

Introduction of Machine Learning

Hung-yi Lee

Modified by Huan Jin

YouTube Channel



李子柒 Liziqi ✓

@cnliziqi 1740万位订阅者 128 个视频

这里是李子柒YouTube官方频道哦~ 欢迎订阅: <https://goo.gl/nkjpSx> >

订阅

首页

视频

直播

播放列表

社区

频道

简介



我们中国人的开门七件事，柴米油盐酱醋茶 Firewood, rice, oil, ...

25,918,459次观看 · 1年前

Please subscribe to 【李子柒 Liziqi】 Liziqi Channel on YouTube if you like my videos: <https://goo.gl/nkjpSx>

#李子柒#liziqi#lǐzǐqi#ChineseCuisine #ChineseFood

Firewood, rice, oil, salt, soy sauce, vinegar, and tea--

The seven essentials for Chinese to start a day

柴米油盐酱醋茶，少了盐可不行。

作为百味之首，...

了解详情

<https://www.youtube.com/@cnliziqi>

The function we want to find ...

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%

$y = f(\text{no. of views on 2/26})$

1. Function with Unknown Parameters

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%

$$y = f(\text{feature})$$



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)

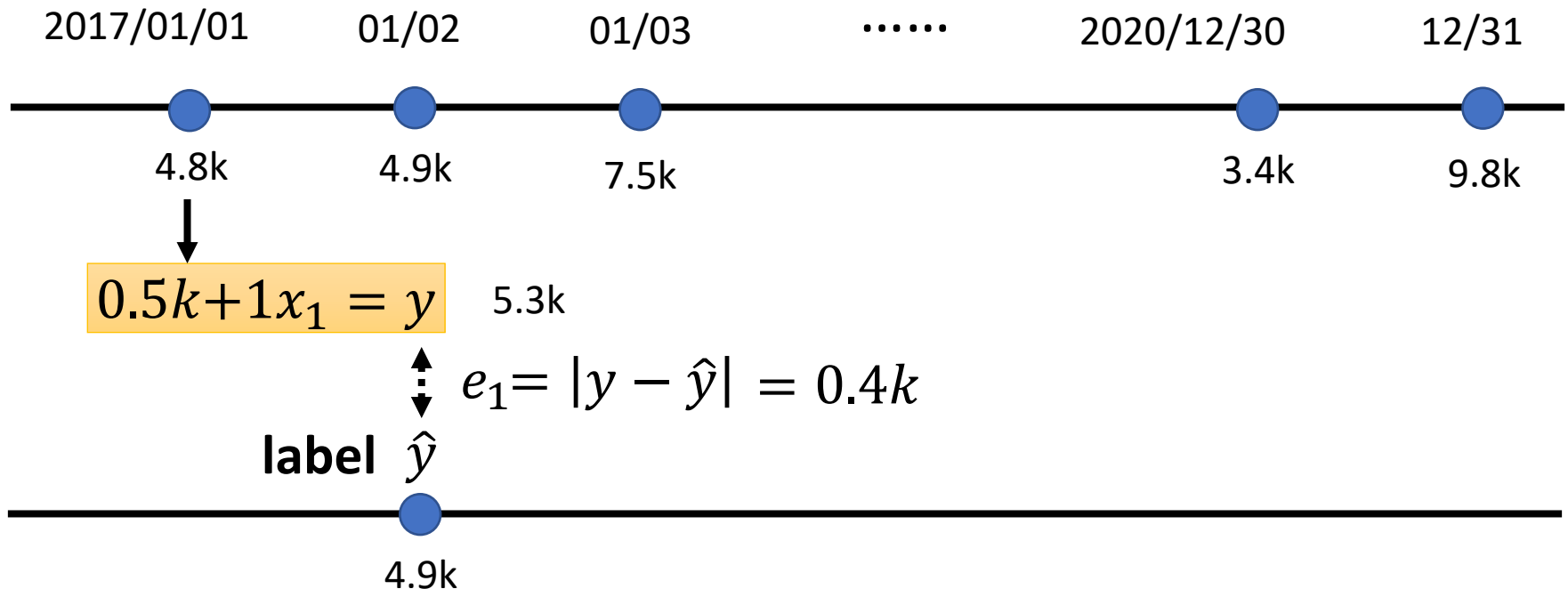
weight **bias**

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

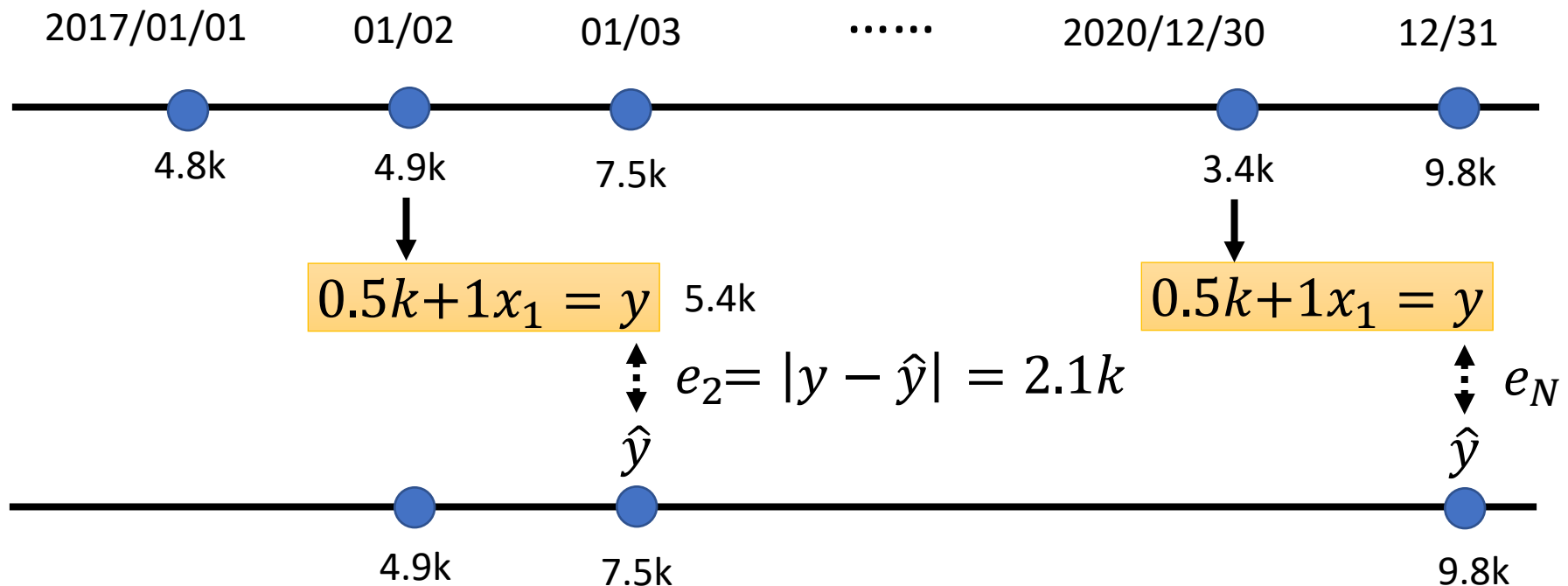


2. Define **Loss** from Training Data

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

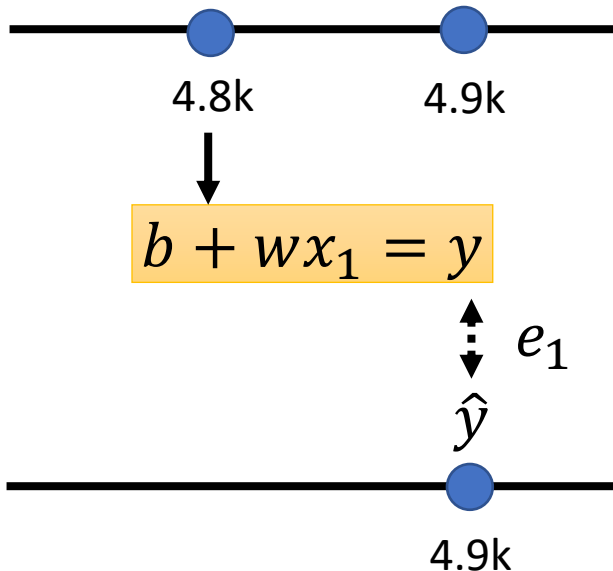
$L(0.5k, 1)$ $y = b + wx_1 \longrightarrow y = 0.5k + 1x_1$ How good it is?

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)

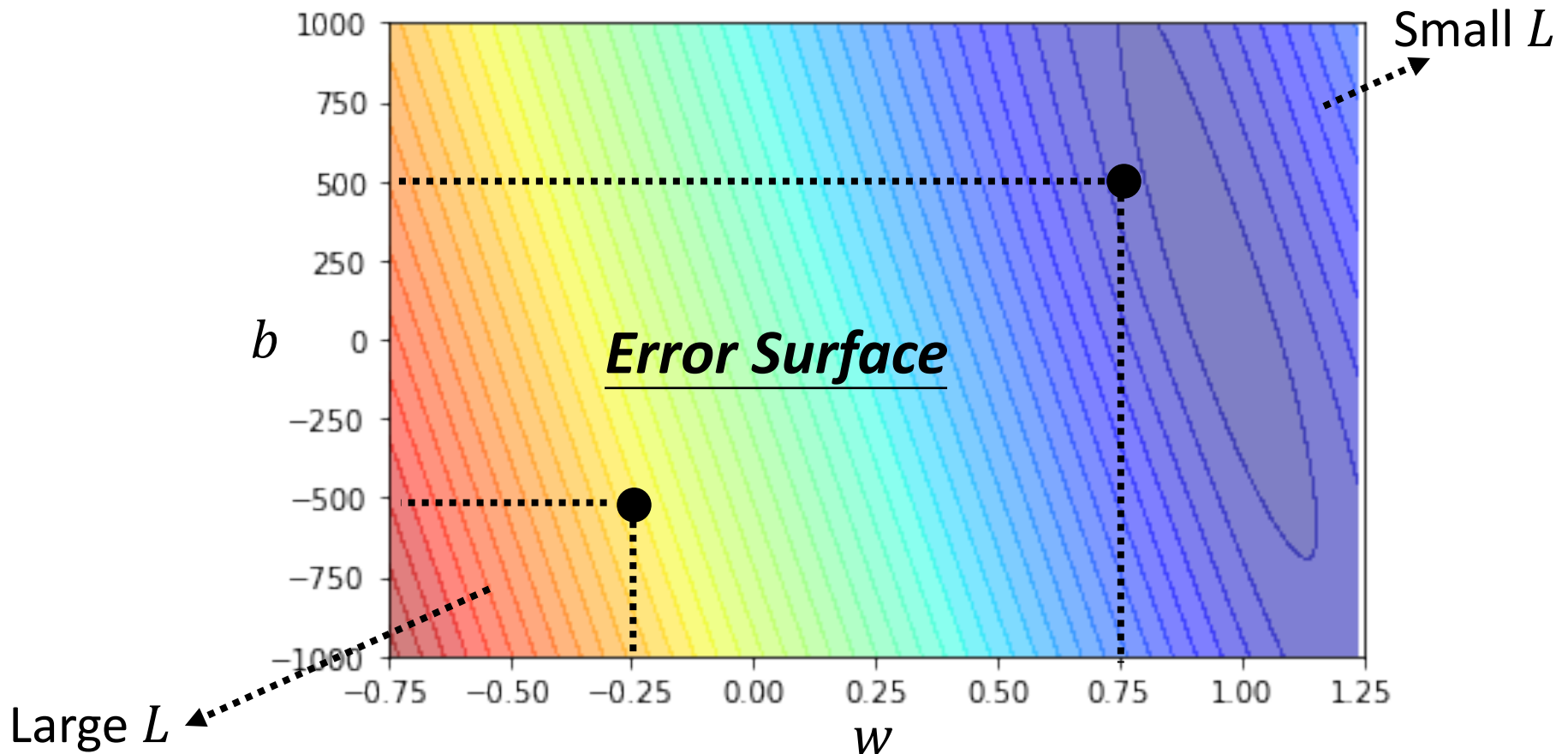
$e = (y - \hat{y})^2$ L is mean square error (MSE)

If y and \hat{y} are both probability distributions \longrightarrow 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 $y = b + wx_1$

