李宏毅 (Hung-yi Lee) · HYLEE | Machine Learning (2021)

HYLEE(2021)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1fM4y137M4

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/ntu-hylee-ml

机器学习 Auto-encoder 生成式对抗网络 学习率
深度学习 卷积神经网络 GAN 自监督 自注意力机制
批次标准化 神经网络压缩 强化学习 元学习 Transformer

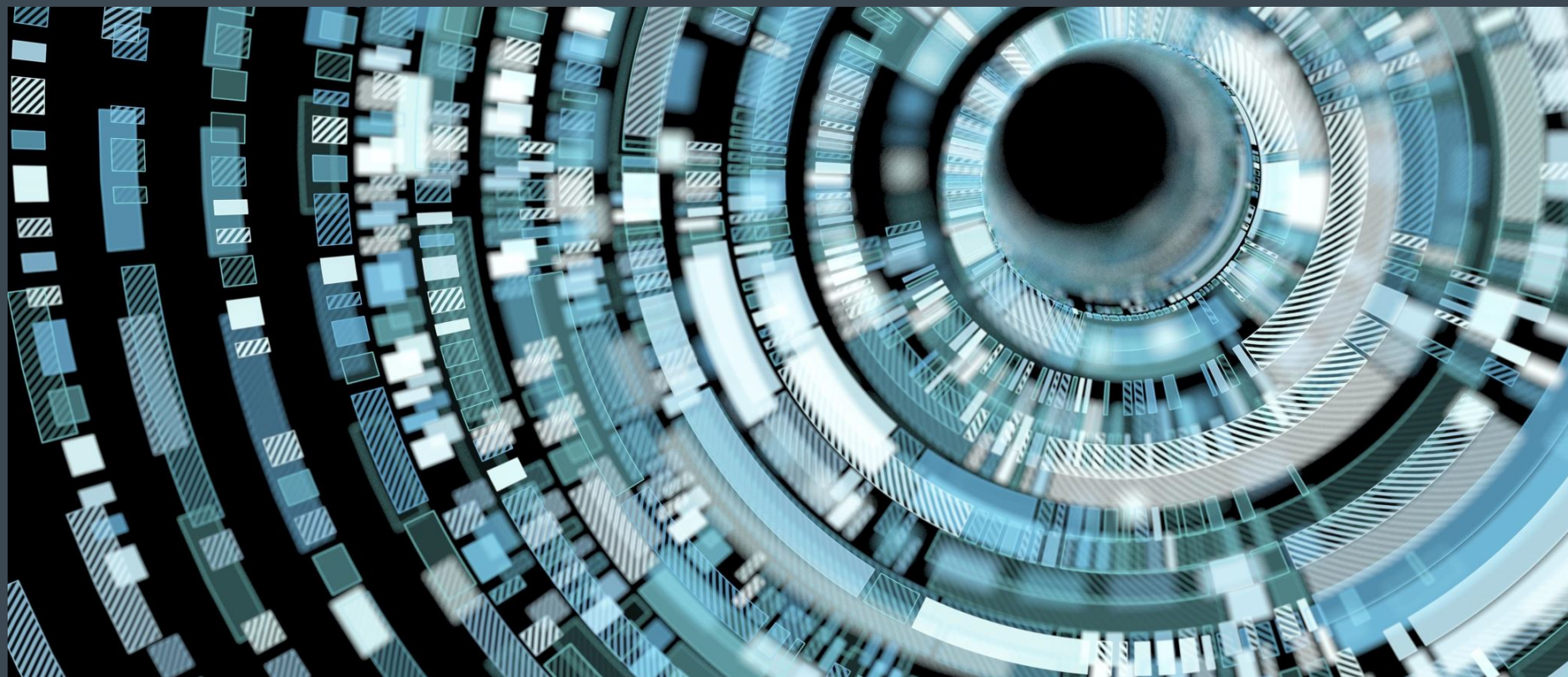Awesome AI Courses Notes Cheatsheets 是 **ShowMeAI** 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n

Awesome AI Courses Notes Cheatsheets· 持续更新中

知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

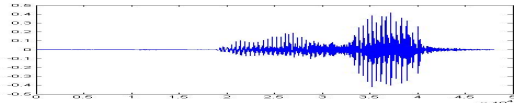**微信公众号**

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]

Introduction of Machine / Deep Learning

Hung-yi Lee 李宏毅

Machine Learning
≈ Looking for Function

- Speech Recognition

$$f(\quad \text{[waveform]} \quad) = \text{“How are you”}$$

- Image Recognition

$$f(\quad \text{[cat image]} \quad) = \text{“Cat”}$$

- Playing Go

$$f(\quad \text{[Go board]} \quad) = \text{“5-5”} \quad \text{(next move)}$$

Different types of Functions

Regression: The function outputs a scalar.
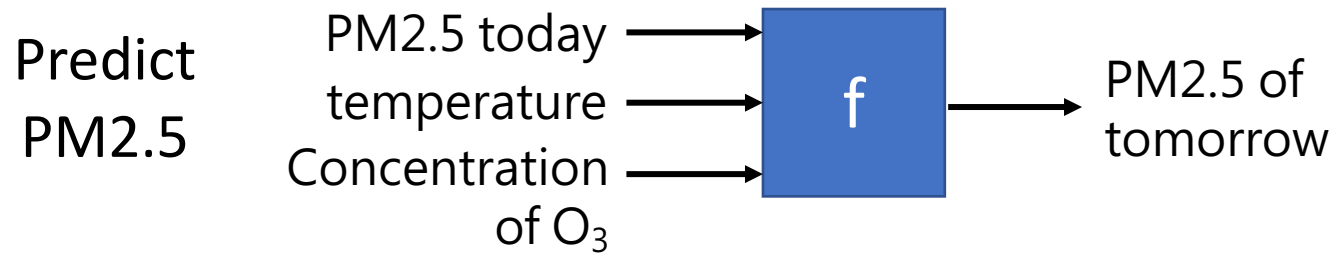
Predict
PM2.5

PM2.5 today $\longrightarrow$
temperature $\longrightarrow$ [f] $\longrightarrow$ PM2.5 of tomorrow
Concentration $\longrightarrow$
of O_3

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

Spam
filtering
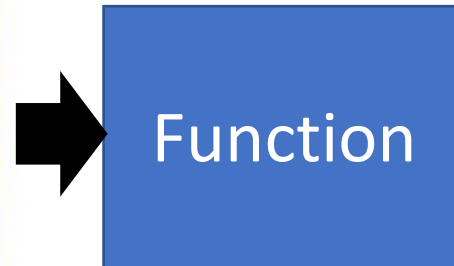
EMAIL $\Rightarrow$ [f] $\Rightarrow$ Yes/No
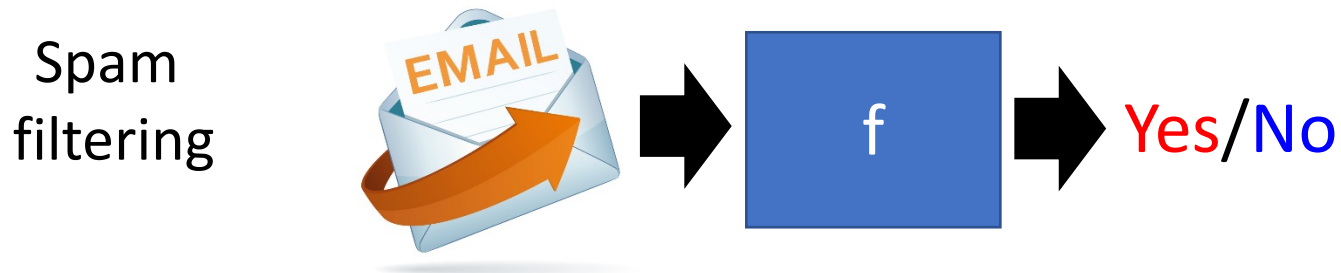
Different types of Functions

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

Playing GO

Each position
is a class
(19 x 19 classes)
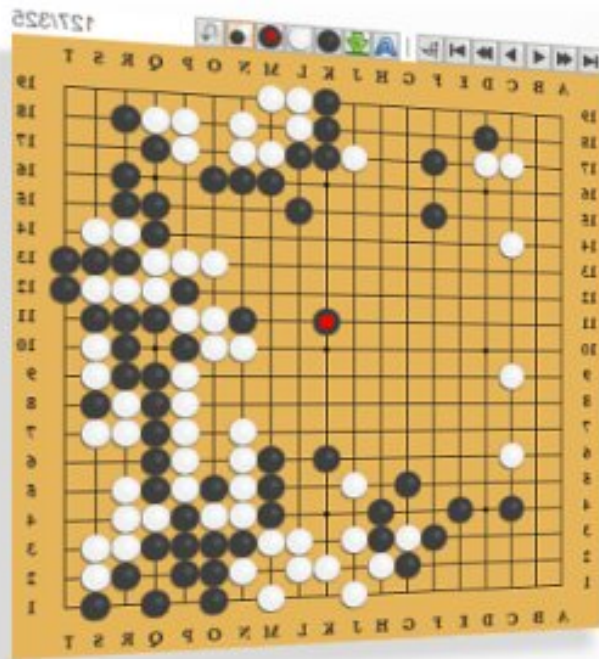
Function

a position on the board

Next move

Structured Learning

create something with structure (image, document)

一小部分啊

Regression, Classification

How to find a function?
A Case Study

YouTube Channel

https://www.youtube.com/c/HungyiLeeNTU
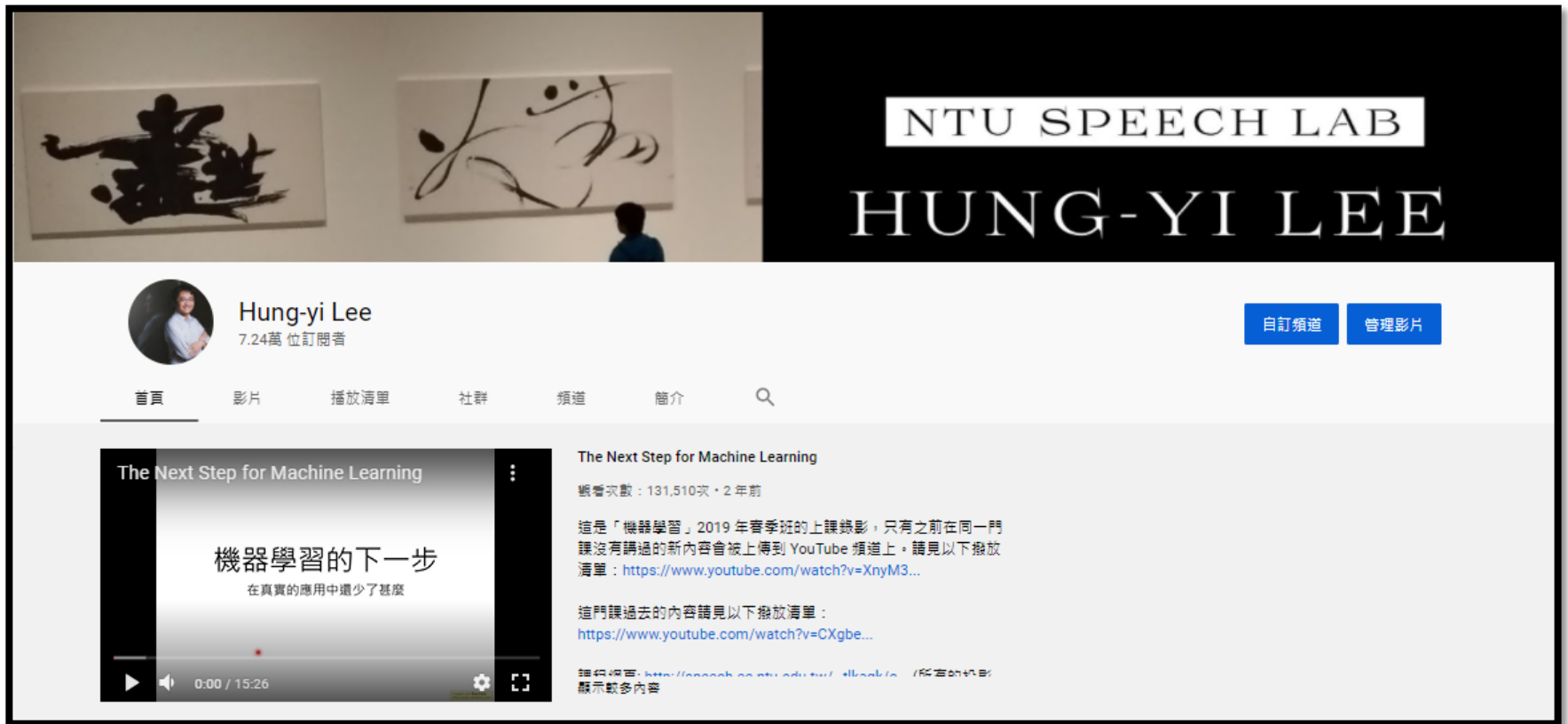
The function we want to find ...

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

1. Function with Unknown Parameters

$$y = f(\qquad\qquad\qquad\qquad\qquad)$$

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.

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

Data from $2017/01/01 - 2020/12/31$

| 2017/01/01 | 01/02 | 01/03 | ...... | 2020/12/30 | 12/31 |

4.8k $\quad$ 4.9k $\quad$ 7.5k $\quad\quad\quad\quad$ 3.4k $\quad$ 9.8k

$0.5k + 1x_1 = y$ $\quad$ 5.3k

$e_1 = |y - \hat{y}| = 0.4k$

label $\hat{y}$

4.9k

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 $2017/01/01 - 2020/12/31$

2017/01/01   01/02   01/03   ......   2020/12/30   12/31

4.8k   4.9k   7.5k   3.4k   9.8k

$0.5k + 1x_1 = y$ 5.4k $0.5k + 1x_1 = y$

$e_2 = |y - \hat{y}| = 2.1k$ e_N

$\hat{y}$ $\hat{y}$

4.9k   7.5k   9.8k

2. Define Loss from Training Data

➢ Loss is a function of parameters $L(b, w)$

➢ Loss: how good a set of values is.

4.8k   4.9k

$$b + wx_1 = y$$

e_1

$\hat{y}$

4.9k

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

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

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

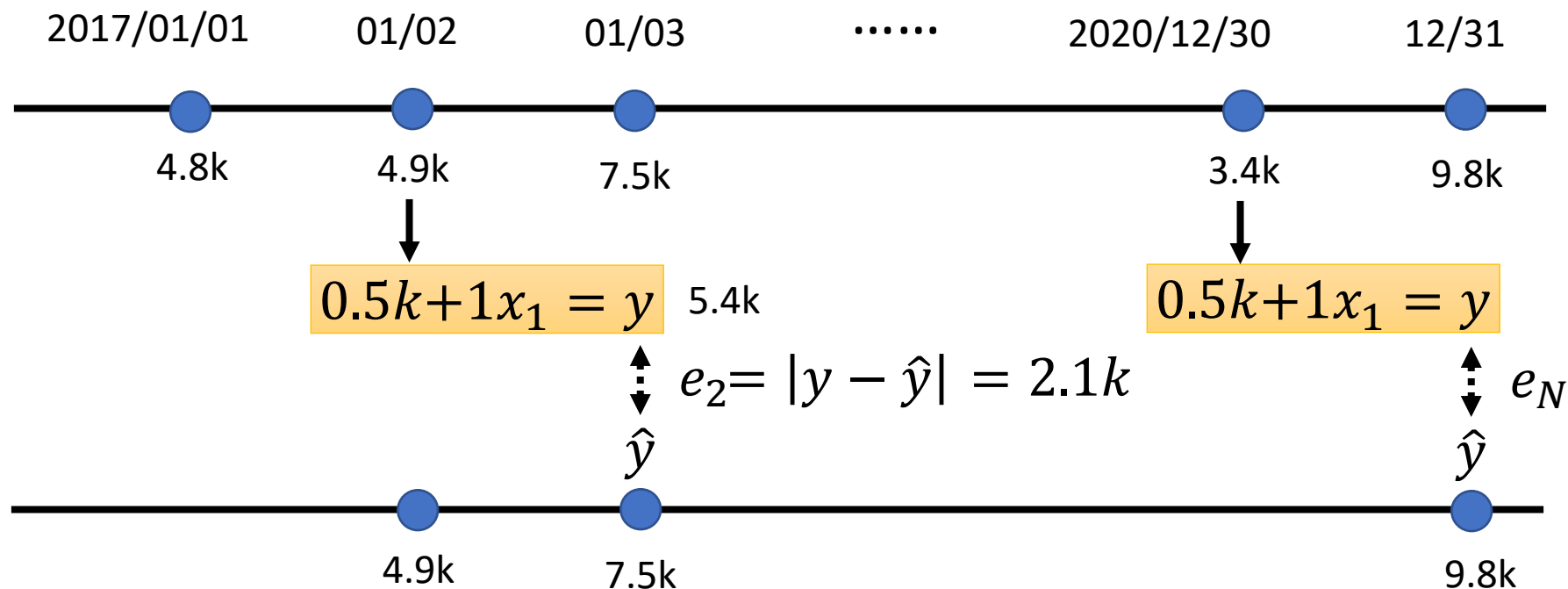If y and $\hat{y}$ are both probability distributions ➡ Cross-entropy

2. Define Loss from Training Data

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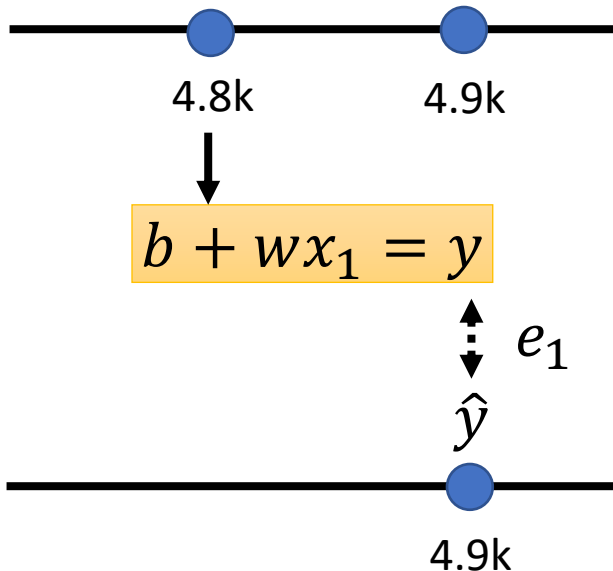
> Loss: how good a set of values is.

