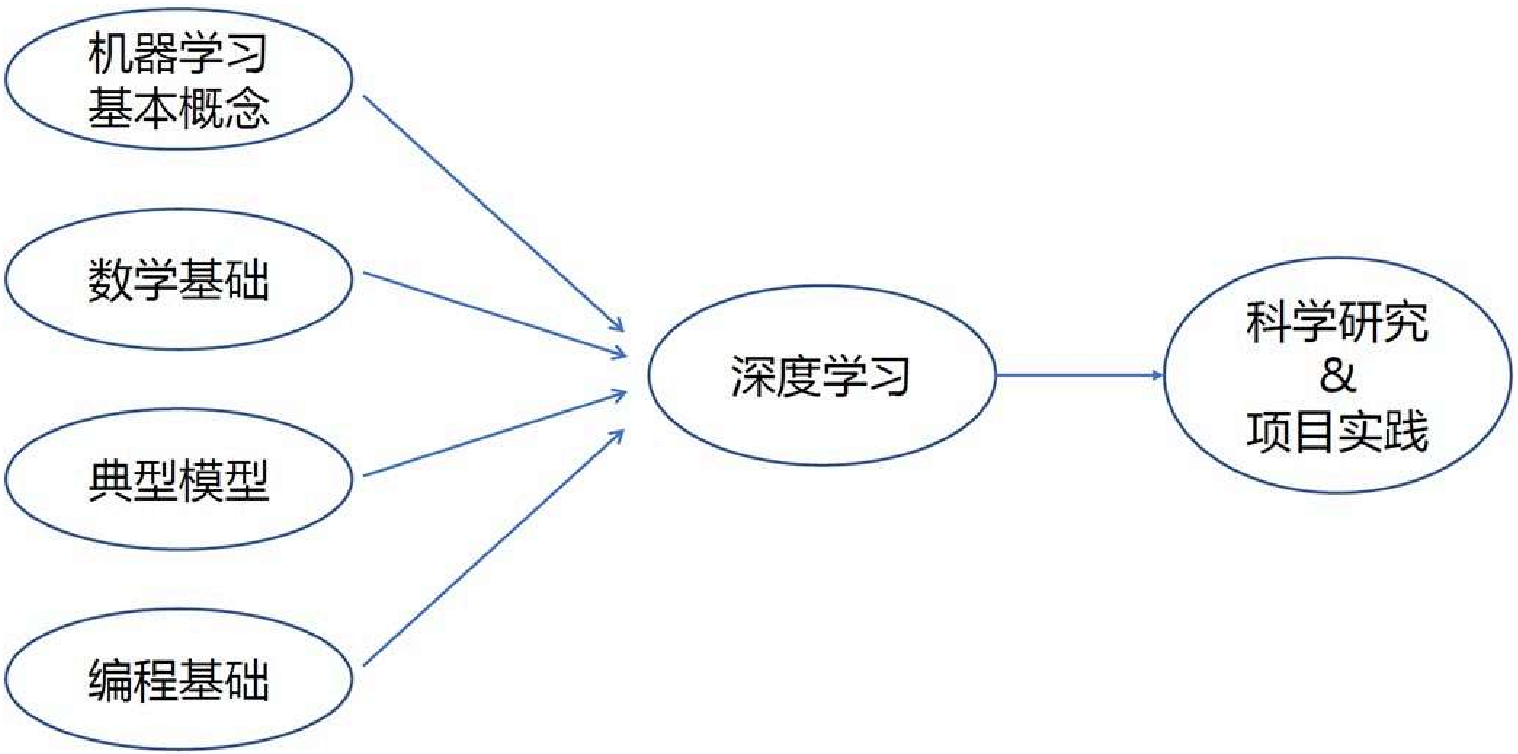
第二讲　基础知识

深度学习基础 & 数据基础



深度学习基础

➢ 深度(机器)学习 ≈ 构建一个映射函数

◆ 语音识别 $f($  $)=$ “你好”

◆ 图像识别 $f($  $)=$ “猫”

◆ 围棋 $f($  $)=$ “5-5” (落子位置)

◆ 对话系统 $f($ “你好” $)=$ “今天天气真不错”

