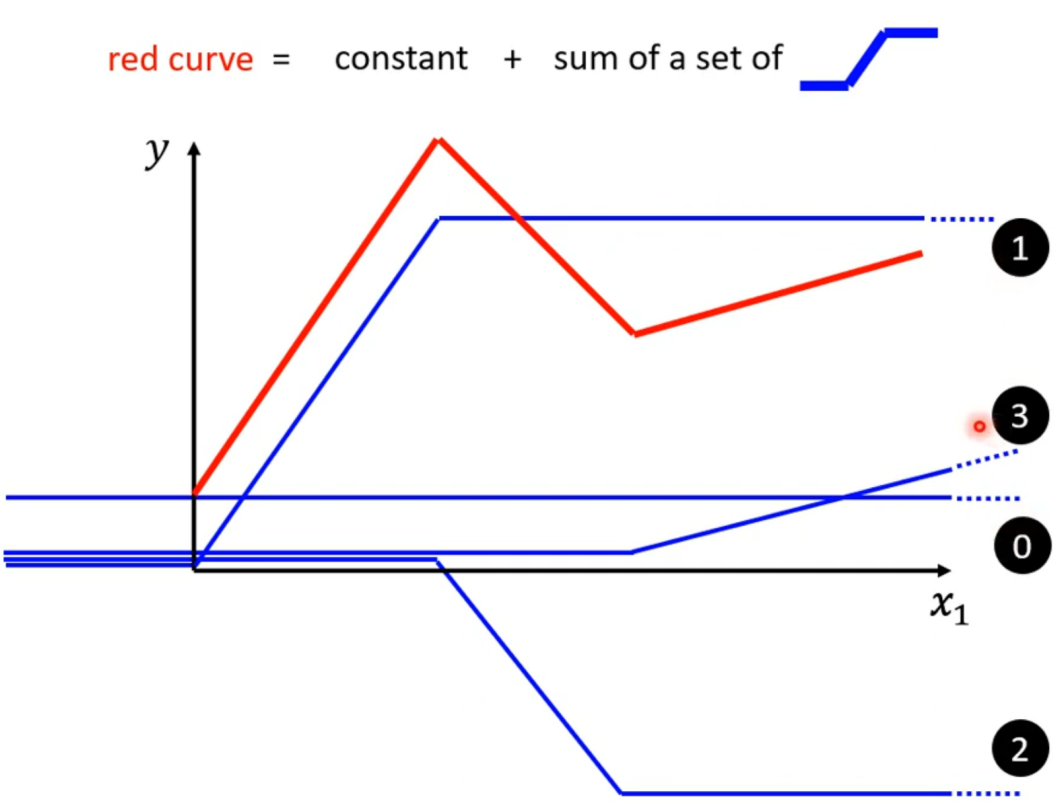
Linear Model

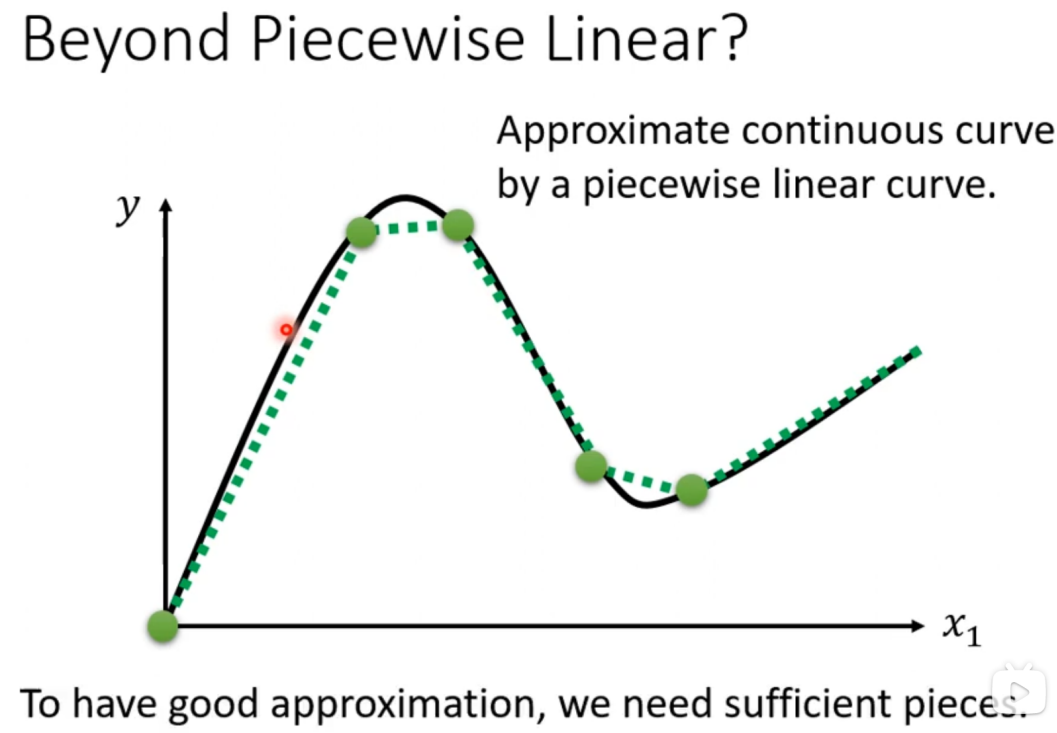
- too simple
- Linear models are too simple … we need more sophisticated modes.