Model $y = b + wx_1$
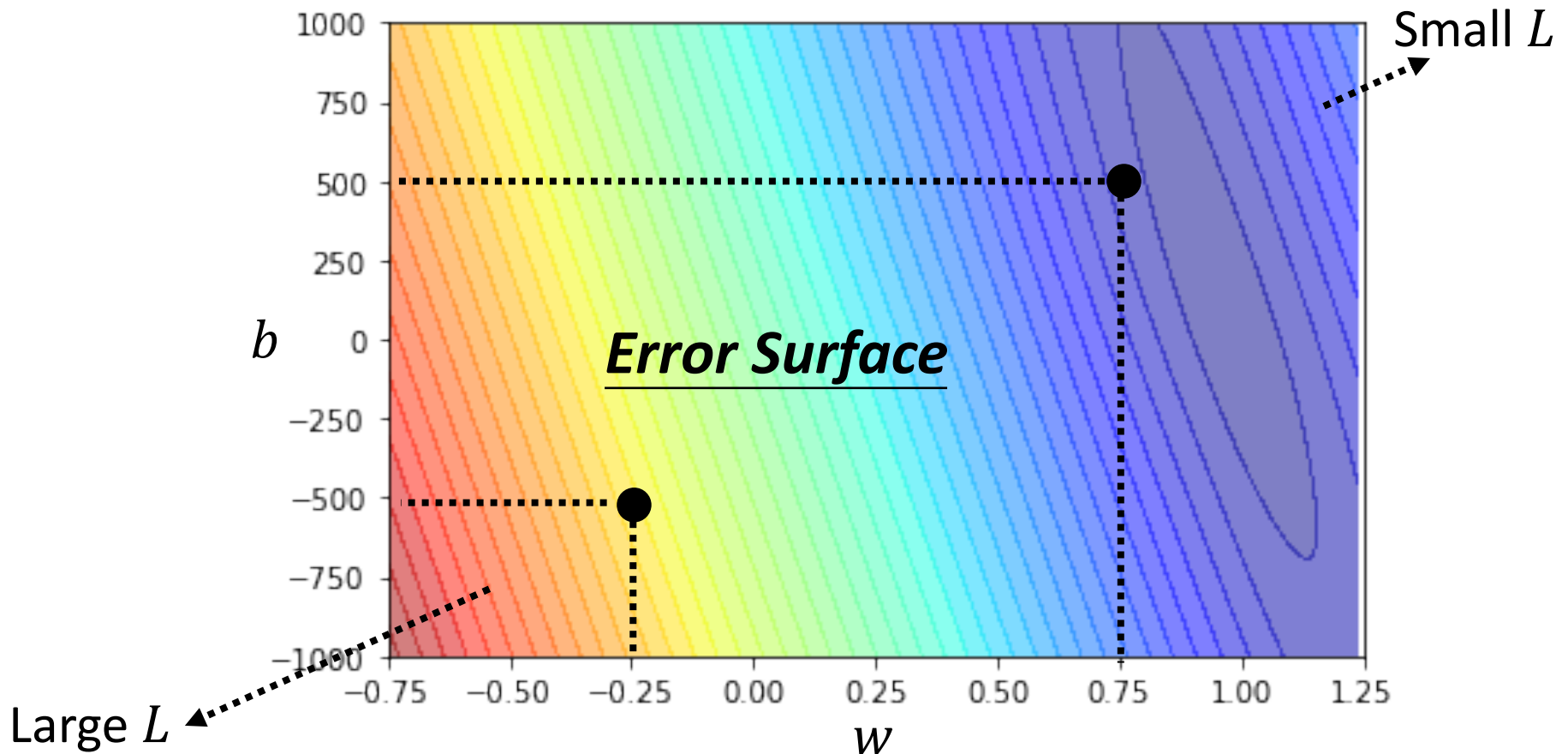
Error Surface

Small L

Large L

3. Optimization

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

Gradient Descent

➢ (Randomly) Pick an initial value w^0

➢ Compute $\dfrac{\partial L}{\partial w}\big|_{w=w^0}$

Negative	➡	Increase w
Positive	➡	Decrease w

Loss L

w^0

w

3. Optimization

$$w^* \blacksquare = arg \min_{w} L \blacksquare$$

Gradient Descent

➢ (Randomly) Pick an initial value w^0

➢ Compute $\left.\dfrac{\partial L}{\partial w}\right|_{w=w^0}$

$$w^1 \leftarrow w^0 - \eta \left.\dfrac{\partial L}{\partial w}\right|_{w=w^0}$$

$\eta \left.\dfrac{\partial L}{\partial w}\right|_{w=w^0}$ η: learning rate

hyperparameters

Loss L

w^0 w^1 w
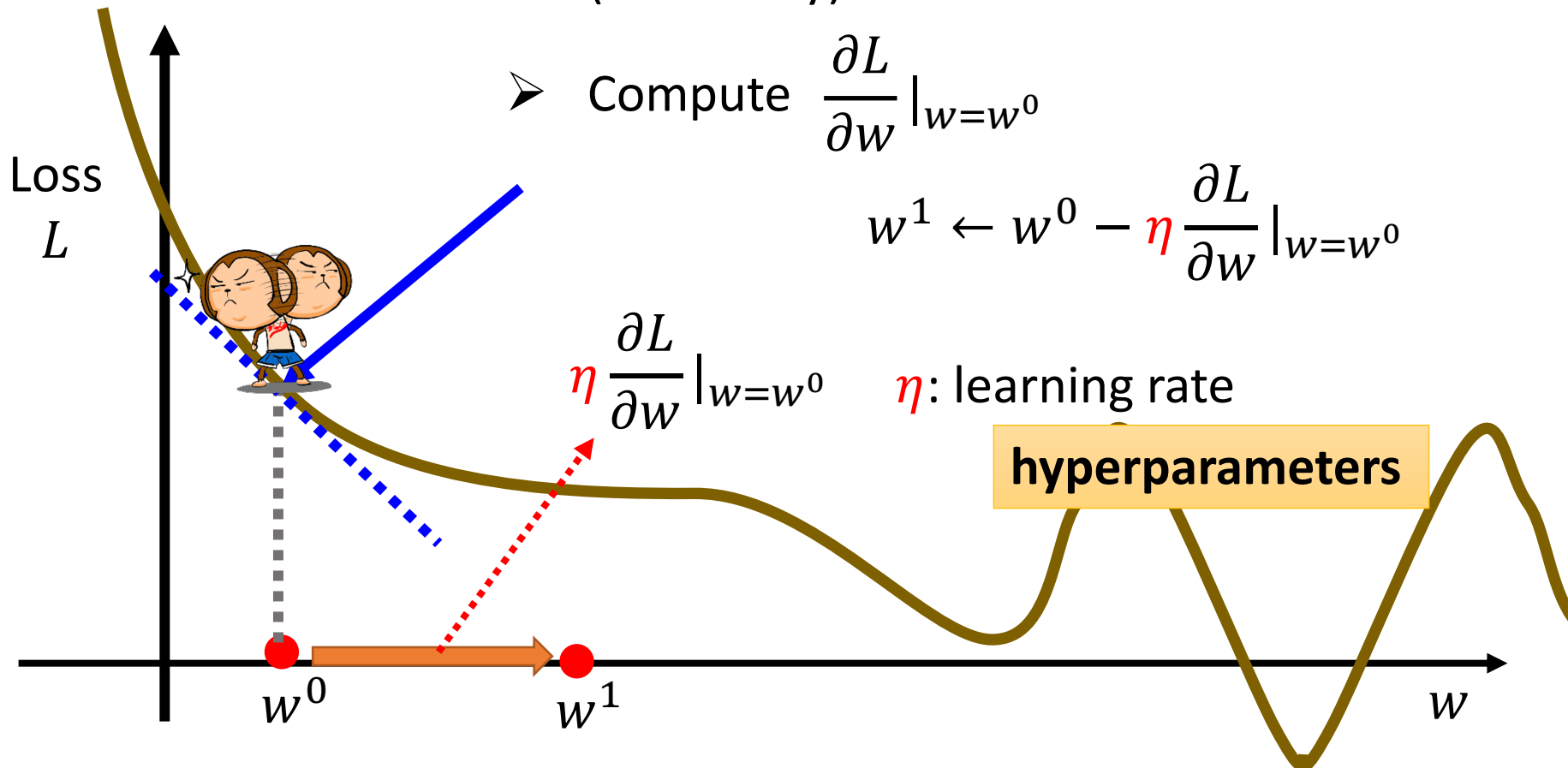
3. Optimization

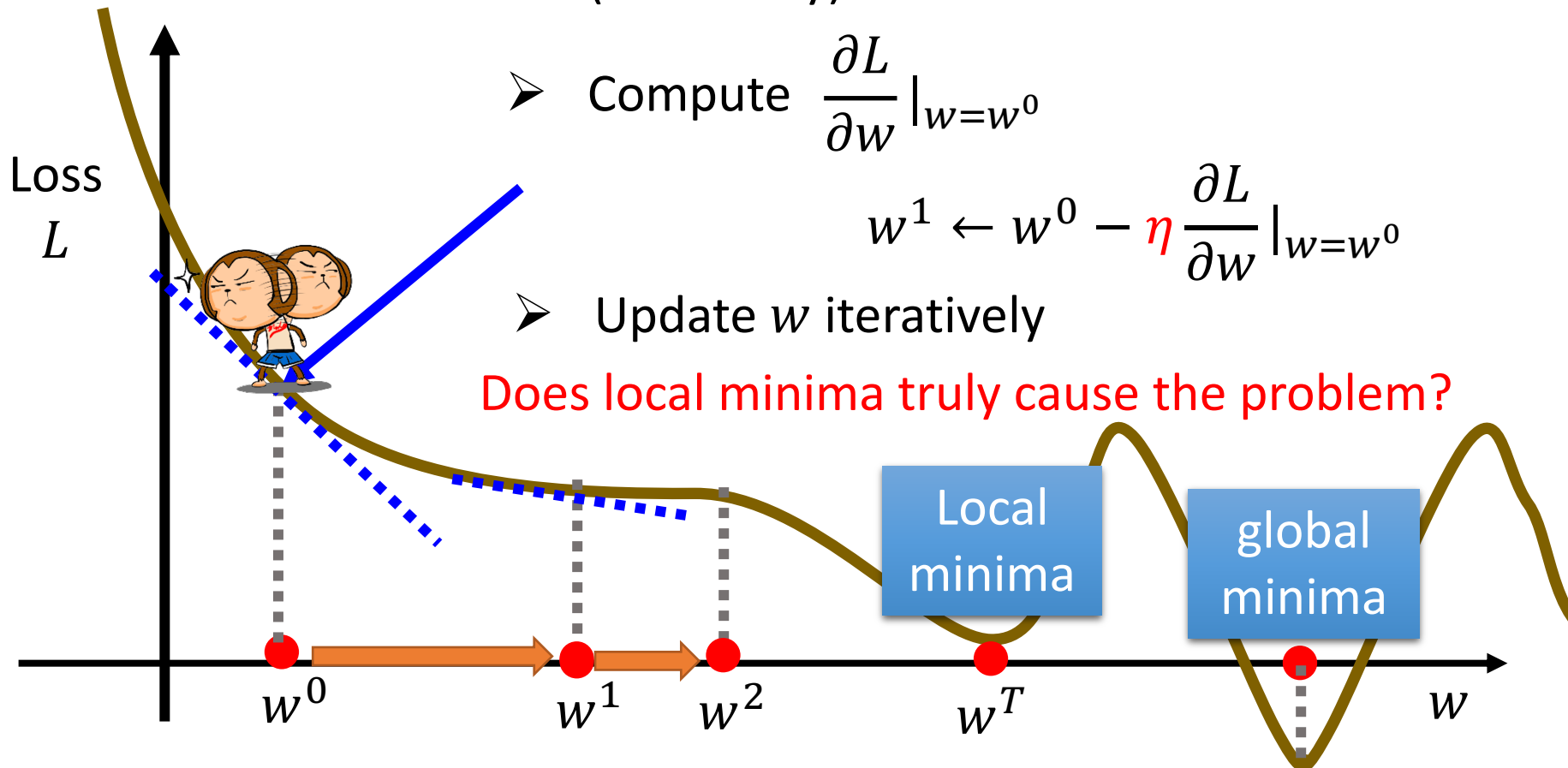$$w^* \blacksquare = arg \min_w L \blacksquare$$
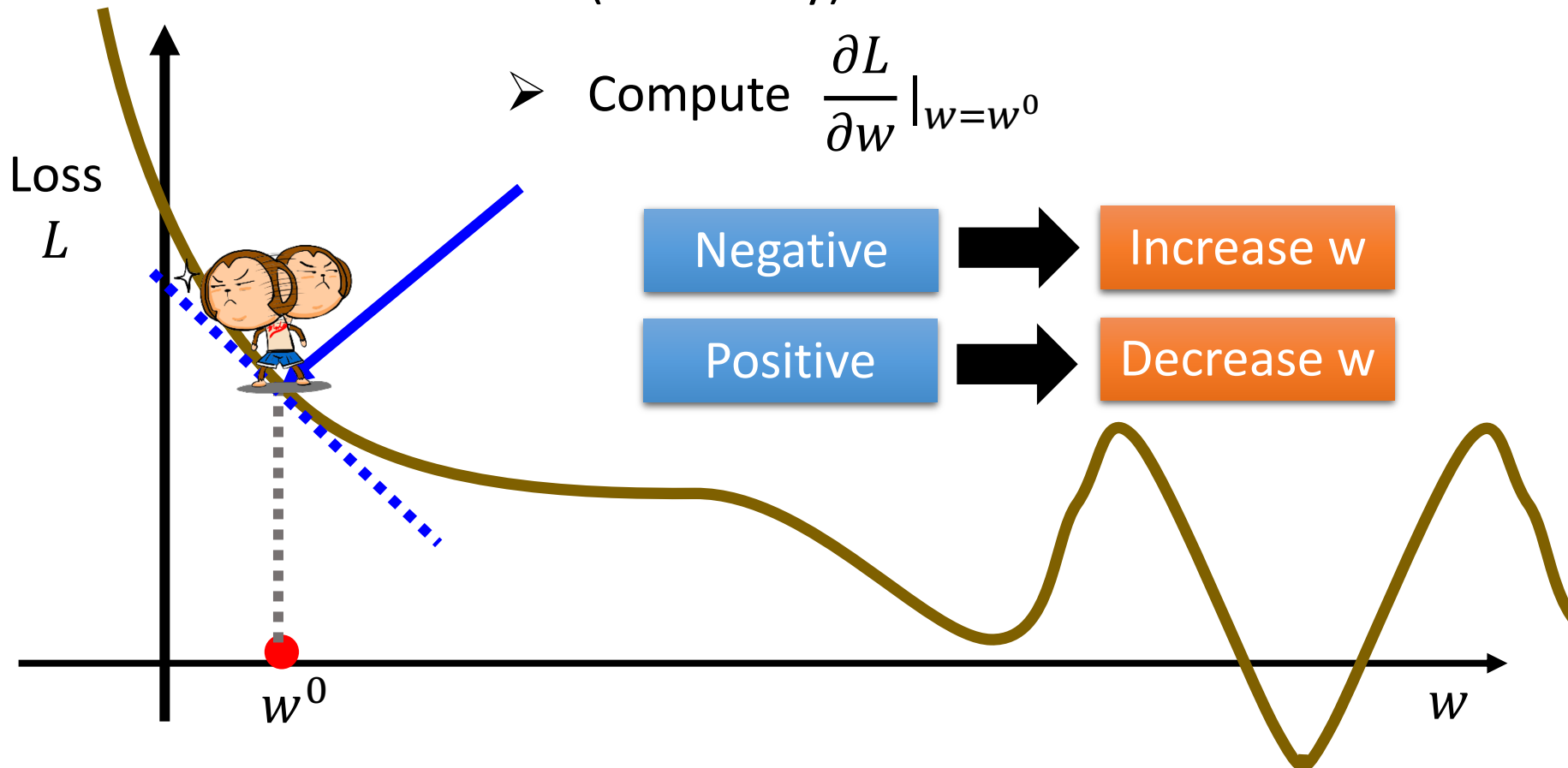
Gradient Descent

➤ (Randomly) Pick an initial value w^0

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

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

➤ Update w iteratively

Does local minima truly cause the problem?