用户输入 机器回应

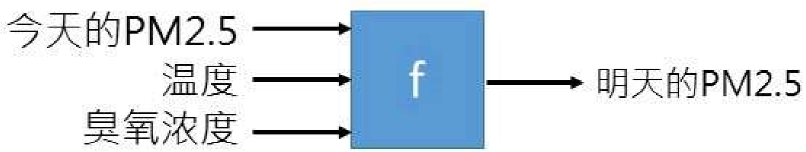
Neural networks and deep learning https://udlbook.github.io/udlbook/

深度学习的类别

➢ **深度学习的类别**(Different types of Functions)

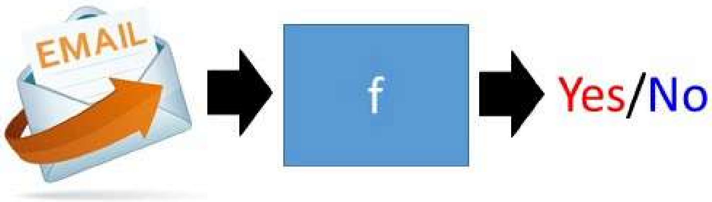
◆ **回归(Regression)**: 函数的输出是标量.

预测
PM2.5



◆ **分类(Classification)**:给定选项(类)，函数输出正确的选项

垃圾邮件
过滤(spam filter)

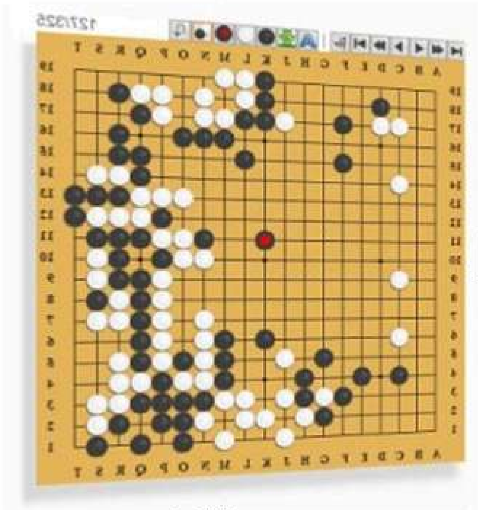


深度学习的类别

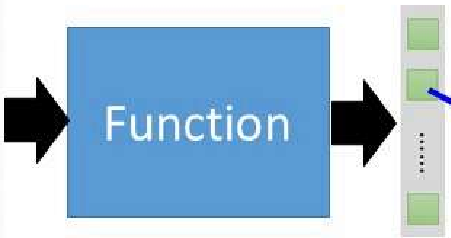
➢ **深度学习的类别**(Different types of Functions)

◆ **分类(Classification)**:给定选项(类)，函数输出正确的选项

每个位置是一个分类
(19 x 19 类)



下棋



棋盘上的一个位置

下一步落子位置

从观看视频人数开始

我们想要找的函数…

$$y = f(\quad)$$

2月27日的
观看人数

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%

深度学习的过程-构建模型

1. 假设带有未知参数(parameters)的函数

$$y = f\Big($$  $$\Big)$$

模型 $y = b + wx_1$ 根据先验知识

y: 2月26号的观看人数, x_1: 2月25号的观看人数 **特征**

w 和 b 是未知的参数(从数据中学习得到)