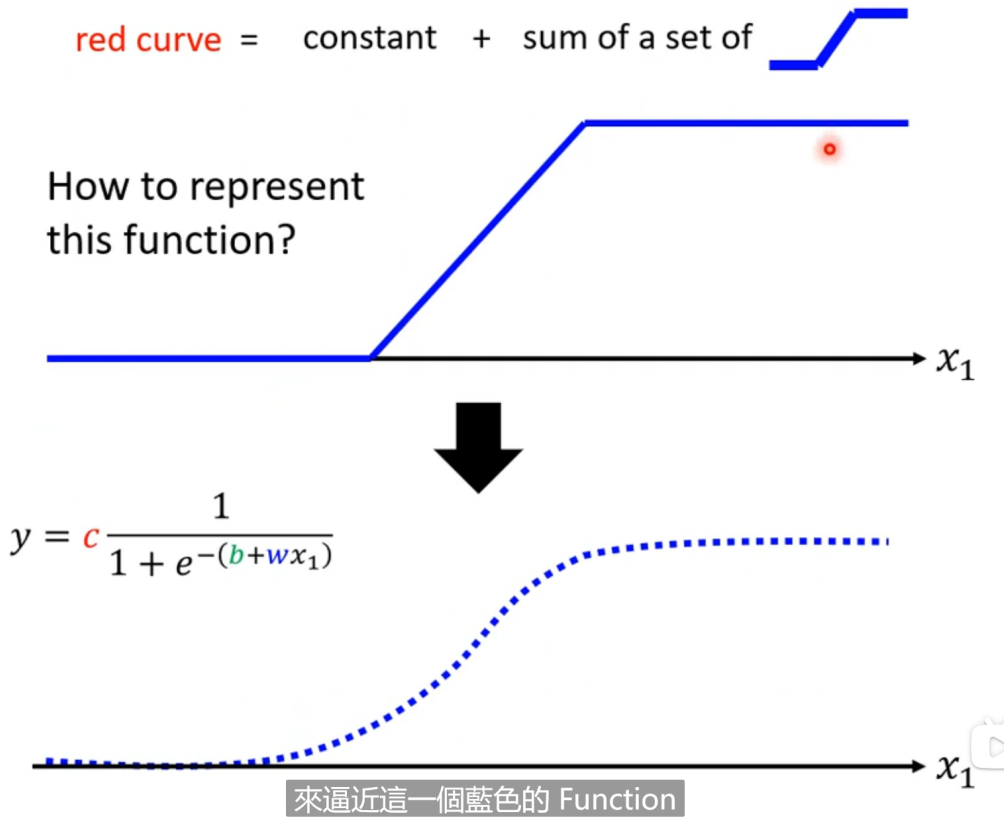
Linear models have severe limitation.

- **Model's Bias** -->more flexible model

- 

- (Fourier)
- **All piecewice Linear Curves= constant+sum of a set of "blue function"**
- 