Loss L

Local minima

global minima

w^0 w^1 w^2 w^T w

3. Optimization

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

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

➤ Compute

$$\left.\frac{\partial L}{\partial w}\right|_{w=w^0,b=b^0}$$

$$\left.\frac{\partial L}{\partial b}\right|_{w=w^0,b=b^0}$$

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

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

Can be done in one line in most deep learning frameworks

➤ Update w and b interatively
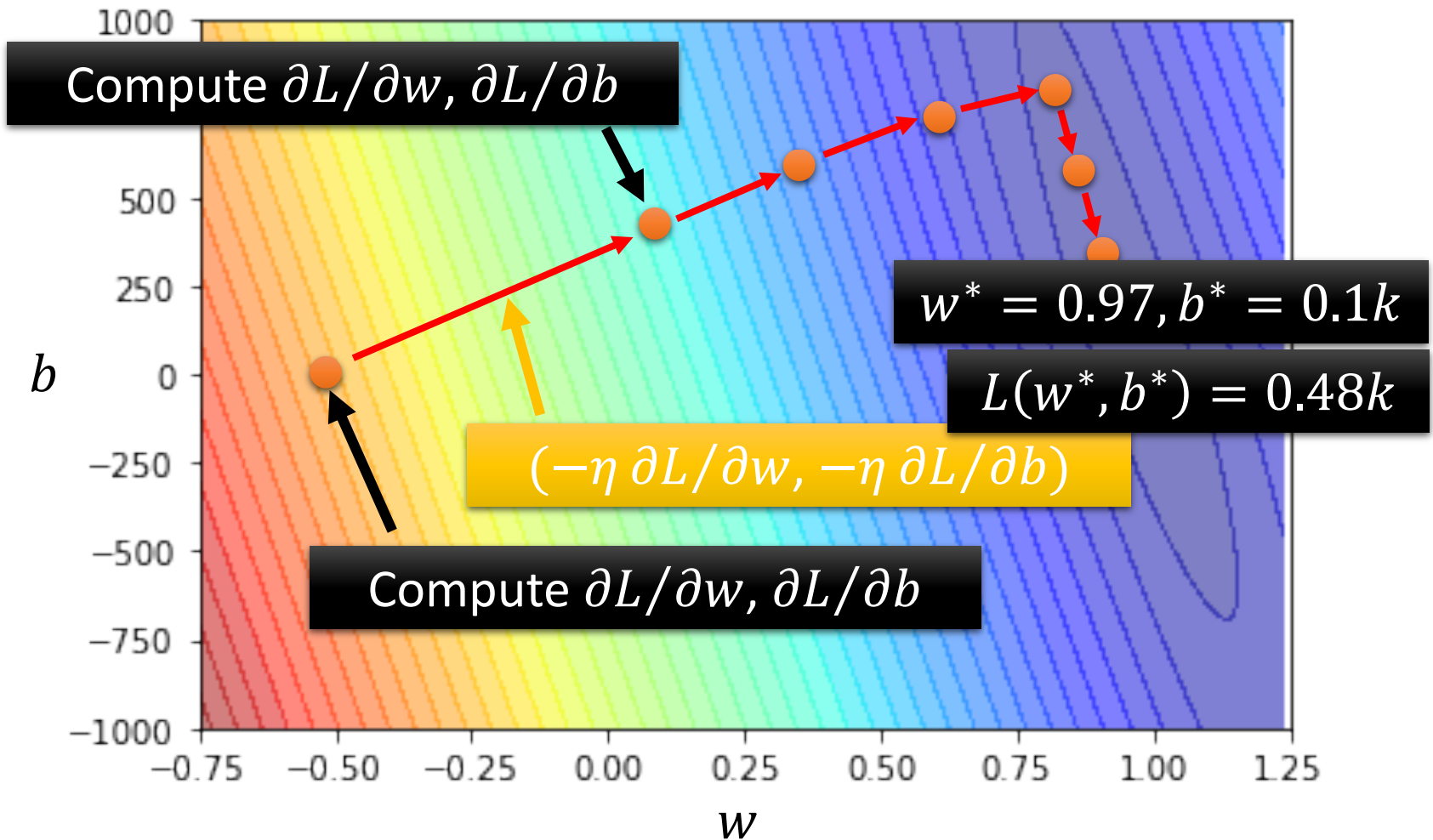
3. Optimization

Model $y = b + wx_1$

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

Compute $\partial L/\partial w, \partial L/\partial b$

Compute $\partial L/\partial w, \partial L/\partial b$

$(-\eta\, \partial L/\partial w, -\eta\, \partial L/\partial b)$

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

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

Machine Learning is so simple ……

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

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

$$y = b + wx_1$$

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

Machine Learning is so simple ……

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

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

$$y = b + wx_1$$

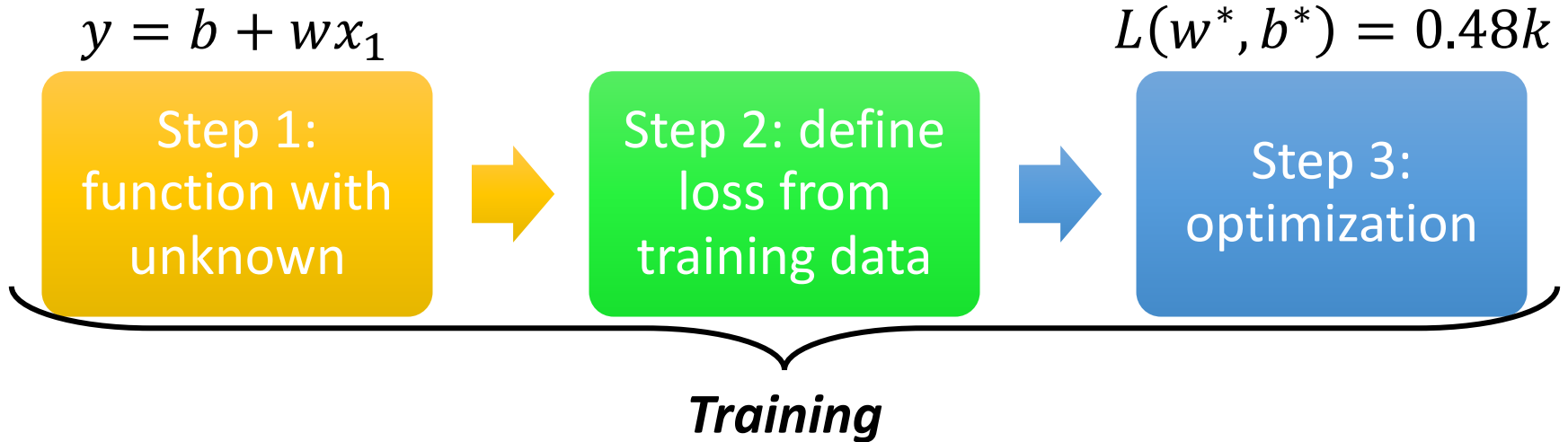| Step 1: function with unknown	→	Step 2: define loss from training data	→	Step 3: optimization
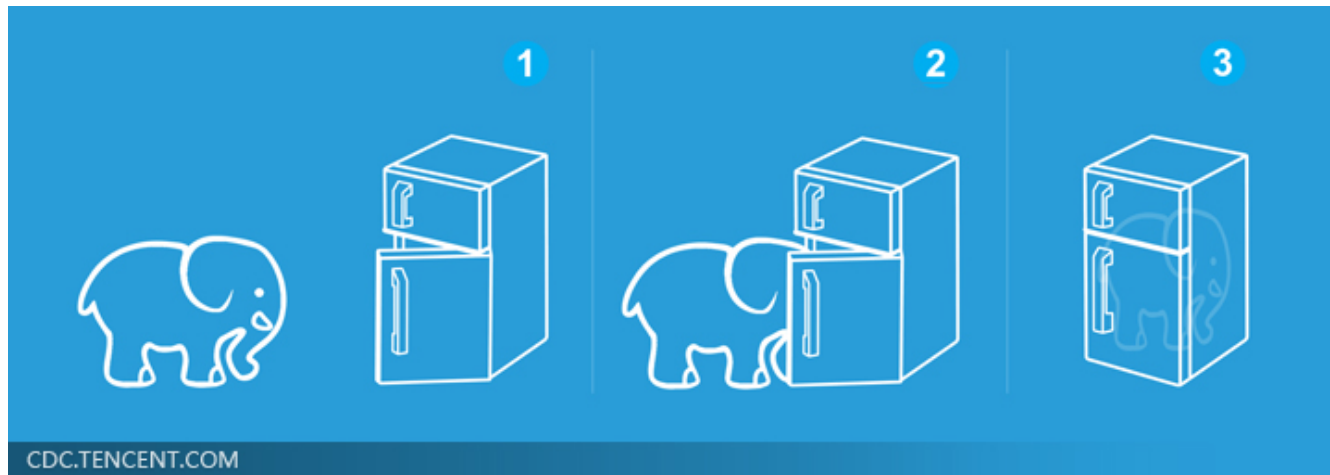
Training

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

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

$$L' = 0.58k$$

$$y = 0.1k + 0.97x_1$$

Views
(k)

2021/01/01   2021/02/14
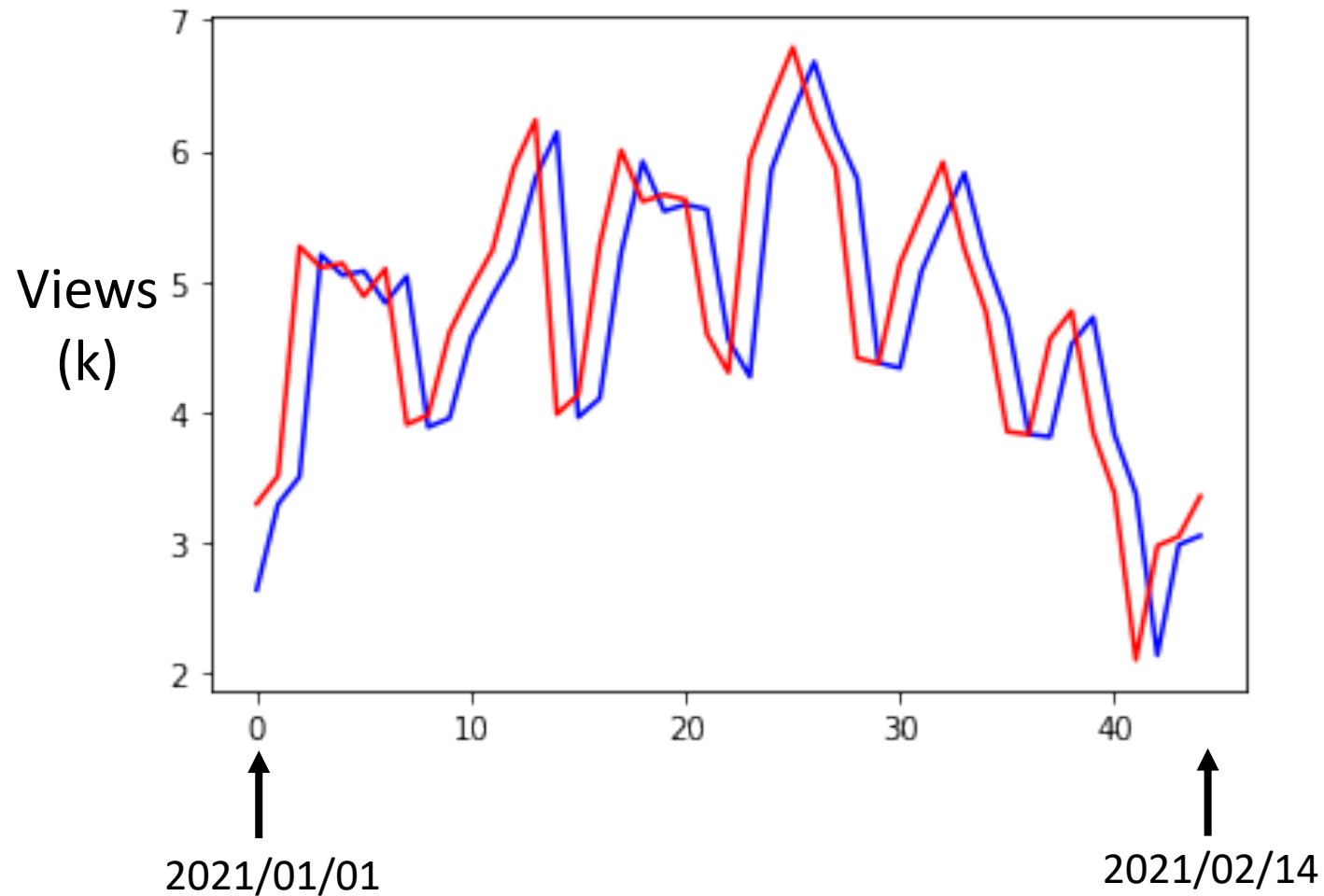
$$y = b + wx_1$$

