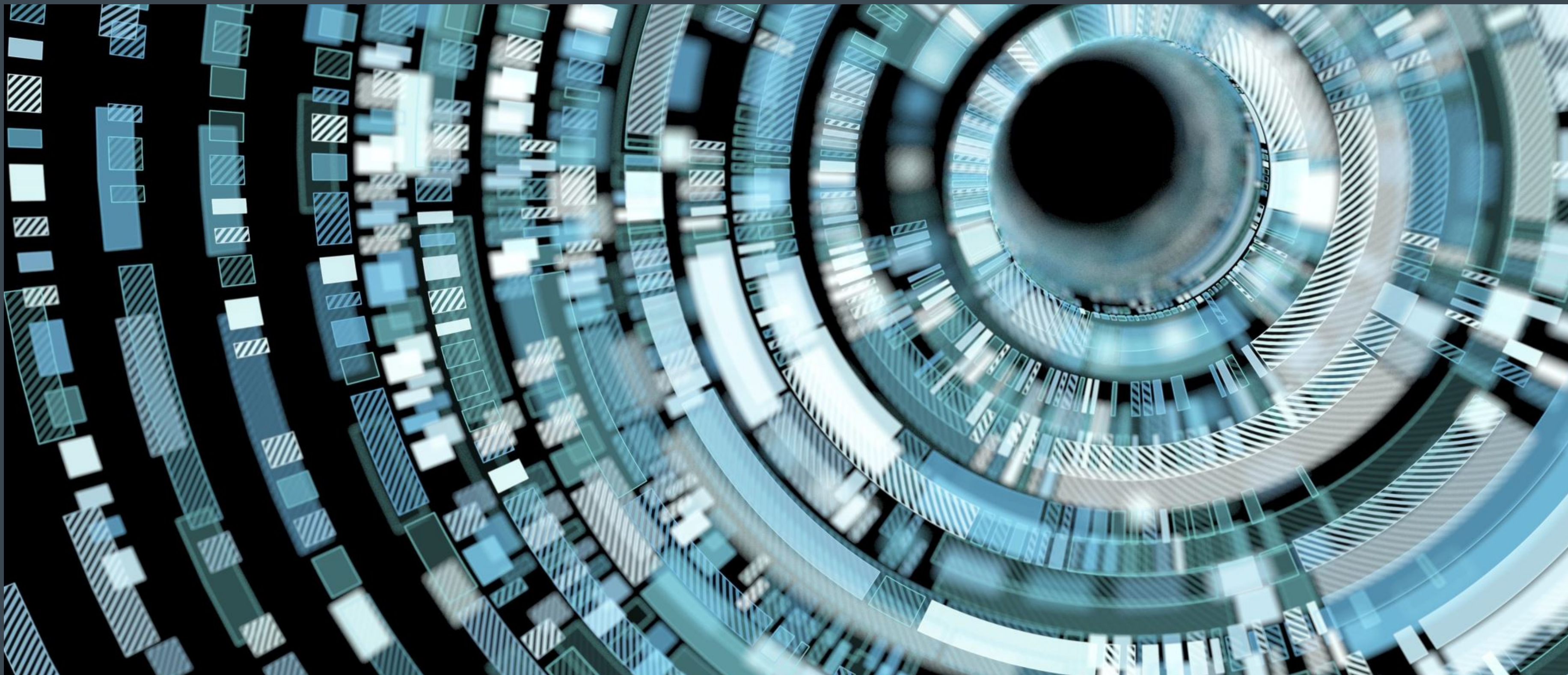
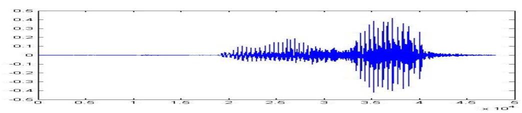


Introduction of Machine / Deep Learning

Hung-yi Lee 李宏毅

Machine Learning
≈ Looking for Function

- Speech Recognition

$f($  $)=$ “How are you”

- Image Recognition

$f($  $)=$ “Cat”

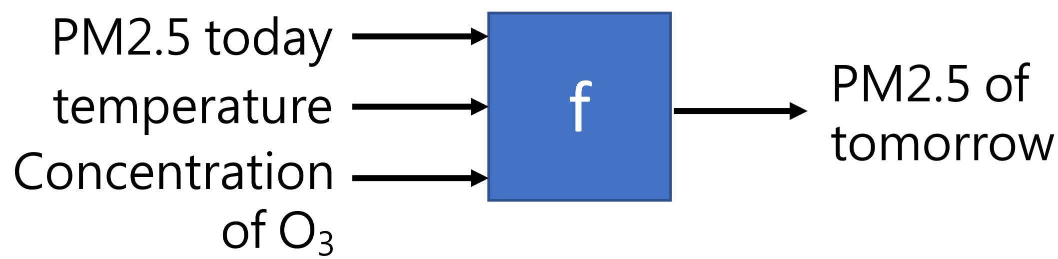
- Playing Go

$f($  $)=$ “5-5” (next move)

Different types of Functions

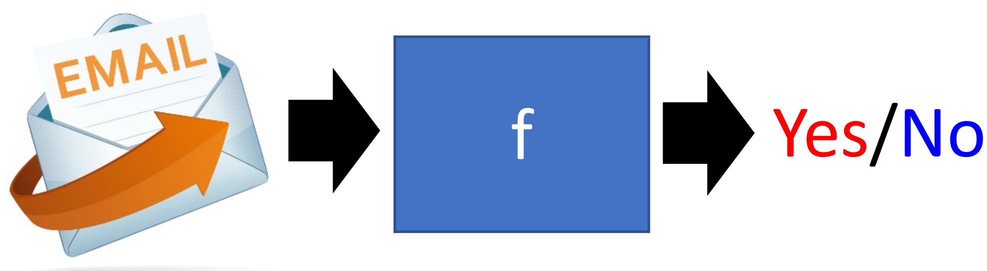
Regression: The function outputs a scalar.

Predict PM2.5



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

Spam filtering



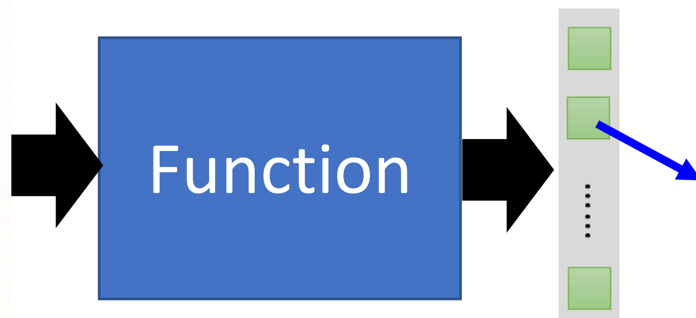
Different types of Functions

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

Each position is a class

(19 x 19 classes)

Playing GO



a position on the board

Next move

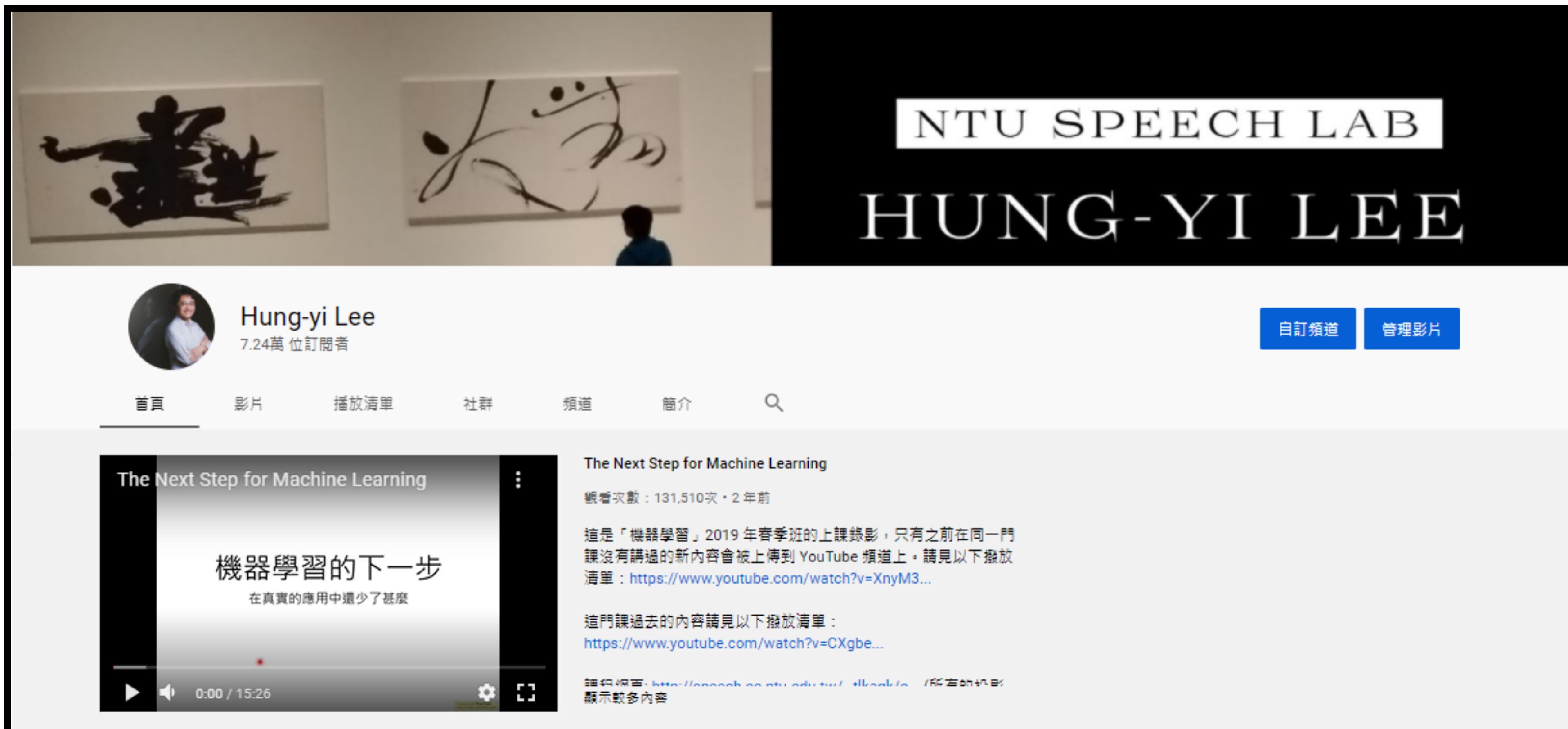
Structured Learning

create something with structure (image, document)

Regression, Classification

How to find a function?
A Case Study

YouTube Channel



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

The function we want to find …

$y = f($ … $)$

no. of views on 2/26

頻道

Hung-yi Lee

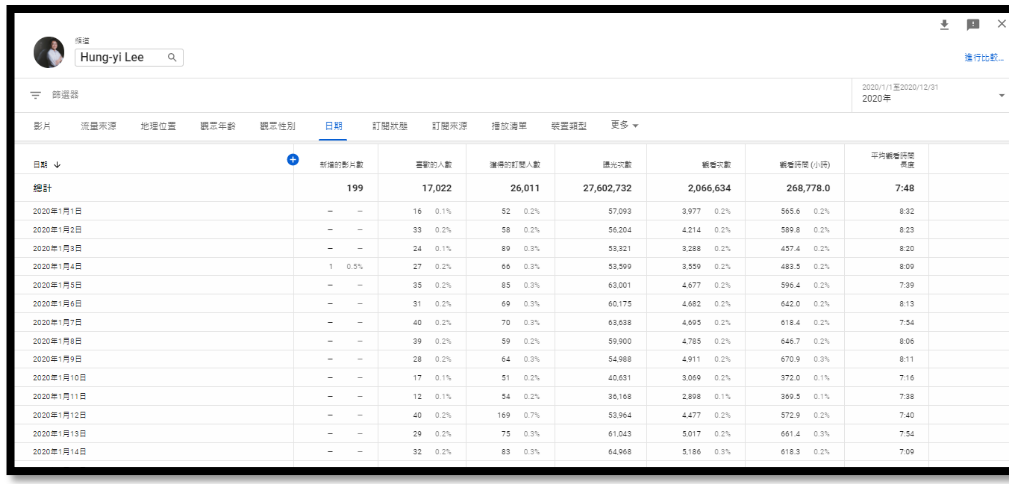
篩選器

影片 流量來源 地理位置 觀眾年齡 觀眾性別 日期 訂閱狀態 訂閱來源 播放清單

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

$y = f($



頻道
Hung-yi Lee
進行比較...
篩選器
2020/1/1至2020/12/31
2020年
影片 流量來源 地理位置 觀眾年齡 觀眾性別 日期 訂閱狀態 訂閱來源 播放清單 裝置類型 更多

日期 ↓	新增的影片數	喜歡的人數	獲得的訂閱人數	曝光次數	觀看次數	觀看時間 (小時)	平均觀看時間長度
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	3,288 0.2%	457.4 0.2%	8:20
2020年1月4日	1 0.5%	27 0.2%	66 0.3%	53,599	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%	63,638	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%	54,988	4,911 0.2%	670.9 0.3%	8:11
2020年1月10日	– –	17 0.1%	51 0.2%	40,631	3,069 0.2%	372.0 0.1%	7:16
2020年1月11日	– –	12 0.1%	54 0.2%	36,168	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 0.2%	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)