权重（weight） **偏置（bias）**

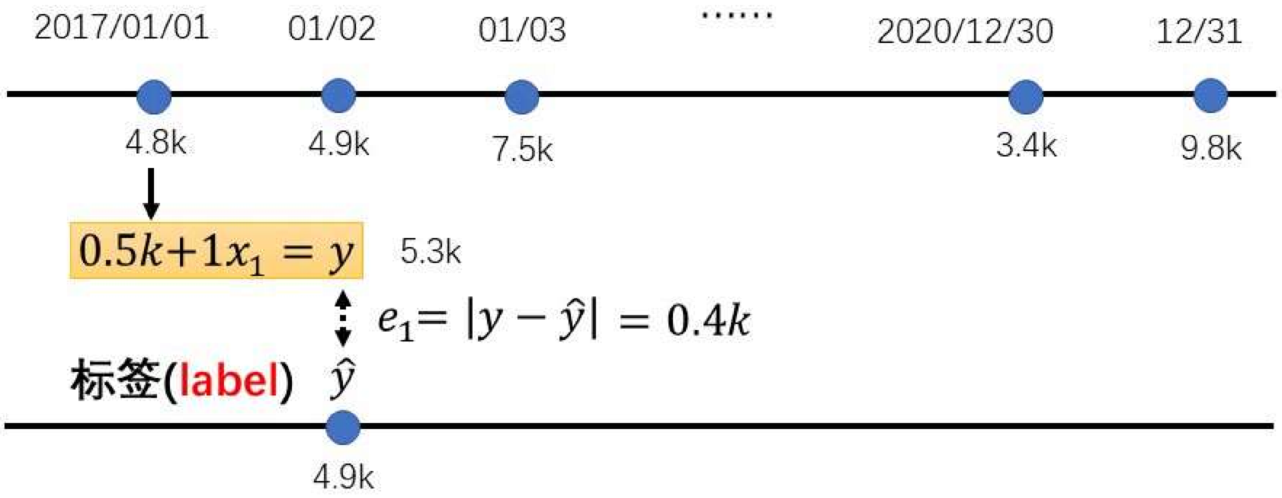
深度学习的过程-定义损失

2. 从训练数据定义损失(Loss)

- Loss 是参数的函数 $L(b, w)$
- Loss: 一组参数有多好的度量标准.

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

从2017/01/01 – 2020/12/31的数据



雨课堂 Rain Classroom

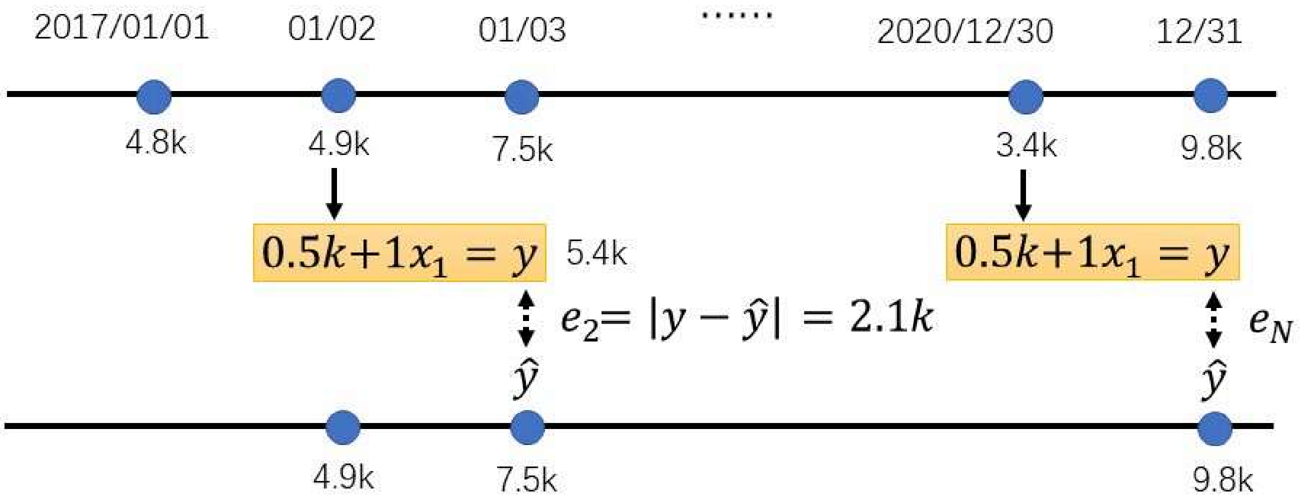
深度学习的过程-定义损失

2. 从训练数据定义损失

- Loss 是参数的函数 $L(b,w)$
- Loss: 一组参数有多好的度量标准.