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

$$L' = 0.46k$$

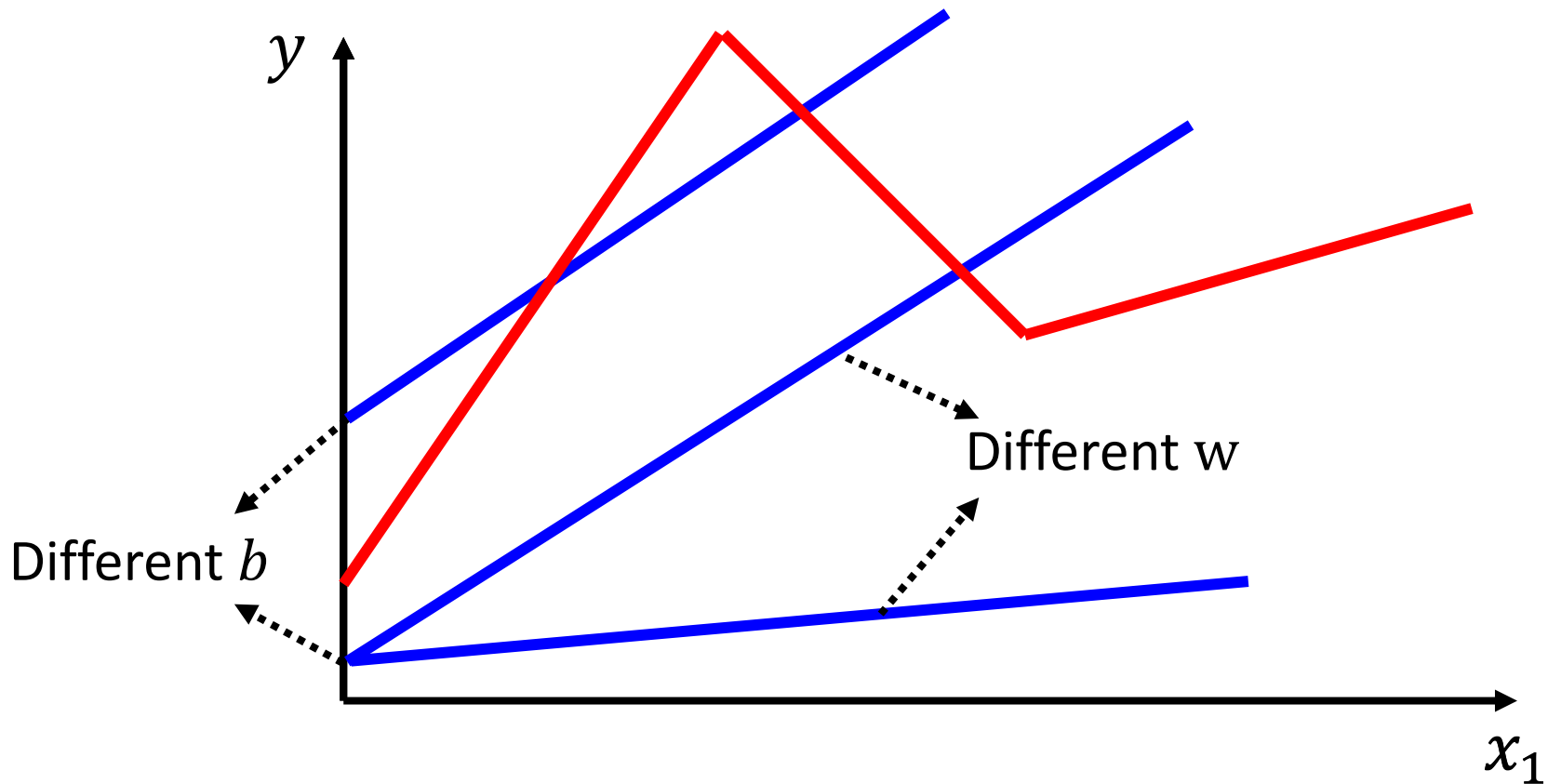$$y = b + \sum_{j=1}^{56} w_j x_j$$

2017 - 2020

$$L = 0.32k$$

2021

$$L' = 0.46k$$
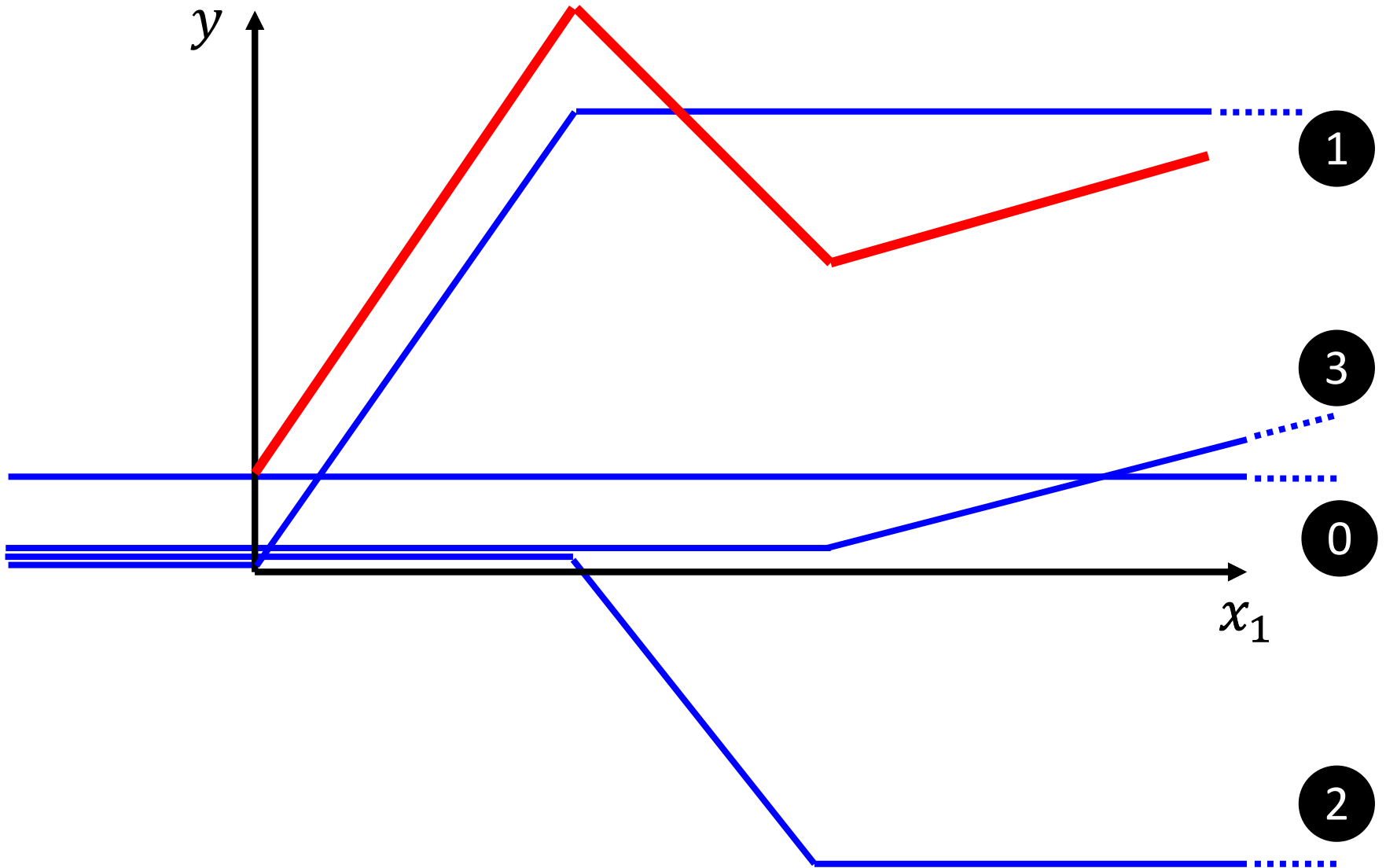
Linear models

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

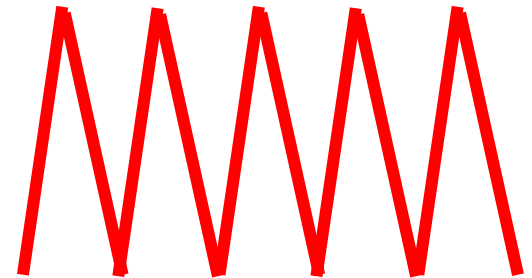Linear models have severe limitation. ***Model Bias***

We need a more flexible model!

red curve = constant + sum of a set of

y

x_1

1

3

0

2

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.
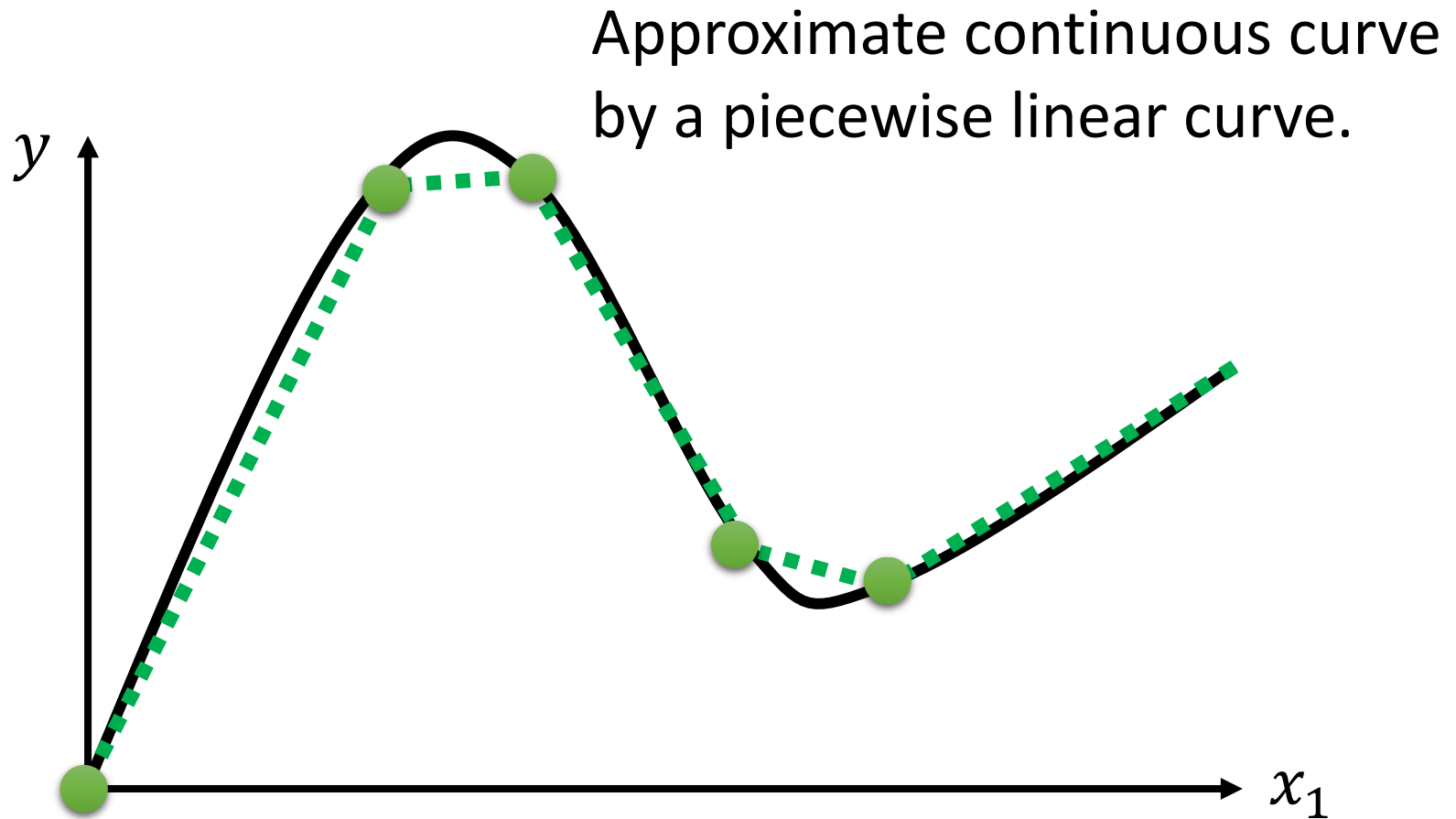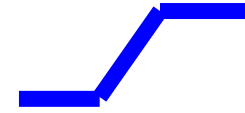

To have good approximation, we need sufficient pieces.

red curve = constant + sum of a set of [Hard Sigmoid]

How to represent this function?

Hard Sigmoid
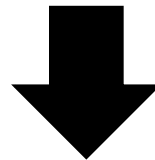
Sigmoid Function

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

$$= c \ sigmoid(b + wx_1)$$

x_1

Different w

Change slopes

Different b

Shift

Different c

Change height
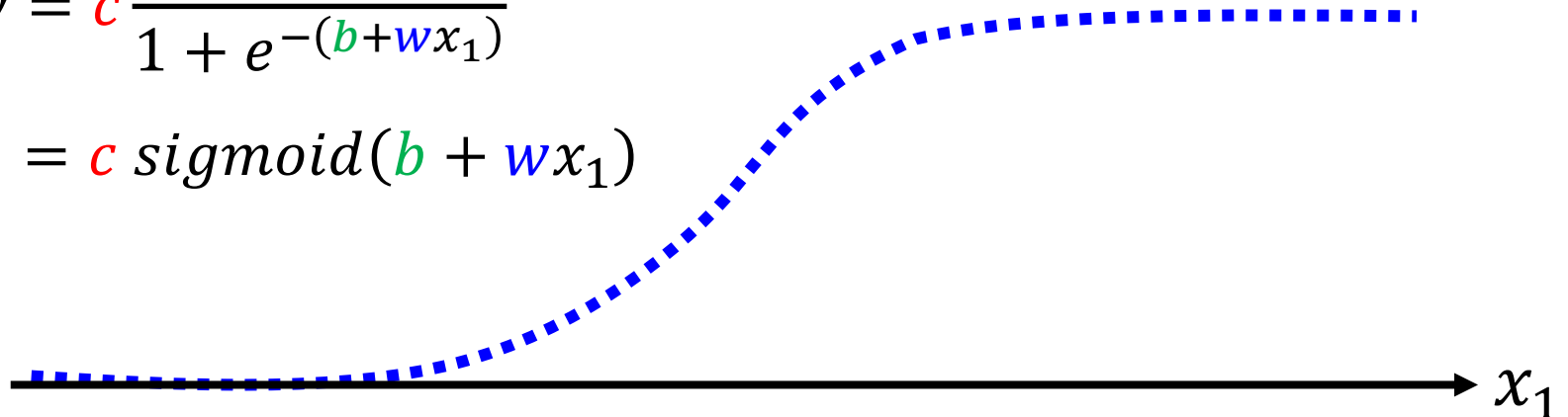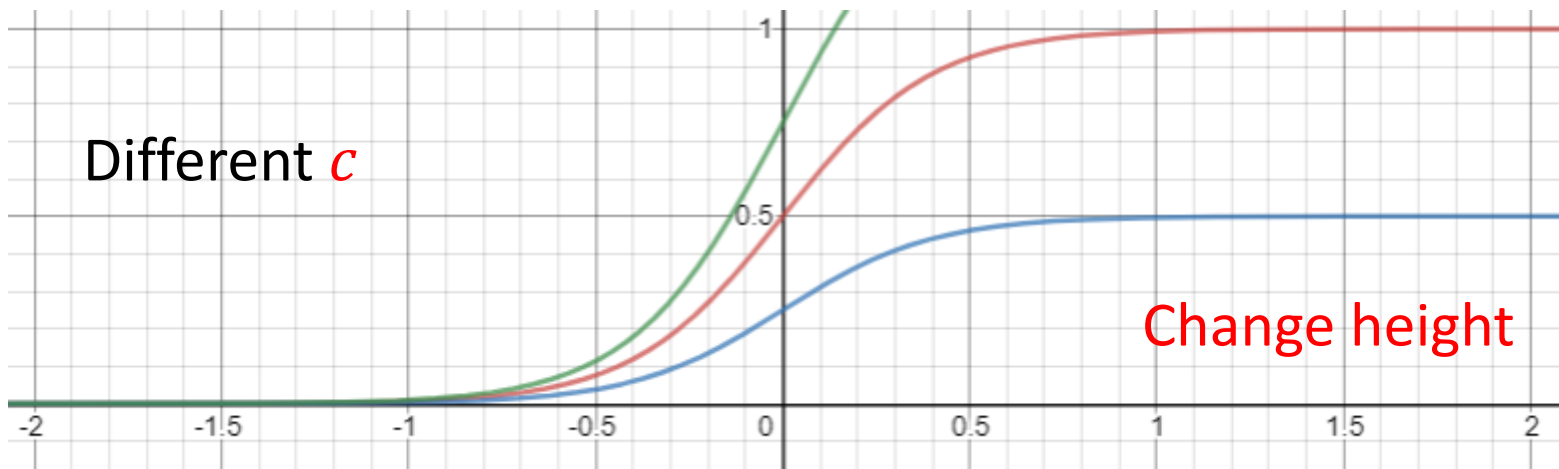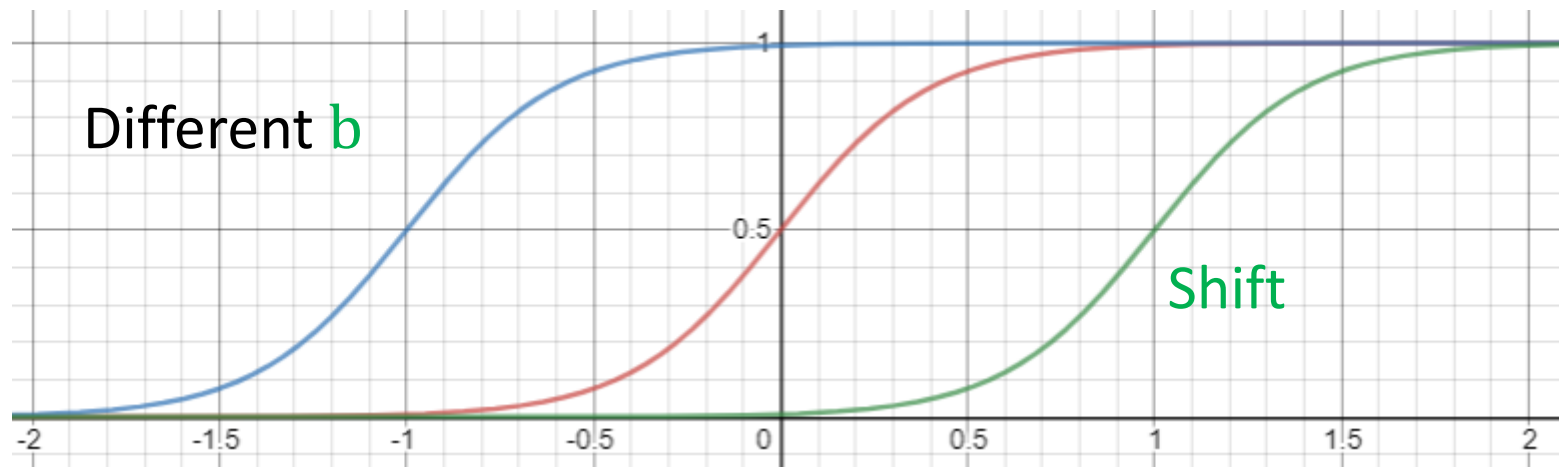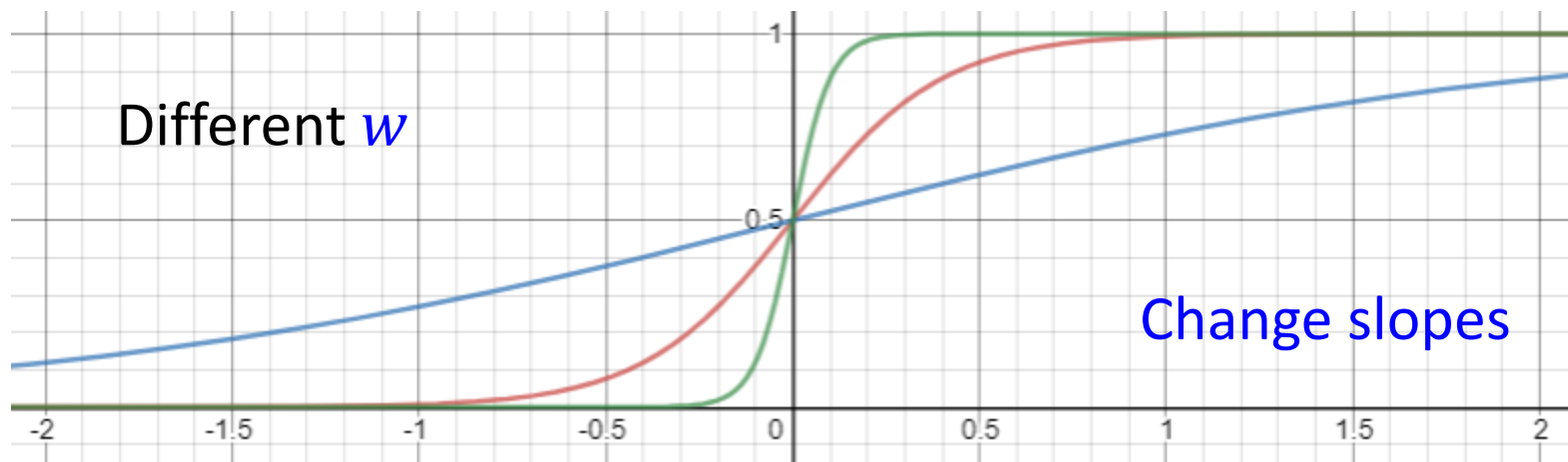
red curve = sum of a set of + constant