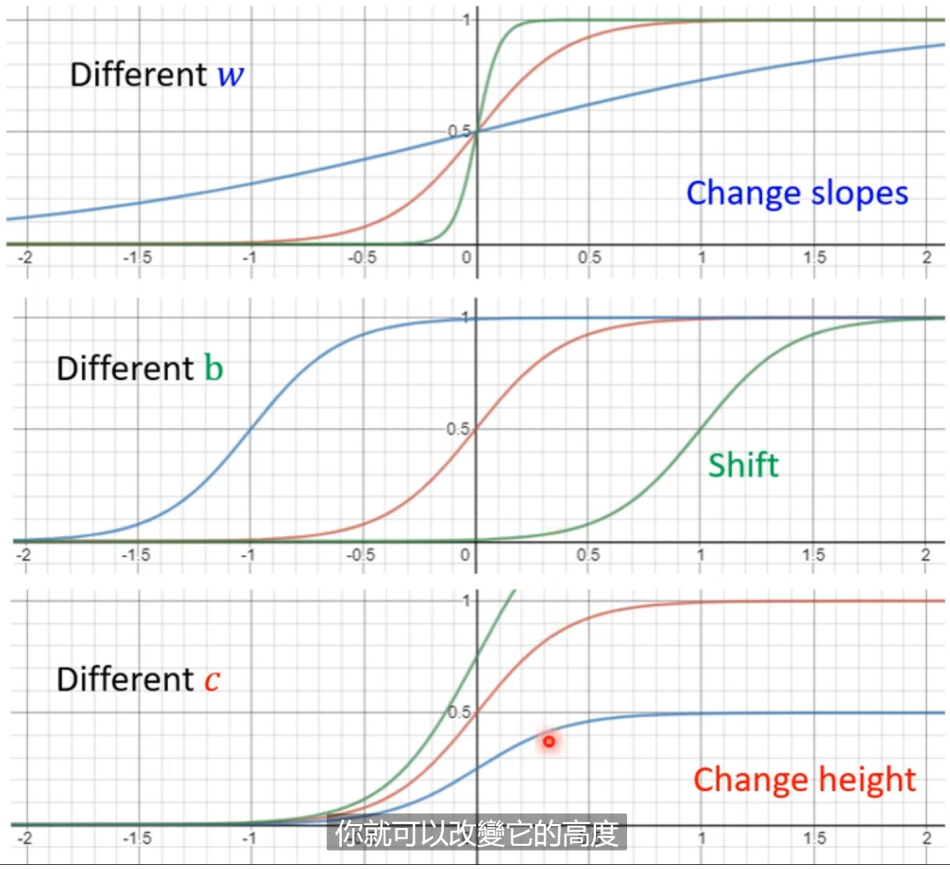
- Continuous data discretization（离散化）
-



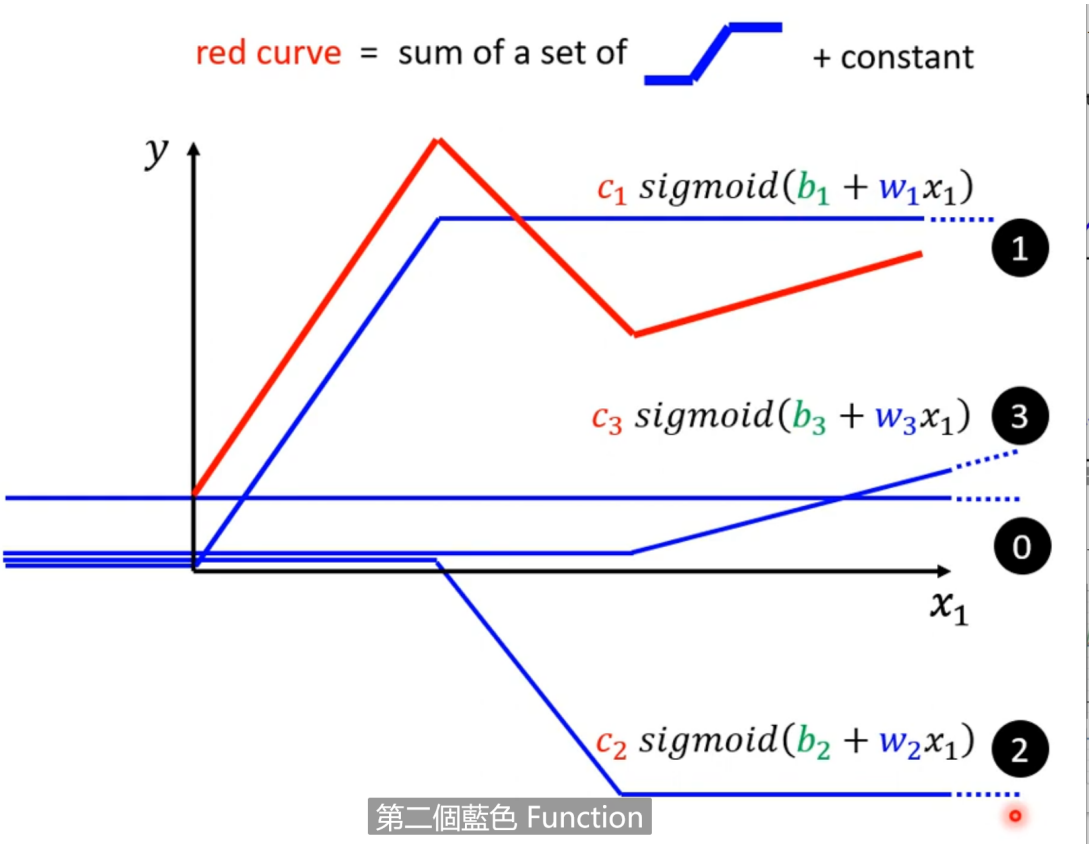
來逼近這一個藍色的 Function

- sigmoid function

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



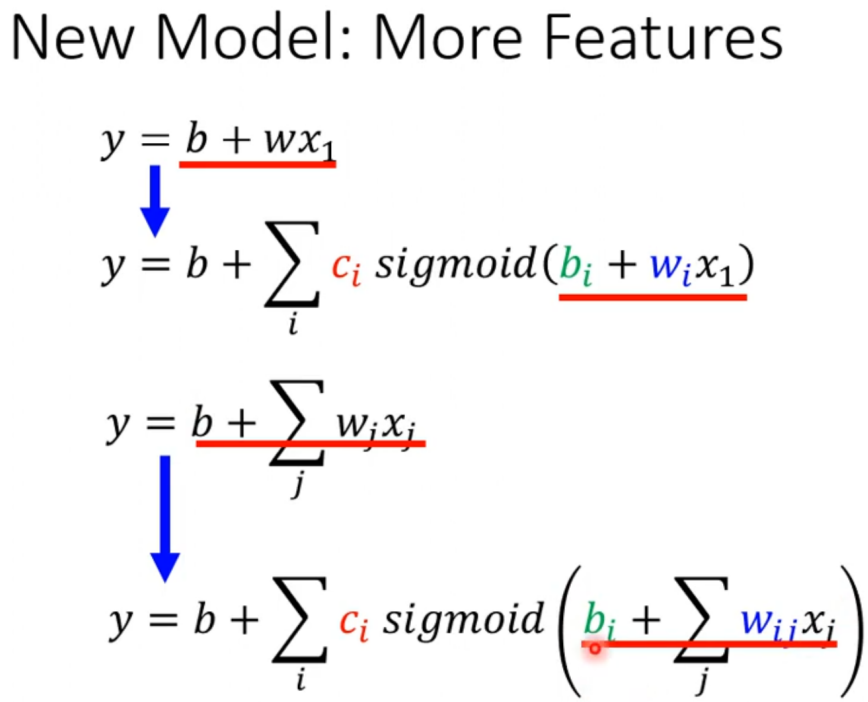
你就可以改變它的高度



第二個藍色 Function

$$y = b + \sum_i c_i\, sinmoid(b_i + w_i x_1)$$

通过sigmoid拟合出任意复杂函数



都有不同的 bi 不同的 wij

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

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

j: 1,2,3 no. of features

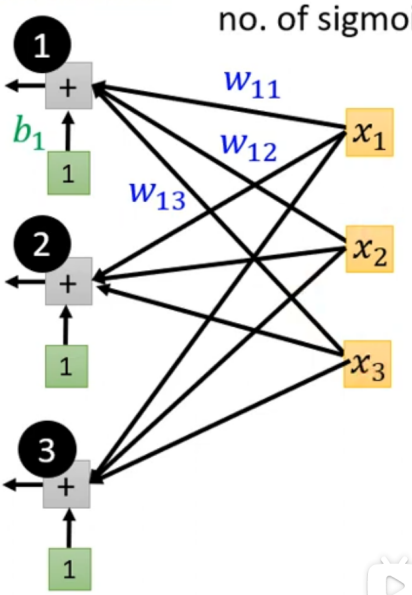
i: 1,2,3 no. of sigmoid

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