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

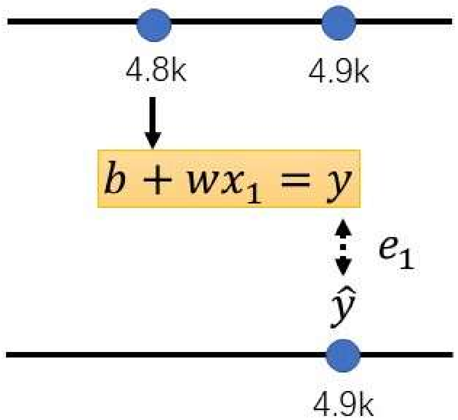
从2017/01/01 – 2020/12/31的数据



深度学习的过程 -定义损失

2. 从训练数据定义损失(Loss)

- Loss 是参数的函数 $L(b,w)$
- Loss: 一组参数有多好的度量标准.



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)

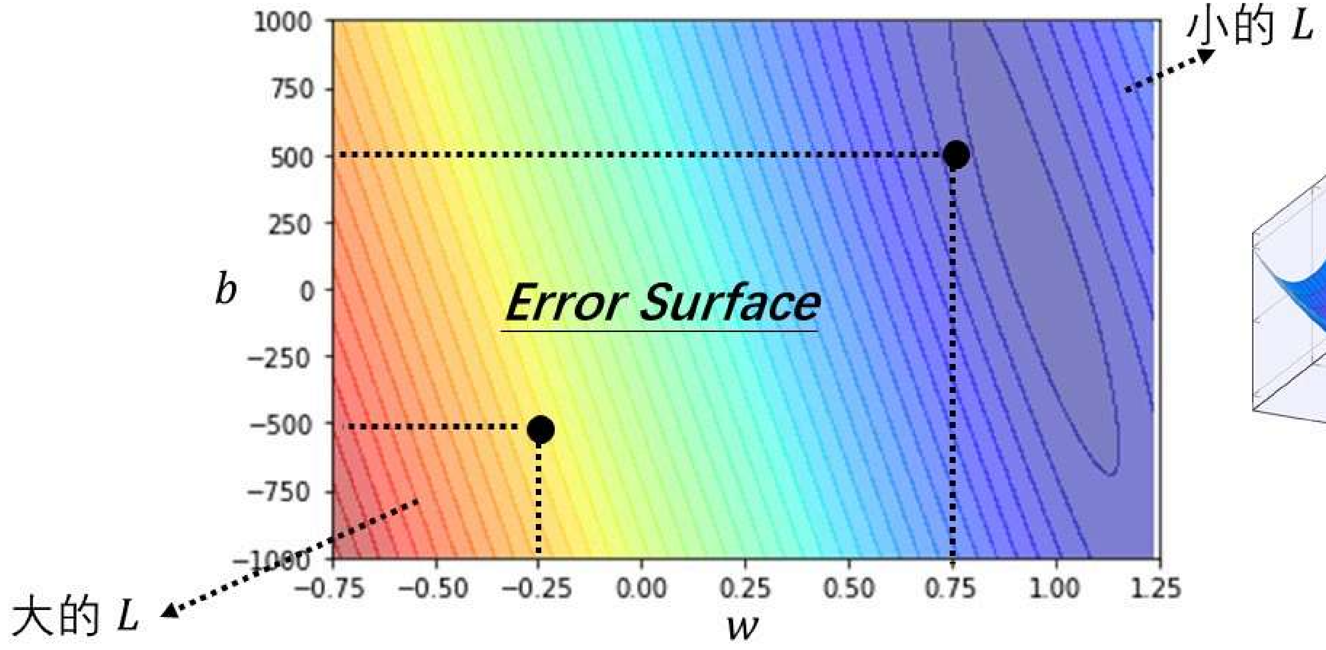
如果 y 和 $\hat{y}$ 都是几率分布 ➡ 交叉熵（Cross-entropy）

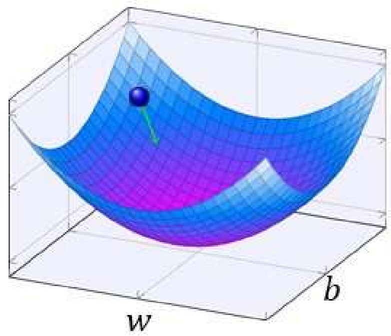
深度学习的过程-定义损失

2. 从训练数据定义损失(Loss)

- Loss 是参数的函数 $L(b, w)$
- Loss: 一组参数有多好的度量标准.