$c_1 \ sigmoid(b_1 + w_1 x_1)$ **1**

$c_3 \ sigmoid(b_3 + w_3 x_1)$ **3**

0

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

0   **1** + **2** + **3**

$c_2 \ sigmoid(b_2 + w_2 x_1)$ **2**
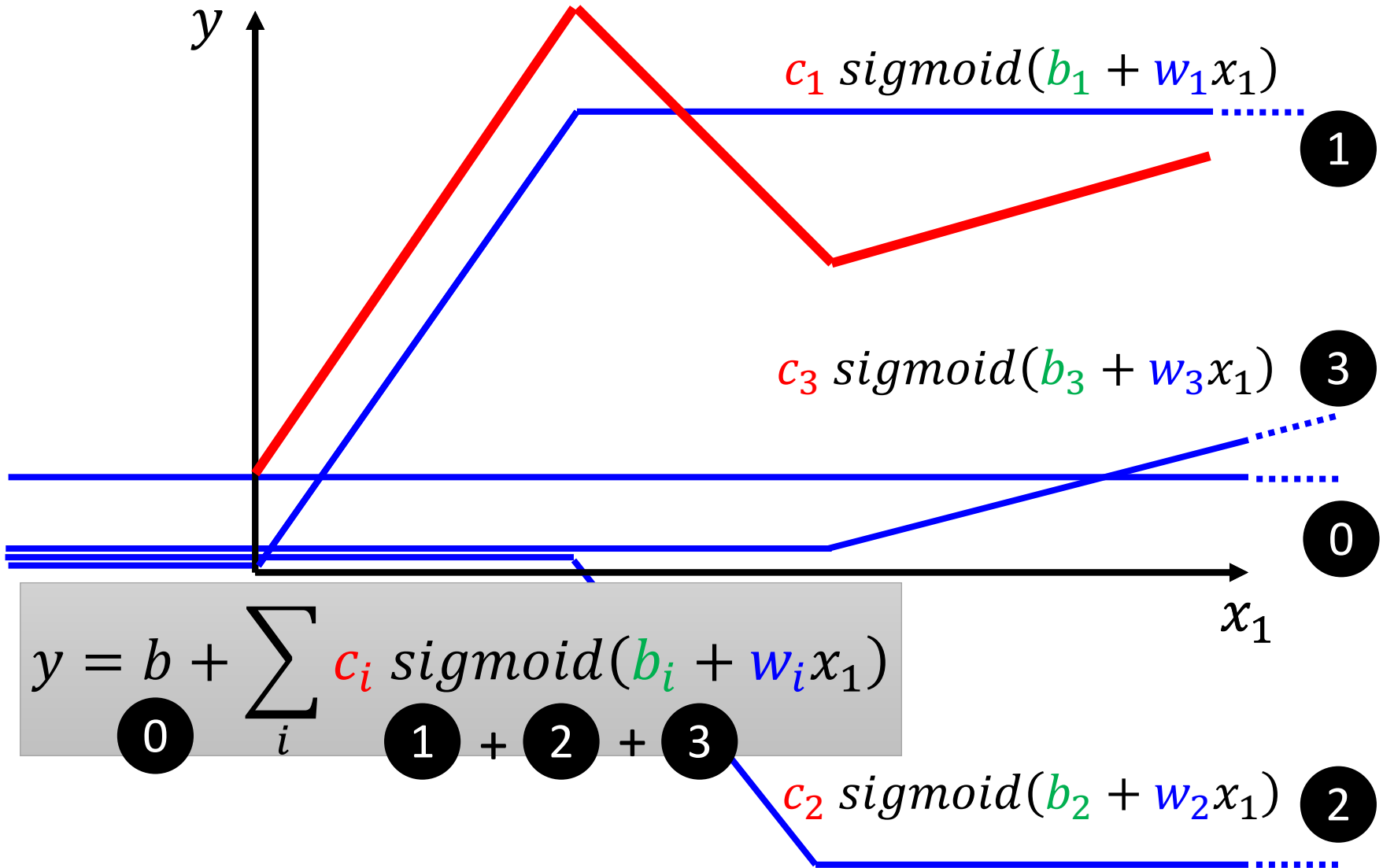
New Model: More Features

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

$$y = b + \sum_i {\color{red}c_i}\ sigmoid(\underline{{\color{green}b_i} + {\color{blue}w_i}x_1})$$

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

$$y = b + \sum_i {\color{red}c_i}\ sigmoid\left({\color{green}b_i} + \underline{\sum_j {\color{blue}w_{ij}}x_i}\right)$$

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

$j: 1, 2, 3$
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

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

$r_3 = b_3 + w_{31} x_1 + w_{32} x_2 + w_{33} x_3$
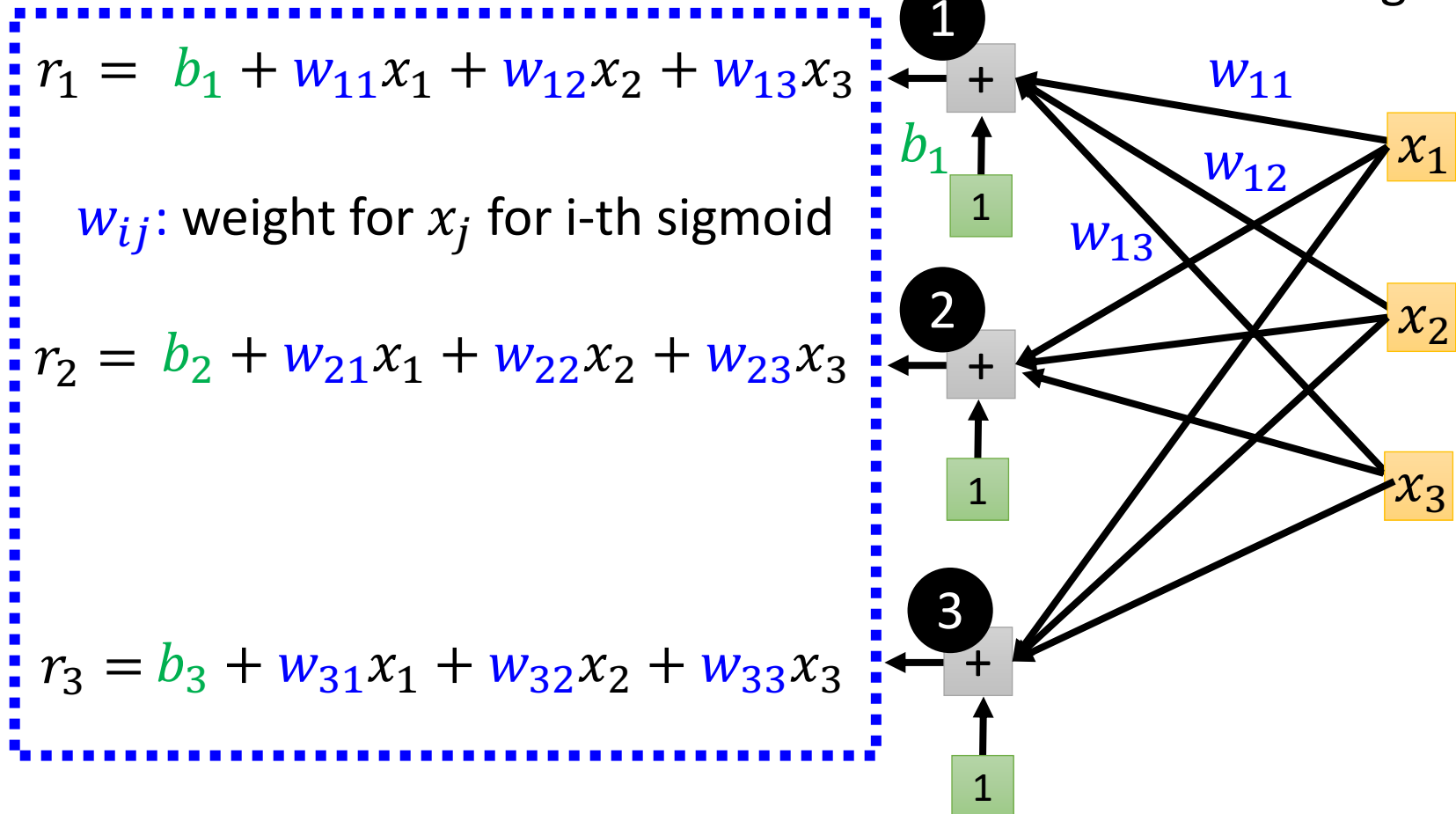
$$y = b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij}x_j\right) \qquad \begin{array}{l} i: 1,2,3 \\ j: 1,2,3 \end{array}$$

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

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

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

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

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

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

$i : 1,2,3$

$j : 1,2,3$

$r = b + W \; x$

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

$i: 1,2,3$

$j: 1,2,3$

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

b_1

w_{11}

w_{12}

w_{13}

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

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

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

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

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

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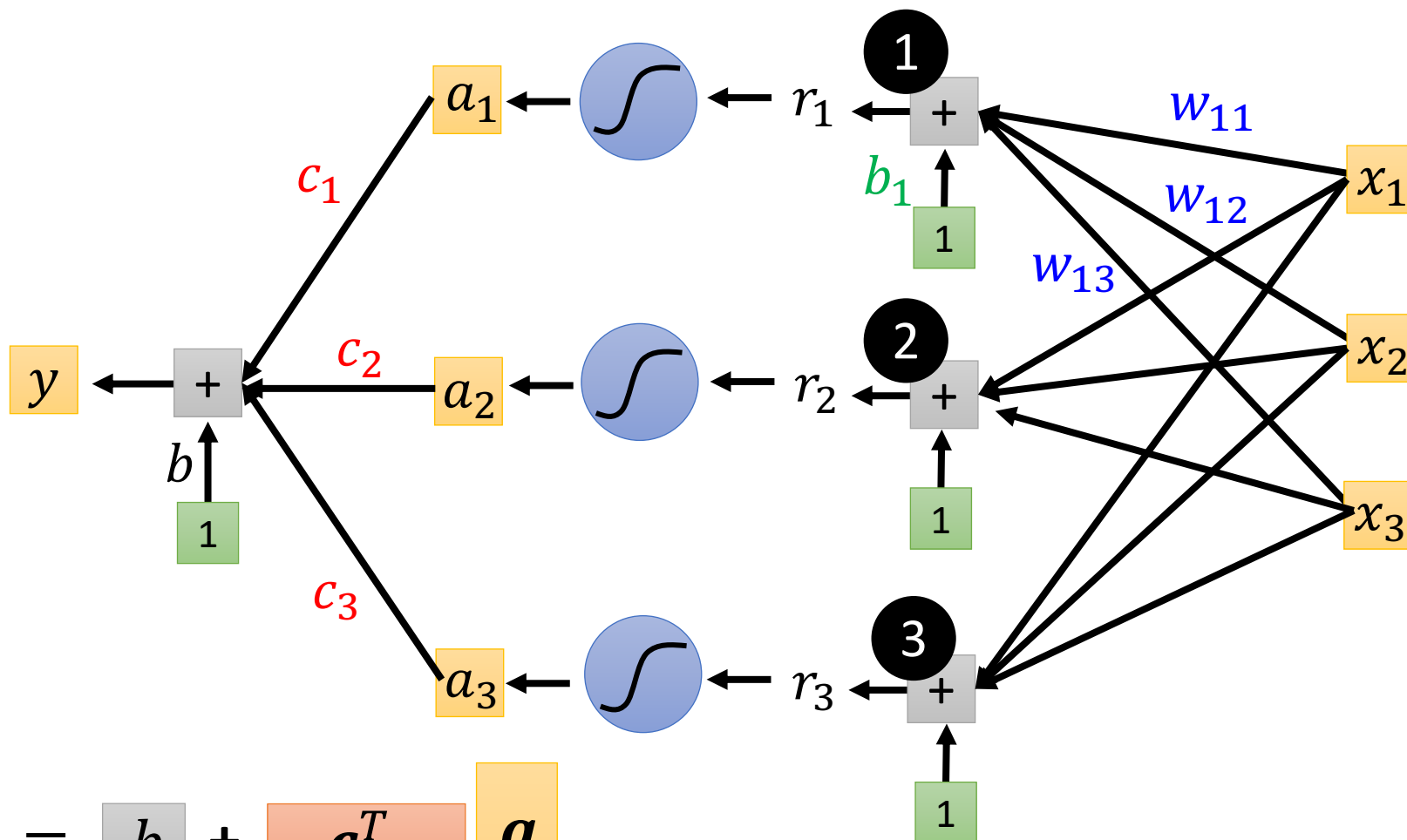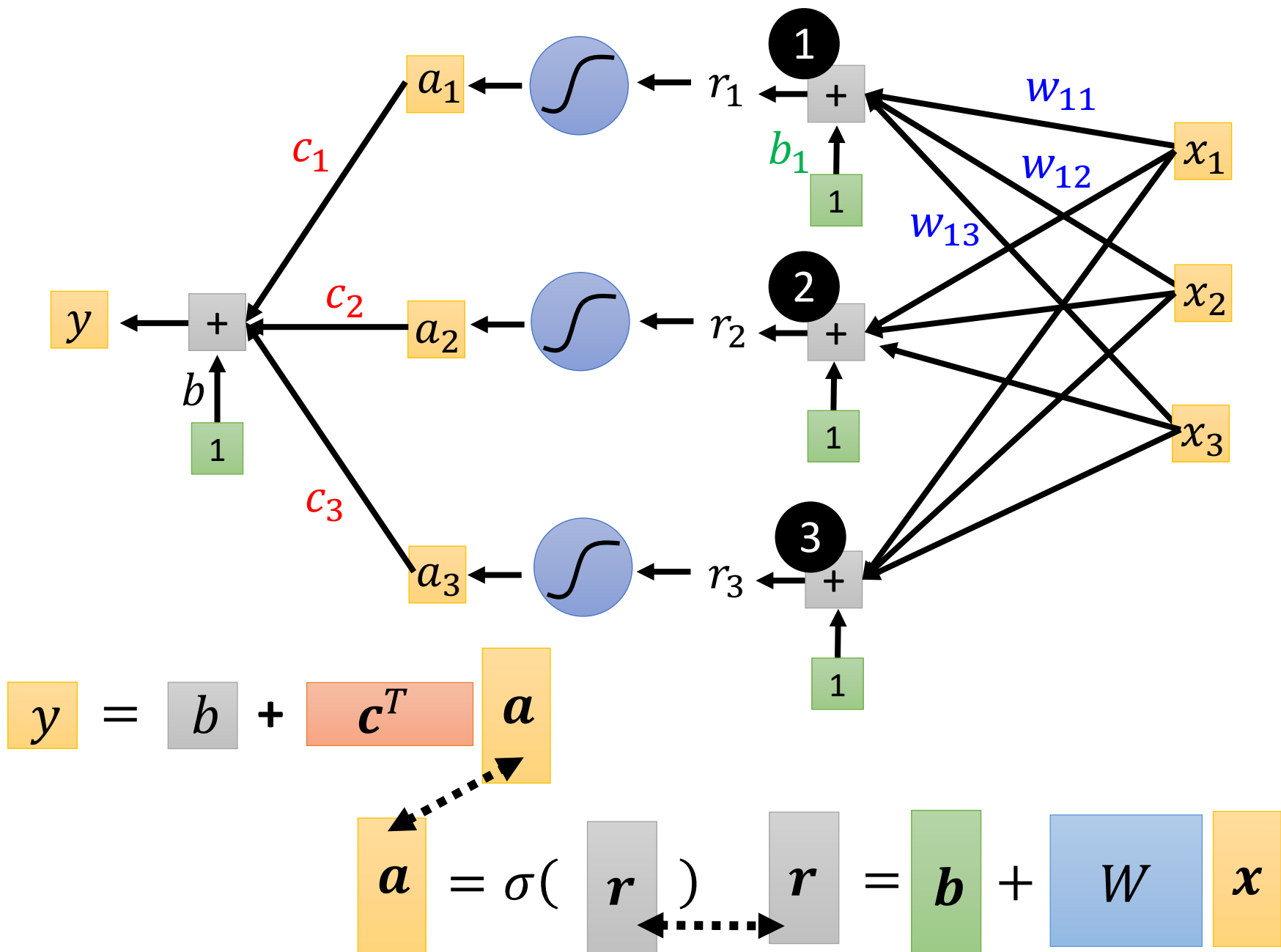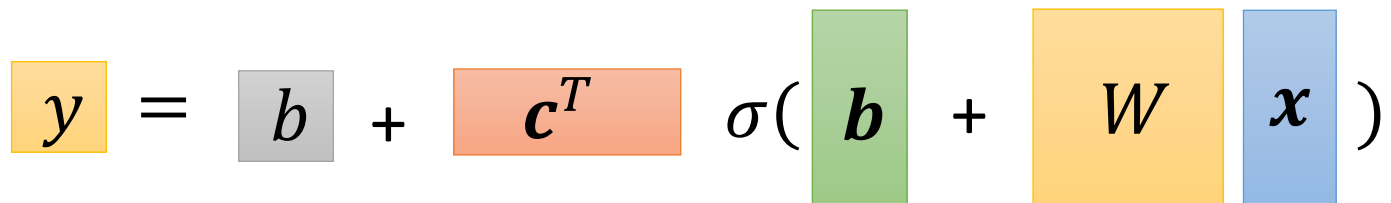
$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W \boldsymbol{x})$$