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

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

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



那這個 x1 x2 跟 x3 和 r1 r2 r3

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

i: 1,2,3

j: 1,2,3

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

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

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

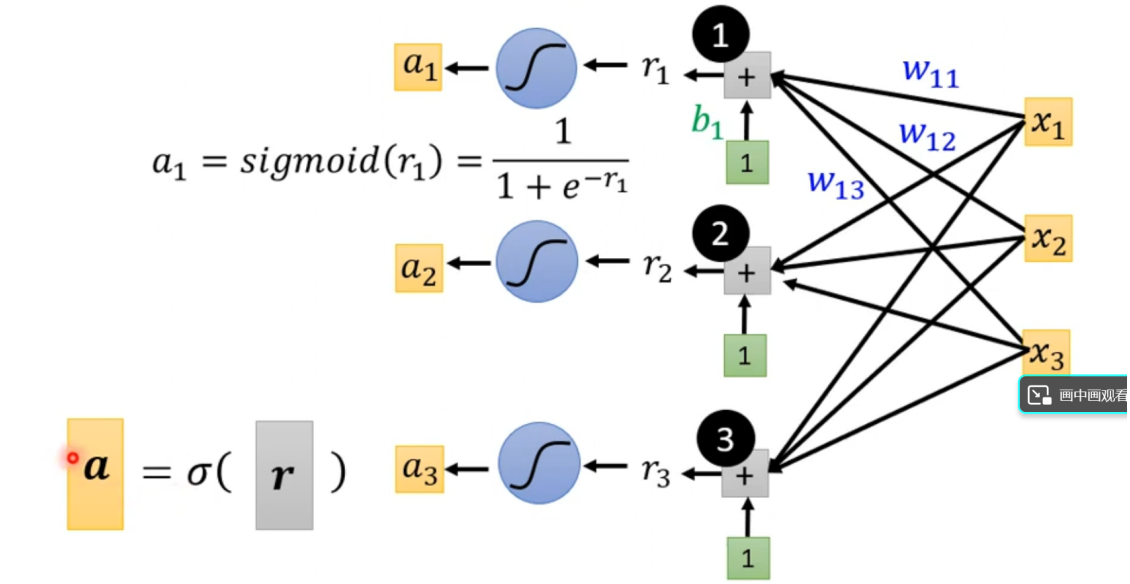
$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} + \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boldsymbol{r} = \boldsymbol{b} + W \quad \boldsymbol{x}$$