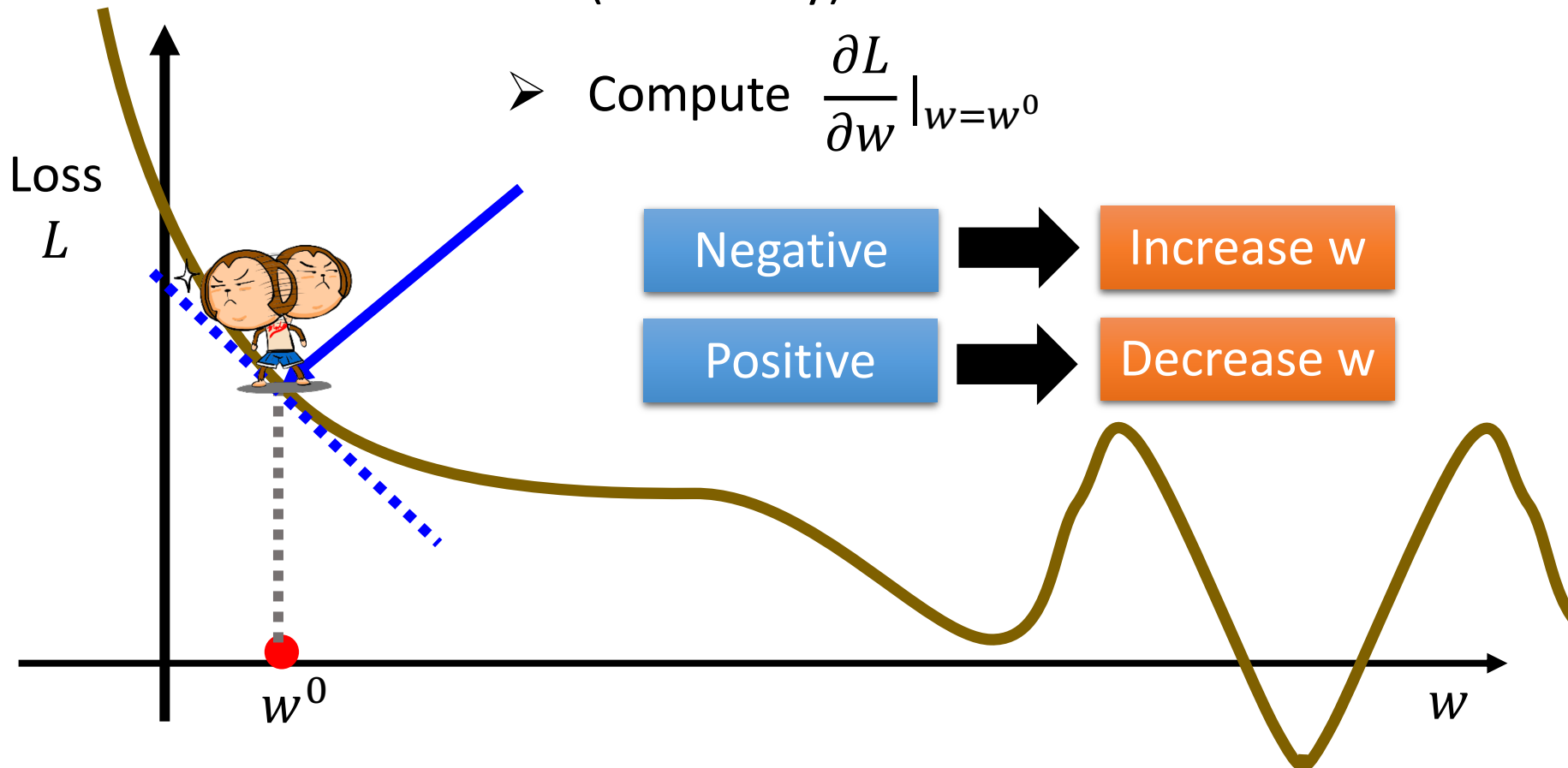


3. Optimization

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

Gradient Descent

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



3. Optimization

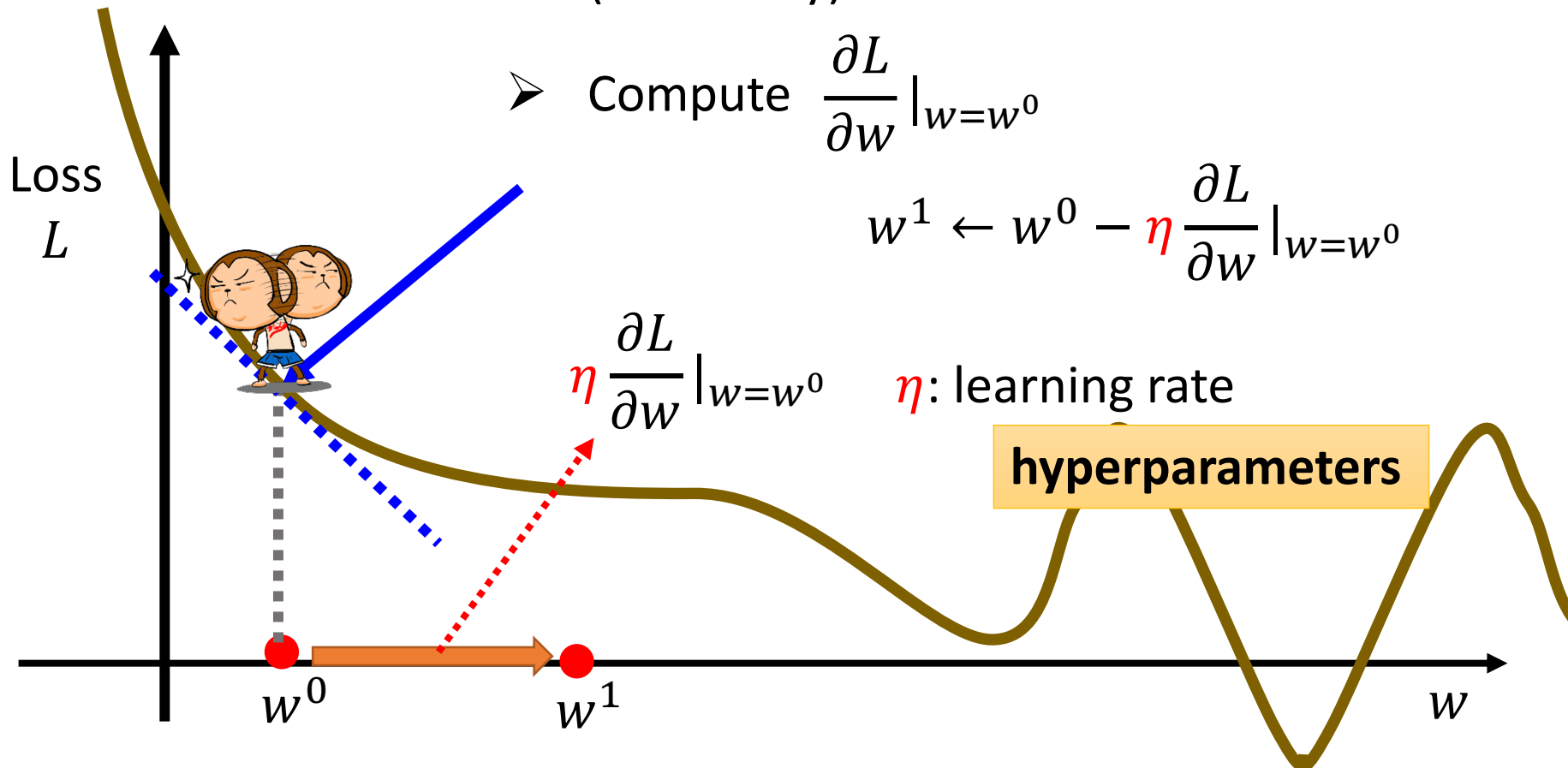
$$w^* = \arg \min_w L$$

Gradient Descent

➤ (Randomly) Pick an initial value w^0

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

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



3. Optimization

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

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

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

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

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

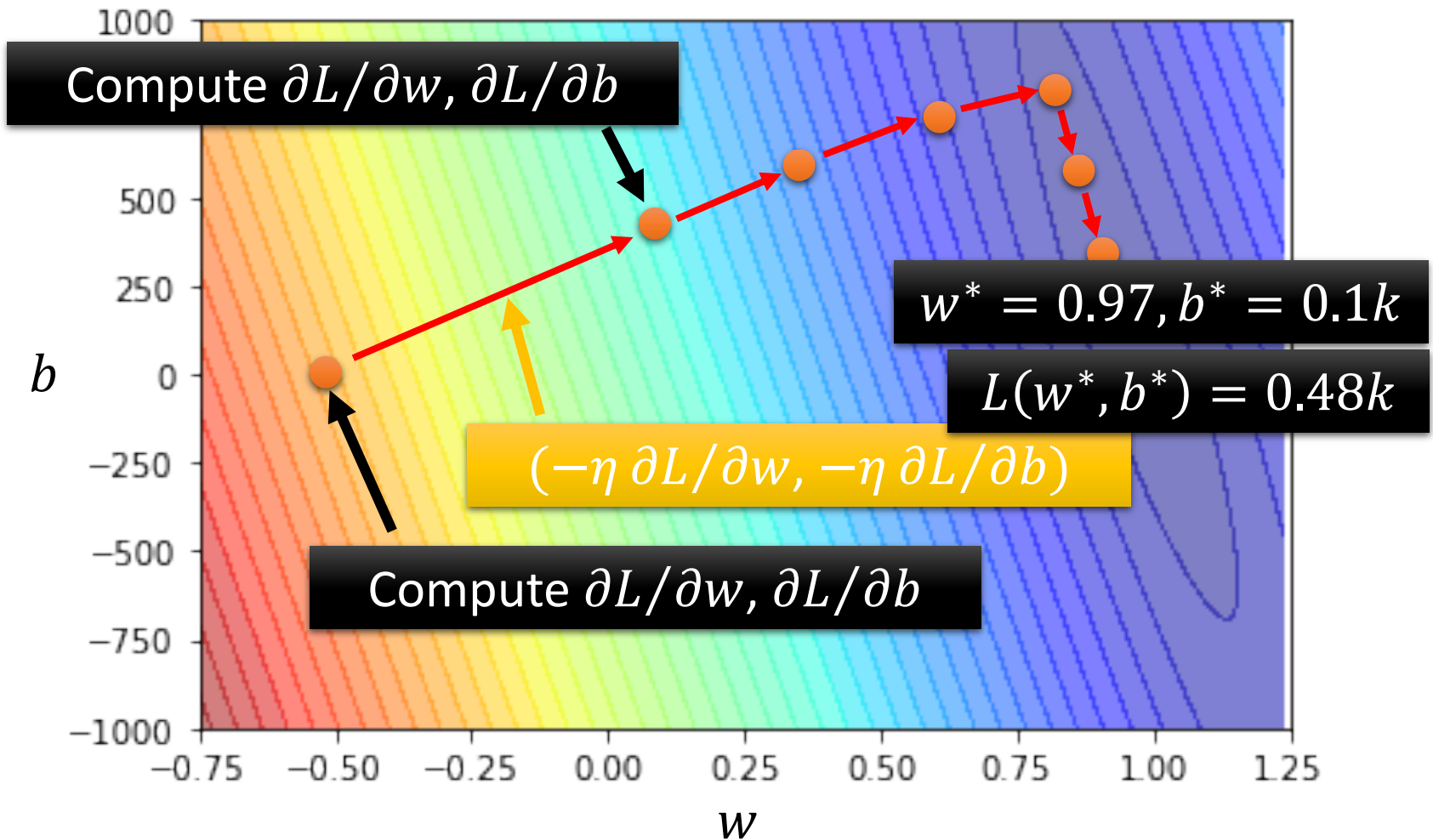
Can be done in one line in most deep learning frameworks

- Update w and b iteratively

Model $y = b + wx_1$

3. Optimization

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



Machine Learning is so simple

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

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

$$y = b + wx_1$$

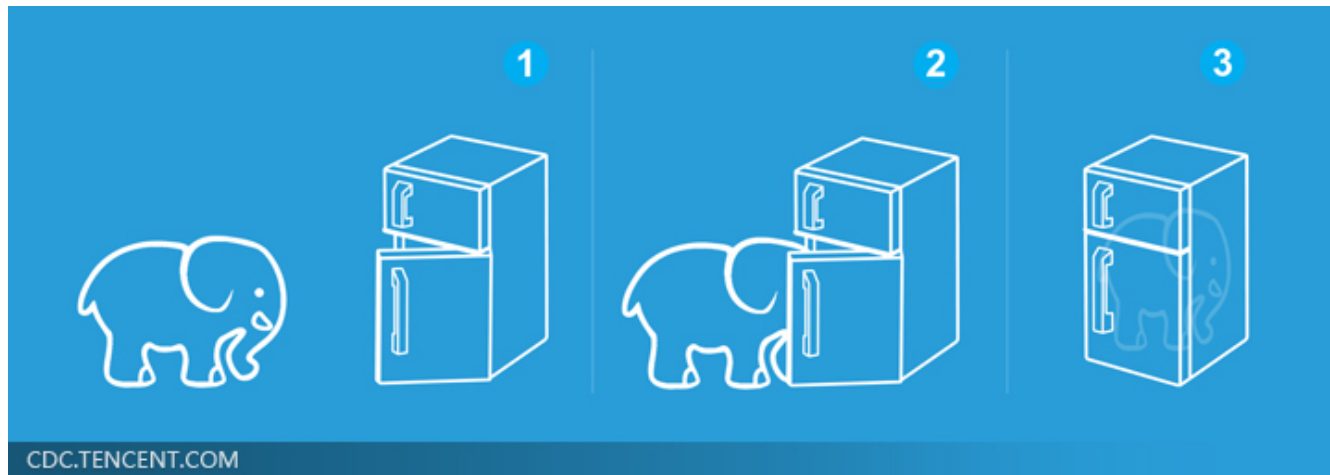
Step 1:
function with
unknown



Step 2: define
loss from
training data



Step 3:
optimization



Machine Learning is so simple

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

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



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

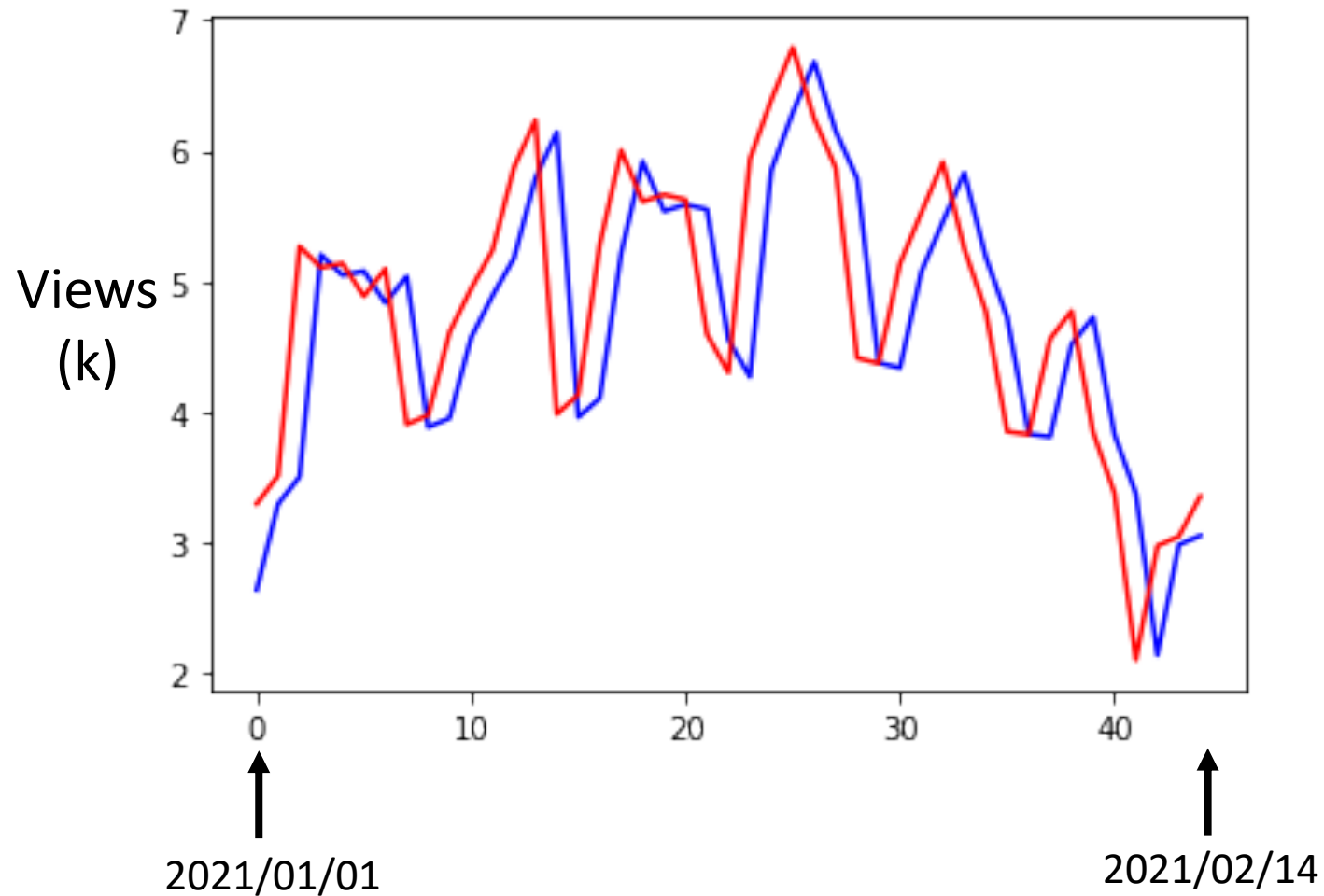
How about data of 2021 (**unseen during training**)?