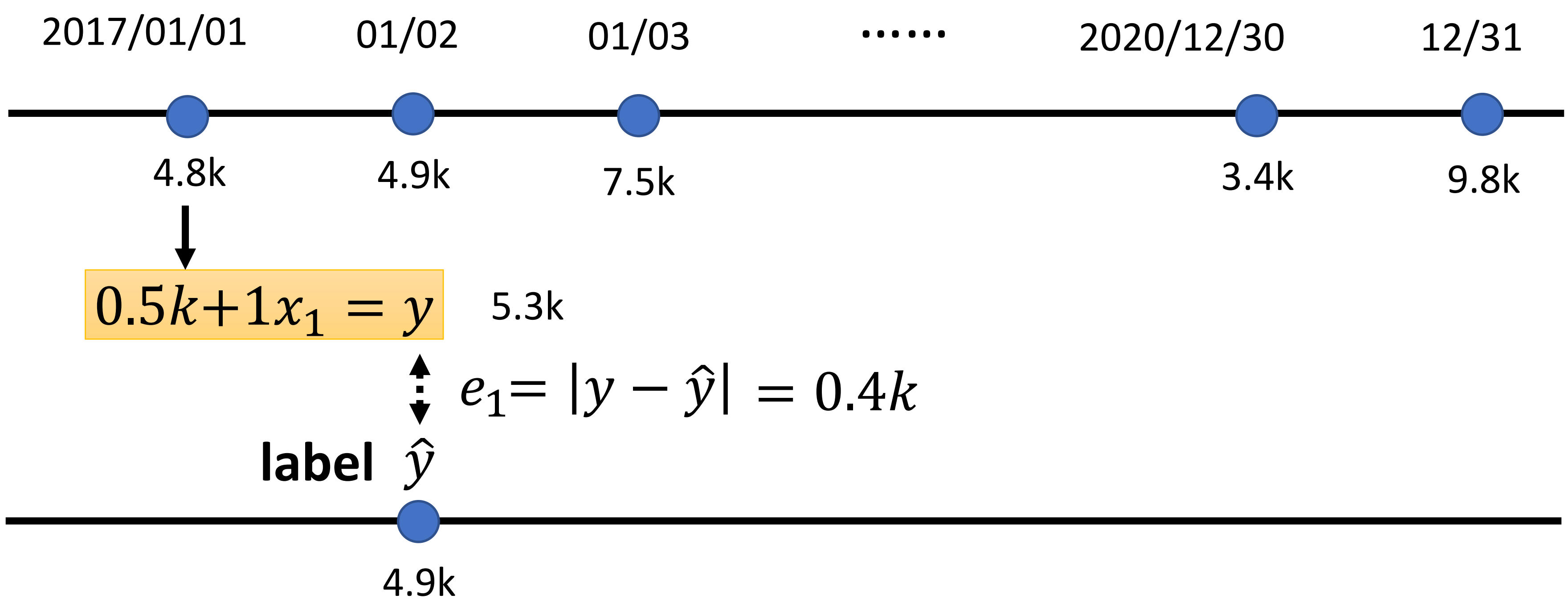
weight **bias**

2. Define Loss from Training Data

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

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

Data from 2017/01/01 – 2020/12/31

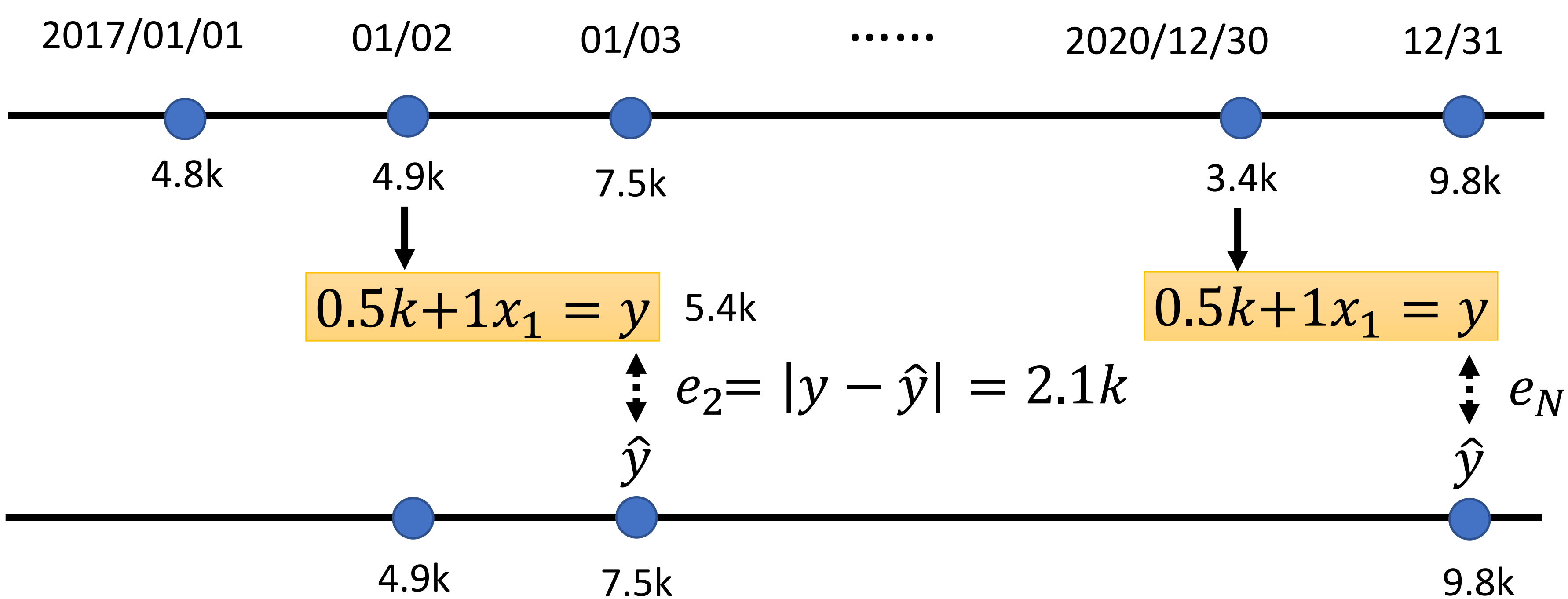


2. Define Loss from Training Data

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

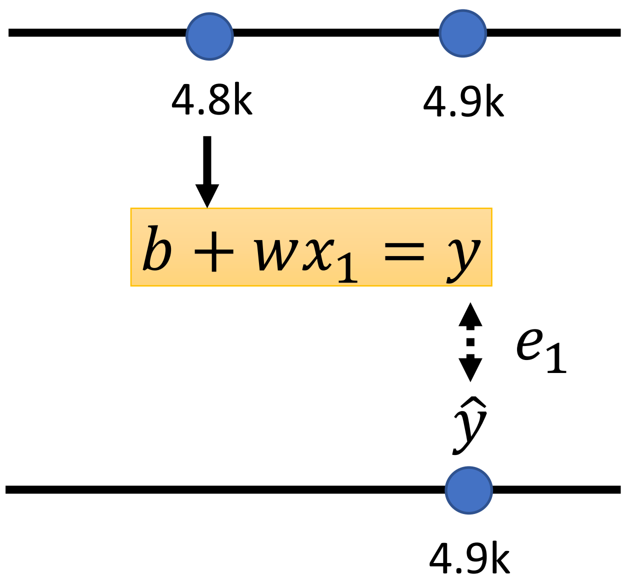
$L(0.5k, 1)$ $\quad y = b + wx_1 \longrightarrow y = 0.5k + 1x_1$ How good it is?

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.



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

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

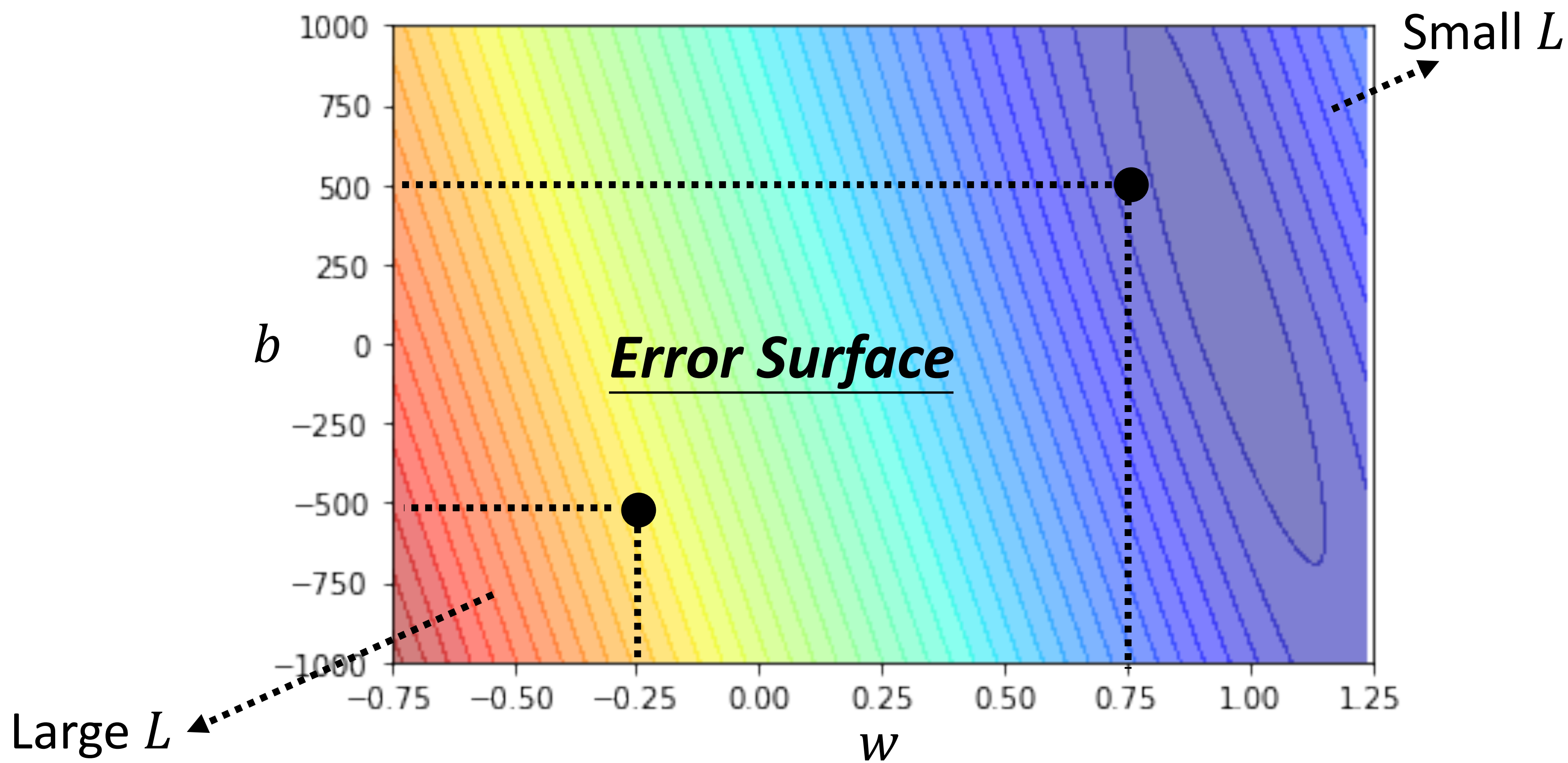
$e = (y - \hat{y})^2$ L is mean square error (MSE)