只是表示的方式不一樣而已

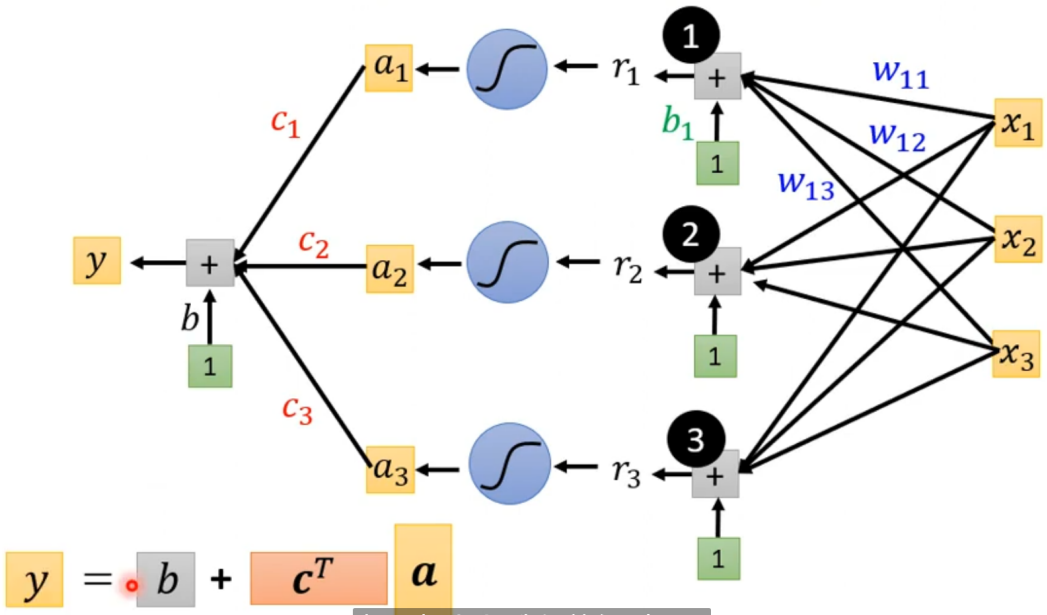
$$\mathbf{r} = \mathbf{b} + \mathbf{w} \cdot \mathbf{x}$$

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



然後呢 所以我們得到了 a 這個向量

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



好再加上 b 我們就得到了 y

$$\begin{cases} \mathbf{y} = \mathbf{b} + \mathbf{c}^{\mathbf{T}} \cdot \mathbf{a} \\ \mathbf{a} = \sigma(\mathbf{r}) \end{cases}$$