模型 $y = b + wx_1$

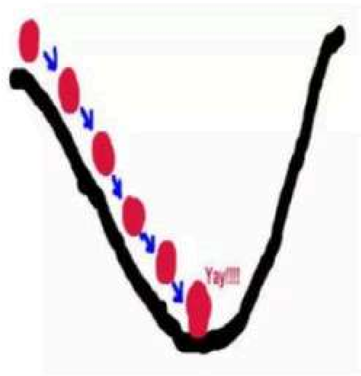




梯度下降法（Gradient Descent ）

- 假设一个人需要从山的某处开始下山，尽快到达山底。在下山之前他需要确认两件事：下山的方向和下山的距离。因为下山的路有很多，他必须利用一些信息，找到从该处开始最陡峭的方向下山，这样可以保证他尽快到达山底。此外，这座山最陡峭的方向并不是一成不变的，每当走过一段规定的距离，他必须停下来，重新利用现有信息找到新的最陡峭的方向。通过反复进行该过程，最终抵达山底。

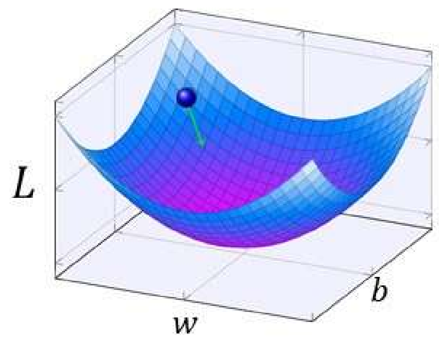
- **梯度下降法用于求解无约束最优化问题**：山代表了需要优化的函数表达式；山的最低点就是该函数的最优值；每次下山的距离代表后面要解释的学习率；寻找方向利用的信息即为样本数据；最陡峭的下山方向则与函数表达式**梯度的方向**有关，之所以要寻找最陡峭的方向，是为了满足最快到达山底的限制条件；某处——代表了我们给优化函数设置的初始值，算法后面正是利用这个初始值进行不断的迭代求出最优解。



梯度下降法（Gradient Descent）

损失函数

$$L(w,b)=\frac{1}{2n}\sum_{i=1}^{n}[y(x_i)-\hat{y}]^2$$



假设要最小化某些函数$L(\theta)$,它可以是任意的多元实值函数，$\theta=\theta_1;\theta_2;\ldots\ldots$。$\theta$代替了$w$ 和b 以强调它可能是任意的函数。

当我们在θ_1 和θ_2 方向分别将球体移动一个很小的量，即$\Delta\theta_1$和$\Delta\theta_2$ 时，球体将会发生什么情况。微积分告诉我们L 将会有如下变化:

$$\Delta L=\frac{\partial L}{\partial\theta_1}\Delta\theta_1+\frac{\partial L}{\partial\theta_2}\Delta\theta_2$$

ΔL符号?

要寻找一种选择$\Delta\theta_1$和$\Delta\theta_2$的方法使得ΔL为**负**；选择**负**是为了让球体滚落。

$$\Delta\boldsymbol{\theta}=(\Delta\theta_1,\Delta\theta_2)^{\mathrm{T}}，\quad \nabla\boldsymbol{L}=\left(\frac{\partial \mathrm{L}}{\partial\theta_1},\frac{\partial \mathrm{L}}{\partial v_2}\right) \longrightarrow \Delta L=\nabla\boldsymbol{L}\cdot\Delta\boldsymbol{\theta}$$

梯度下降法（Gradient Descent ）

$$\Delta L = \boldsymbol{\nabla L} \cdot \Delta\boldsymbol{\theta}$$

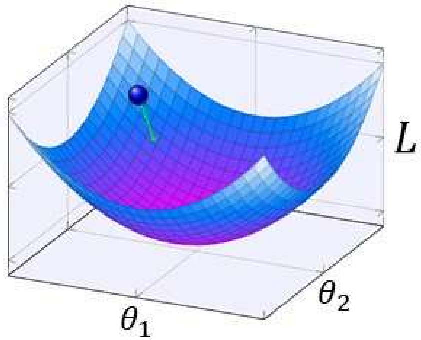
这个方程真正让我们兴奋的是它让我们看到了如何选取Δv 才能让ΔL 为负数。
假设我们选取:

$$\Delta\boldsymbol{\theta} = -\eta\nabla\boldsymbol{L}$$

η 称为学习速率。

$$\Delta L = \Delta\boldsymbol{L} \cdot \Delta\boldsymbol{\theta} = -\eta\nabla\boldsymbol{L} \cdot \Delta\boldsymbol{L} = -\eta\|\Delta\boldsymbol{L}\|^2$$

$$\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}' = \boldsymbol{\theta} - \eta\nabla\boldsymbol{L}$$