$$L' = 0.58k$$

$$y = 0.1k + 0.97x_1$$

Red: real no. of views

blue: estimated no. of views



$$y = b + wx_1$$

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

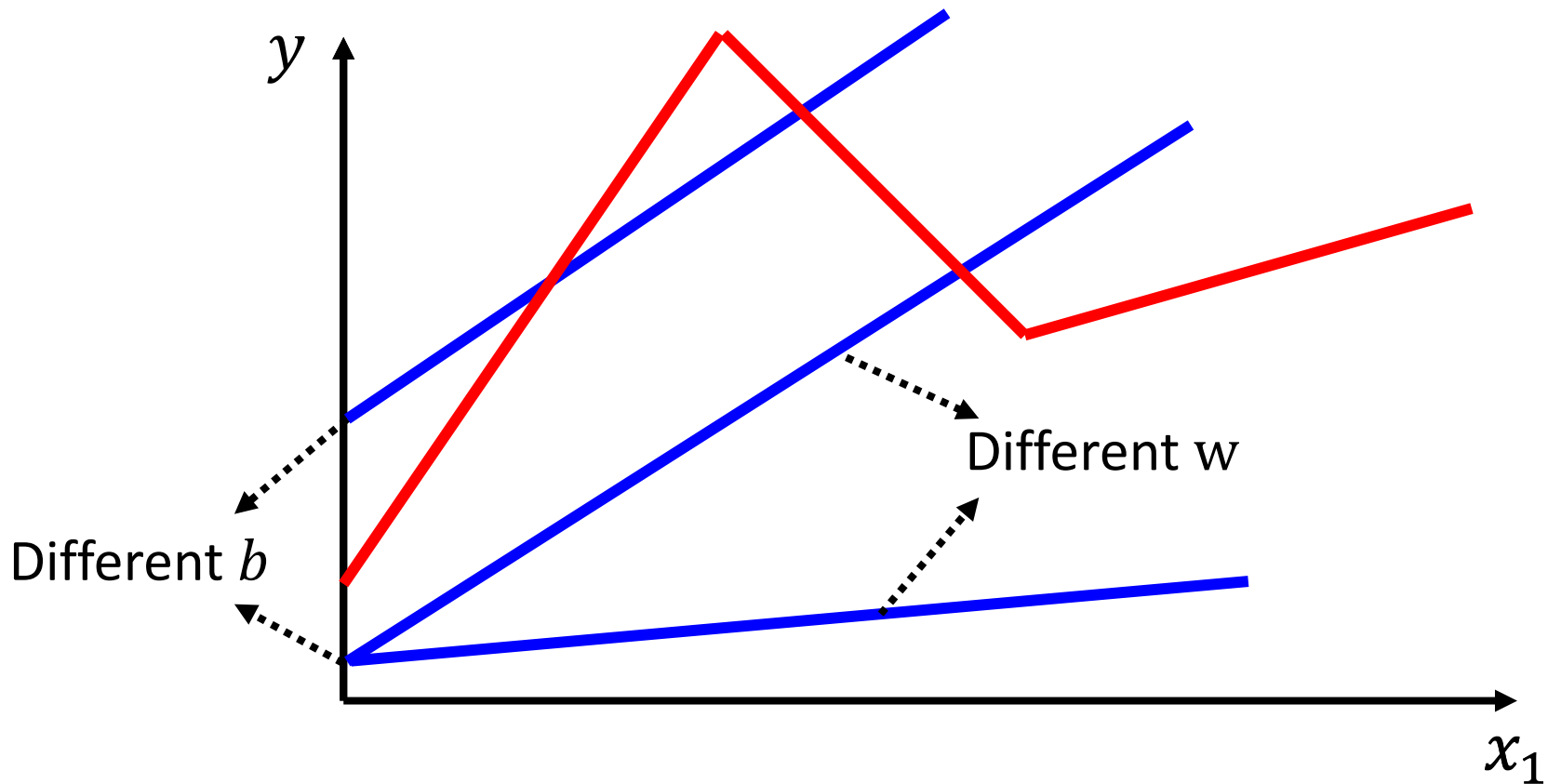
$$L = 0.32k$$

2021

$$L' = 0.46k$$


Linear models

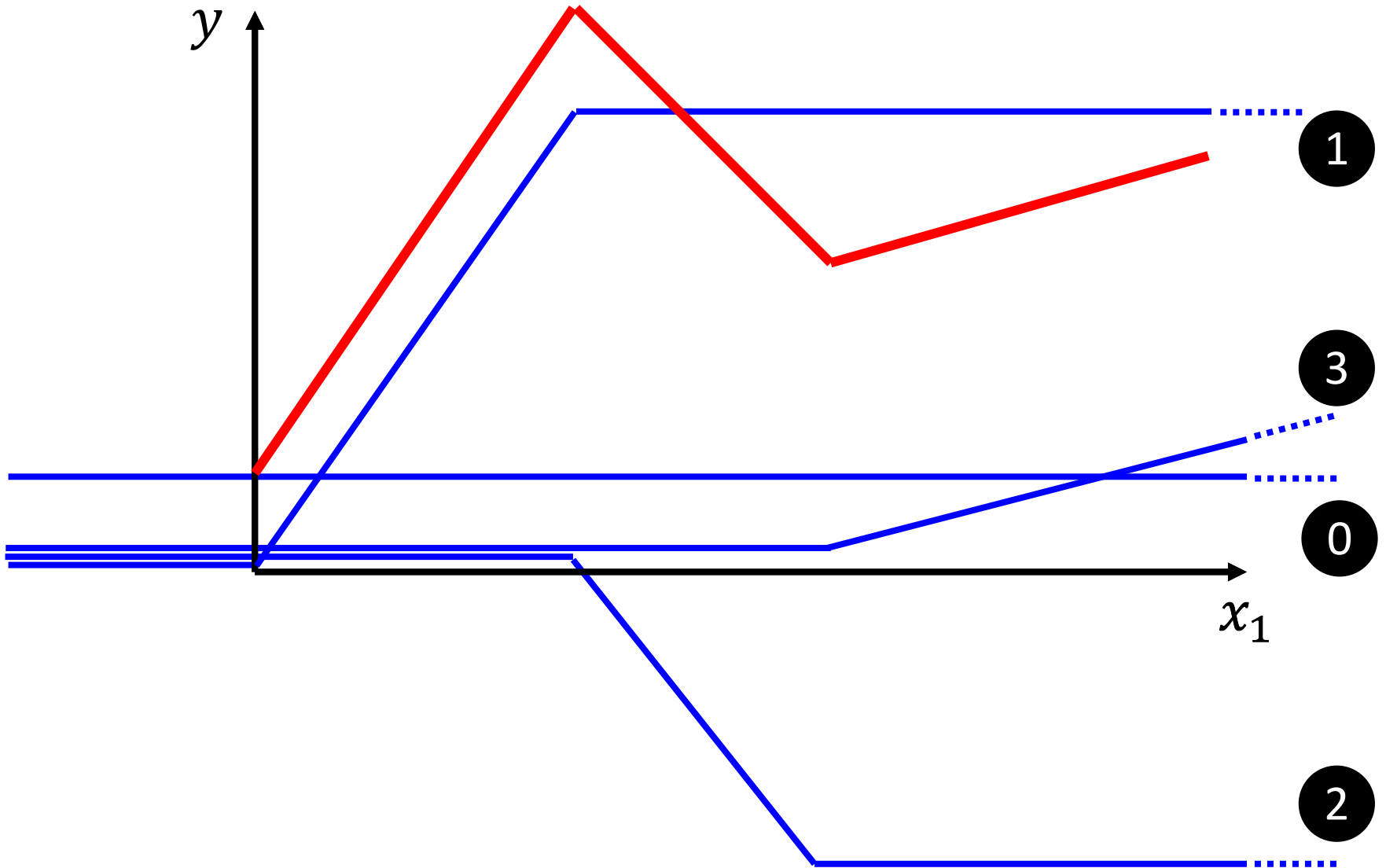
- Linear models are too simple ...
- we need more sophisticated models



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

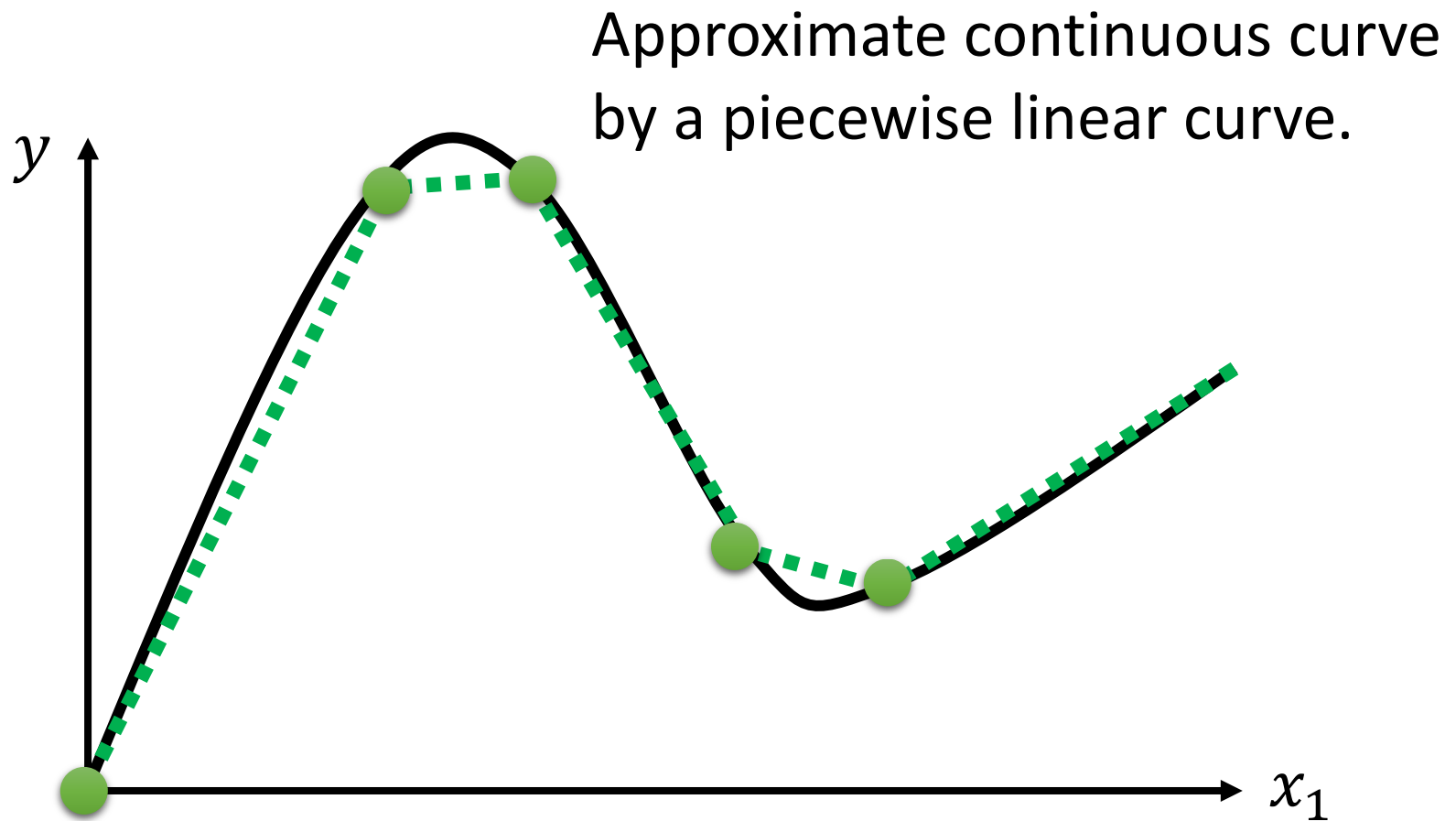
= constant + sum of a set of



More pieces require more

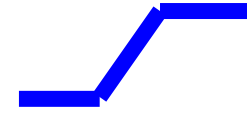


Beyond Piecewise Linear?



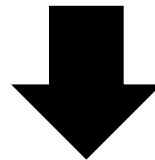
To have good approximation, we need sufficient pieces.

red curve = constant + sum of a set of



How to represent
this function?