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

2. Define Loss from Training Data

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

Model $y = b + wx_1$

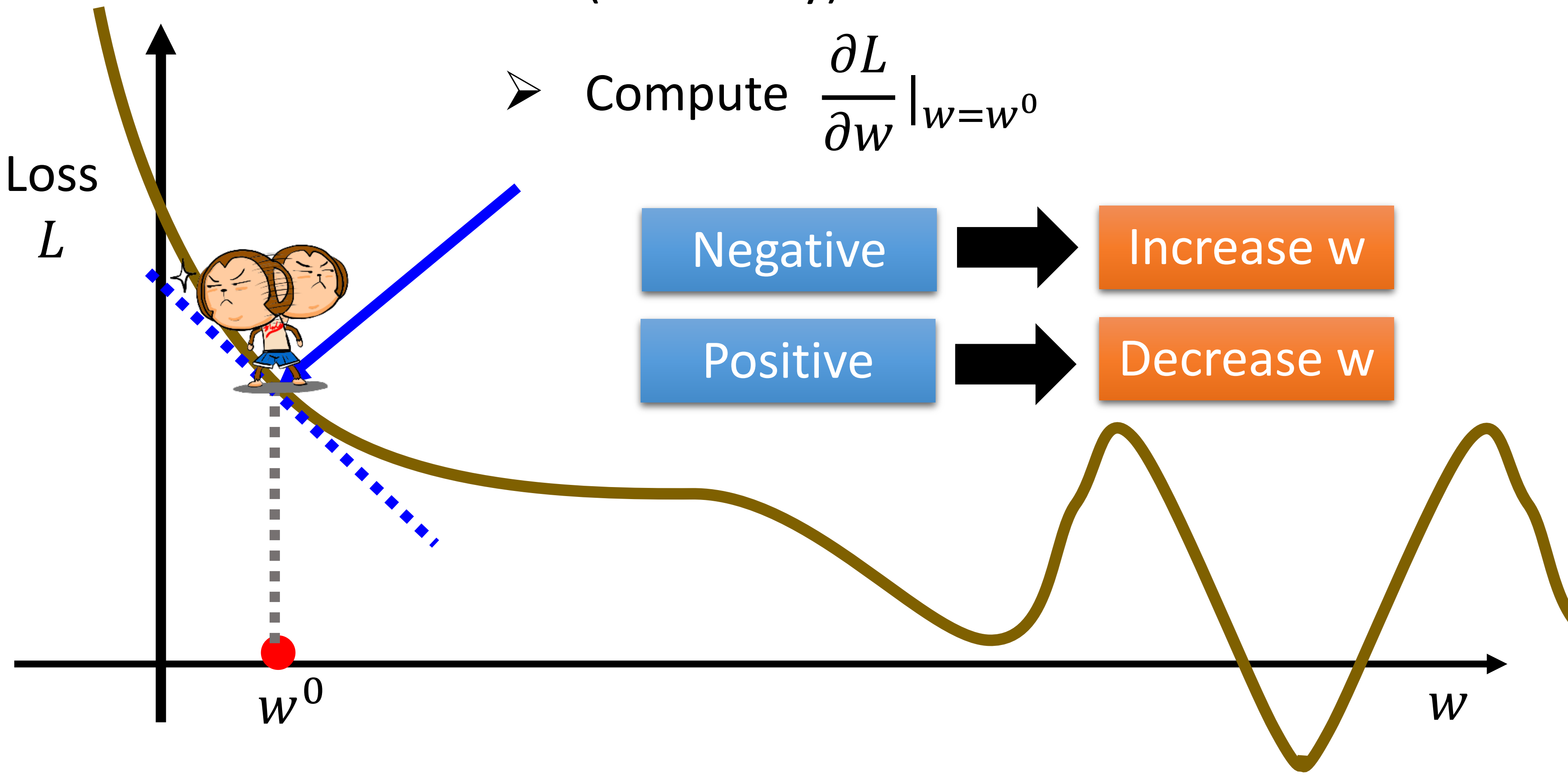


3. Optimization

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

Gradient Descent

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



Source of image: http://chico386.pixnet.net/album/photo/171572850

3. Optimization

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

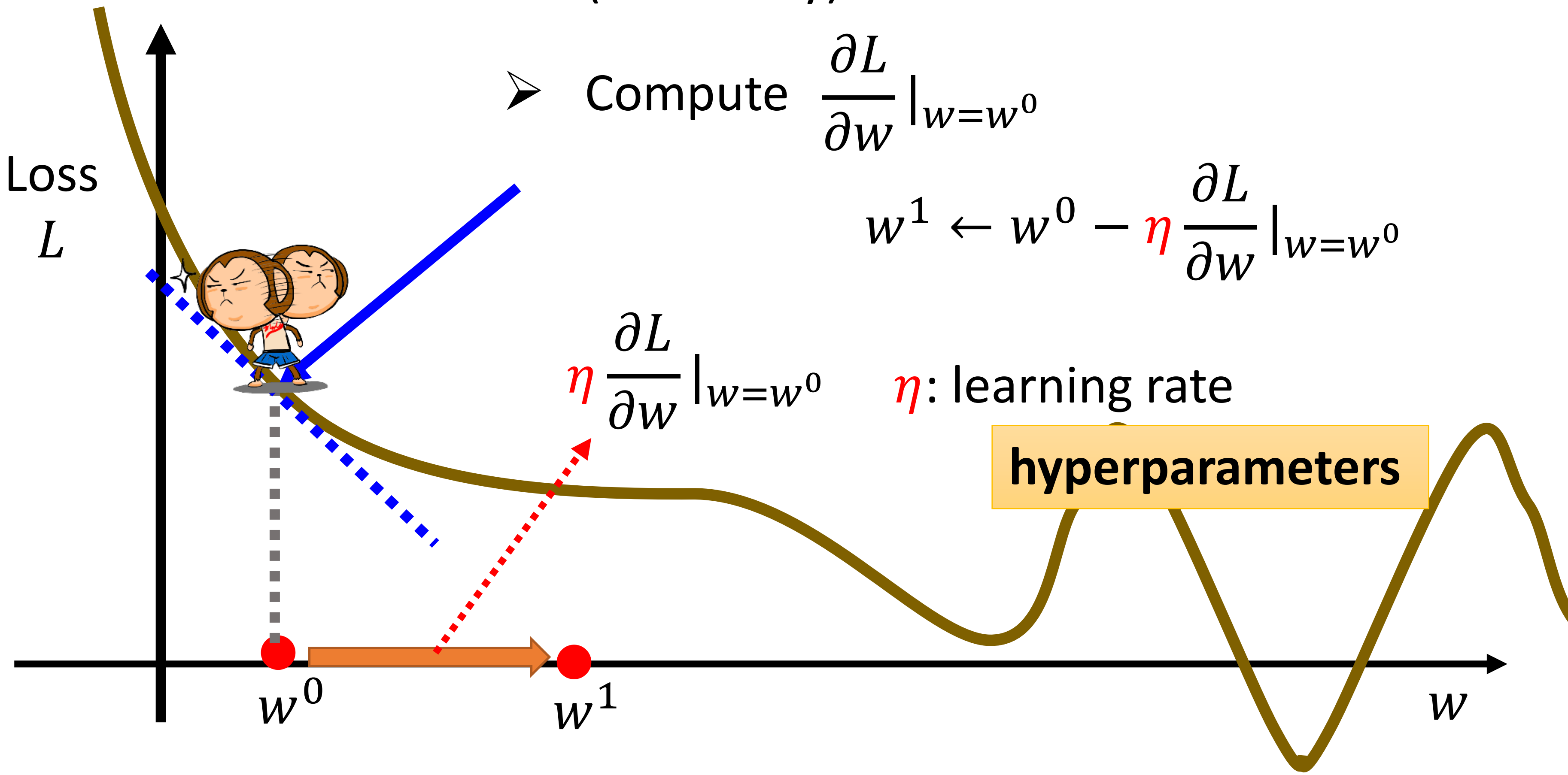
Gradient Descent

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

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

η: learning rate

hyperparameters



Source of image: http://chico386.pixnet.net/album/photo/171572850

3. Optimization

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

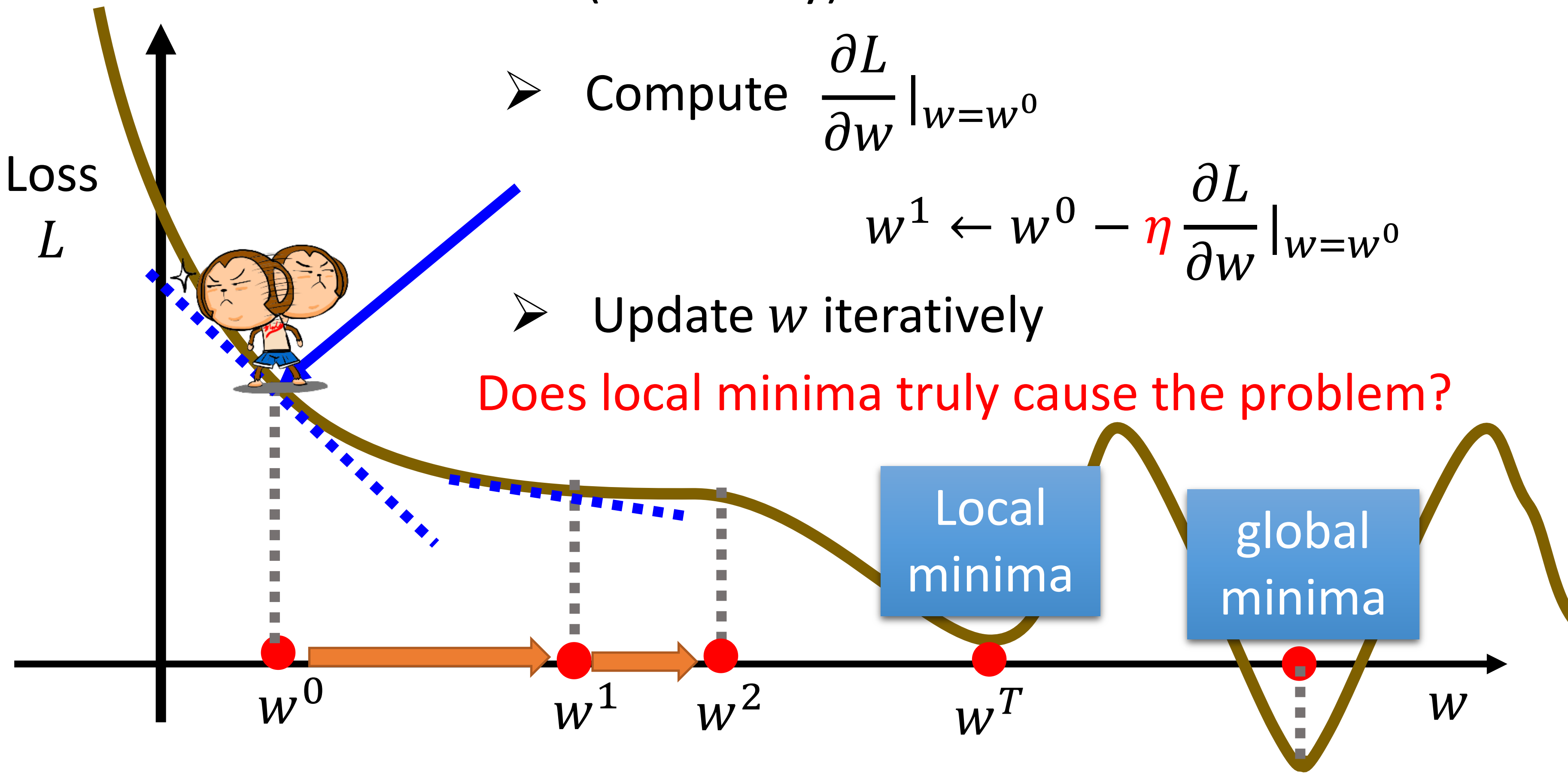
Gradient Descent

- (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?



3. Optimization

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

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

$$\boxed{\begin{array}{l} \dfrac{\partial L}{\partial w}|_{w=w^0,b=b^0} \\ \\ \dfrac{\partial L}{\partial b}|_{w=w^0,b=b^0} \end{array}}$$

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

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

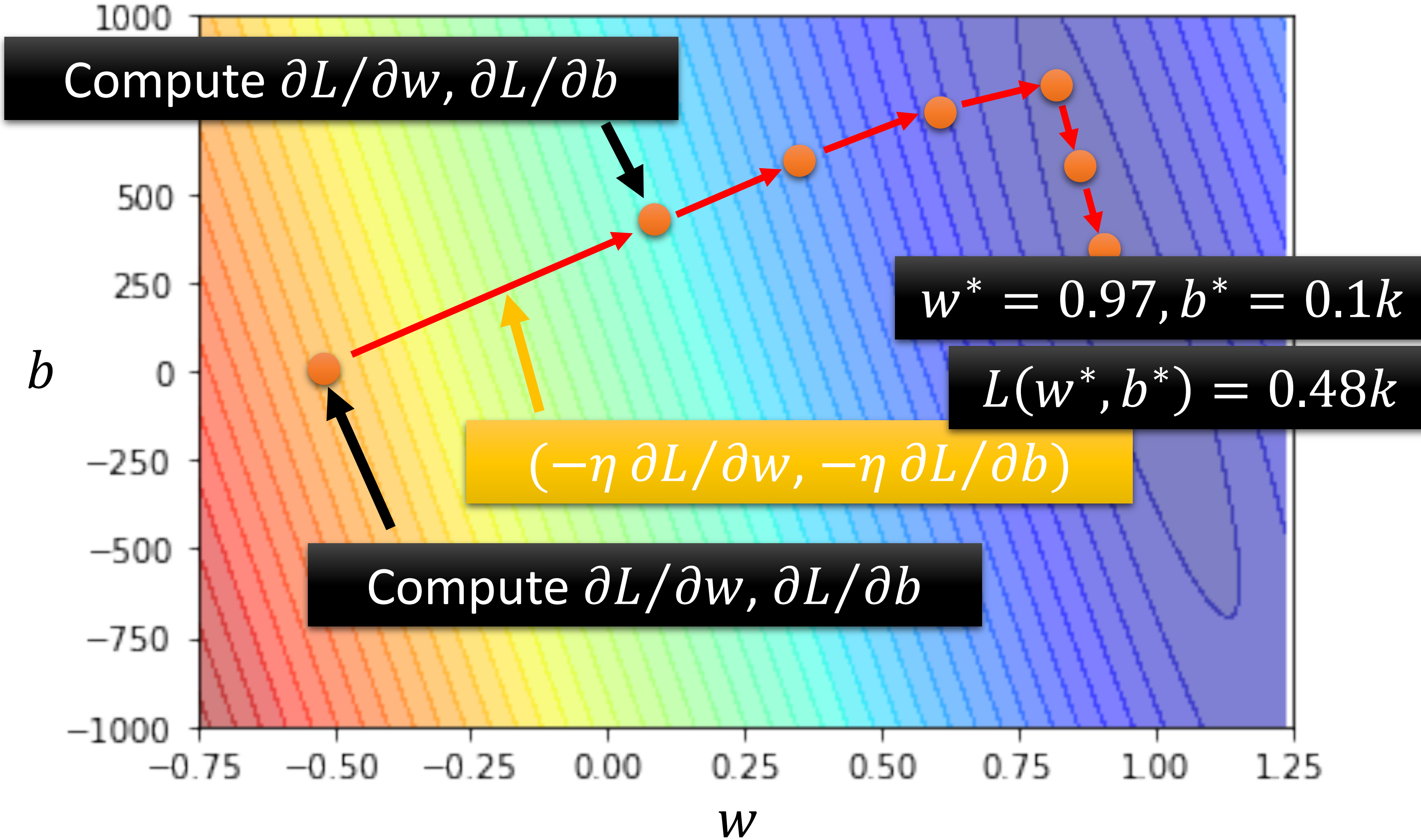
Can be done in one line in most deep learning frameworks

- Update w and b interatively

Model $y = b + wx_1$

3. Optimization

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



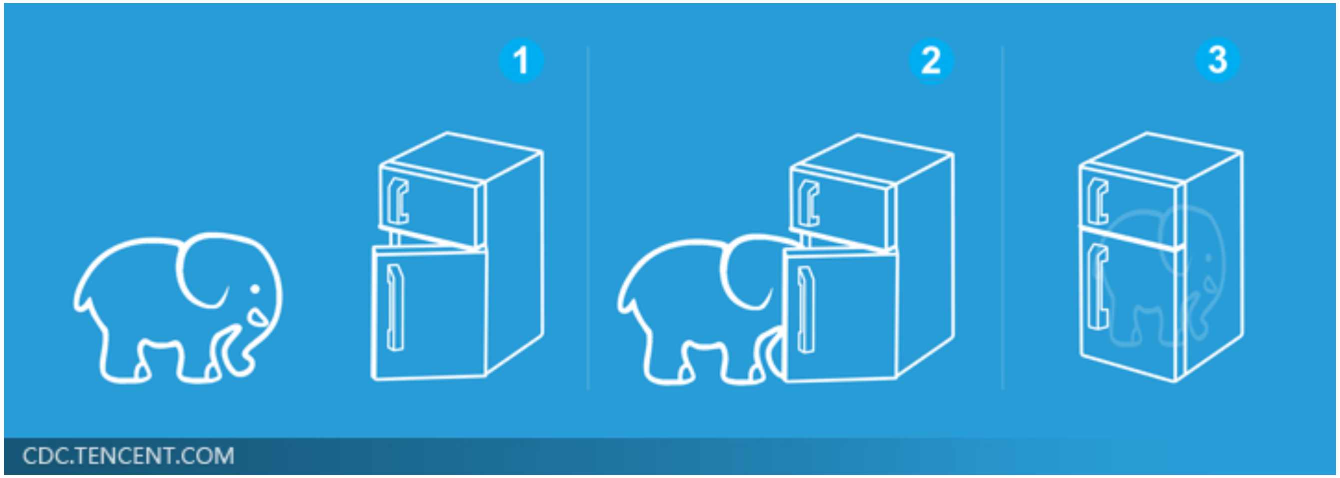
Machine Learning is so simple ……

$$y = b + wx_1$$

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

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

Step 1: function with unknown → Step 2: define loss from training data → Step 3: optimization



Machine Learning is so simple ……

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

$$y = b + wx_1$$

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

Step 1: function with unknown → Step 2: define loss from training data → Step 3: optimization

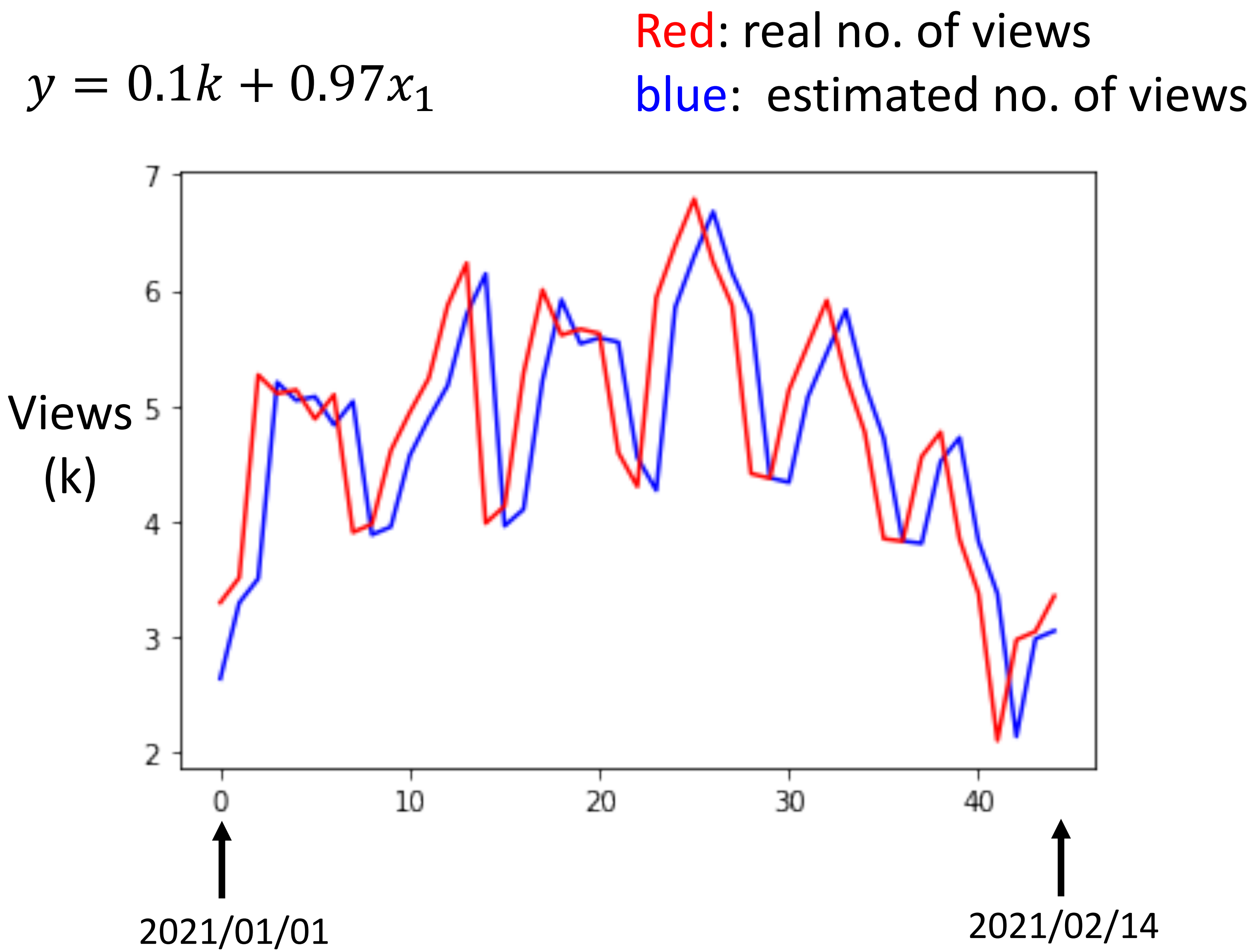
Training

$y = 0.1k + 0.97x_1$ 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
Views (k)
7
6
5
4
3
2
0
10
20
30
40
2021/01/01
2021/02/14