然后我们用它再次更新规则来计算下一次移动。如果我们反复持续这样做，我们将持续减小$\boldsymbol{L}$直到—— 正如我们希望的—— 获得一个全局的最小值。

总结一下，梯度下降算法工作的方式就是重复计算梯度$\nabla\boldsymbol{L}$，然后沿着相反的方向移动,沿着山谷“滚落”。我们可以想象它像这样:

$$\boldsymbol{\nabla L} = \left(\frac{\partial \mathrm{L}}{\partial\theta_1}, \cdots\cdots, \frac{\partial \mathrm{L}}{\partial\theta_m}\right) \qquad \Delta\boldsymbol{\theta} = -\eta\nabla\boldsymbol{L}$$

深度学习的过程-优化

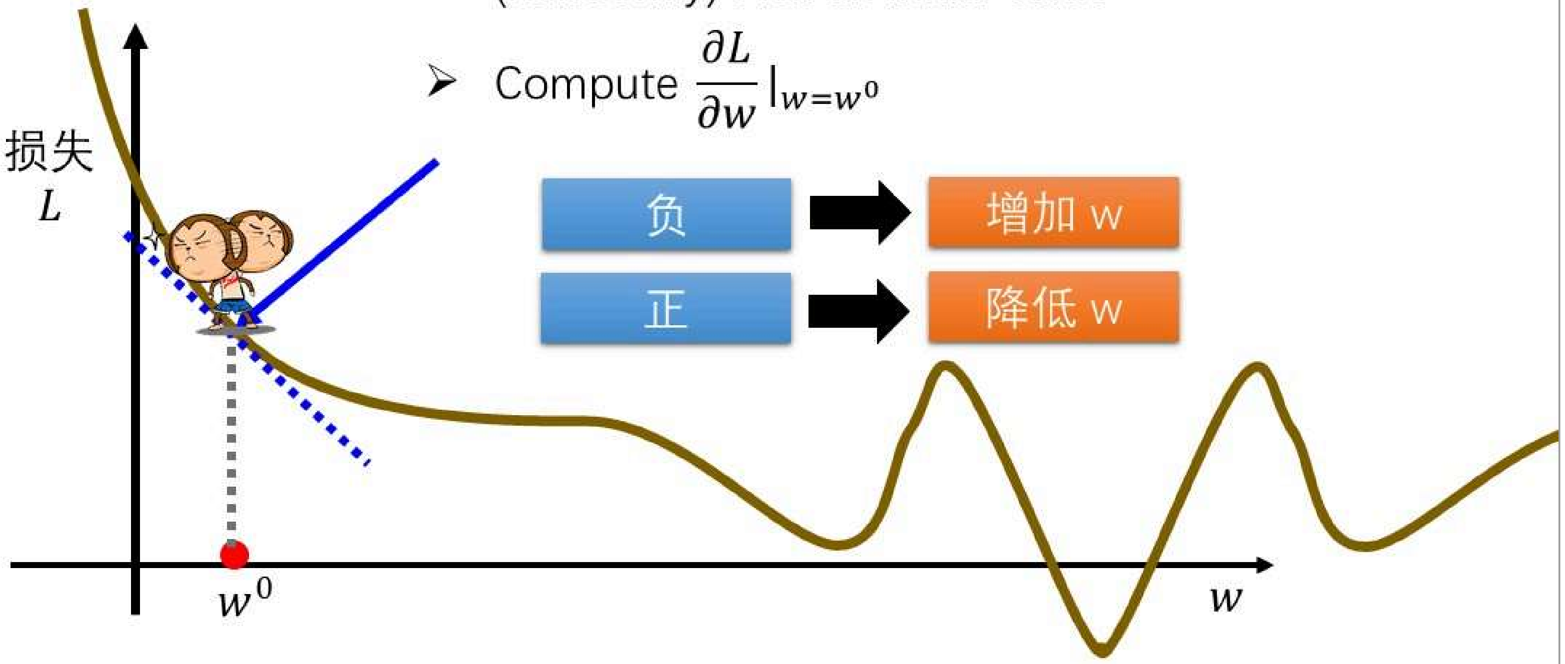
Source of image: http://chico386.pixnet.net/album/photo/171572850