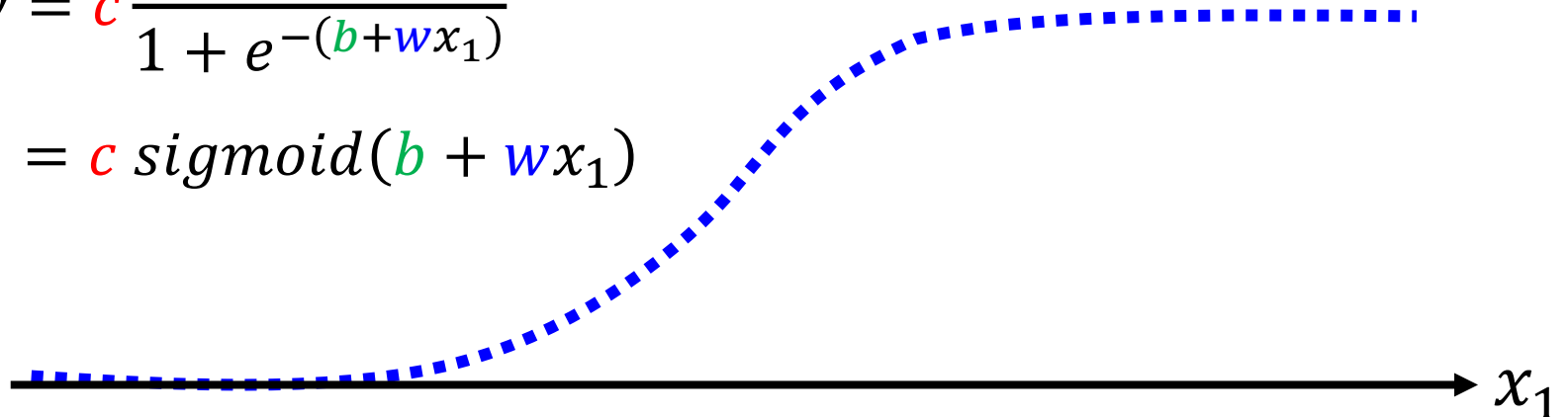
Hard Sigmoid

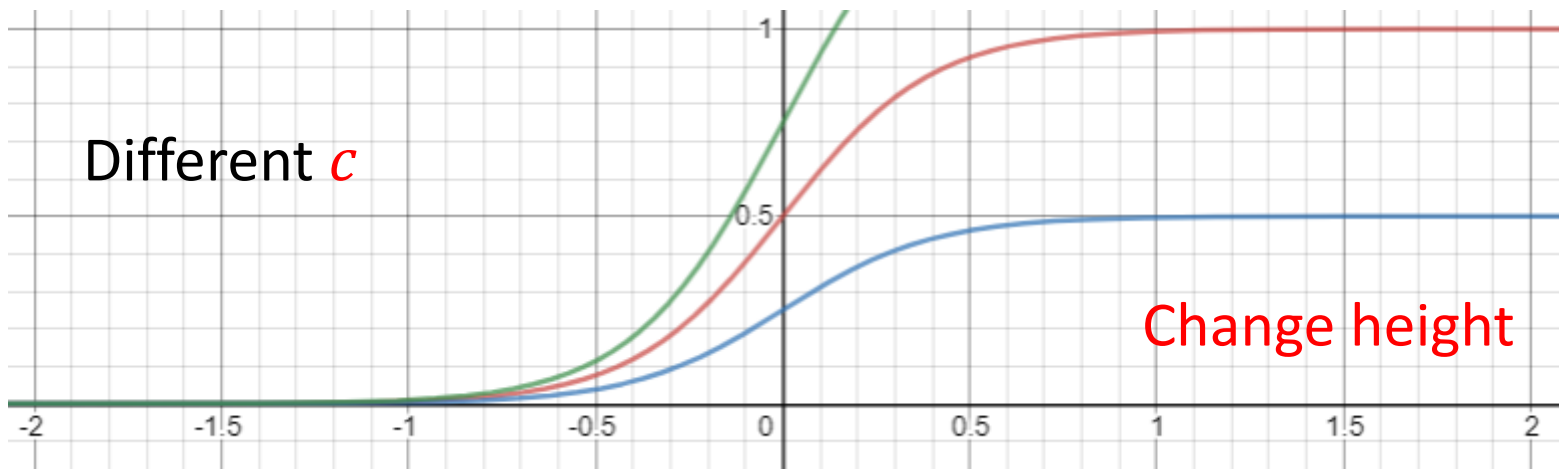
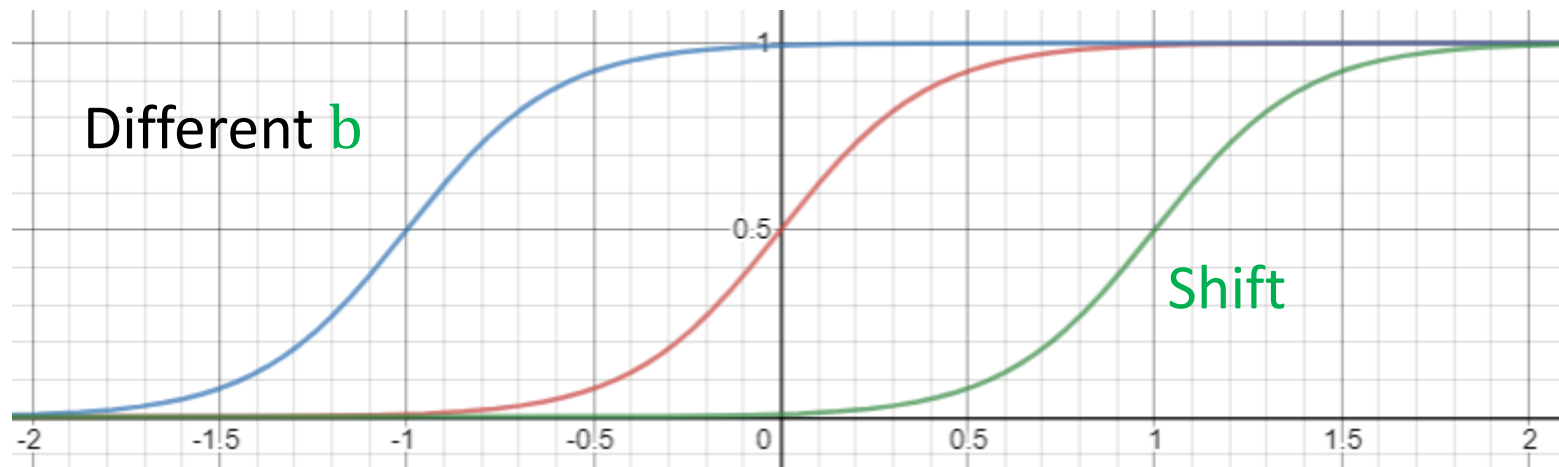
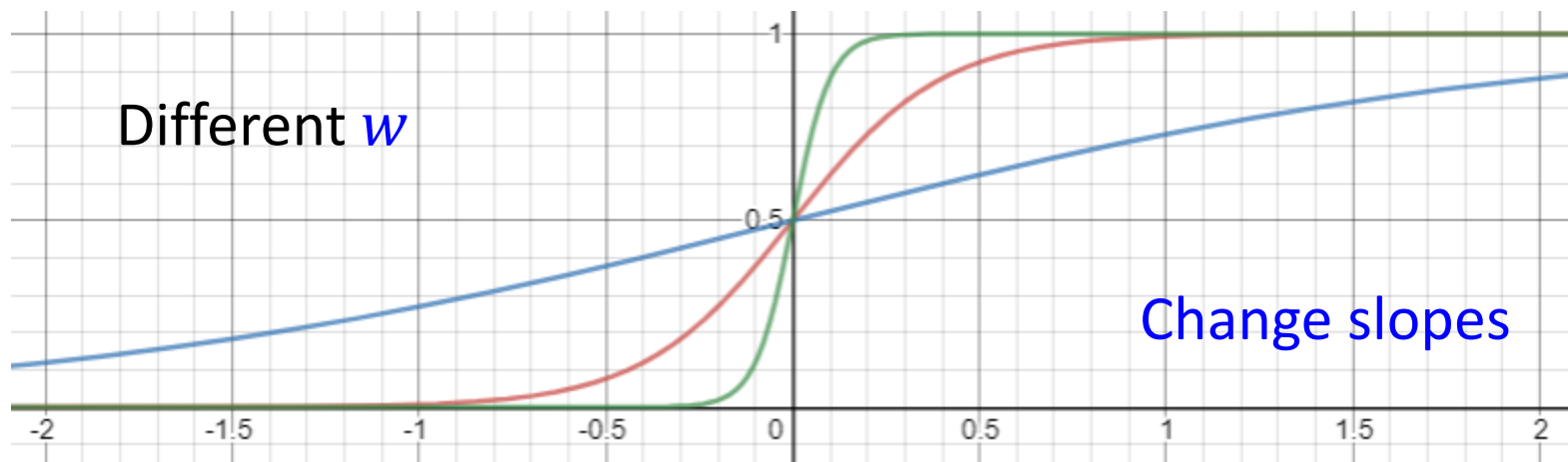


Sigmoid Function

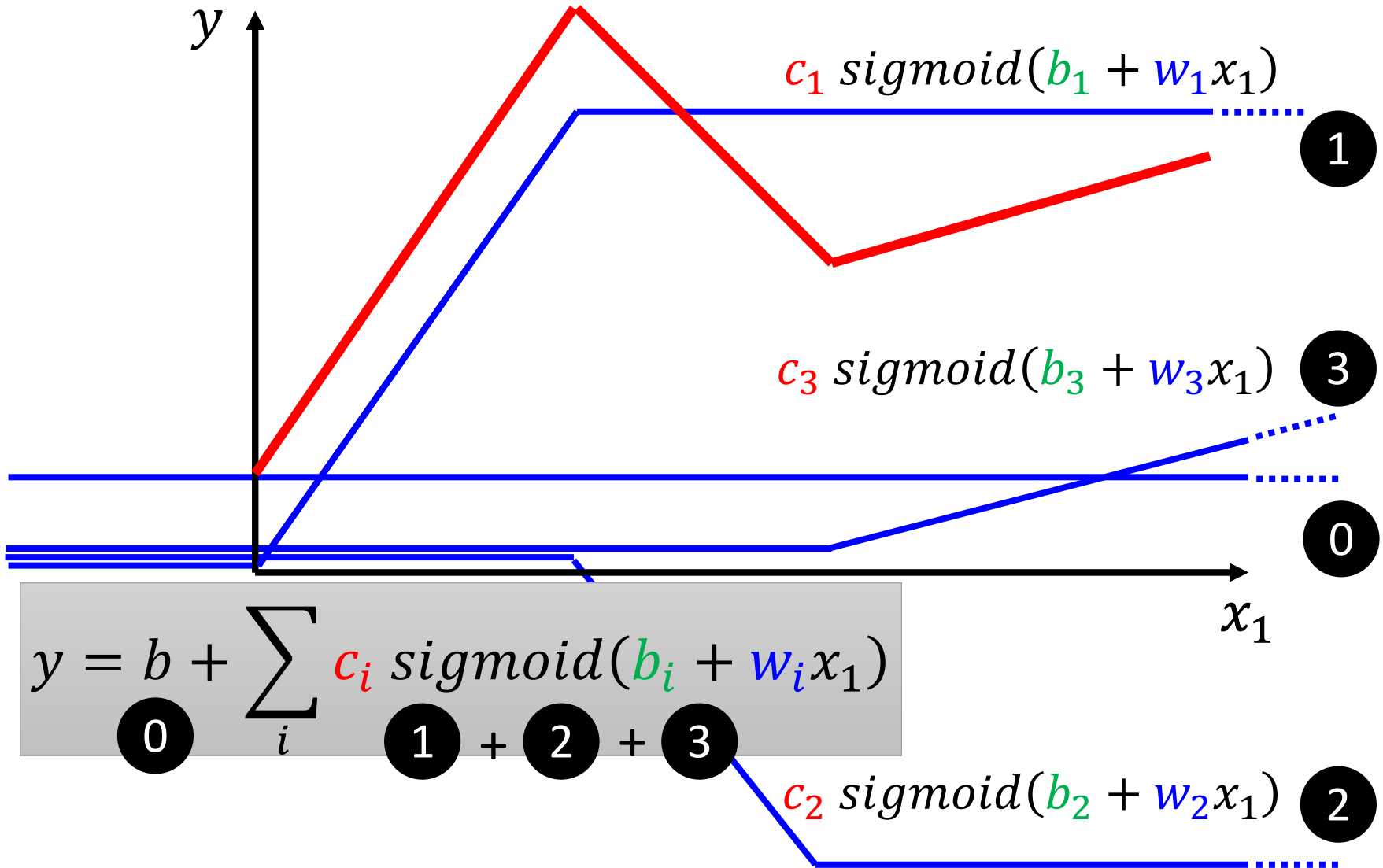
$$y = c \frac{1}{1 + e^{-(b + wx_1)}}$$

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






red curve = sum of a set of  + constant




New Model: More Features

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

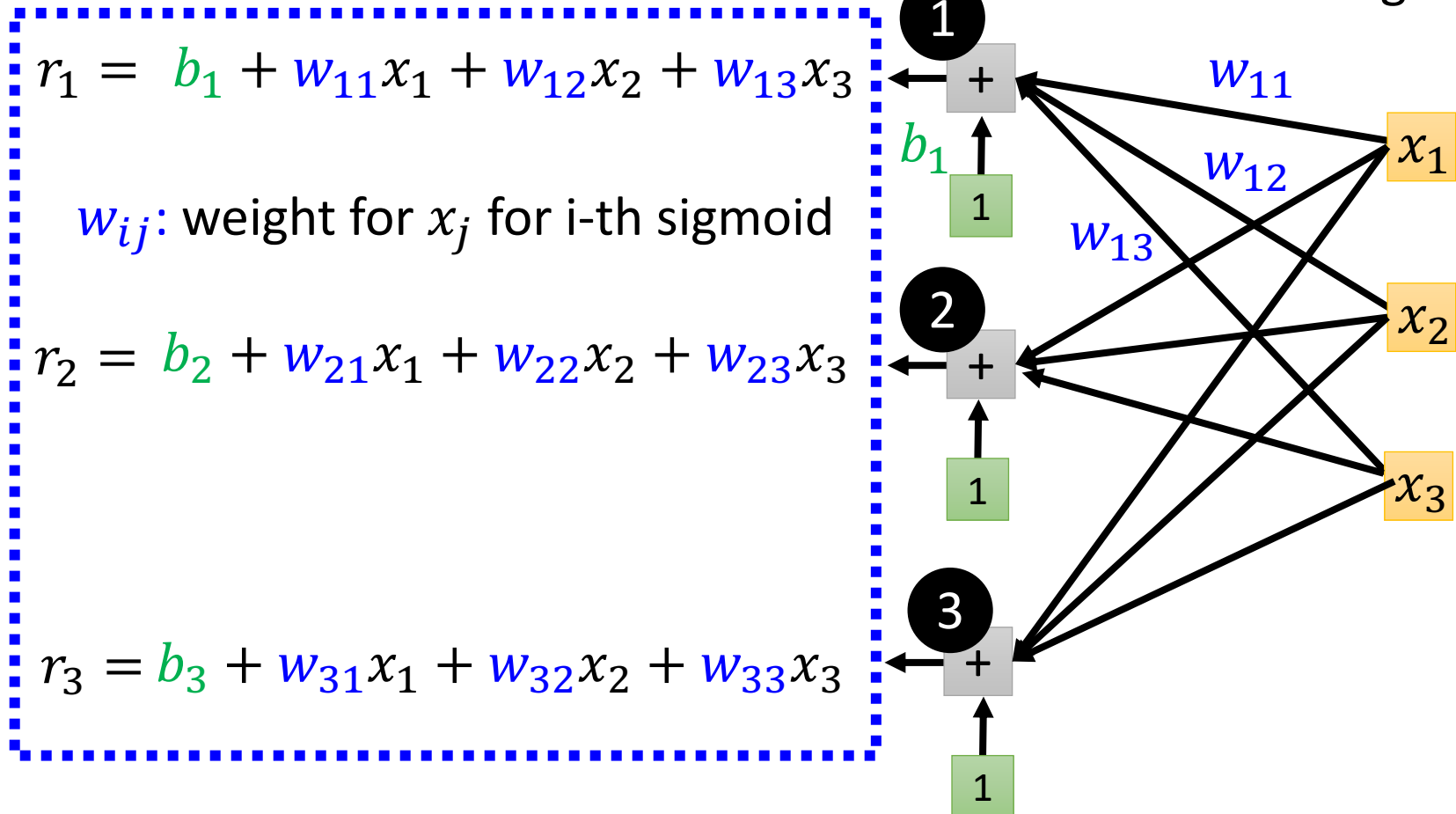

$$y = b + \sum_i \textcolor{red}{c_i} \textit{sigmoid}(\underline{\textcolor{green}{b_i} + \textcolor{blue}{w_i}x_1})$$

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


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

$$y = b + \sum_i \textcolor{red}{c}_i \textit{sigmoid} \left(\textcolor{green}{b}_i + \sum_j \textcolor{blue}{w}_{ij} x_j \right)$$

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



$$y = b + \sum_i \textcolor{red}{c}_i \textit{sigmoid} \left(\textcolor{green}{b}_i + \sum_j \textcolor{blue}{w}_{ij} x_j \right) \quad \begin{array}{l} i: 1,2,3 \\ j: 1,2,3 \end{array}$$