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

Rows of W

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

Back to ML Framework

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

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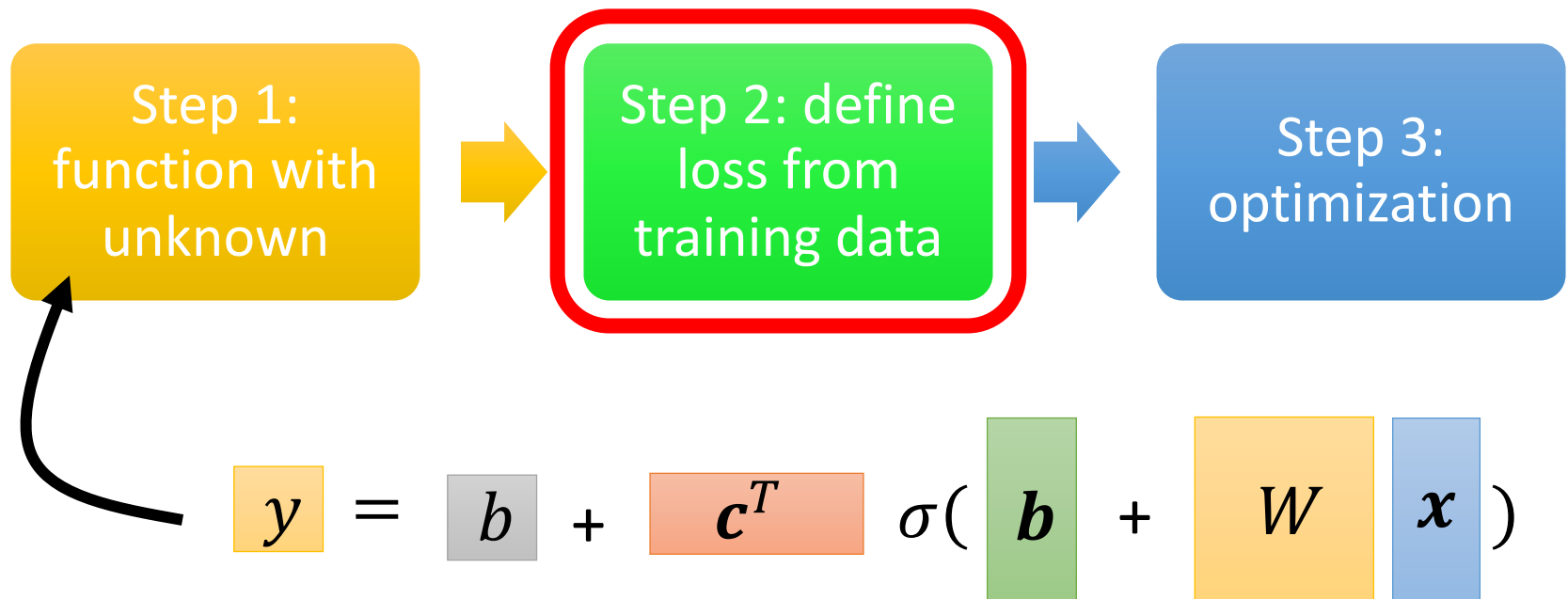
Loss

➢ Loss is a function of parameters $L(\theta)$

➢ Loss means how good a set of values is.

feature

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

label $\hat{y}$

e

Given a set of values
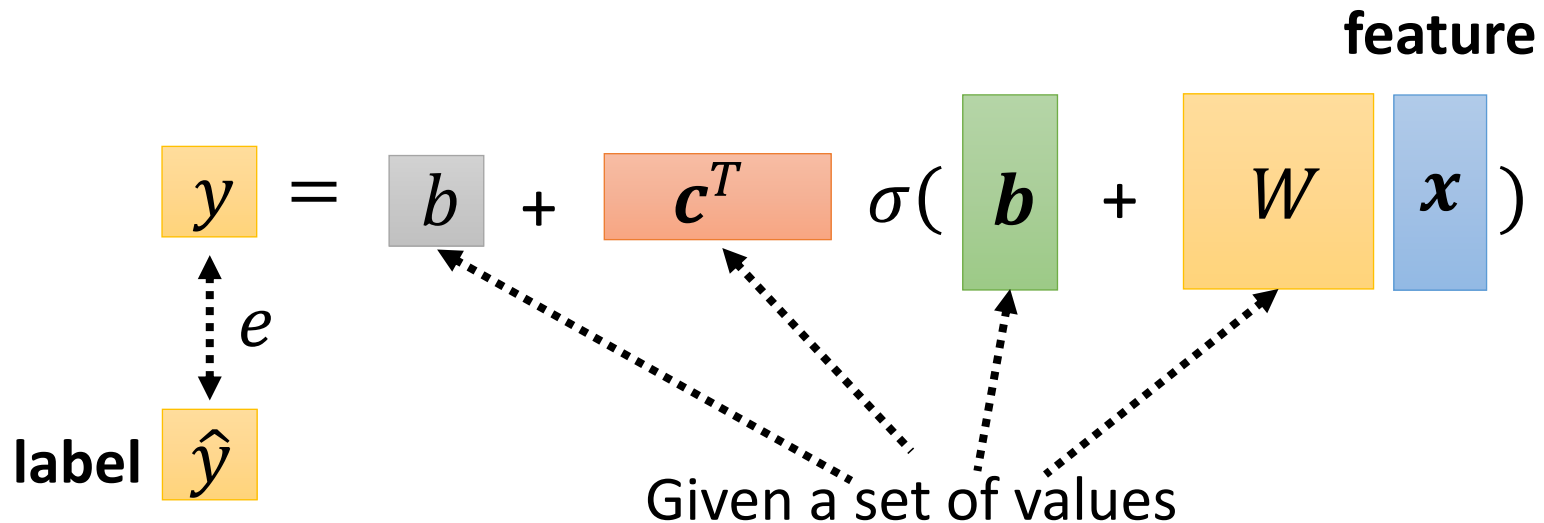
Loss: $\quad L = \dfrac{1}{N} \sum_n e_n$

Back to ML Framework

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

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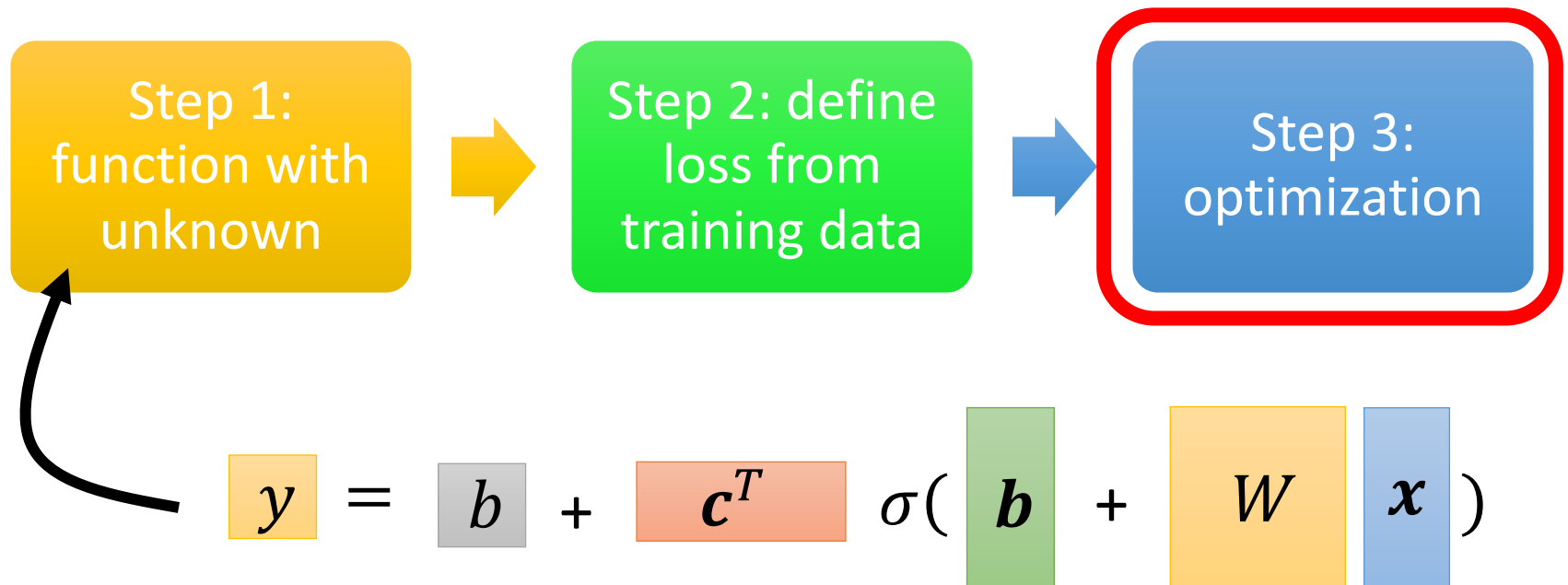
Optimization of New Model

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

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

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

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

gradient

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

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

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 $\boldsymbol{g} = \nabla L^1(\boldsymbol{\theta}^0)$ L^1

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

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

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

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

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

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

B

batch

batch

batch

batch

batch

L

N
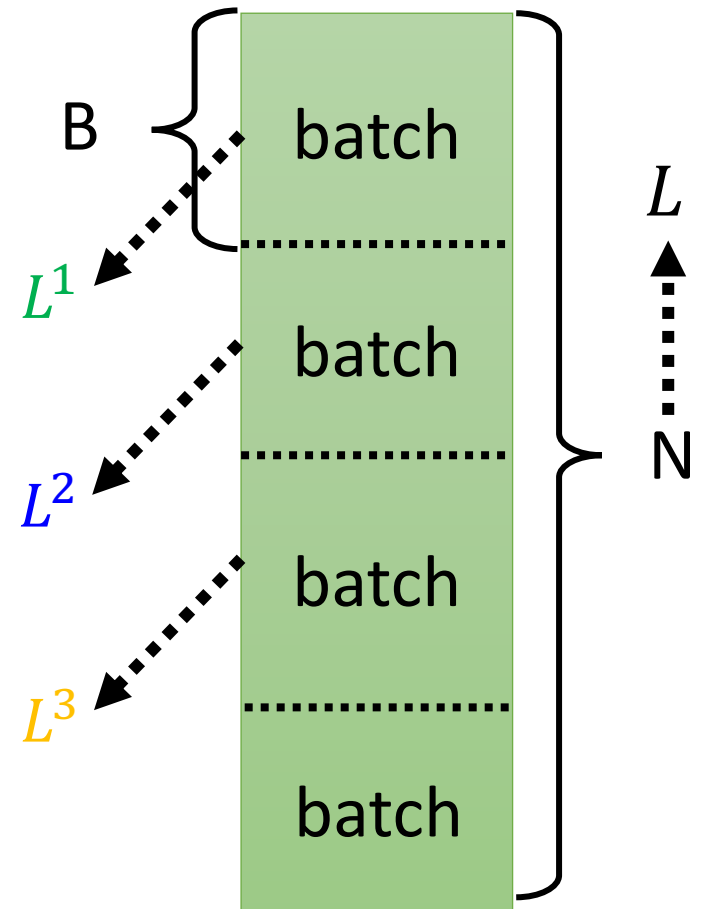
Optimization of New Model

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)

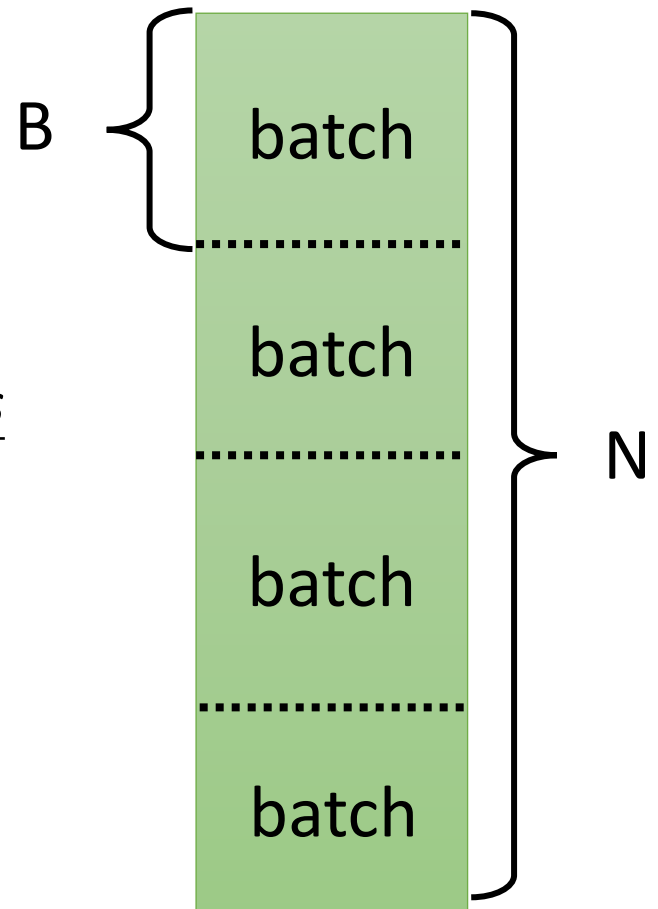How many update in **1 epoch**?