$$y = b + wx_1$$

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 $L' = 0.46k$

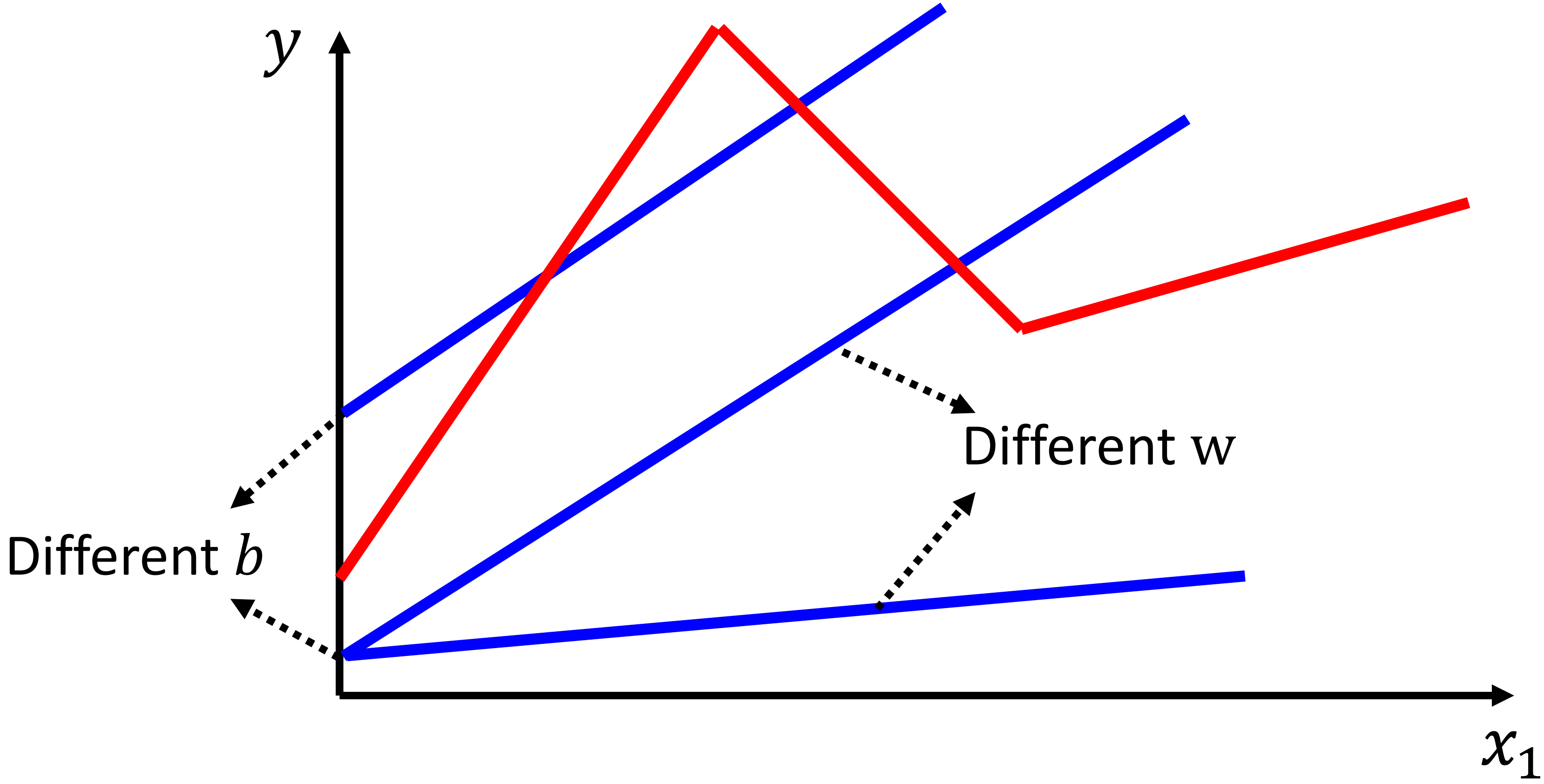
$$y = b + \sum_{j=1}^{56} w_j x_j$$

2017 - 2020 $L = 0.32k$

2021 $L' = 0.46k$

Linear models

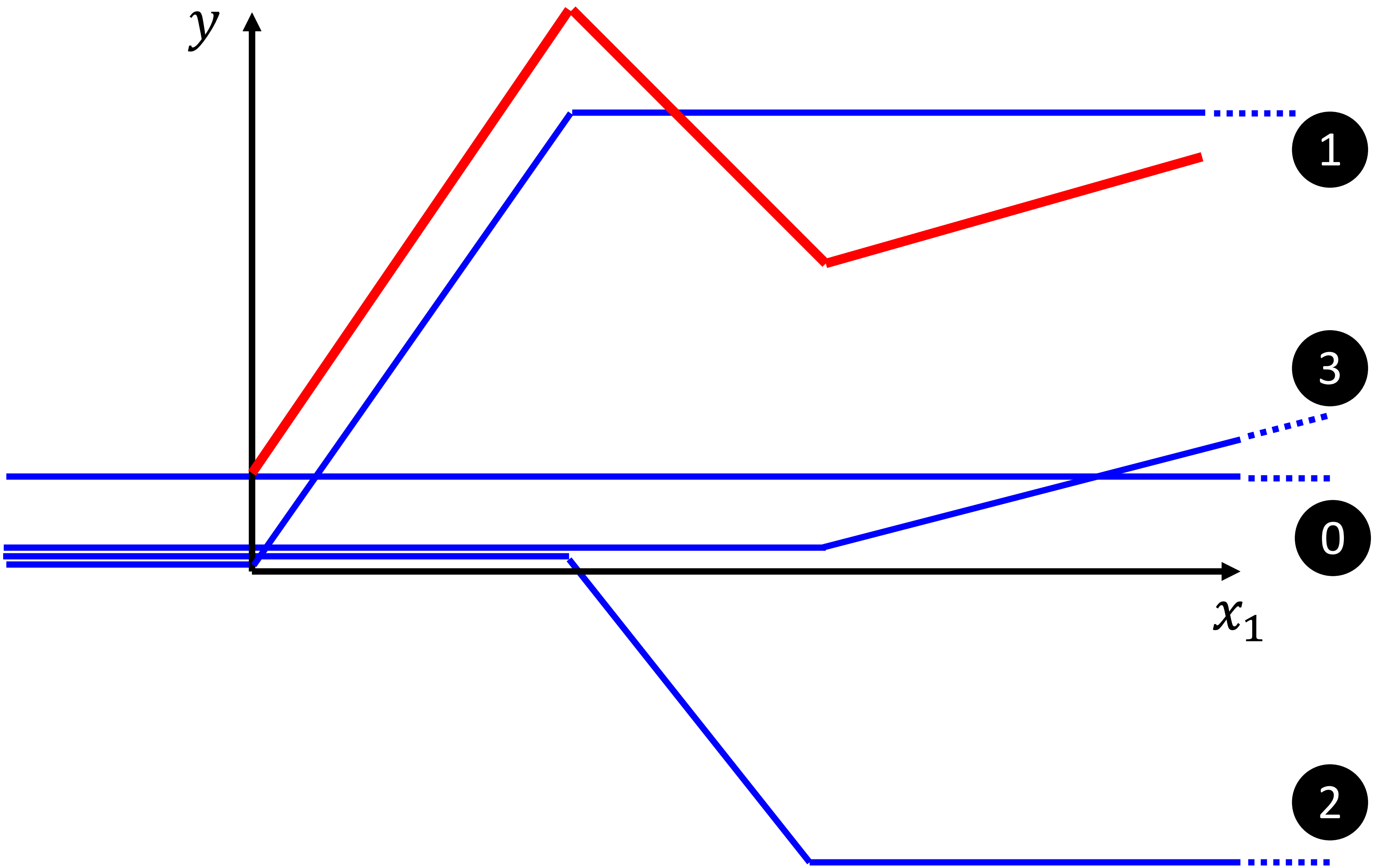
Linear models are too simple … we need more sophisticated modes.



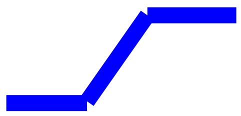
Linear models have severe limitation. ***Model Bias***

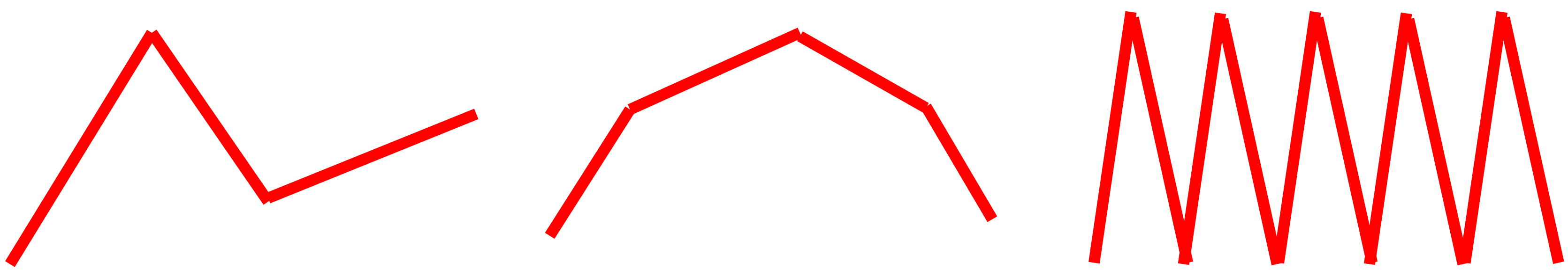
We need a more flexible model!

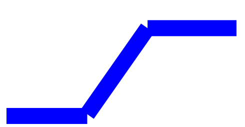
red curve = constant + sum of a set of



All Piecewise Linear Curves

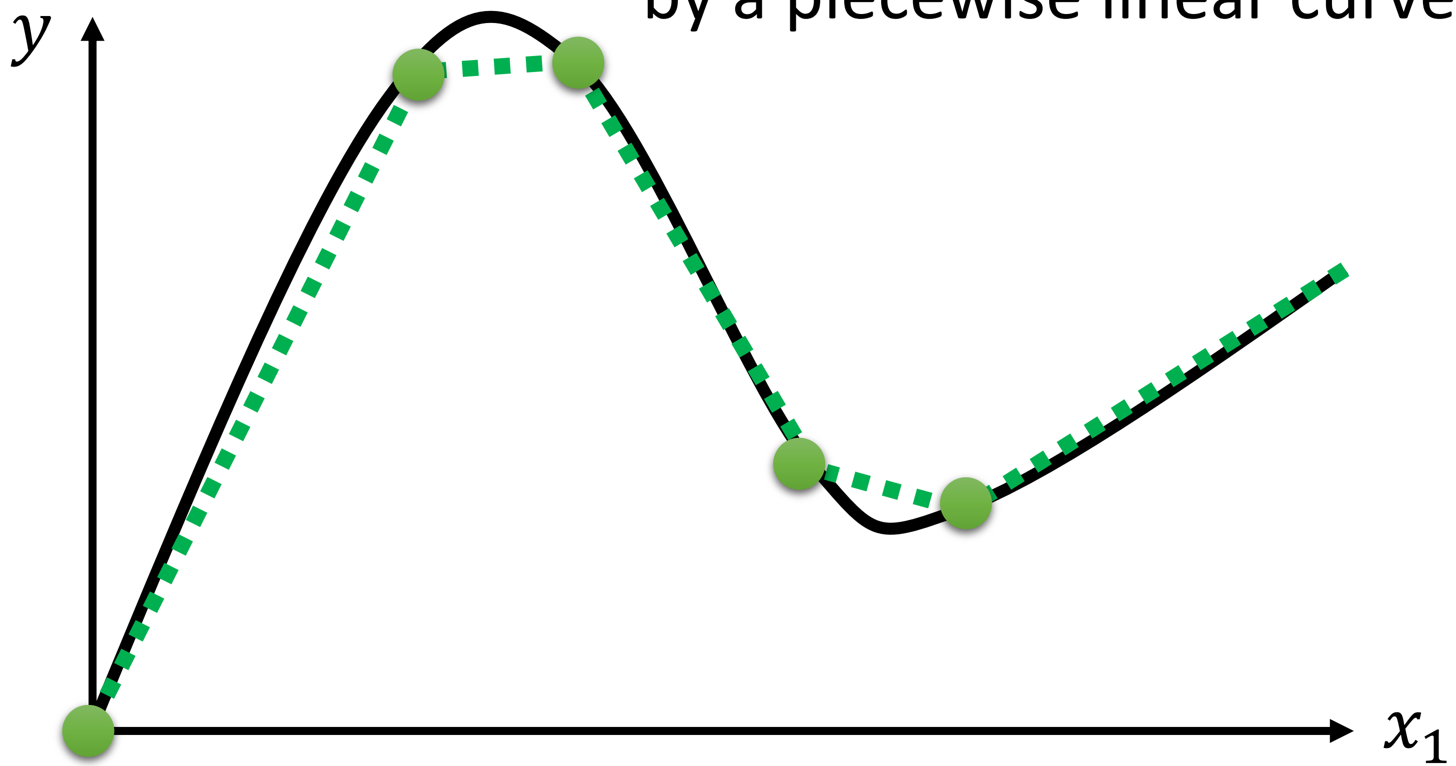
= constant + sum of a set of 



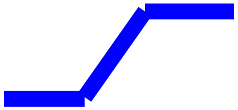
More pieces require more 

Beyond Piecewise Linear?

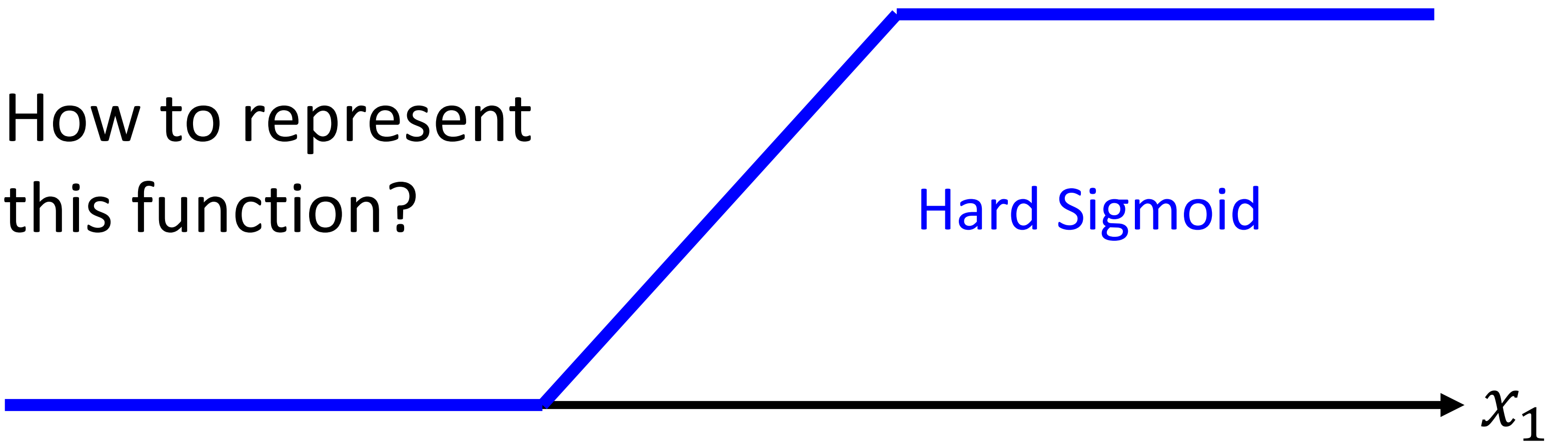
Approximate continuous curve by a piecewise linear curve.