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

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

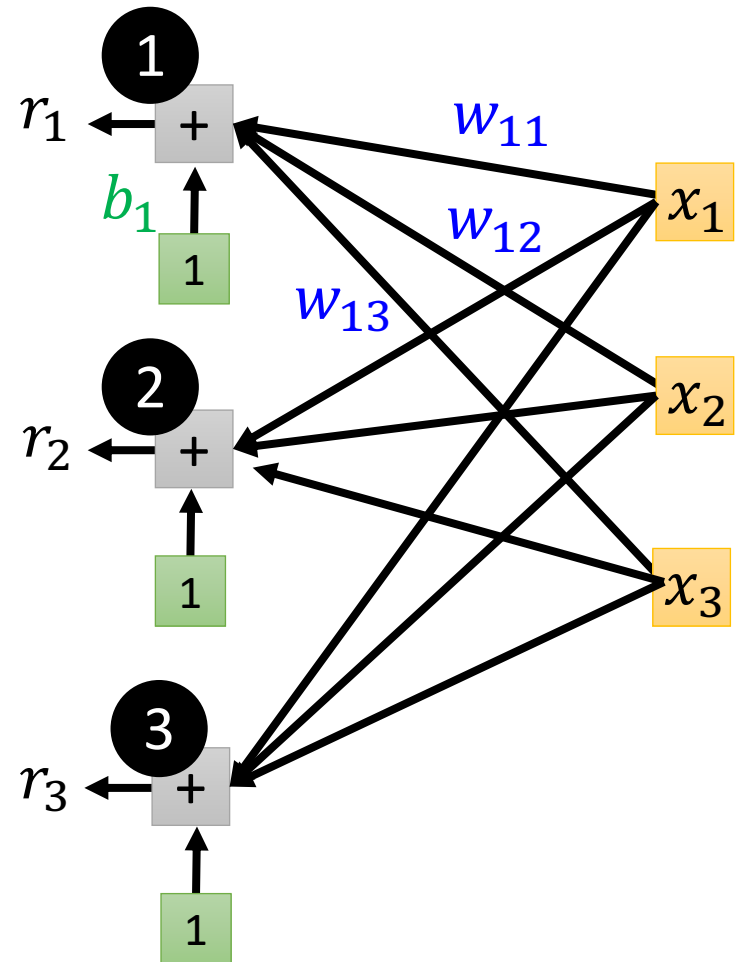
$$r_3 = \textcolor{green}{b}_3 + \textcolor{blue}{w}_{31}x_1 + \textcolor{blue}{w}_{32}x_2 + \textcolor{blue}{w}_{33}x_3$$

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

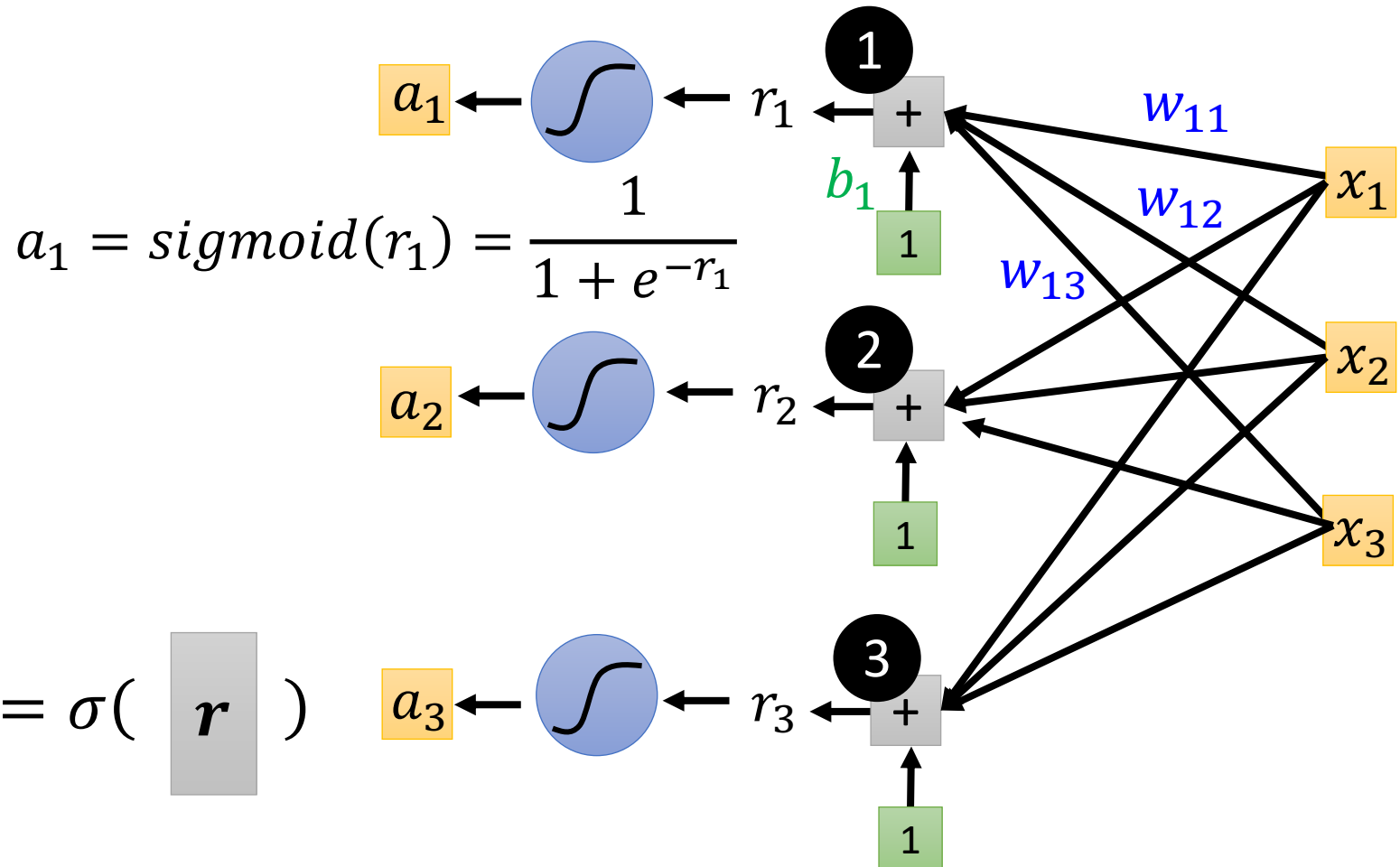
$$\boxed{\textcolor{gray}{r}} = \boxed{\textcolor{green}{b}} + \boxed{\textcolor{blue}{W}} \boxed{\textcolor{orange}{x}}$$

$$y = b + \sum_i \textcolor{red}{c}_i \textit{sigmoid} \left(\textcolor{green}{b}_i + \sum_j \textcolor{blue}{w}_{ij} x_j \right) \quad \begin{array}{l} i: 1,2,3 \\ j: 1,2,3 \end{array}$$

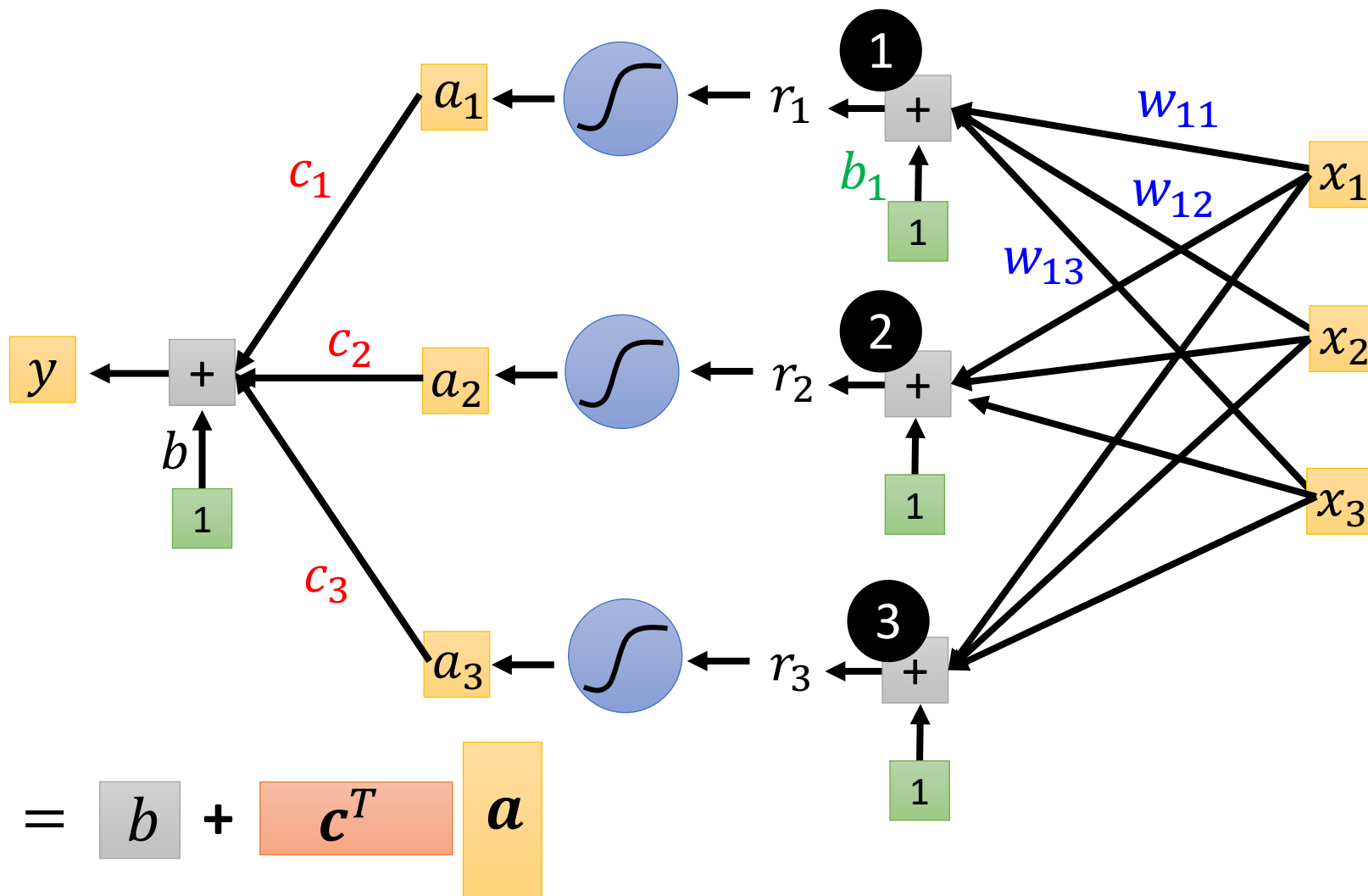
$$\mathbf{r} = \mathbf{b} + \mathbf{W} \mathbf{x}$$