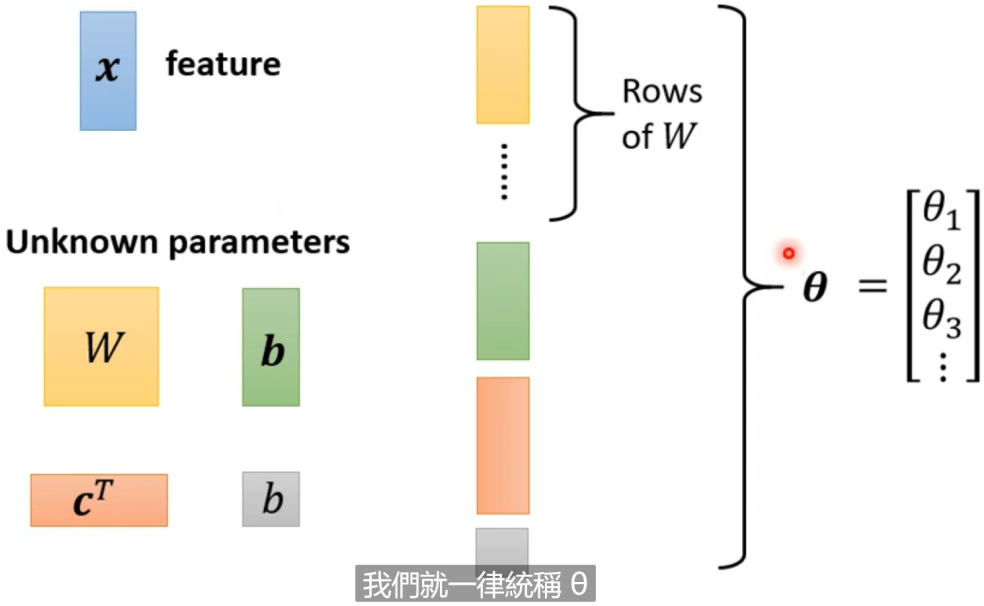
$$y = b + c^T \sigma(b + W x)$$

比較有彈性的這個 Function

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

Function with unknown parameters

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

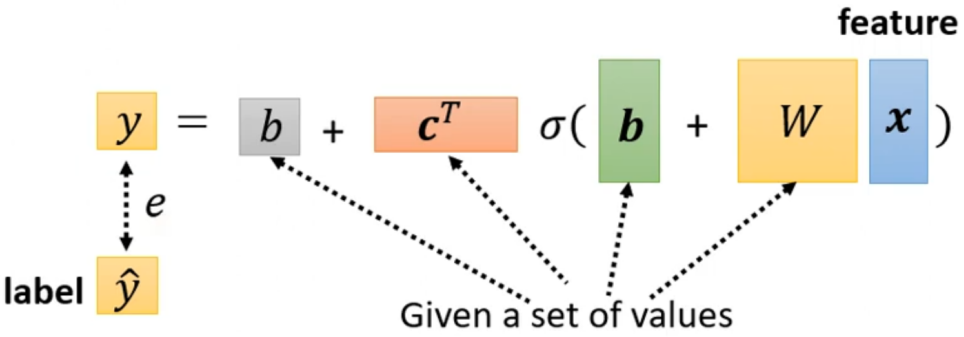


我們就一律統稱 θ

- **Loss**
-

Loss

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



Loss: $L = \frac{1}{N}\sum_{n} e_n$

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

- **gradient** $= \Delta L(\theta^0)$, $\theta^1 \leftarrow \theta^0 - \eta g$

Optimization of New Model

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

- (Randomly) Pick initial values θ^0

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

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

那假設你這邊參數有 1000 個

Optimization of New Model

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

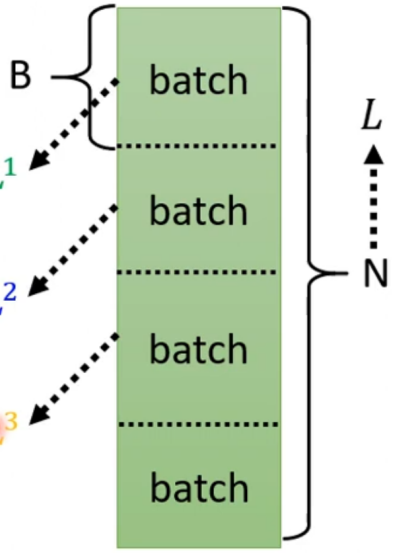
- (Randomly) Pick initial values θ^0
- Compute gradient $g = \nabla L^1(\theta^0)$ L^1

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

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

  update $\theta^3 \leftarrow \theta^2 - \eta g$

1 **epoch** = see all the batches once



把所有的 Batch 都看過一遍

- Random Gradient Decent
  - 1 **epoch** = traverse each batches once (历元、时期)
  - HyperParamet: Learning rate, Sigmoid, Batch Size ...
  - ***Example 1***

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

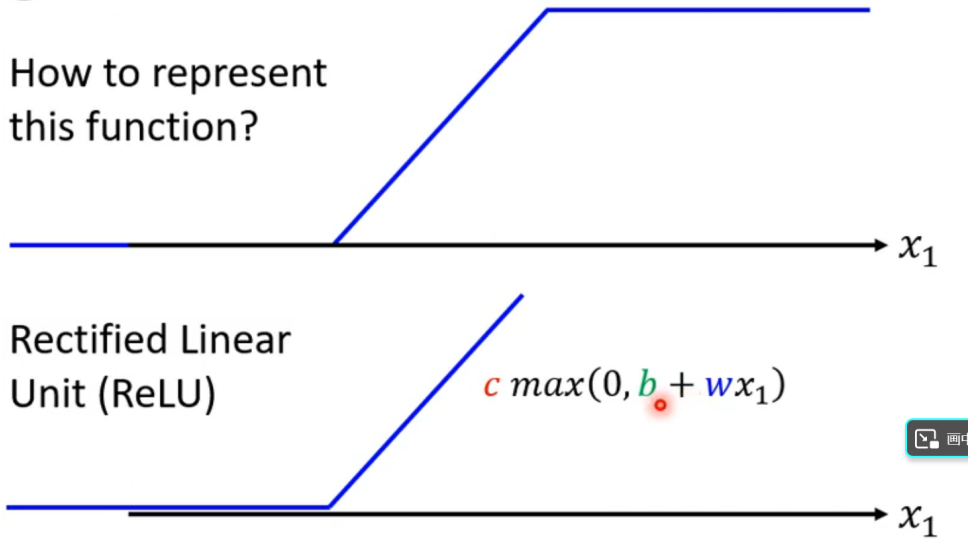
    How many update in **1 epoch**?