To have good approximation, we need sufficient pieces.

red curve = constant + sum of a set of 

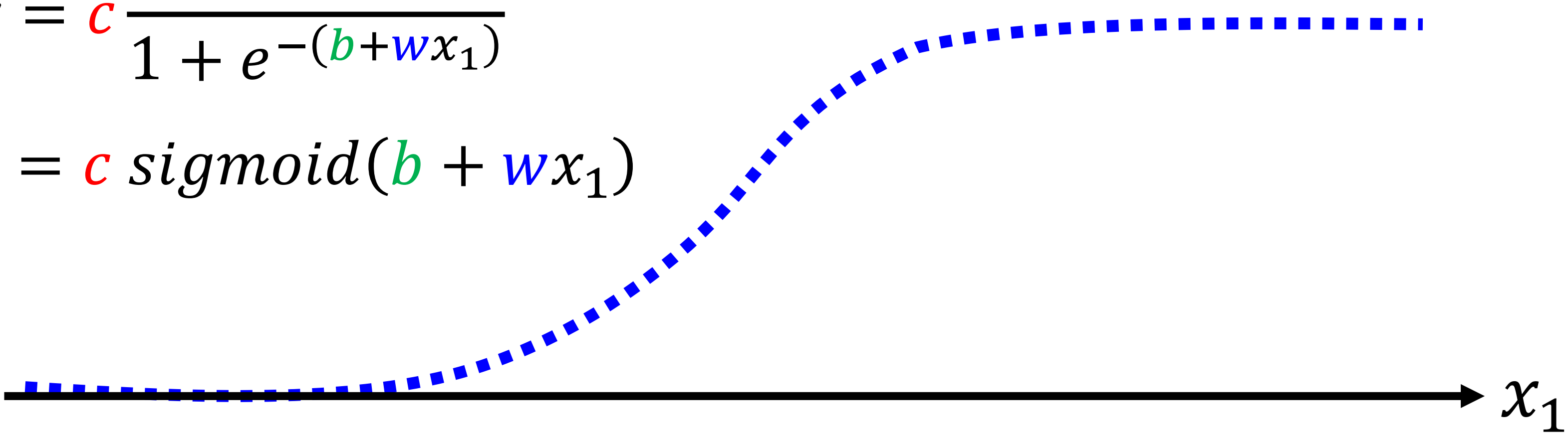
How to represent this function?

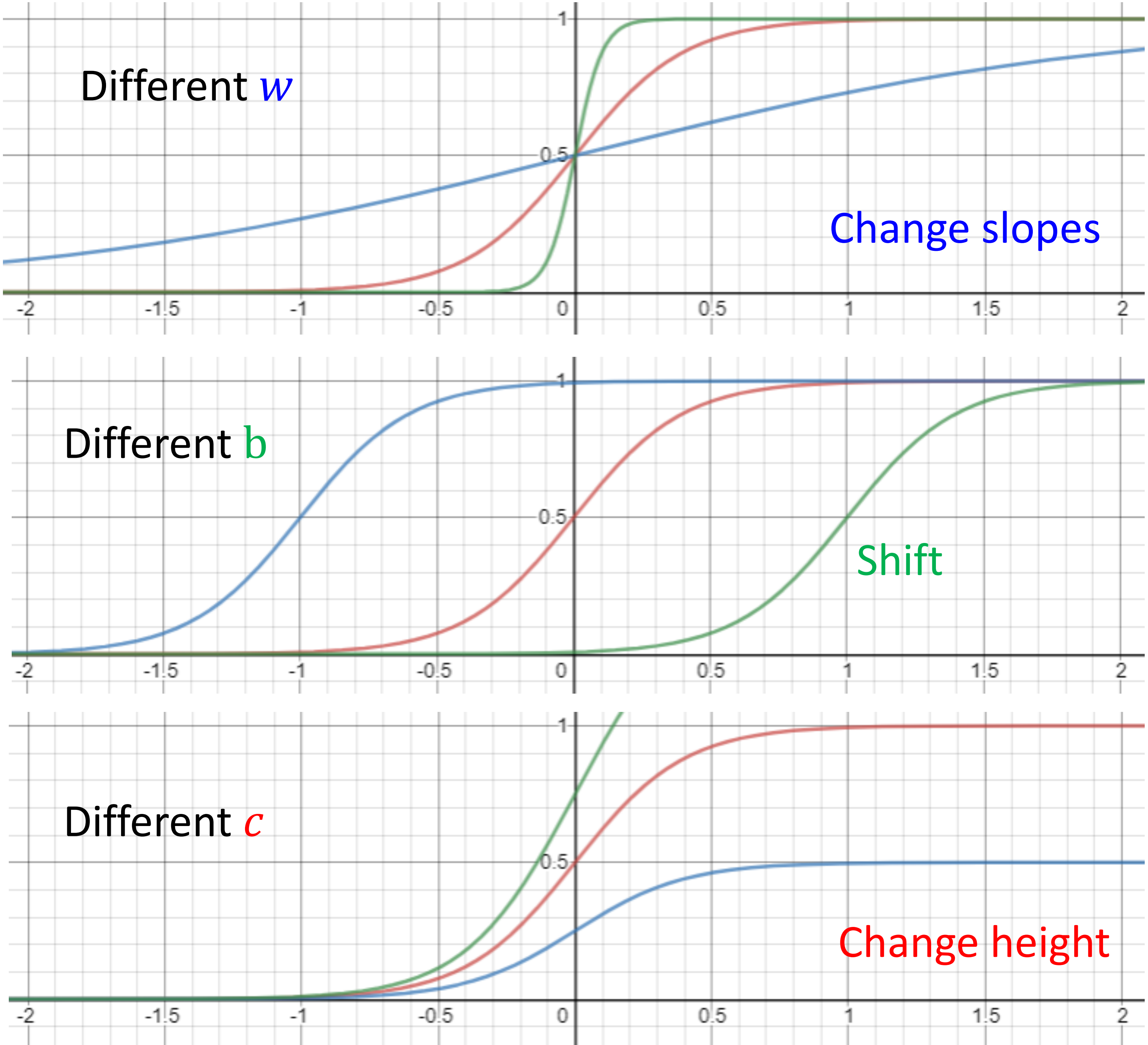


<u>Sigmoid Function</u>

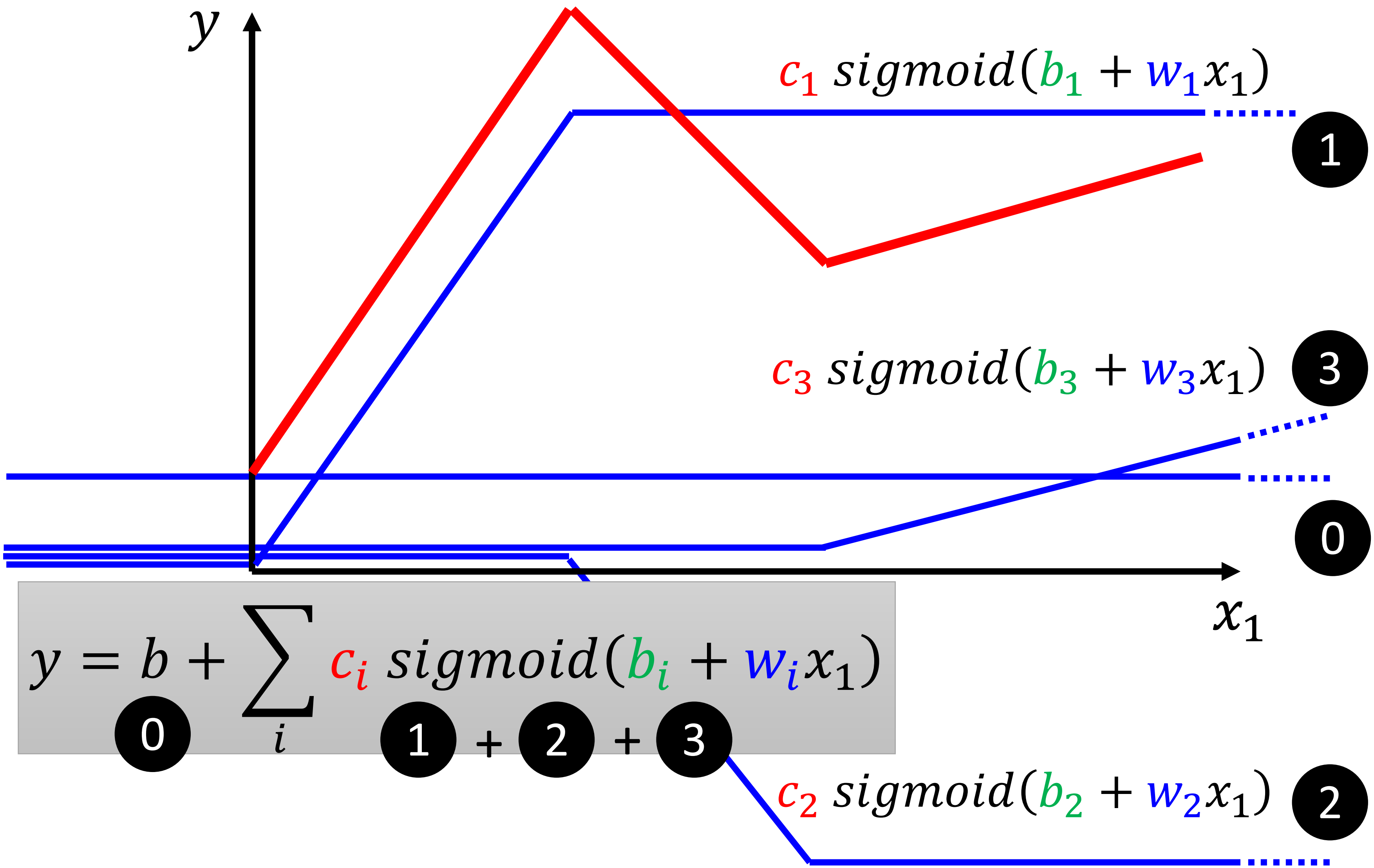
$$y = c \frac{1}{1 + e^{-(b + w x_1)}}$$

$$= c \; sigmoid(b + w x_1)$$




Different w
Change slopes
Different b
Shift
Different c
Change height
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

red curve = sum of a set of + constant



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

0 1 + 2 + 3

New Model: More Features

$$y = b + \underline{w x_1}$$

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

$$y = b + \underline{\sum_j w_j x_j}$$

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

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

j: 1,2,3
no. of features

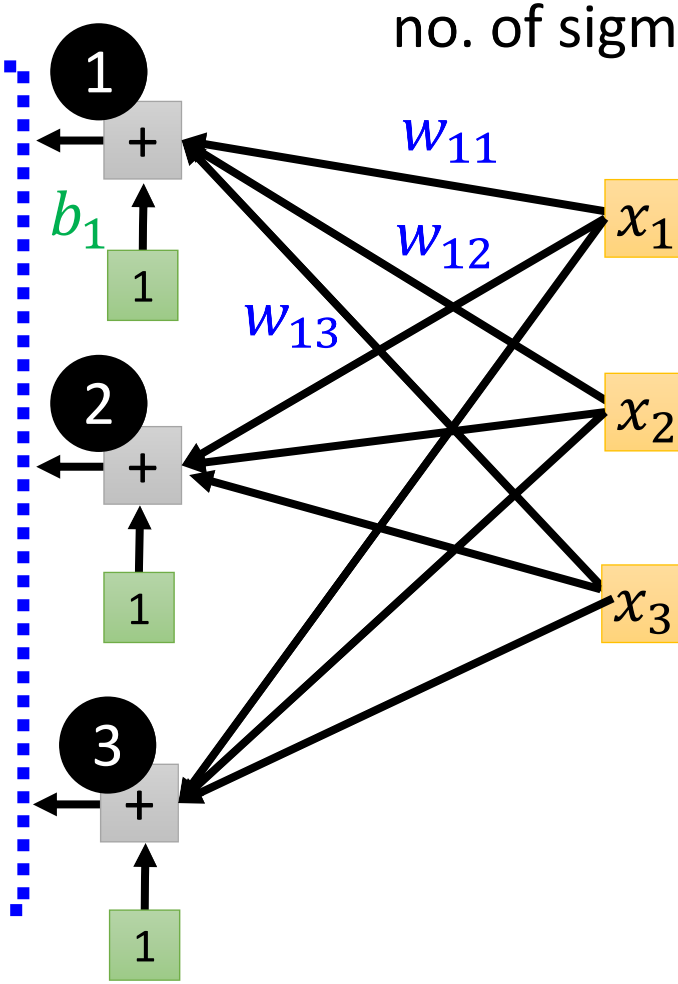
i: 1,2,3
no. of sigmoid

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

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

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

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



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

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

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

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

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

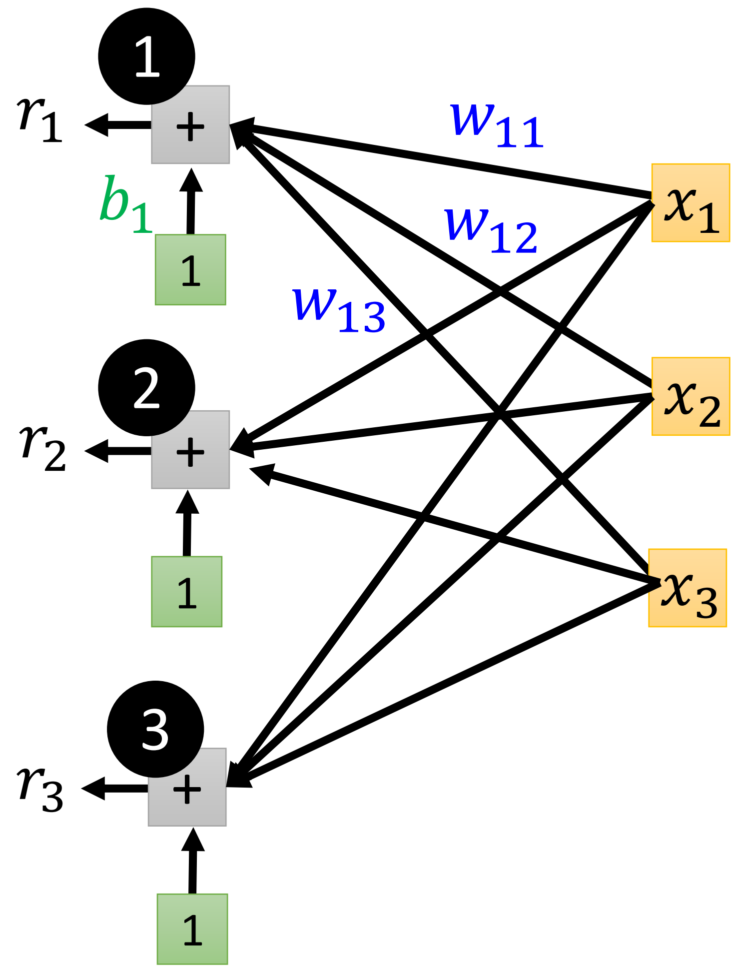
$$\boldsymbol{r} = \boldsymbol{b} + W \boldsymbol{x}$$