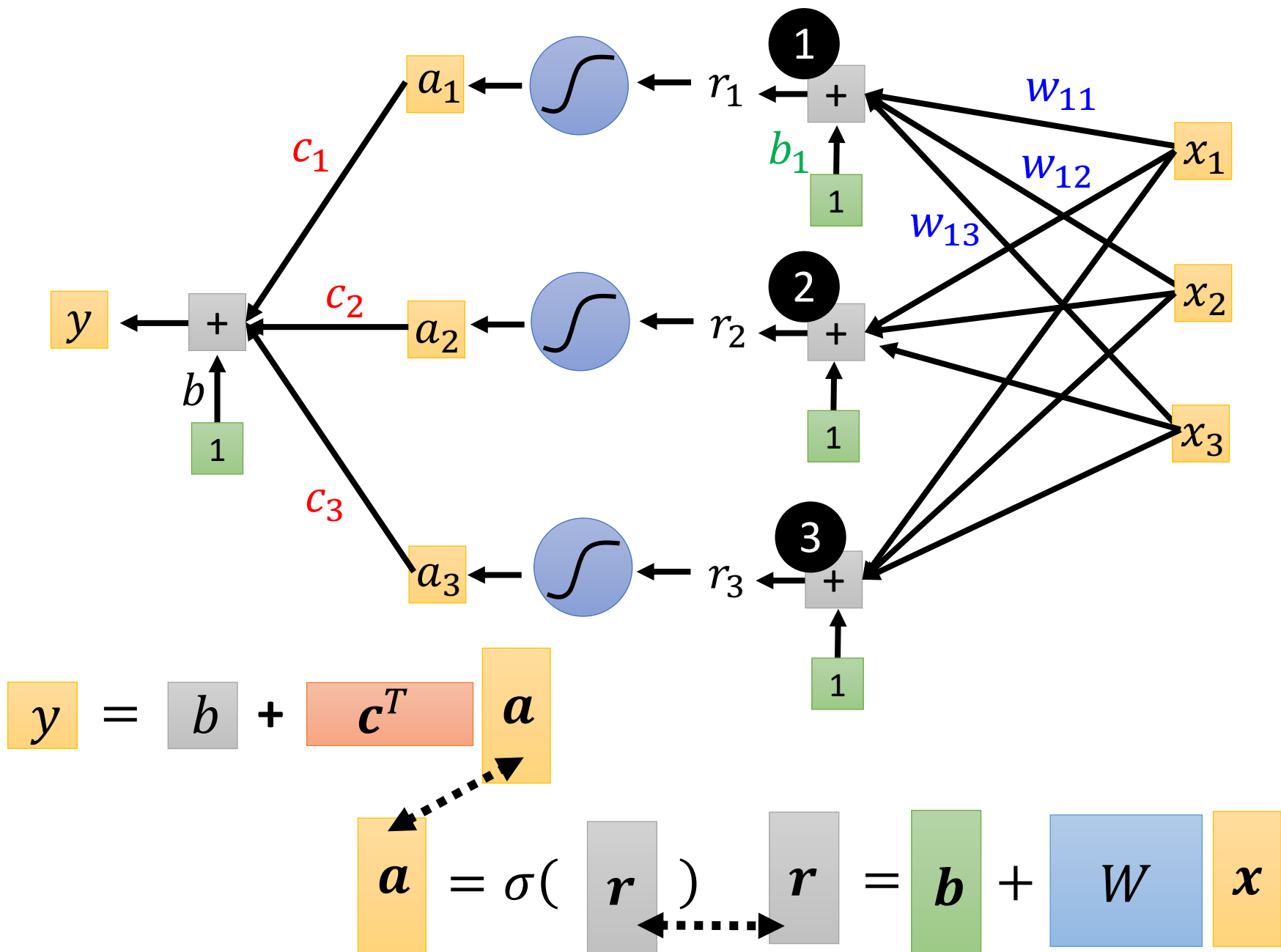


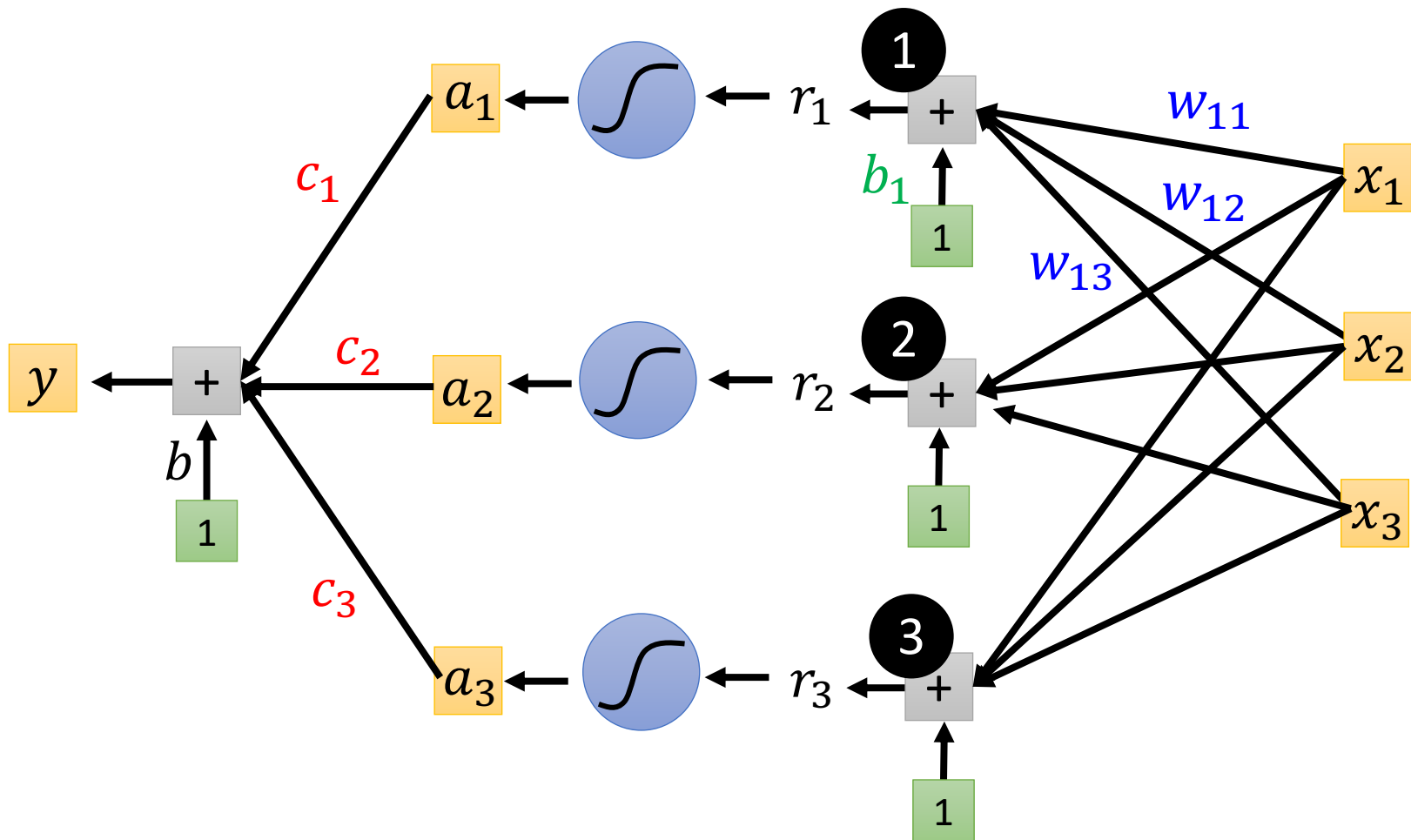
$$y = b + \sum_i \textcolor{red}{c}_i \textcolor{blue}{sigmoid} \left(\textcolor{green}{b}_i + \sum_j \textcolor{blue}{w}_{ij} x_j \right) \quad \begin{array}{l} i: 1,2,3 \\ j: 1,2,3 \end{array}$$



$$y = b + \sum_i \mathbf{c}_i \operatorname{sigmoid} \left(\mathbf{b}_i + \sum_j \mathbf{w}_{ij} x_j \right) \quad \begin{array}{l} i: 1,2,3 \\ j: 1,2,3 \end{array}$$







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

Function with unknown parameters

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

x feature

Unknown parameters

W

b

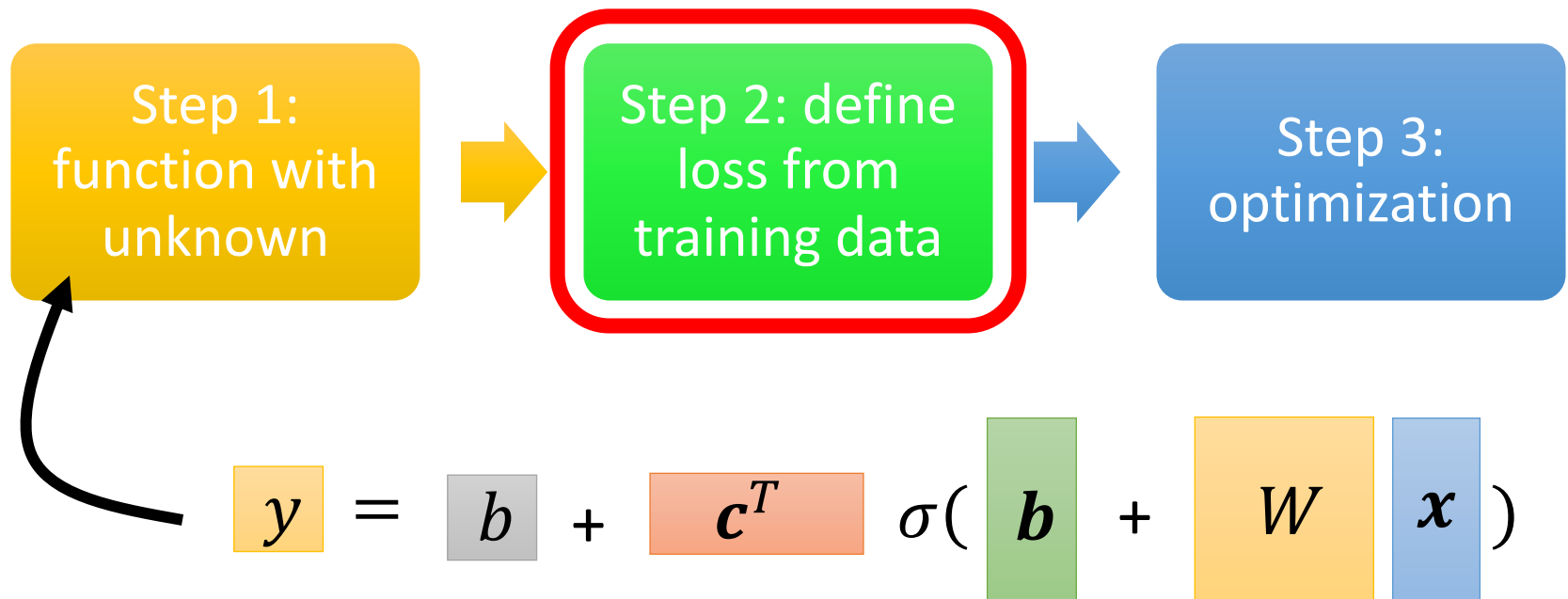
c^T

b

Rows of W

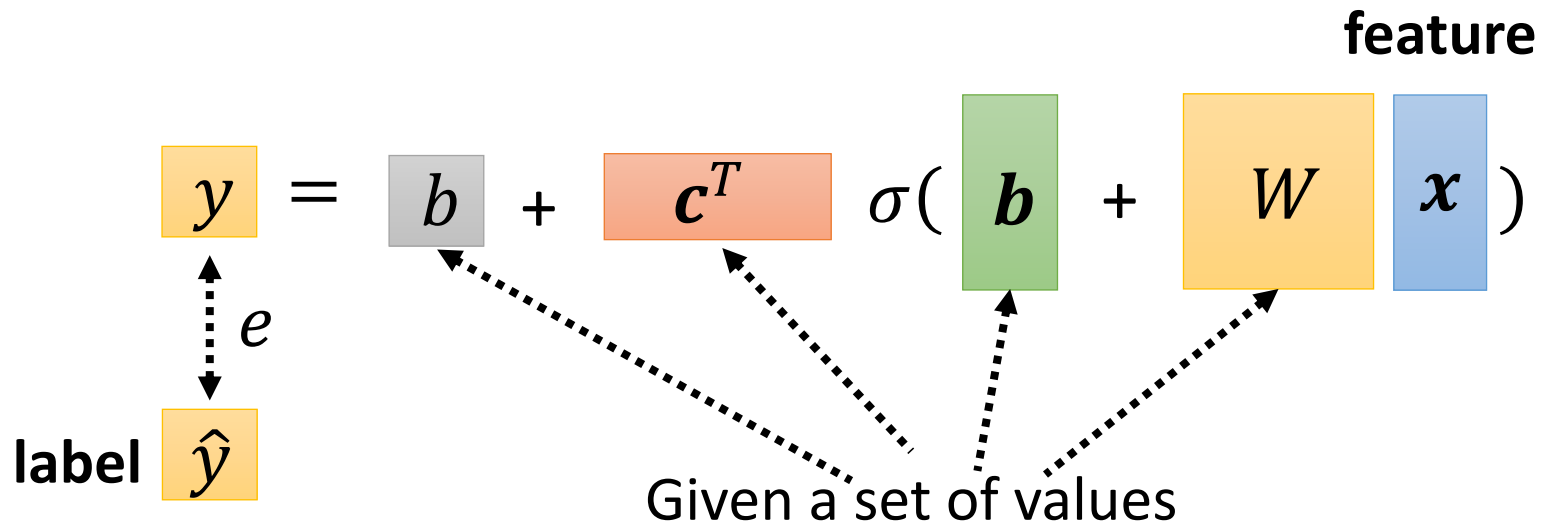
$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{bmatrix}$$

Back to ML Framework



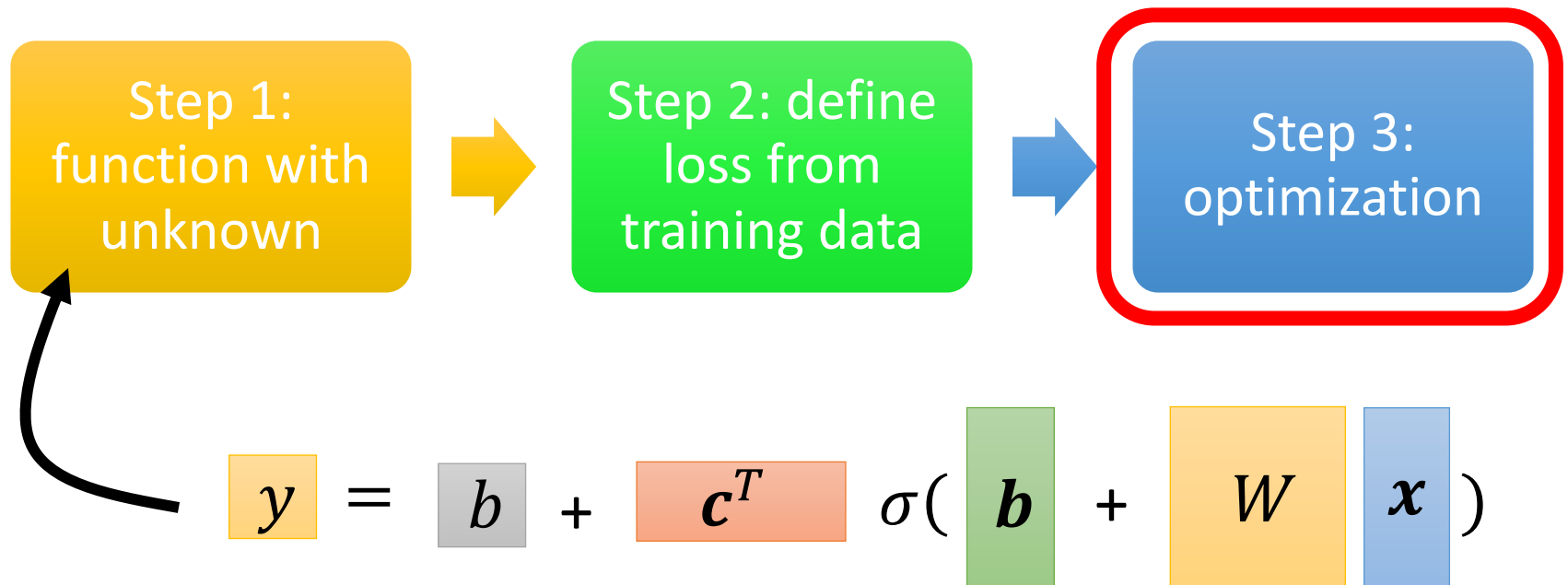
LOSS

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



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

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 $\boldsymbol{\theta}^0$

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

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

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

Optimization of New Model

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

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

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

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

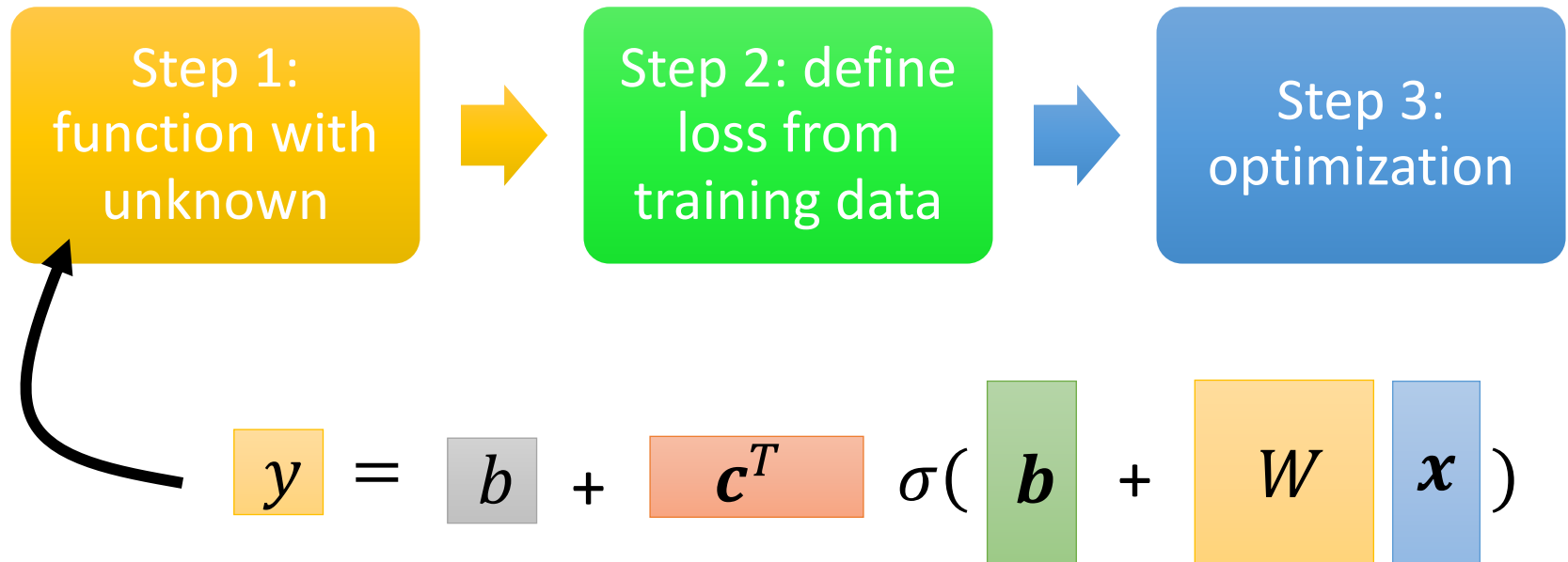
➤ Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^1)$