10 updates

B

batch

batch

batch

batch

N

Back to ML Framework

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

$$y = b + \boldsymbol{c}^T \, \sigma(\boldsymbol{b} + W\boldsymbol{x})$$
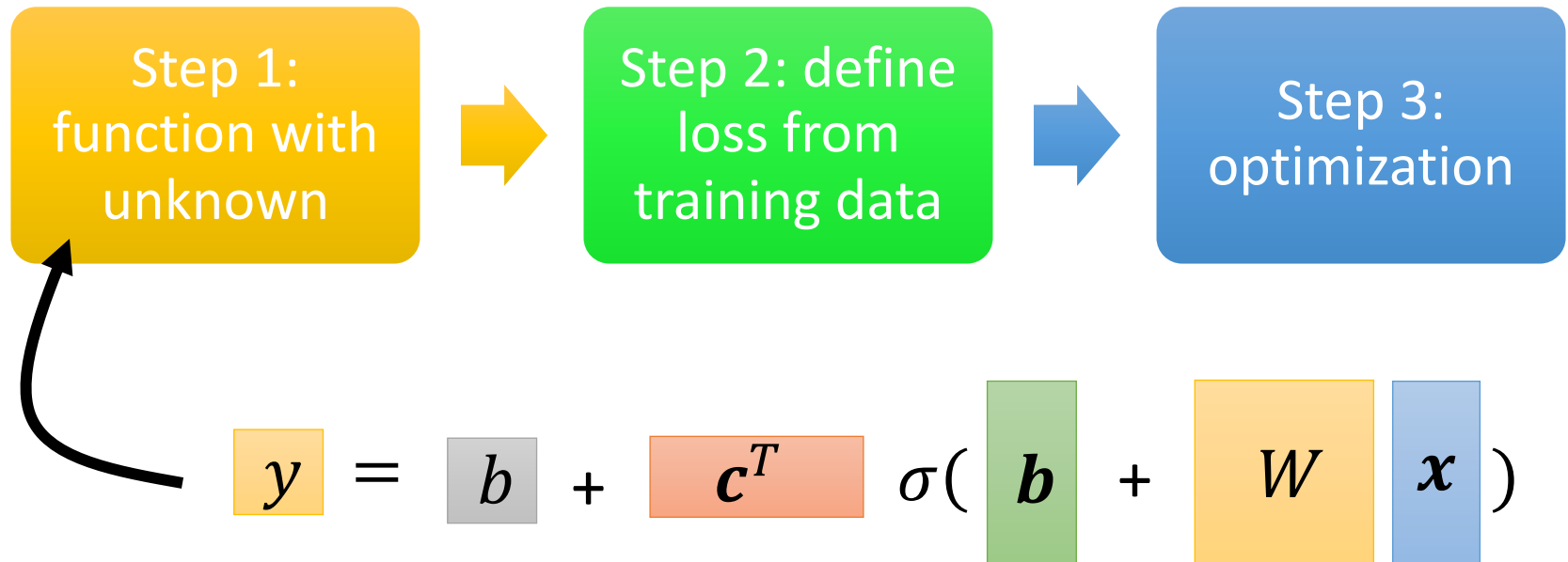
More variety of models …

Sigmoid → ReLU

How to represent
this function?
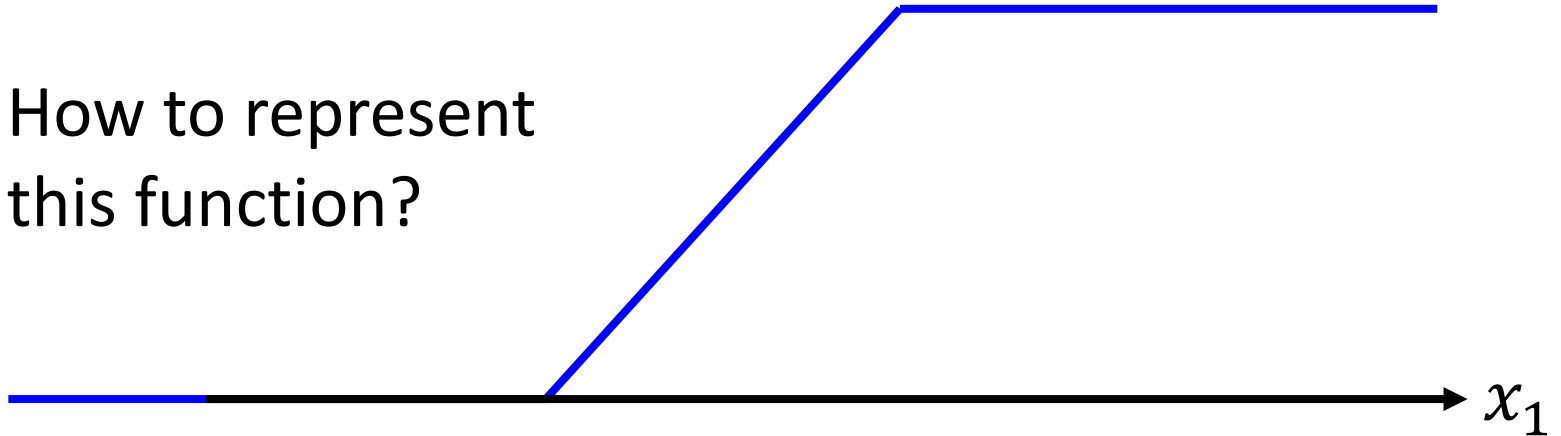
Rectified Linear
Unit (ReLU)

$c \, max(0, b + wx_1)$

$c' \, max(0, b' + w'x_1)$
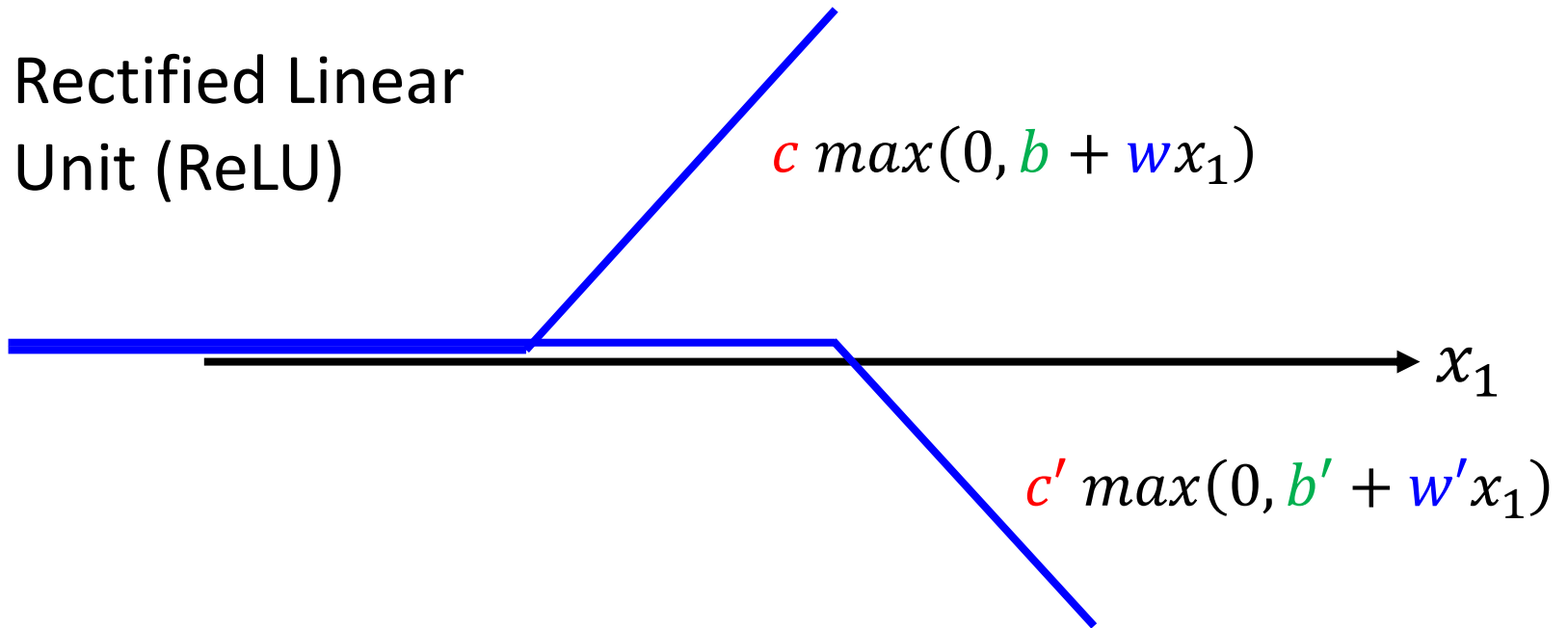
Sigmoid → ReLU

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

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

Back to ML Framework

Step 1: function with unknown $\rightarrow$ Step 2: define loss from training data $\rightarrow$ Step 3: optimization

$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W \boldsymbol{x})$$
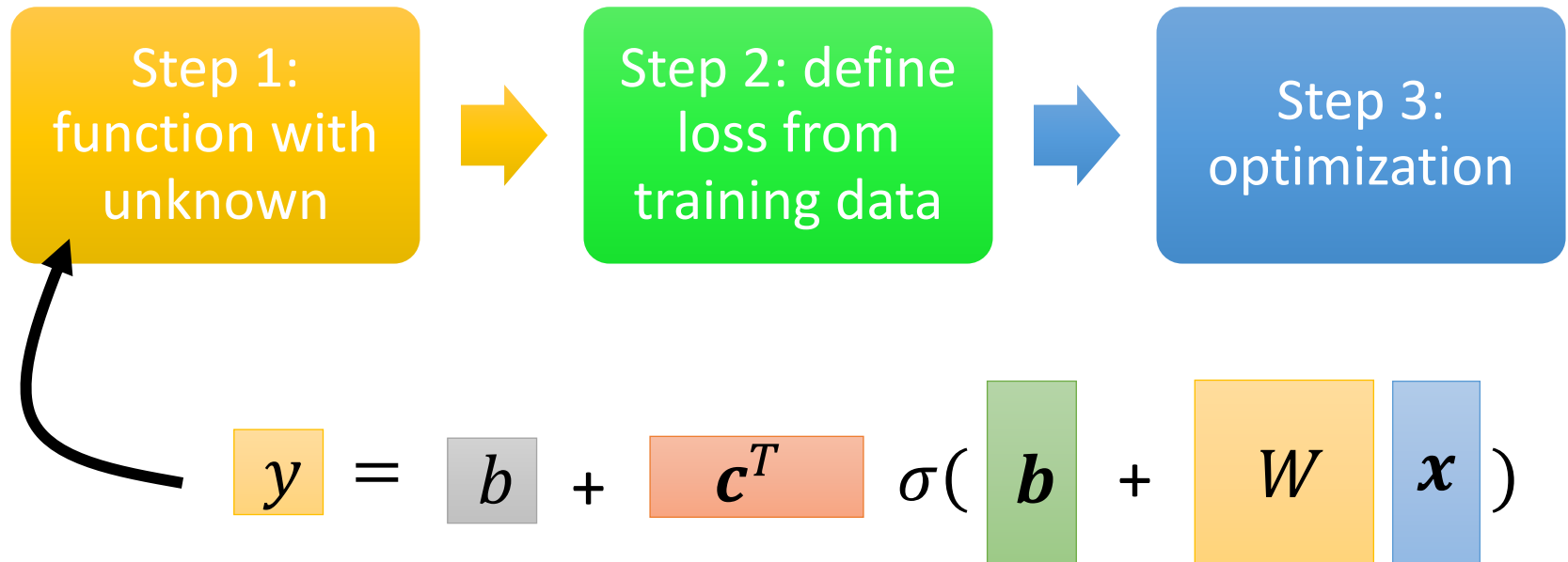
Even more variety of models …

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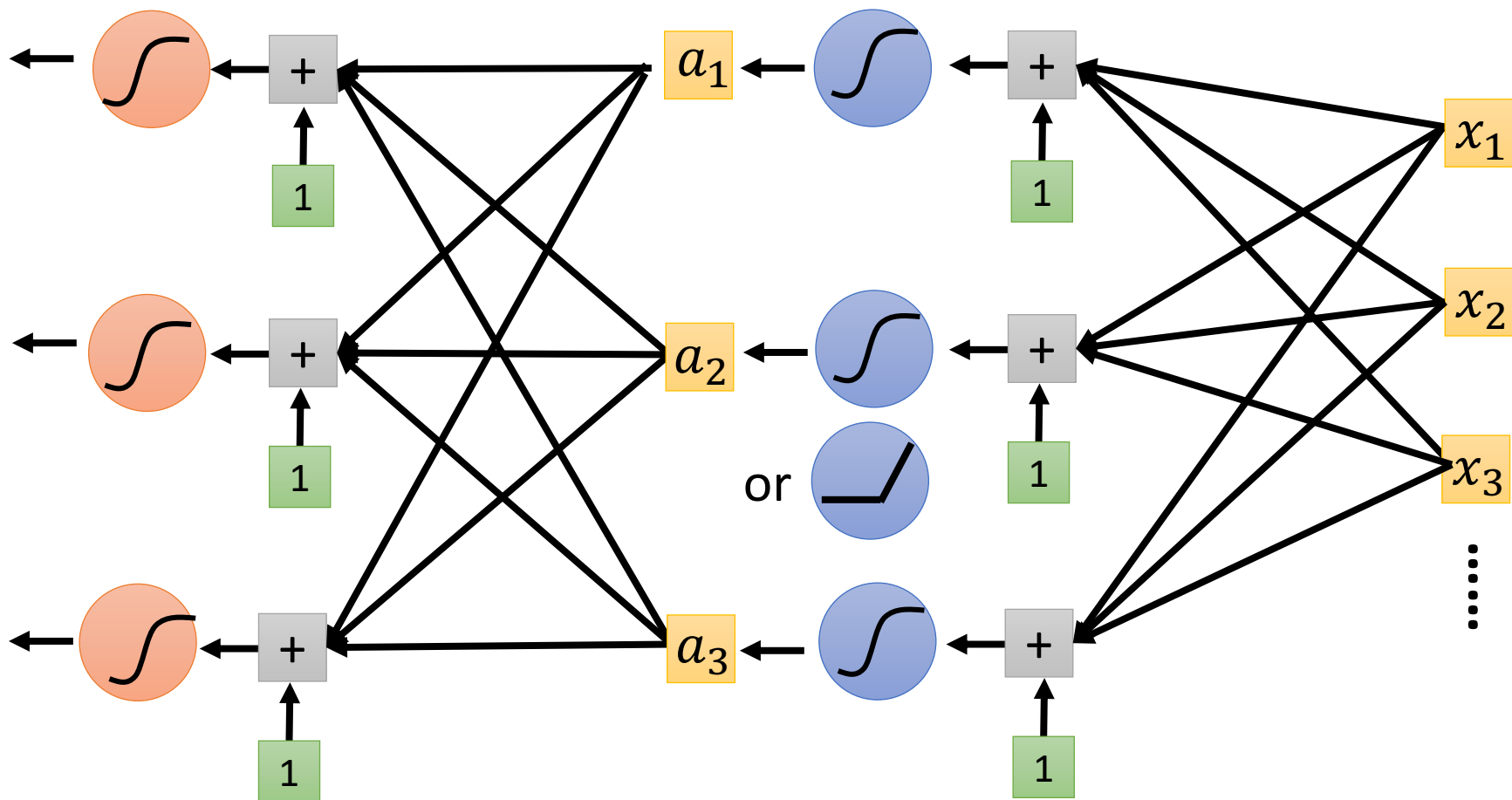
Experimental Results

- 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

Views (k)

2021/01/01

2021/02/14

?