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

i: 1,2,3

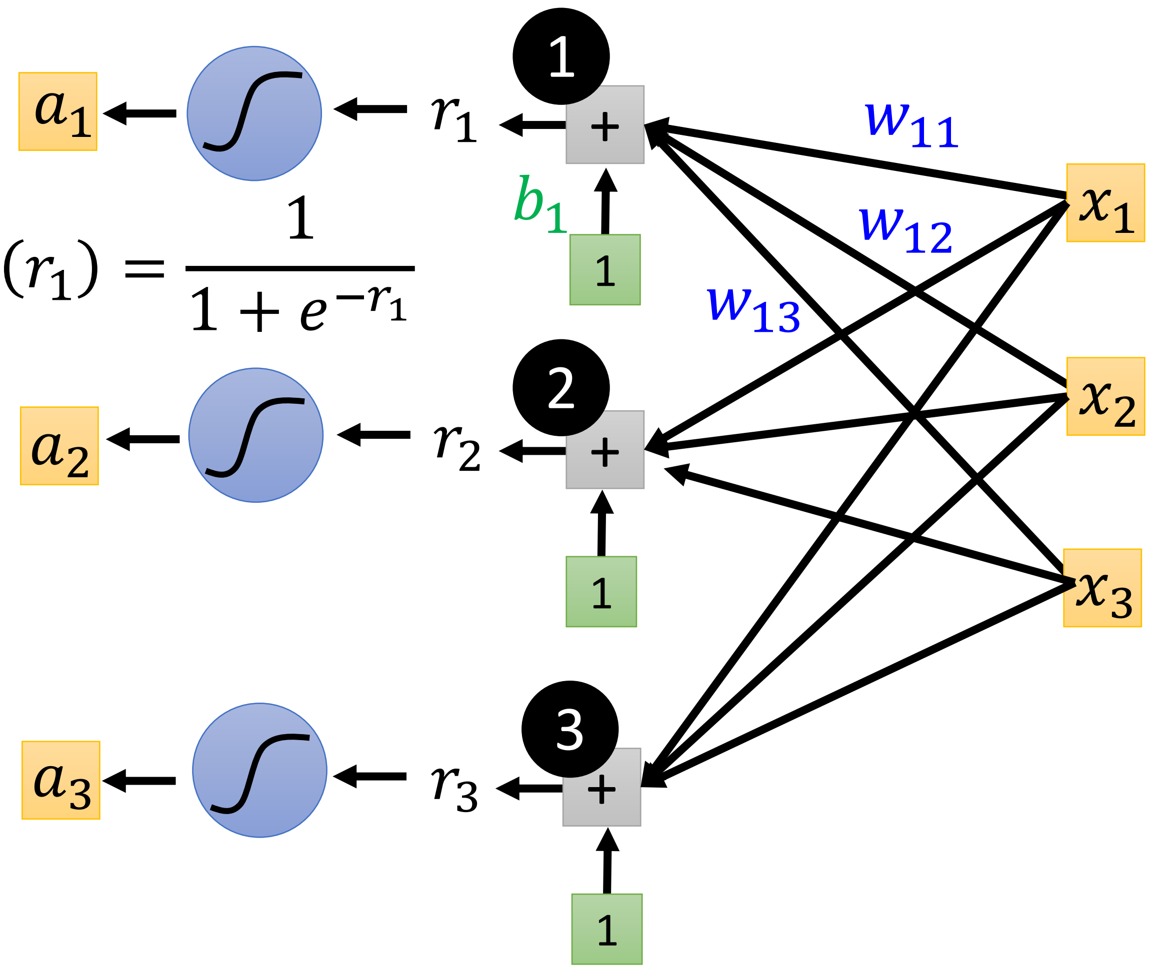
j: 1,2,3

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



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

$$a_1 = sigmoid(r_1) = \frac{1}{1 + e^{-r_1}}$$

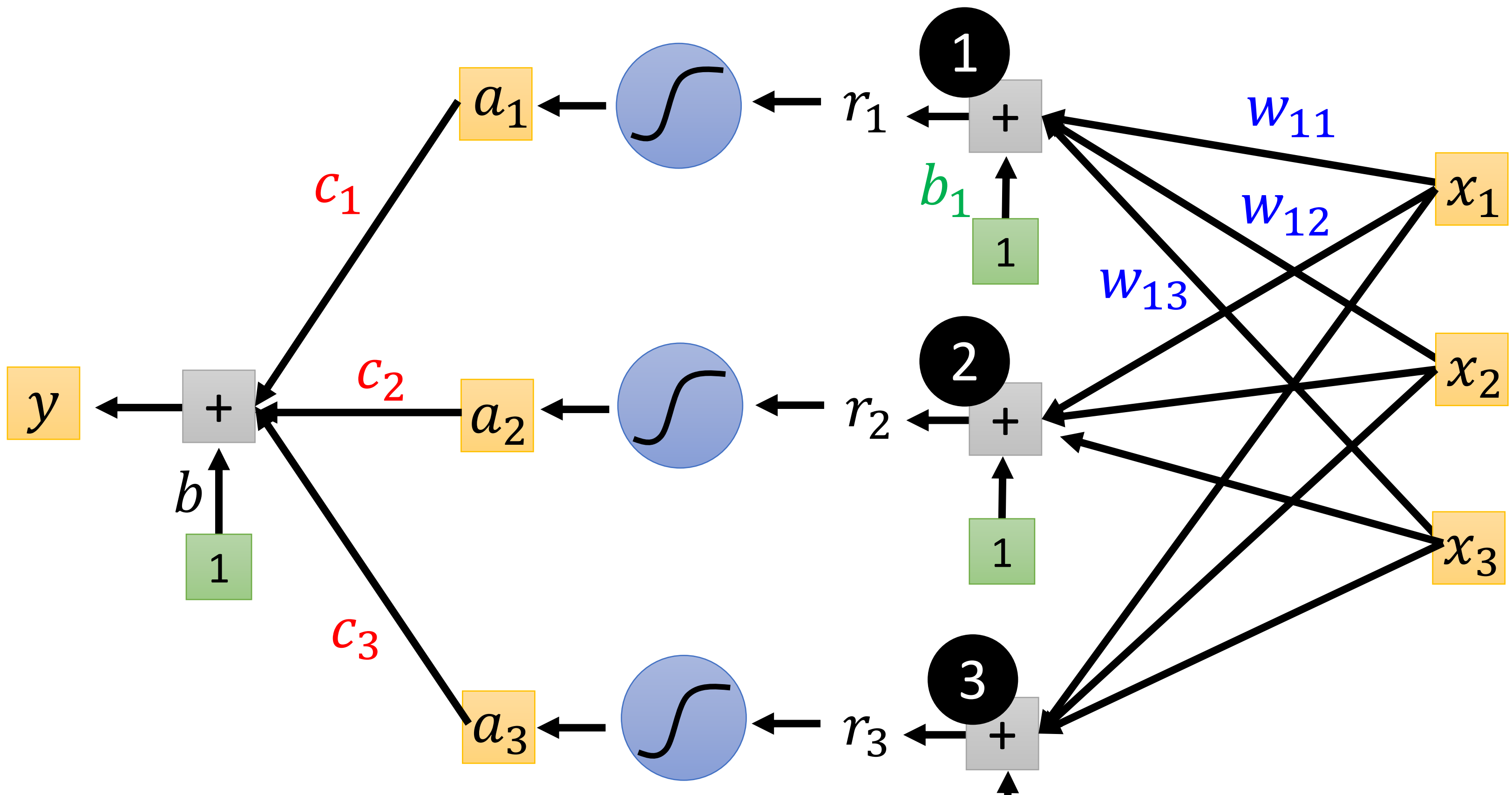


$$\boldsymbol{a} = \sigma(\boldsymbol{r})$$

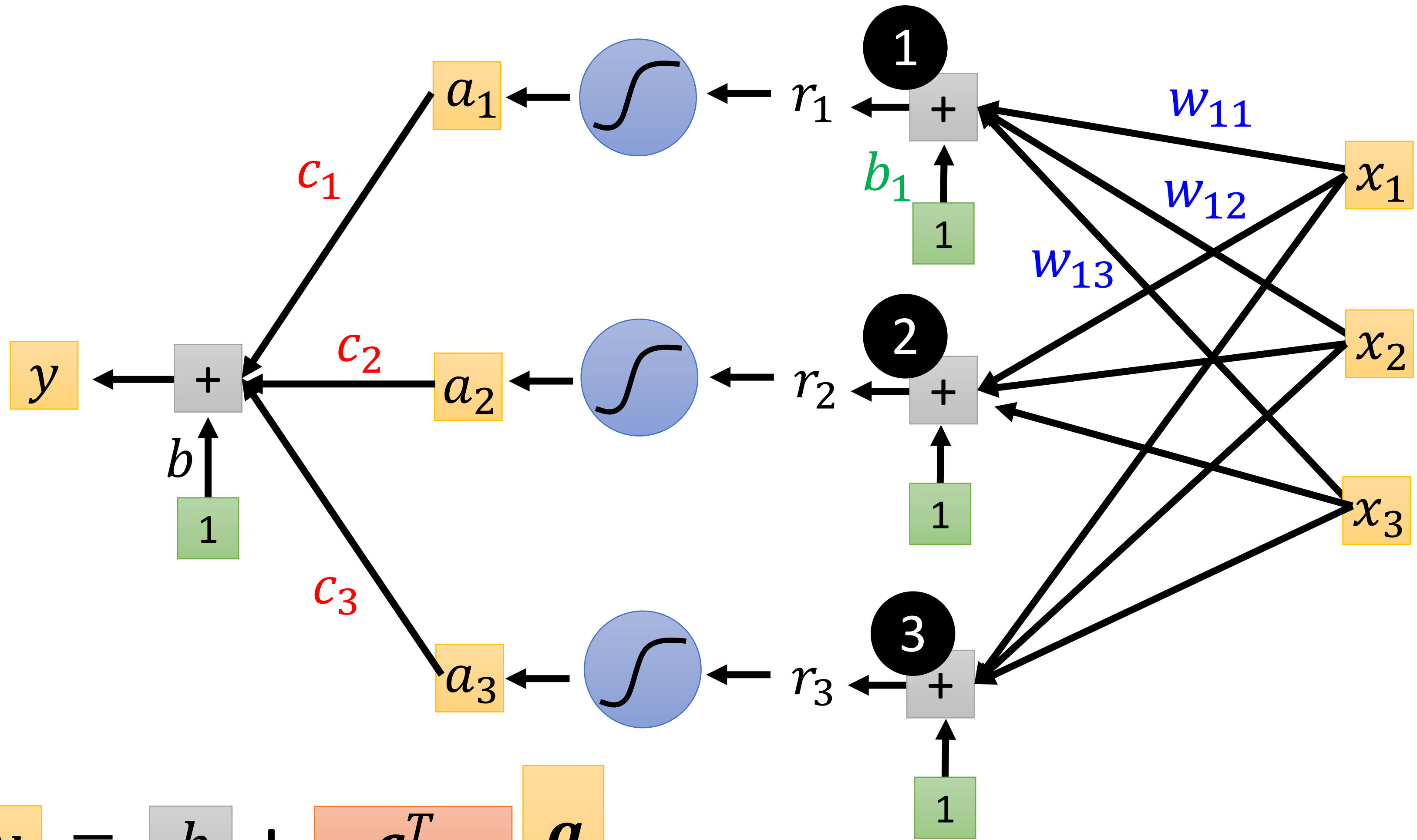
$$y = \boxed{b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right)}$$