    1,000 updates

    B
  - number of epoch = number of batches

- **ReLU**

Sigmoid → ReLU



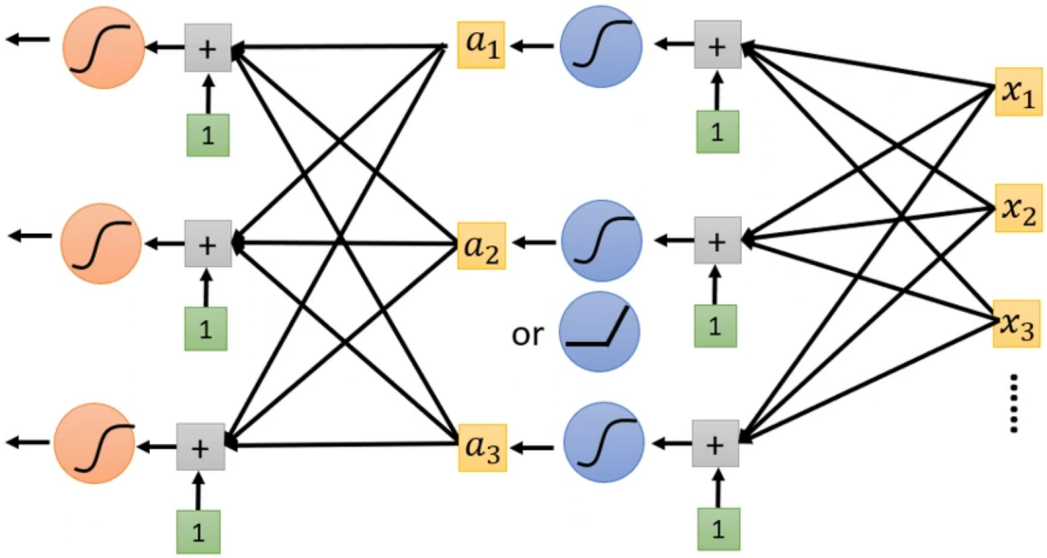
$$y = b + \sum_{2i} c_i max\{0, (b_i + \sum_j w_{ij} x_j)\}$$

- 2ReLU -> hard Sigmoid
- **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



- **MLP**
  - result:
  -

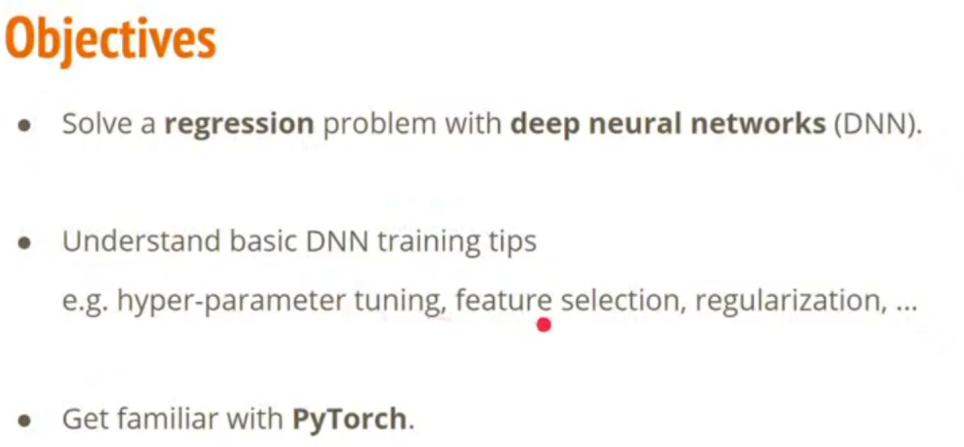
- Deep Learning

We use ReLu or Sigmoid to fitting a flexible Fuction, but why it needs deeper layers instead of wider layers ?

- over fitting

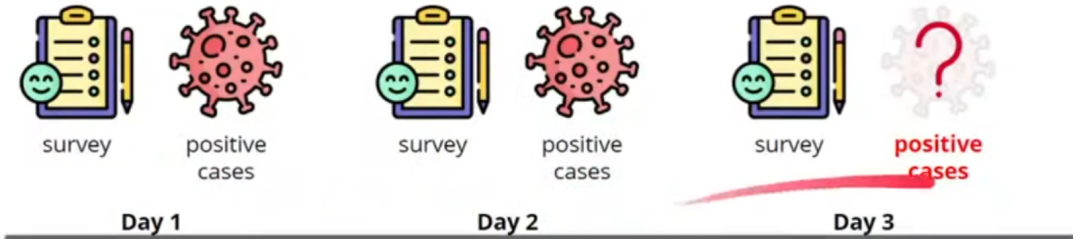
HW1：COVID-19 Cases Prediction

> ML2021Spring-hw1 | Kaggle



Task Description

- Given survey results in the **past 3 days** in a specific **state** in U.S., then predict the percentage of **new tested positive cases** in the 3rd day.