Back to ML Framework

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

$$y = b + \boldsymbol{c}^T \sigma(\boldsymbol{b} + W\boldsymbol{x})$$
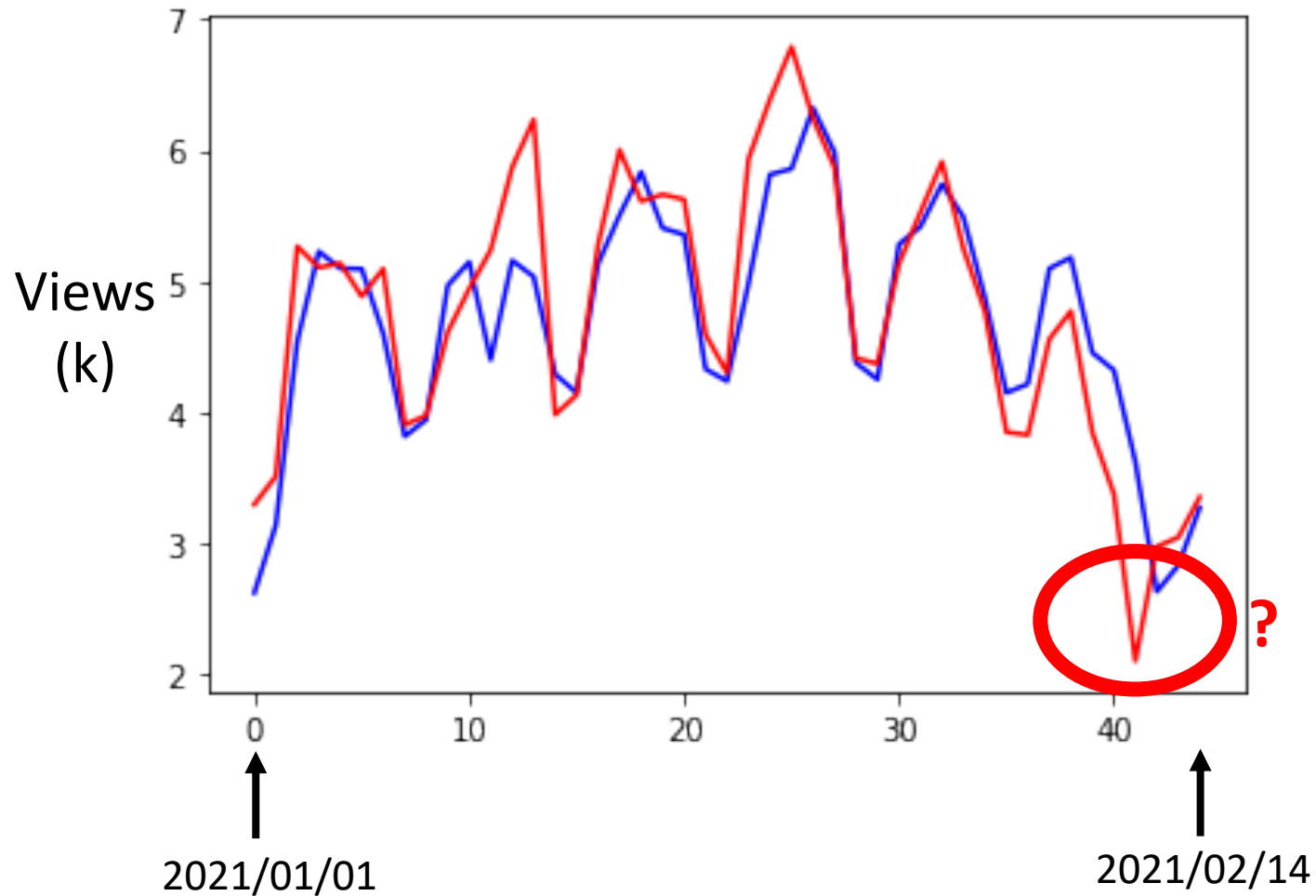
It is not *fancy* enough.

Let's give it a *fancy* name!

hidden layer

hidden layer

a_1

a_2

a_3

x_1

x_2

x_3

Neuron
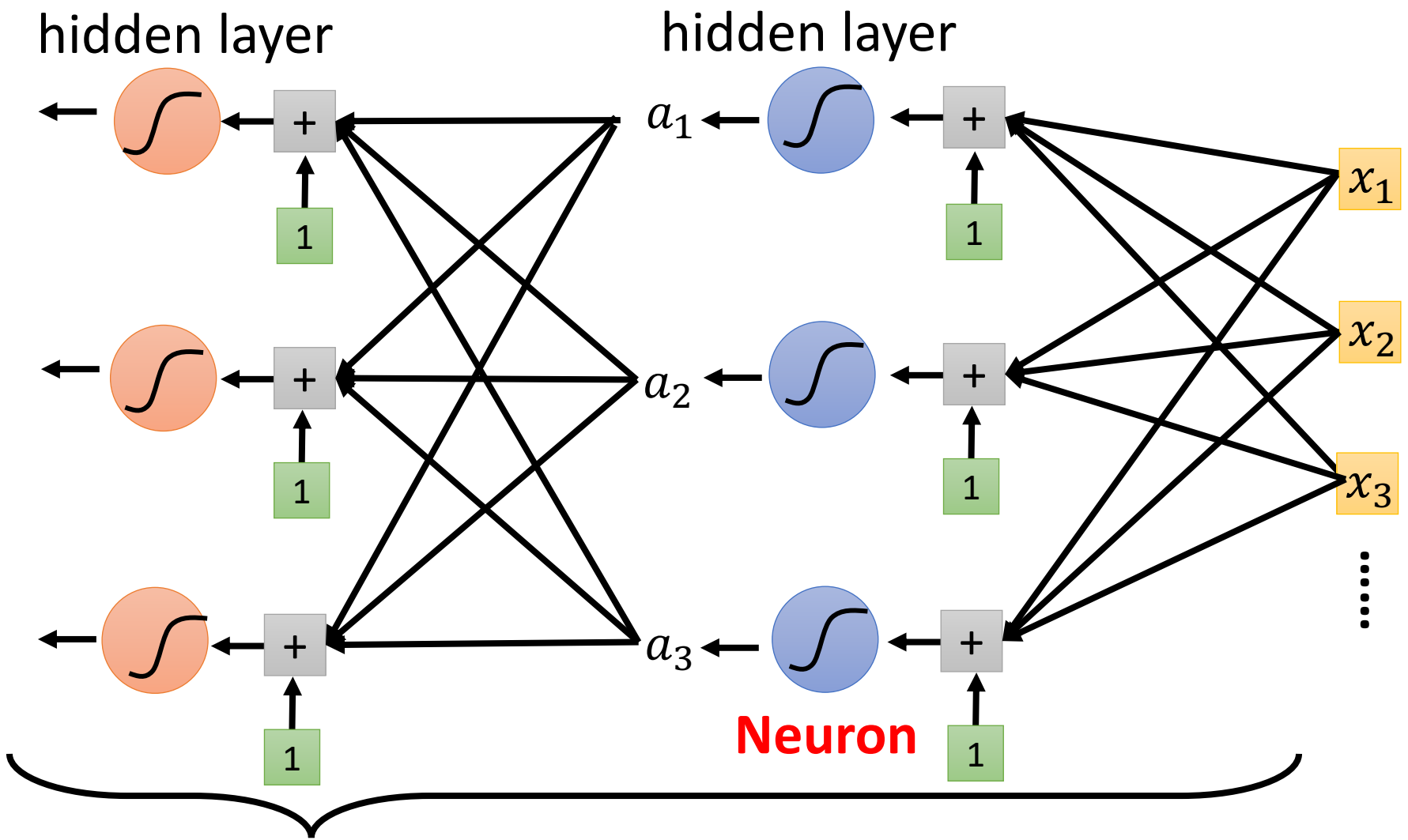
Neural Network This mimics human brains ... (???)

Many layers means **Deep** ➡ Deep Learning

Deep = Many hidden layers

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

22 layers

19 layers

8 layers

7.3%

16.4%

6.7%
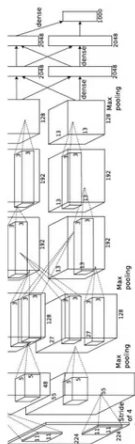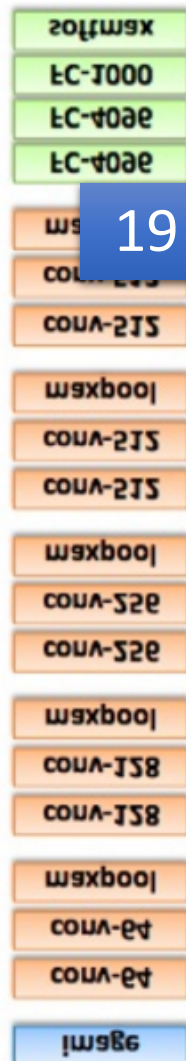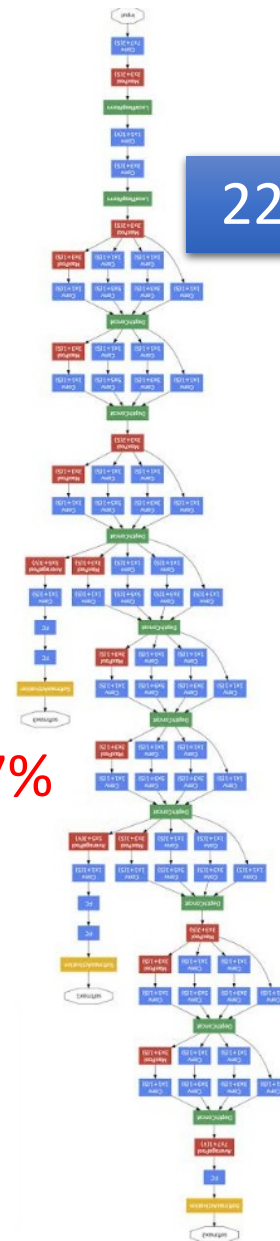
AlexNet (2012)　　　VGG (2014)　　　GoogleNet (2014)

Deep = Many hidden layers
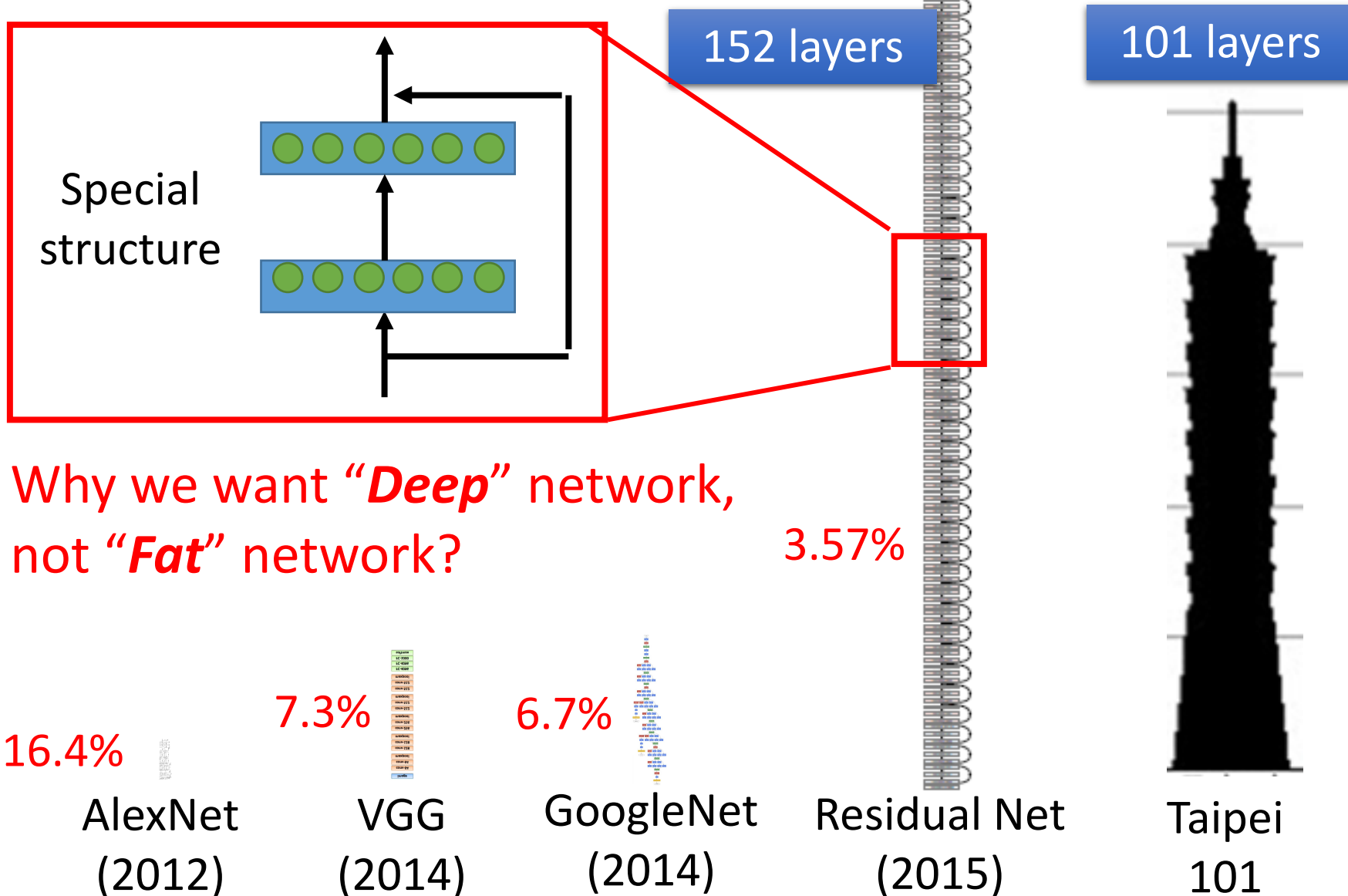
Special structure

152 layers

101 layers

Why we want "*Deep*" network, not "*Fat*" network?

3.57%

16.4%

7.3%

6.7%

AlexNet (2012)

VGG (2014)

GoogleNet (2014)

Residual Net (2015)

Taipei 101

Why don't we go deeper?

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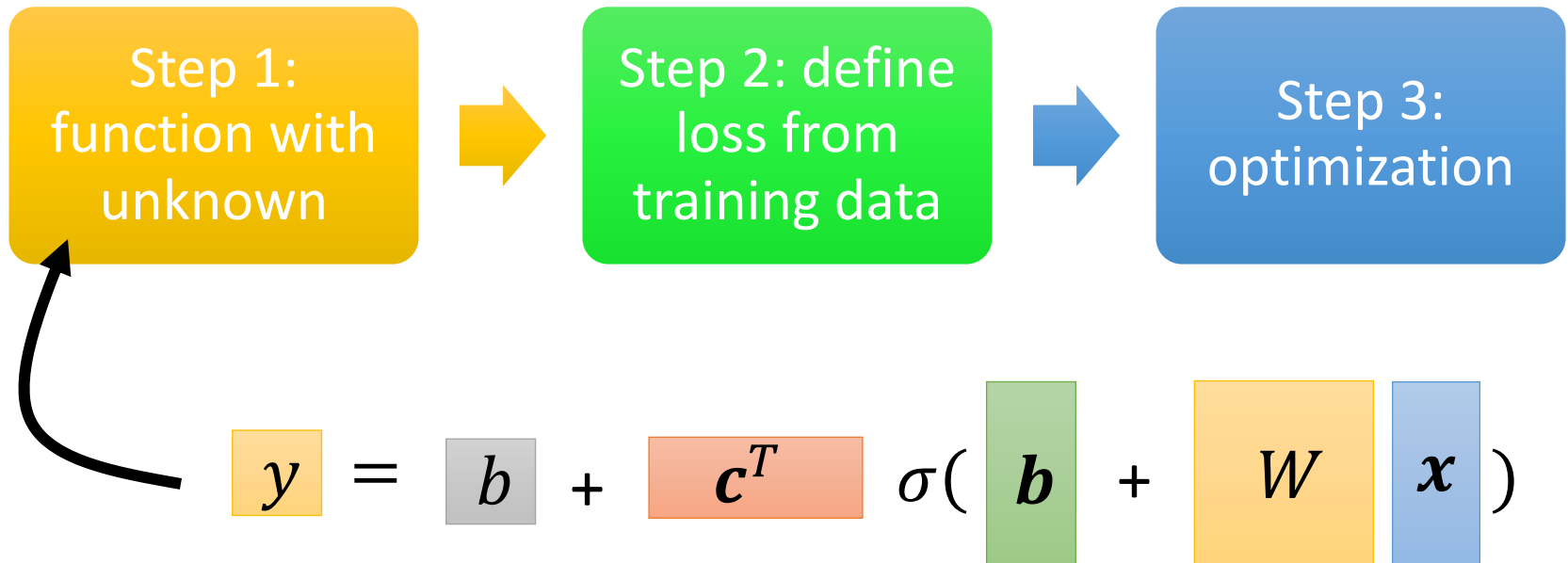   - input features are the no. of views in the past 56 days

	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

HYLEE(2021)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频·B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1fM4y137M4

课件 & 代码·博客 [扫码或点击链接]
http://blog.showmeai.tech/ntu-hylee-ml

机器学习 Auto-encoder 生成式对抗网络 学习率
深度学习 卷积神经网络 GAN 自监督 自注意力机制
批次标准化 神经网络压缩 强化学习 元学习 Transformer

Awesome AI Courses Notes Cheatsheets 是 **ShowMeAI** 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n

Awesome AI Courses Notes Cheatsheets· 持续更新中

知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]