i: 1,2,3

j: 1,2,3

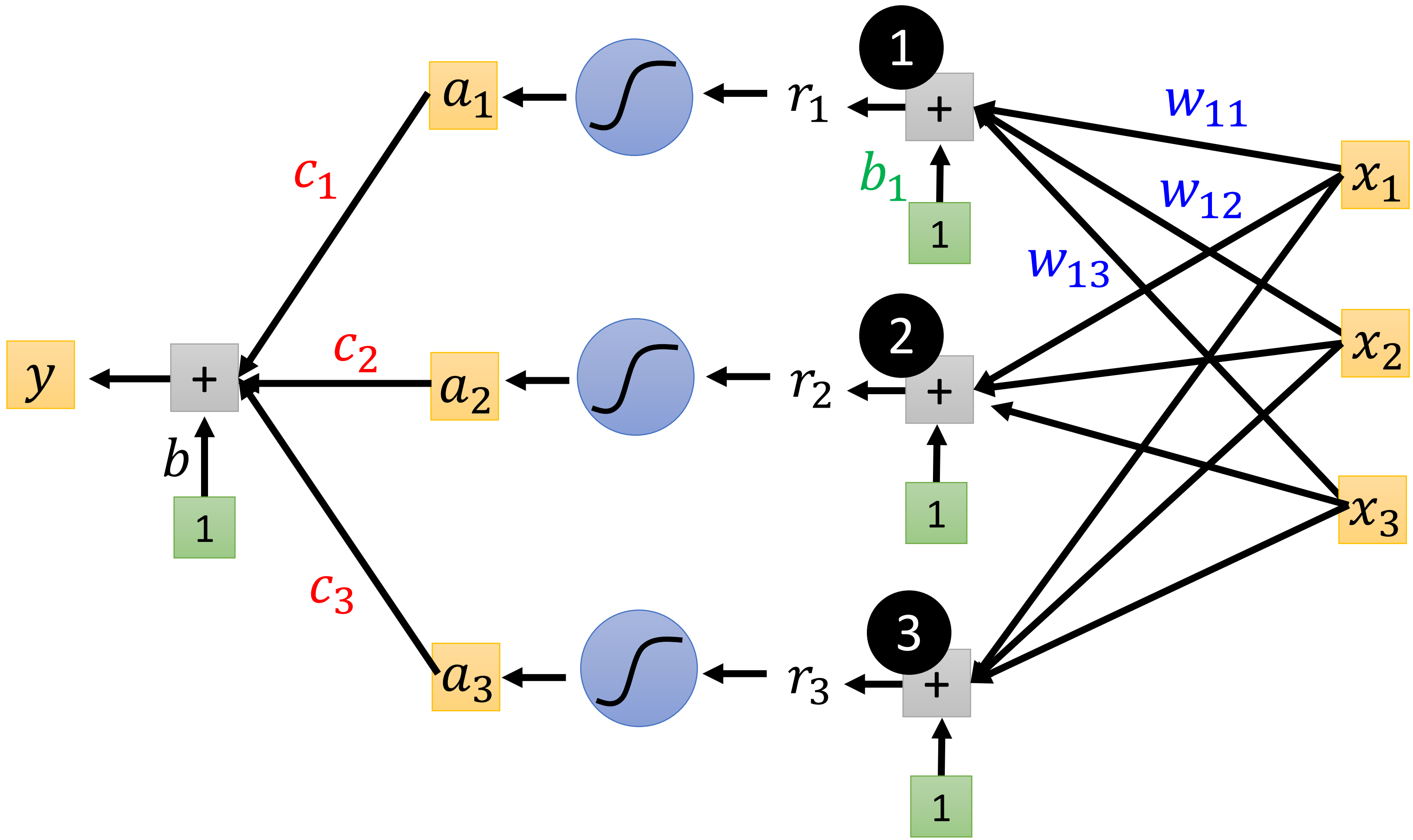


$$y = b + c^T a$$



$$y = b + \boldsymbol{c}^T \boldsymbol{a}$$

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



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

Function with unknown parameters

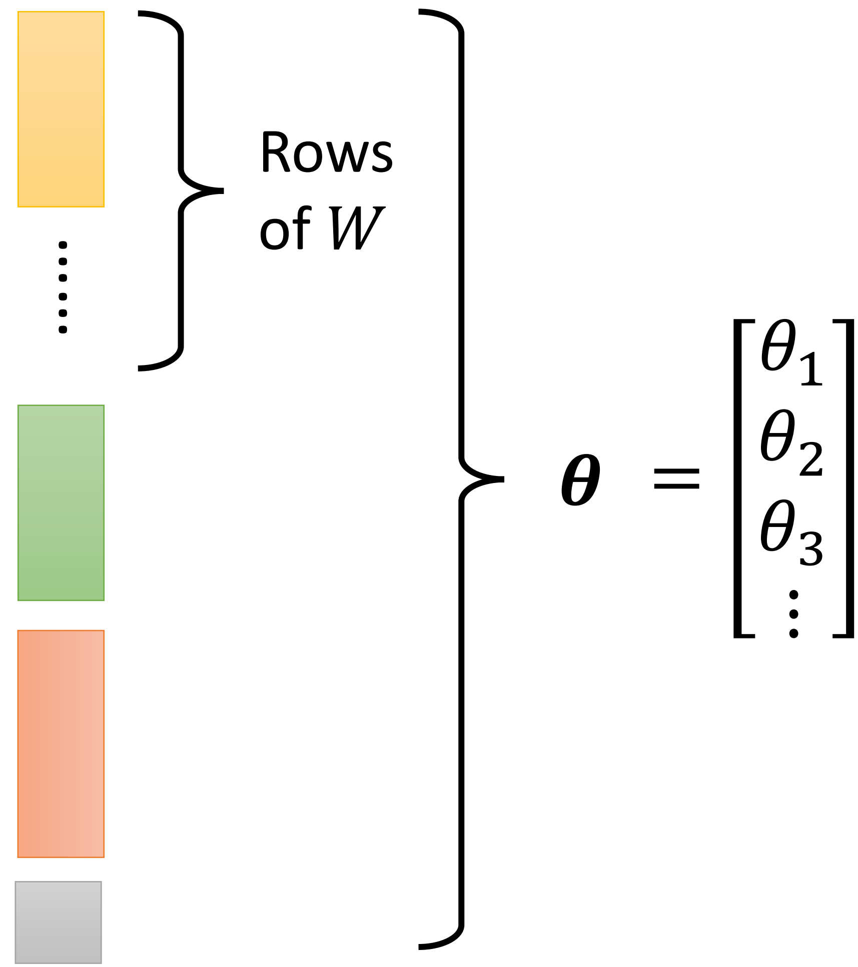
$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W\boldsymbol{x})$$

$\boldsymbol{x}$ **feature**

Unknown parameters

W $\boldsymbol{b}$

$\boldsymbol{c}^T$ b



Back to ML Framework

- Step 1: function with unknown
- Step 2: define loss from training data
- Step 3: optimization

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

Loss

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

feature

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

e

label $\hat{y}$

Given a set of values

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

Back to ML Framework

Step 1: function with unknown

Step 2: define loss from training data

Step 3: optimization

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

Optimization of New Model

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

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

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

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

gradient

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

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

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

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

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

Optimization of New Model

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

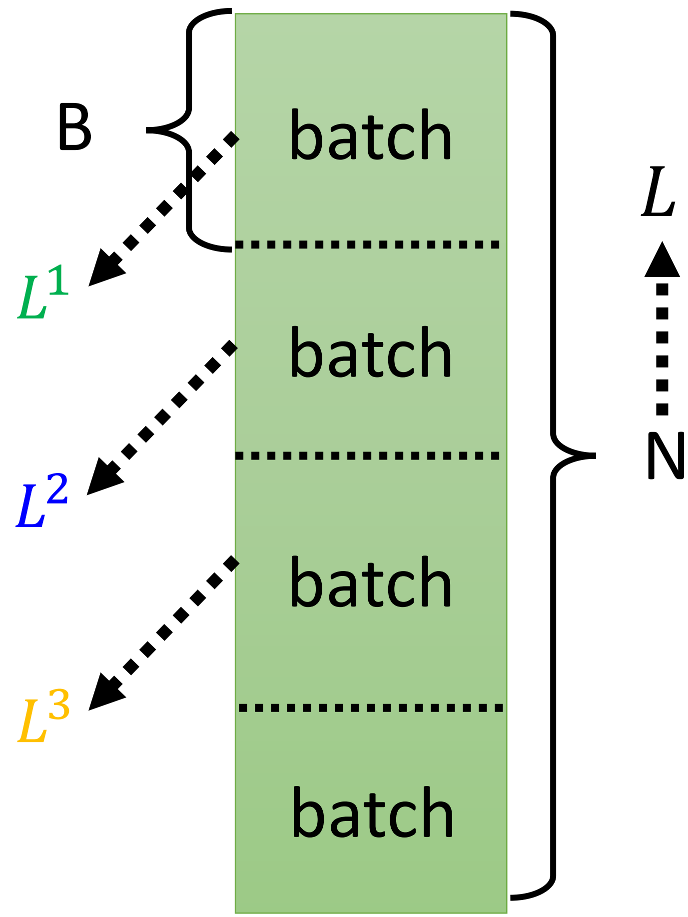
- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- Compute gradient $\boldsymbol{g} = \nabla L^1(\boldsymbol{\theta}^0)$

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

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

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

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**?

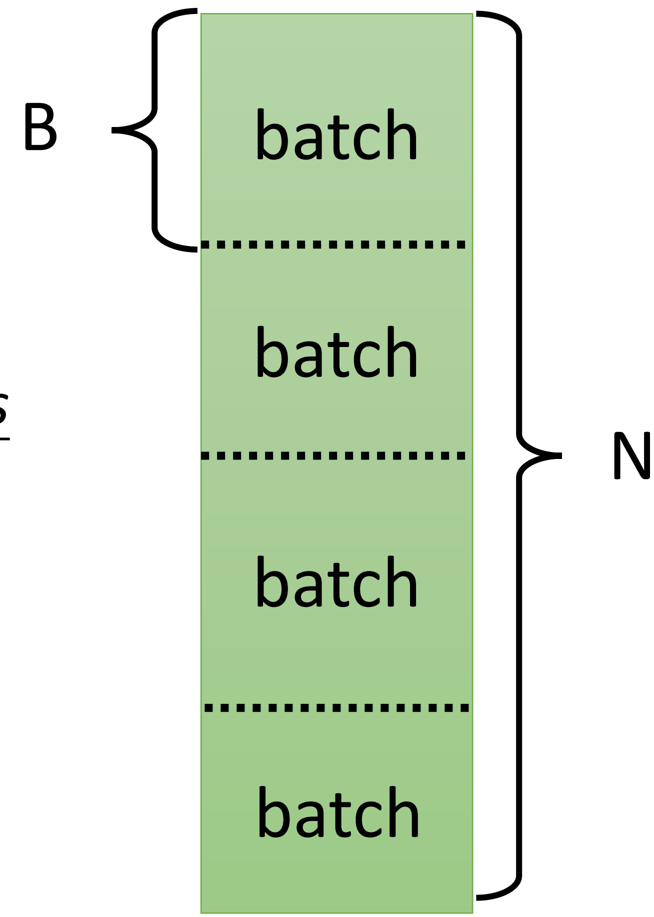
1,000 updates

Example 2

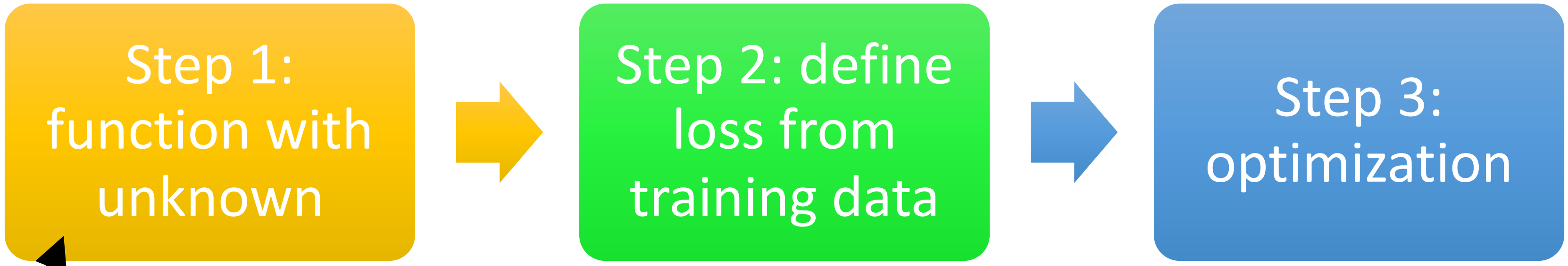
- 1,000 examples (N = 1,000)
- Batch size is 100 (B = 100)

How many update in **1 epoch**?

10 updates



Back to ML Framework

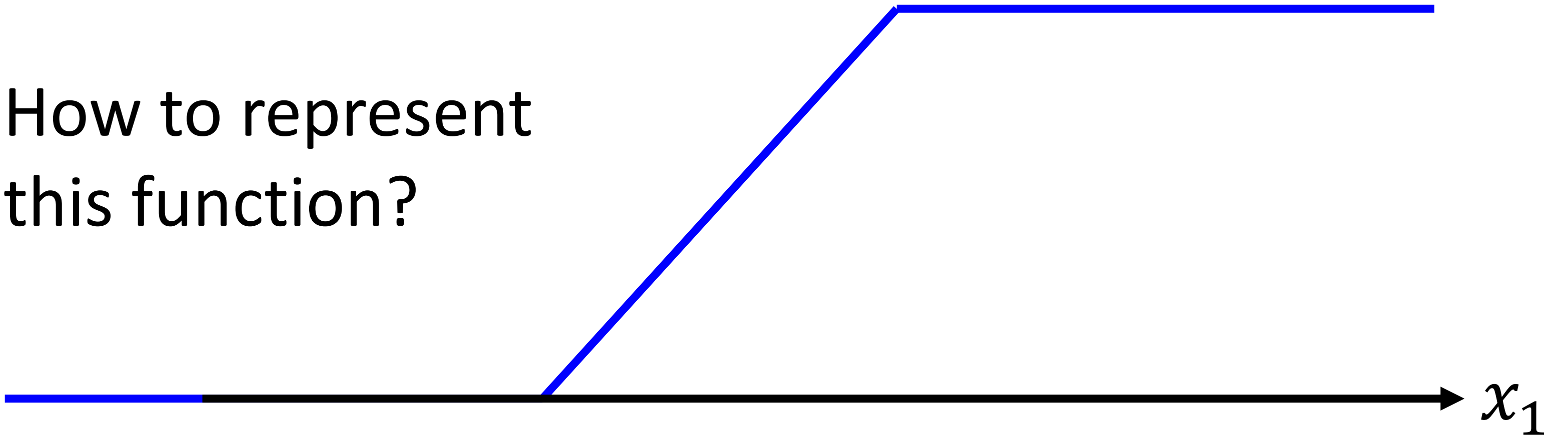


$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W\boldsymbol{x})$$

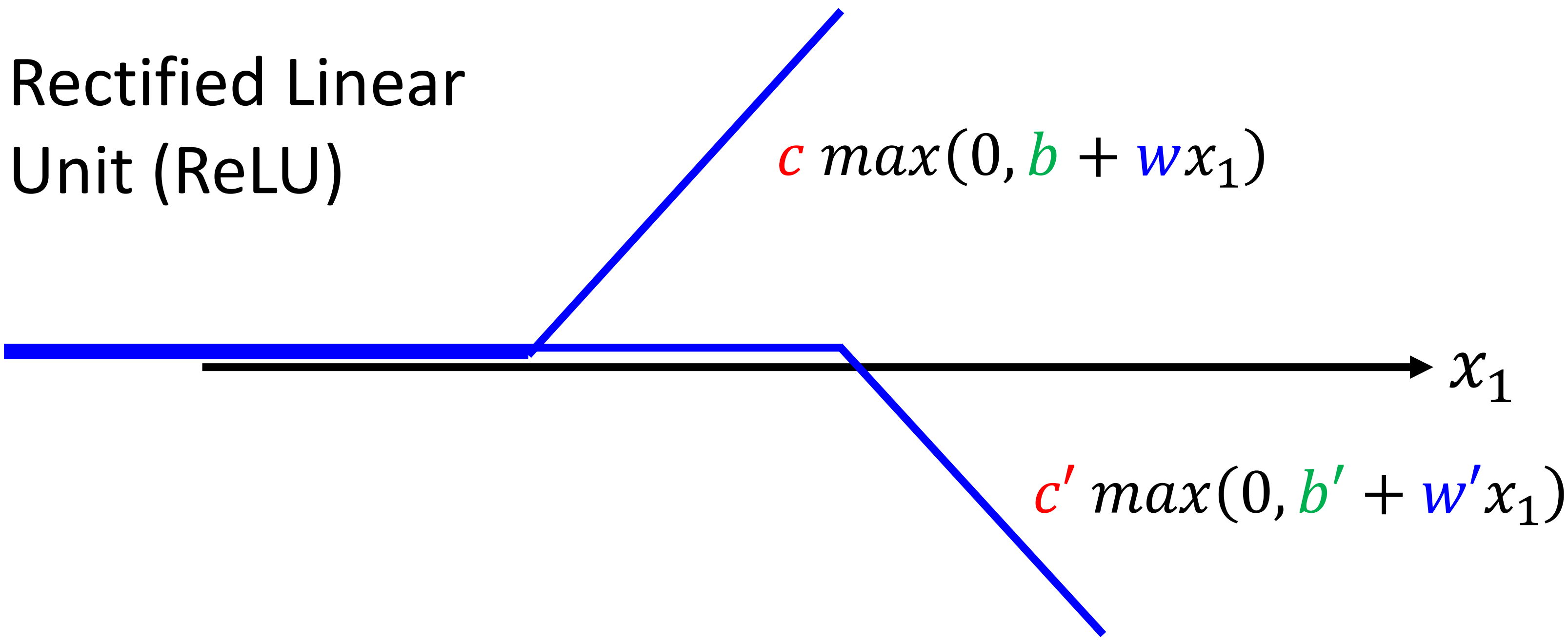
More variety of models …

Sigmoid → ReLU

How to represent this function?



Rectified Linear Unit (ReLU)



Sigmoid → ReLU

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

Activation function

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

Which one is better?