Data -- Delphi's COVID-19 Surveys

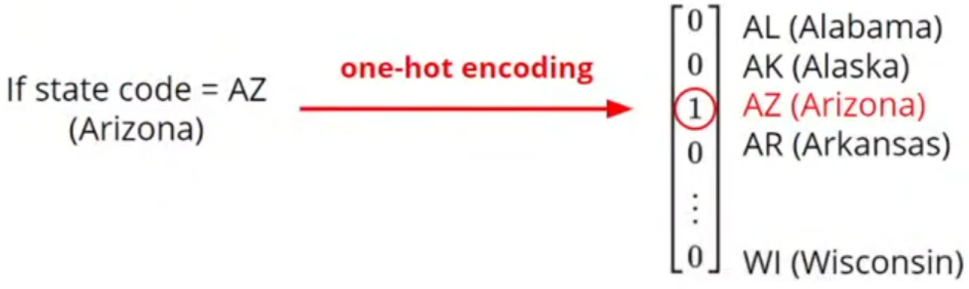
- **States** (40, encoded to **one-hot** vectors)
  - e.g. `AL, AK, AZ, ...`
- **COVID-like illness** (4)
  - e.g. `cli,ili` (influenza-like illness), ...
- **Behavior Indicators** (8)
  - e.g. `wearing_mask, travel_outside_state, ...`
- **Mental Health Indicators** (5)
  - e.g. `anxious, depressed, ...`
- **Tested Positive Cases** (1)
  - **`tested_positive`** (this is what we want to predict)

Data -- One-hot Vector

- **One-hot vectors**:

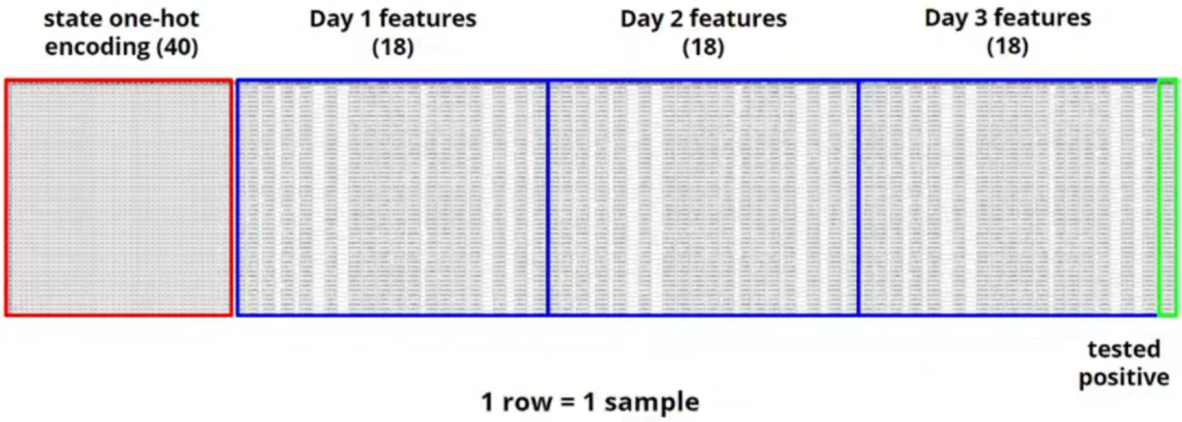
  Vectors with **only one element equals to one** while others are zero.

  Usually used to encode discrete values.



Data -- Training

`covid.train.csv` (2700 samples)



Evaluation Metric

- **Root Mean Squared Error (RMSE)**

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(f(\mathbf{x}^n) - \hat{y}^n\right)^2}$$

your model

input features (testing data)

ground truth label (correct answer)

Grading

Simple baseline (public)	+1 pt (sample code)
Simple baseline (private)	+1 pt (sample code)
Medium baseline (public)	+1 pt
Medium baseline (private)	+1 pt
Strong baseline (public)	+1 pt
Strong baseline (private)	+1 pt
Upload code to NTU COOL	+4 pts

Total: **10** pts

Grading -- Bonus

- If you got 10 points, we make your code **public** to the whole class.

- In this case, if you also submit **a PDF report briefly describing your methods** (<100 words in English), you get a bonus of **0.5 pt**.
  (your report will also be available to all students)

- Report template

Code Submission

- **NTU COOL** (4pts)
  - Compress your code and report into

    `<student ID>_hw1.zip`

    `e.g. b06901020_hw1.zip`

  - We can only see your last submission.
  - Do not submit your model or dataset.
  - If your code is not reasonable, your semester grade x 0.9.

Hints

- **Simple Baseline**
  - Sample code
- **Medium Baseline**
  - *Feature selection*: 40 states + 2 `tested_positive`
    (will be demonstrated in class)
- **Strong Baseline**
  - *Feature selection* (what other features are useful?)
  - *DNN architecture* (layers? dimension? activation function?)
  - *Training* (mini-batch? optimizer? learning rate?)
  - *L2 regularization*
  - There are some mistakes in the sample code, can you find them?