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

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

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

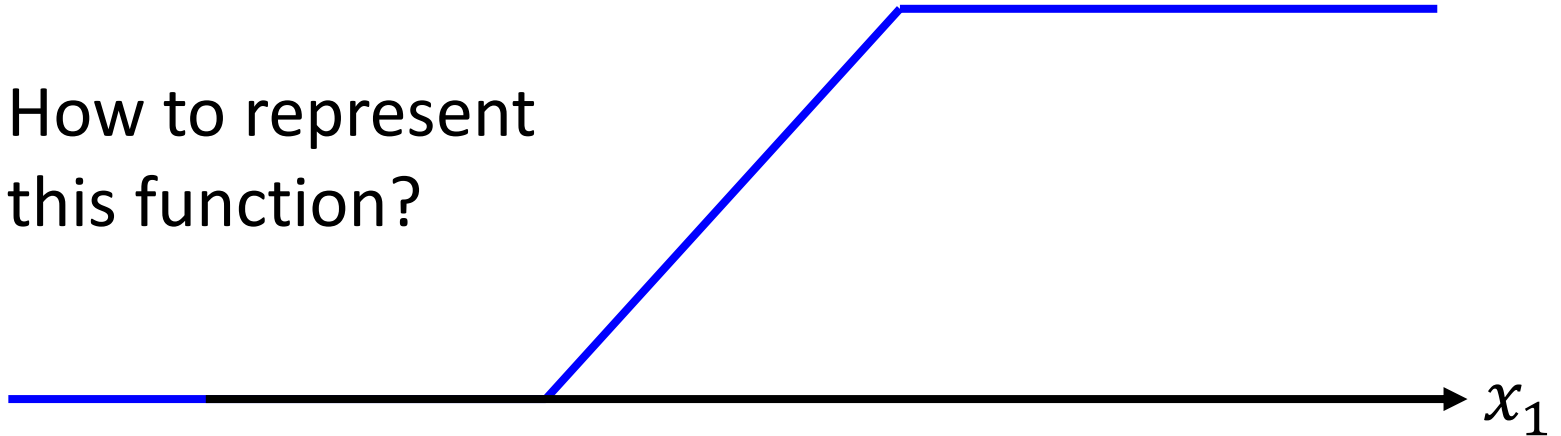
Back to ML Framework



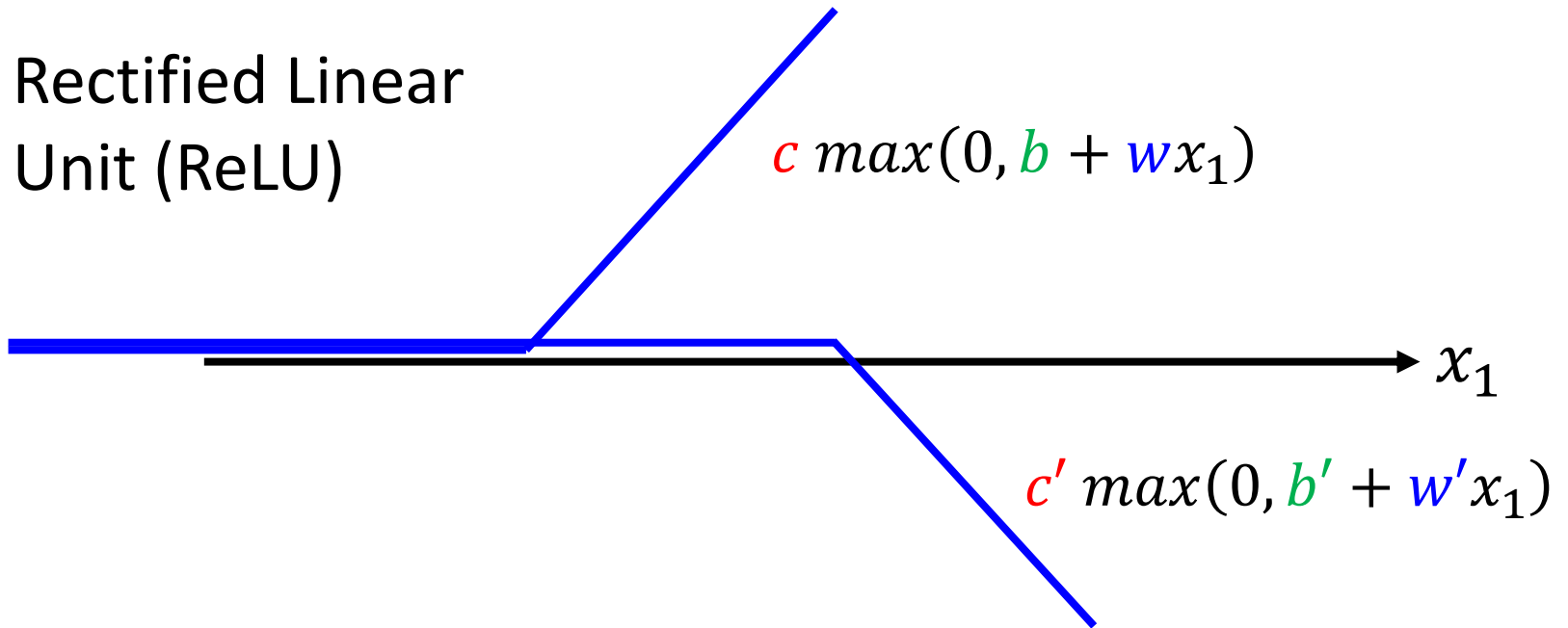
More variety of models ...

Sigmoid \rightarrow ReLU


How to represent
this function?



Rectified Linear
Unit (ReLU)



Sigmoid \rightarrow ReLU

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


Activation function

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

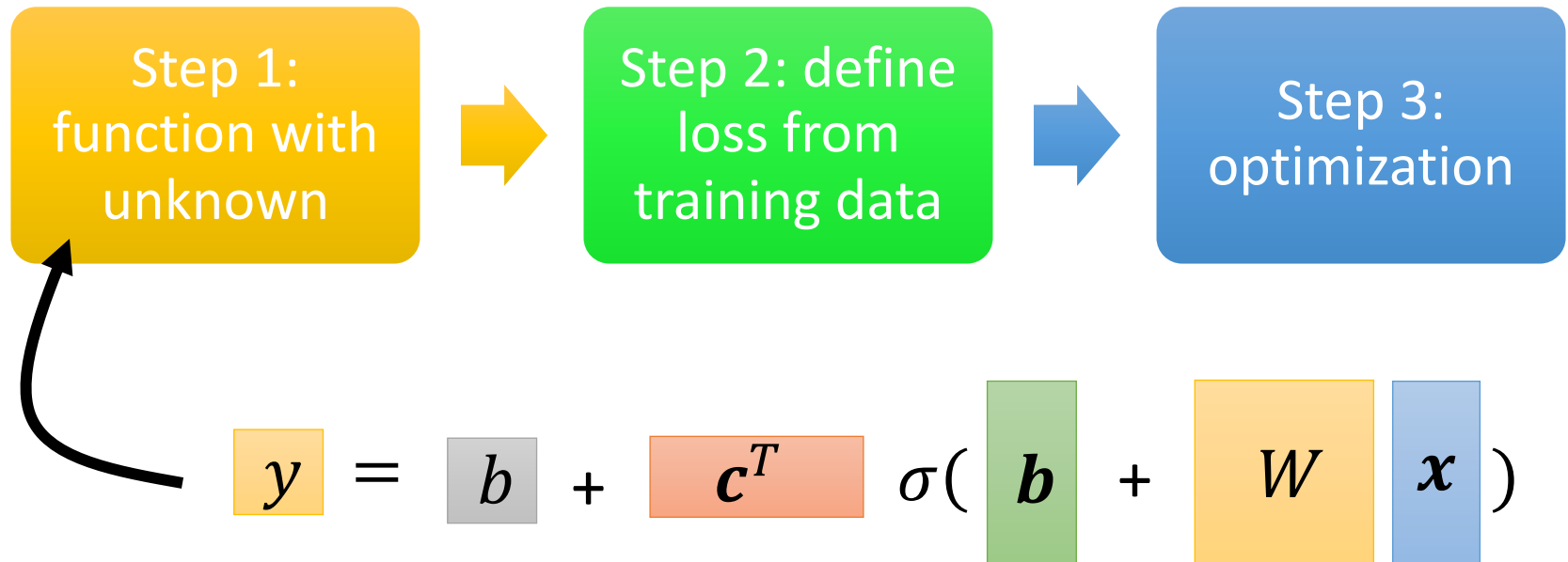
Which one is better?

Experimental Results

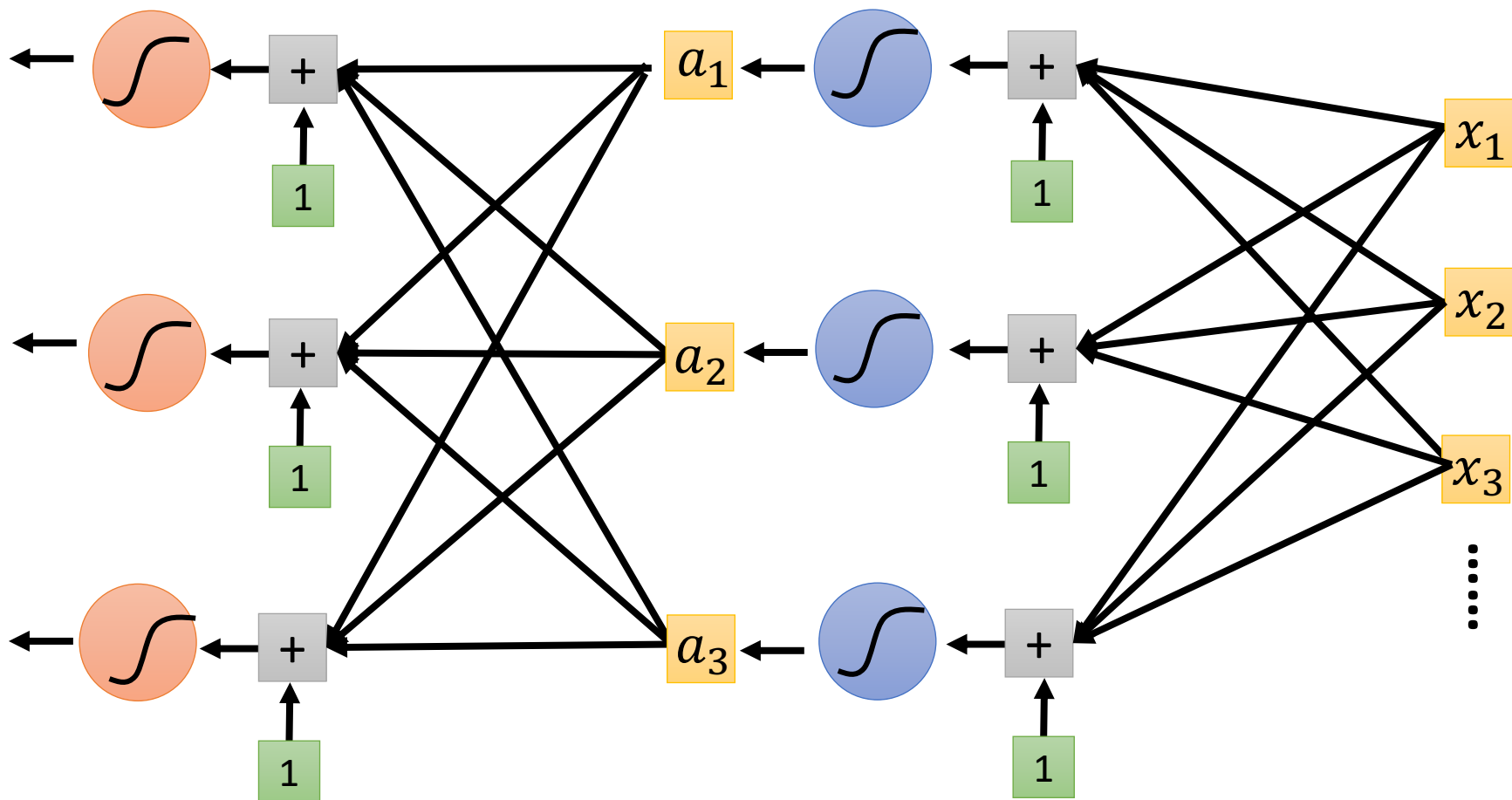
$$y = b + \sum_{\textcolor{red}{i}} \textcolor{red}{c}_i \max \left(0, \textcolor{green}{b}_i + \sum_j \textcolor{blue}{w}_{ij} x_j \right)$$

	linear
2017 – 2020	0.32k
2021	0.46k

Back to ML Framework



Even more variety of models ...



$$\begin{aligned}
 \boxed{a'} &= \sigma \left(\boxed{b'} + \boxed{W'} \boxed{a} \right) & \boxed{a} &= \sigma \left(\boxed{b} + \boxed{W} \boxed{x} \right)
 \end{aligned}$$

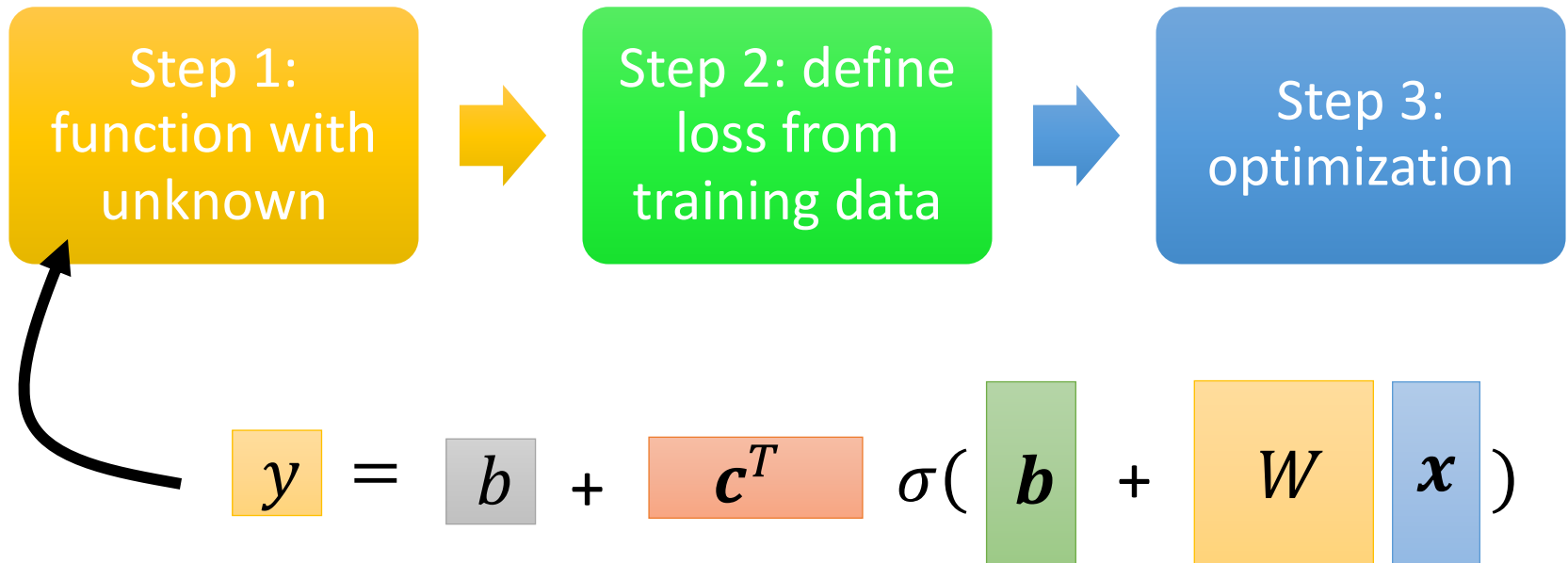
A dashed arrow points from the \boxed{a} term in the second equation to the \boxed{a} term in the first equation, indicating the sequential flow of information from input to hidden to output.