Experimental Results

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

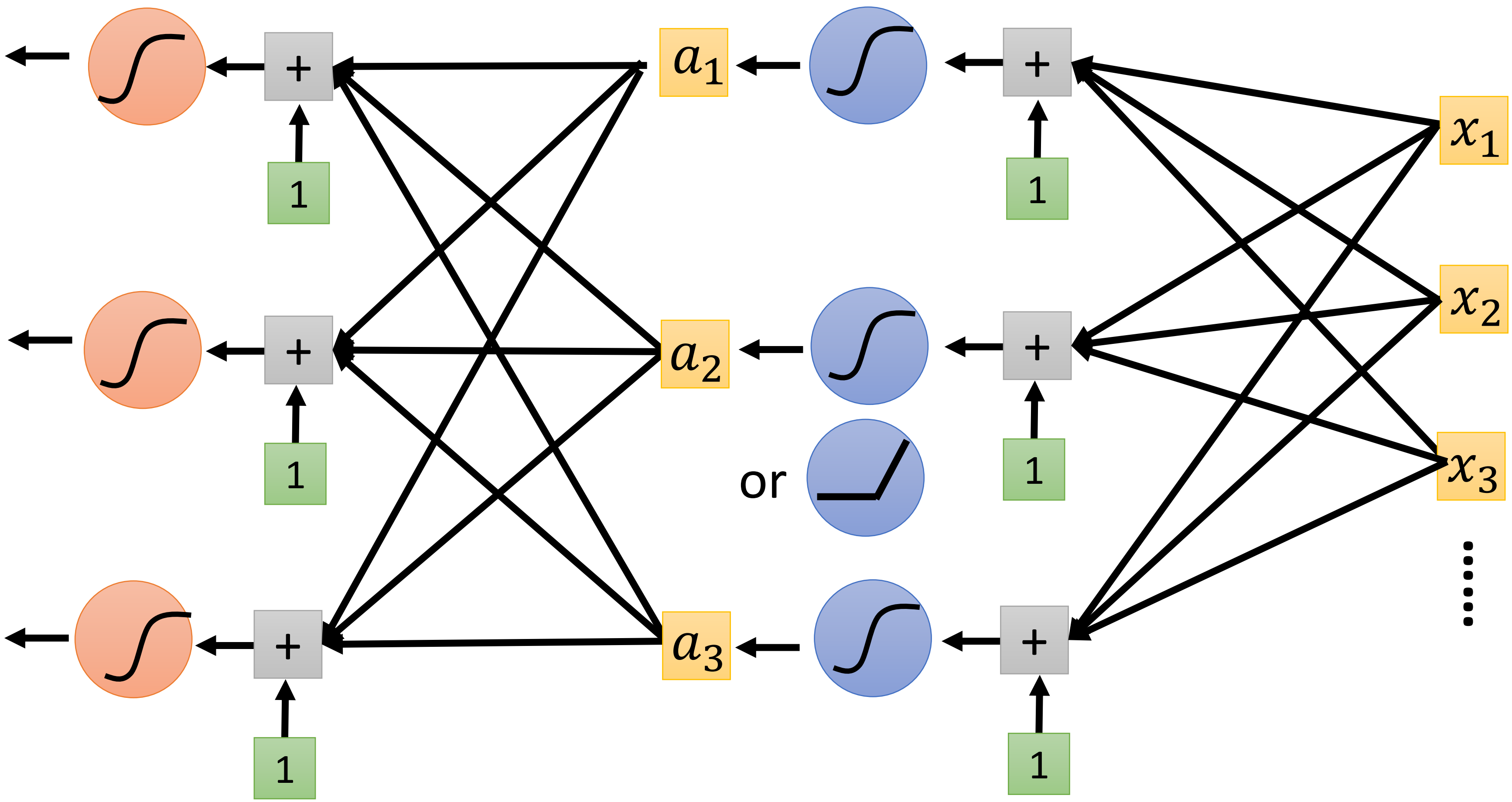
	linear
2017 – 2020	0.32k
2021	0.46k

Back to ML Framework

Step 1: function with unknown → Step 2: define loss from training data → Step 3: optimization

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

Even more variety of models …



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

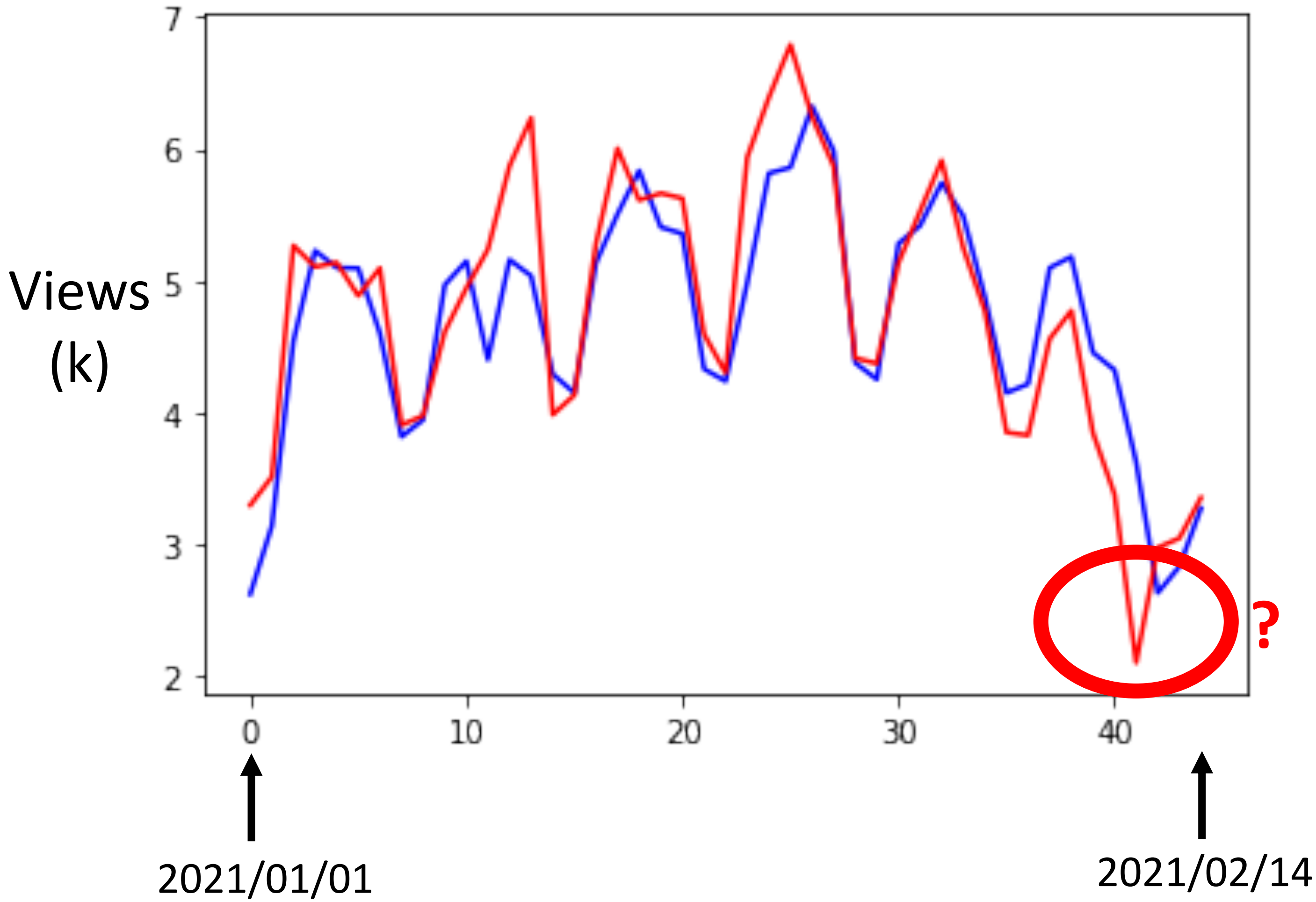
Experimental Results

- 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

3 layers

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



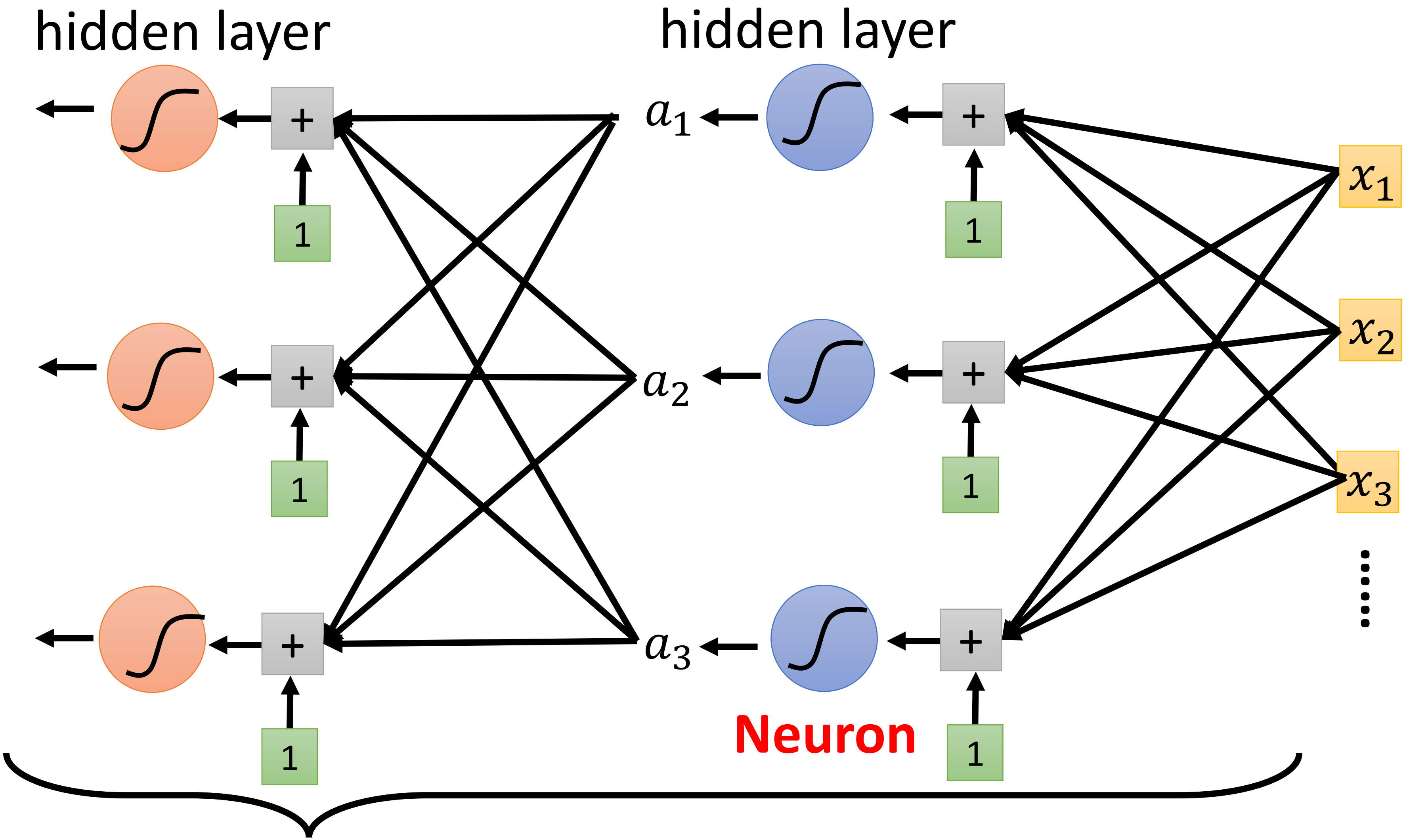
Back to ML Framework

Step 1: function with unknown → Step 2: define loss from training data → Step 3: optimization

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

It is not ***fancy*** enough.

Let's give it a ***fancy*** name!



Neural Network This mimics human brains … (???)

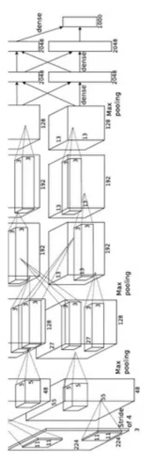
Many layers means **Deep** ➡ Deep Learning

Deep = Many hidden layers

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

8 layers

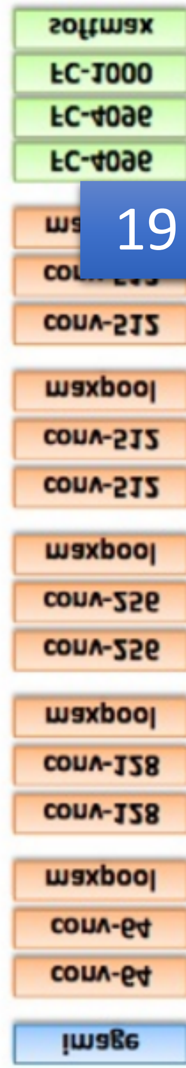
16.4%



AlexNet (2012)

19 layers

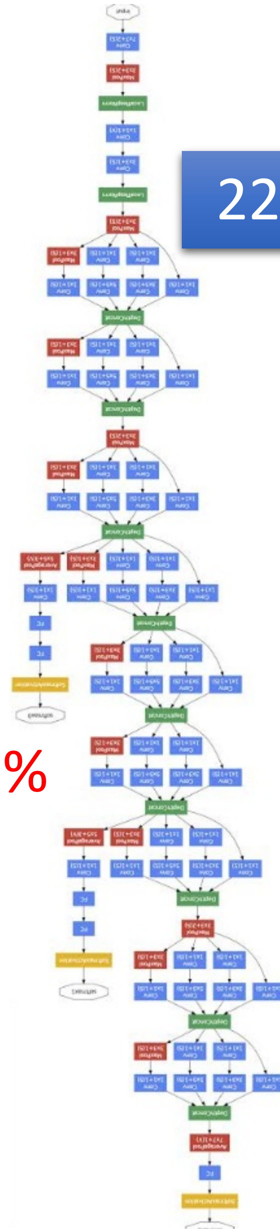
7.3%



VGG (2014)

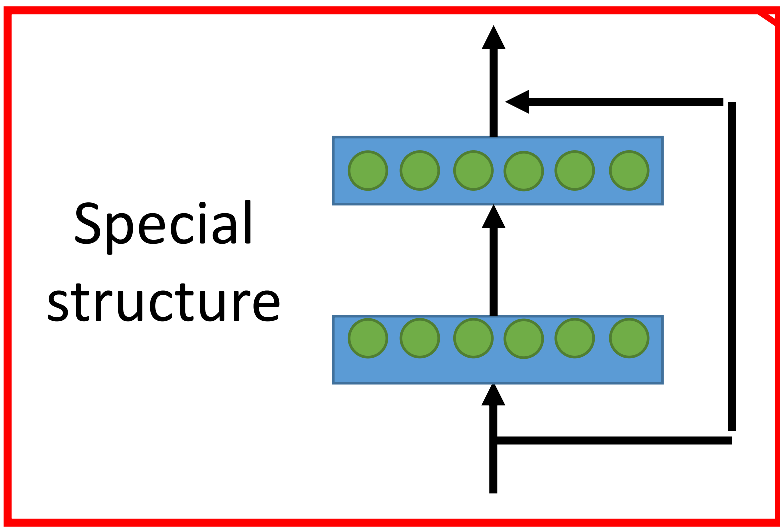
22 layers

6.7%

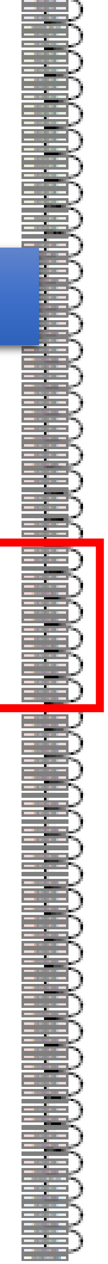


GoogleNet (2014)

Deep = Many hidden layers



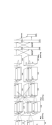
152 layers



101 layers

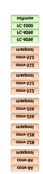
Why we want “***Deep***” network,
not “***Fat***” network?

16.4%



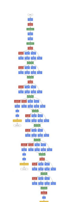
AlexNet
(2012)

7.3%



VGG
(2014)

6.7%



GoogleNet
(2014)

3.57%

Residual Net
(2015)

Taipei
101

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-WRlEFefw

Backpropagation
Computing gradients in an efficient way

https://youtu.be/ibJpTrp5mcE