Experimental Results

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

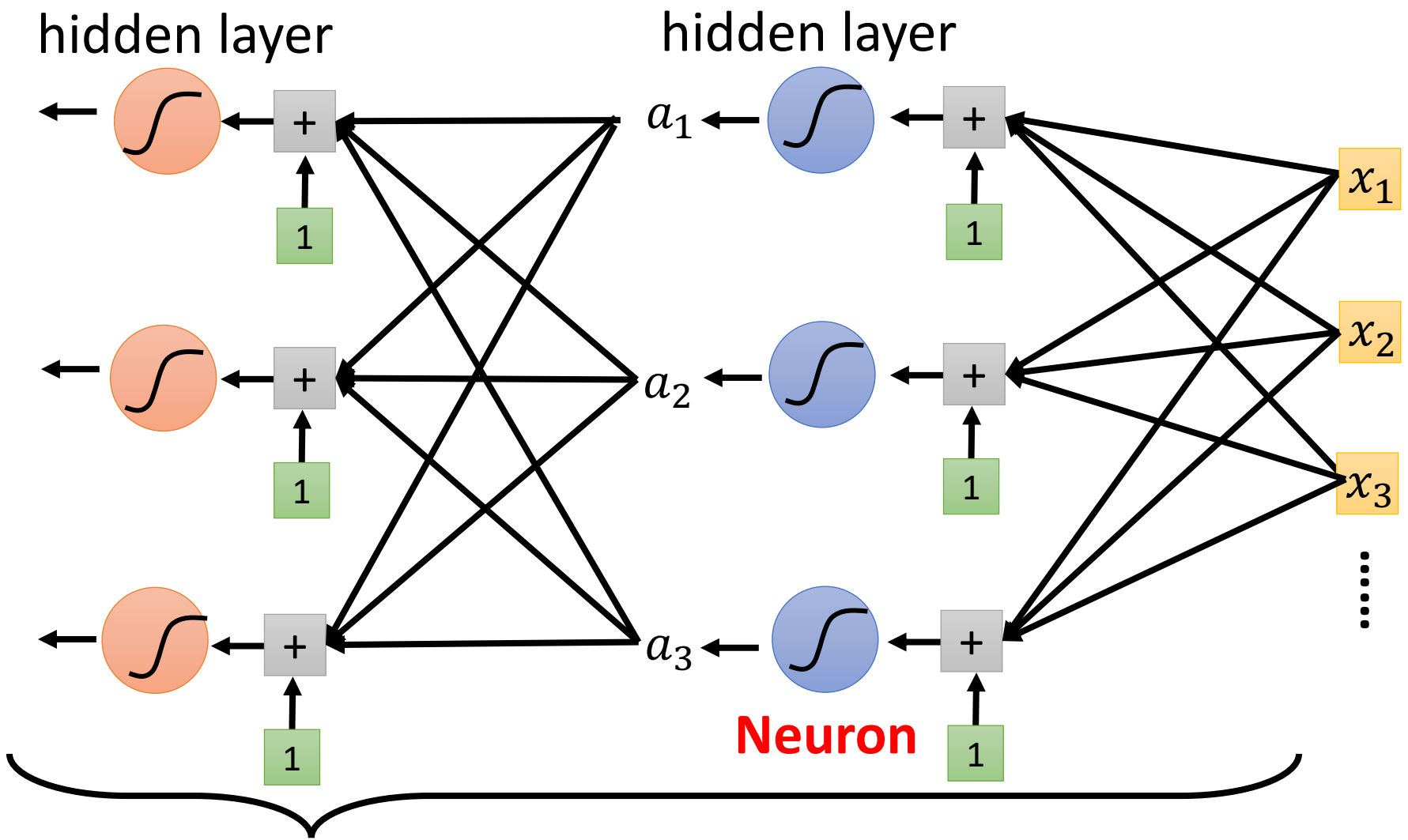
	1 layer
2017 – 2020	0.28k
2021	0.43k

Back to ML Framework



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

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



Neural Network

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

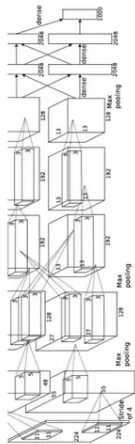
Many layers means **Deep** ➡ Deep Learning

Deep = Many hidden layers

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

8 layers

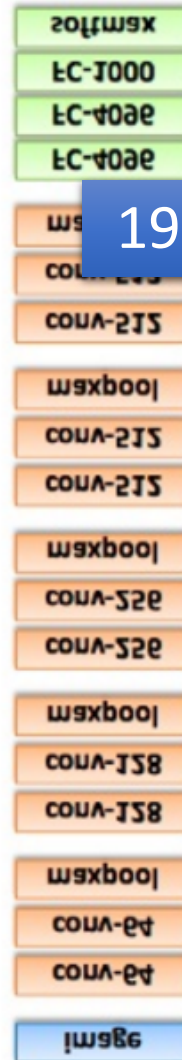
16.4%



AlexNet (2012)

19 layers

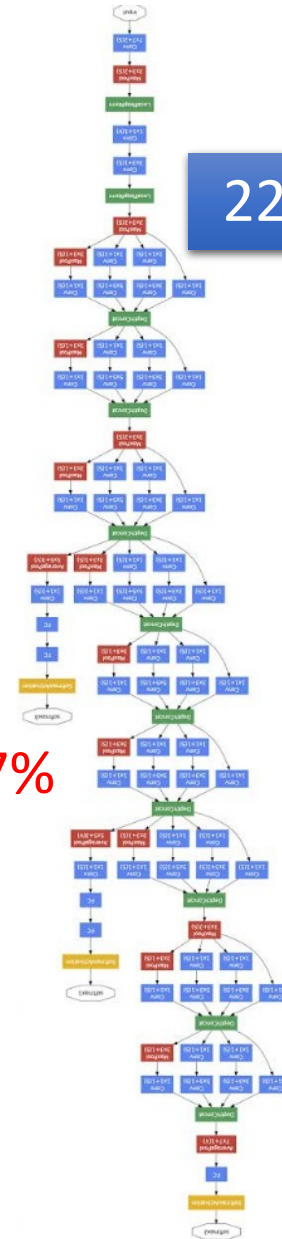
7.3%



VGG (2014)

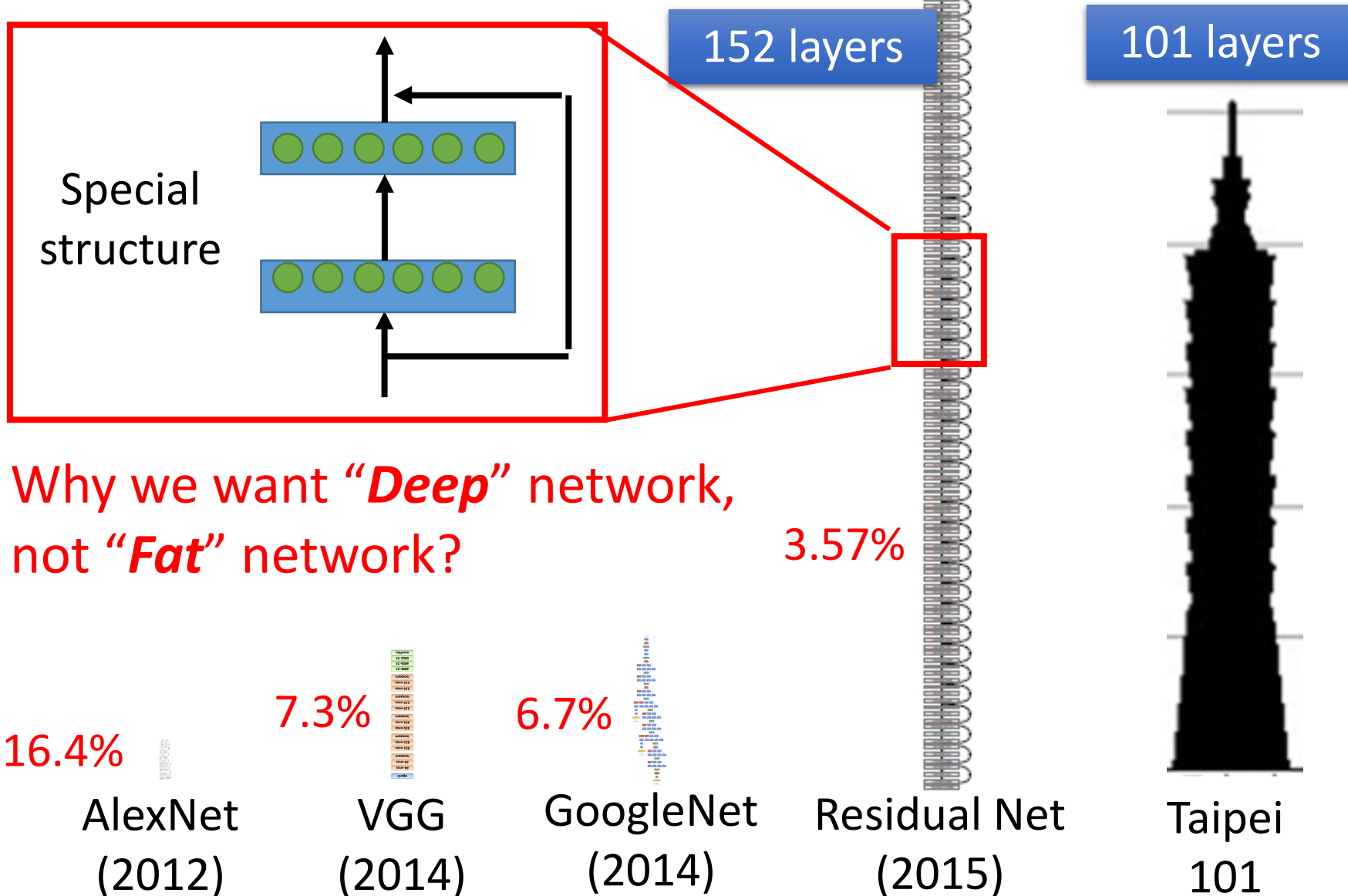
22 layers

6.7%



GoogleNet (2014)

Deep = Many hidden layers



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**

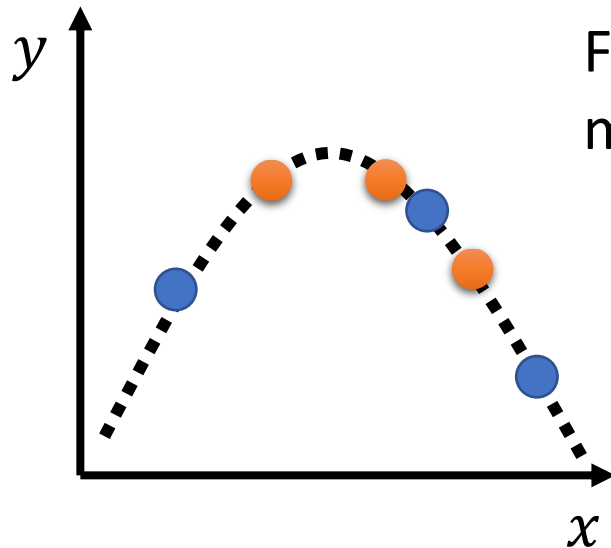
Let's predict no. of views today!

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

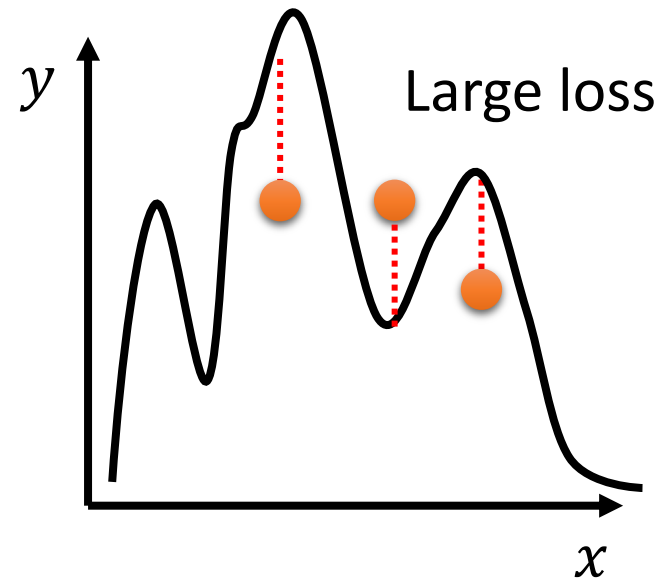
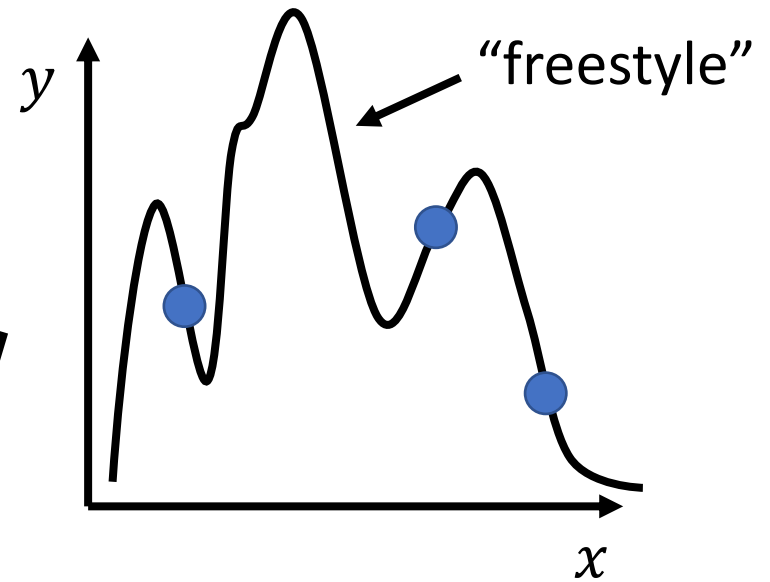
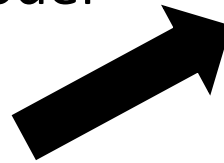
	1 layer	2 layer	3 layer	4 layer
2017 – 2020	0.28k	0.18k	0.14k	0.10k
2021	0.43k	0.39k	0.38k	0.44k

We will talk about model selection next time. 😊

Overfitting



Flexible
model



---- Real data distribution
(not observable)

● Training data

● Testing data

Acknowledgments

- I would like to express my sincere gratitude to the online course of Machine learning delivered by Professor Hyung-Yi Lee from National Taiwan University.