3. 优化(Optimization)

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

梯度下降（Gradient Descent）

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



深度学习的过程-优化

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

3. 优化(Optimization)

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

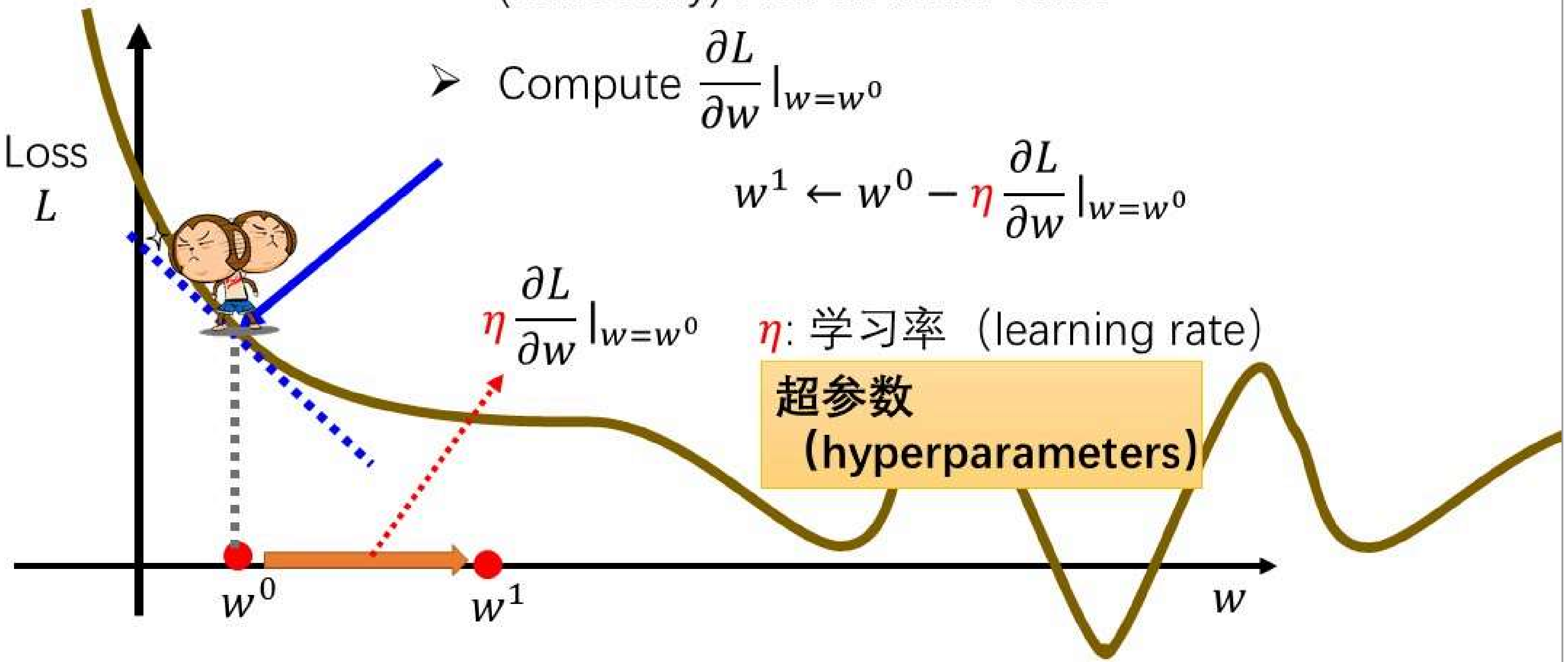
Gradient Descent

- (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: 学习率 (learning rate)

超参数 (hyperparameters)



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

3. 优化Optimization

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

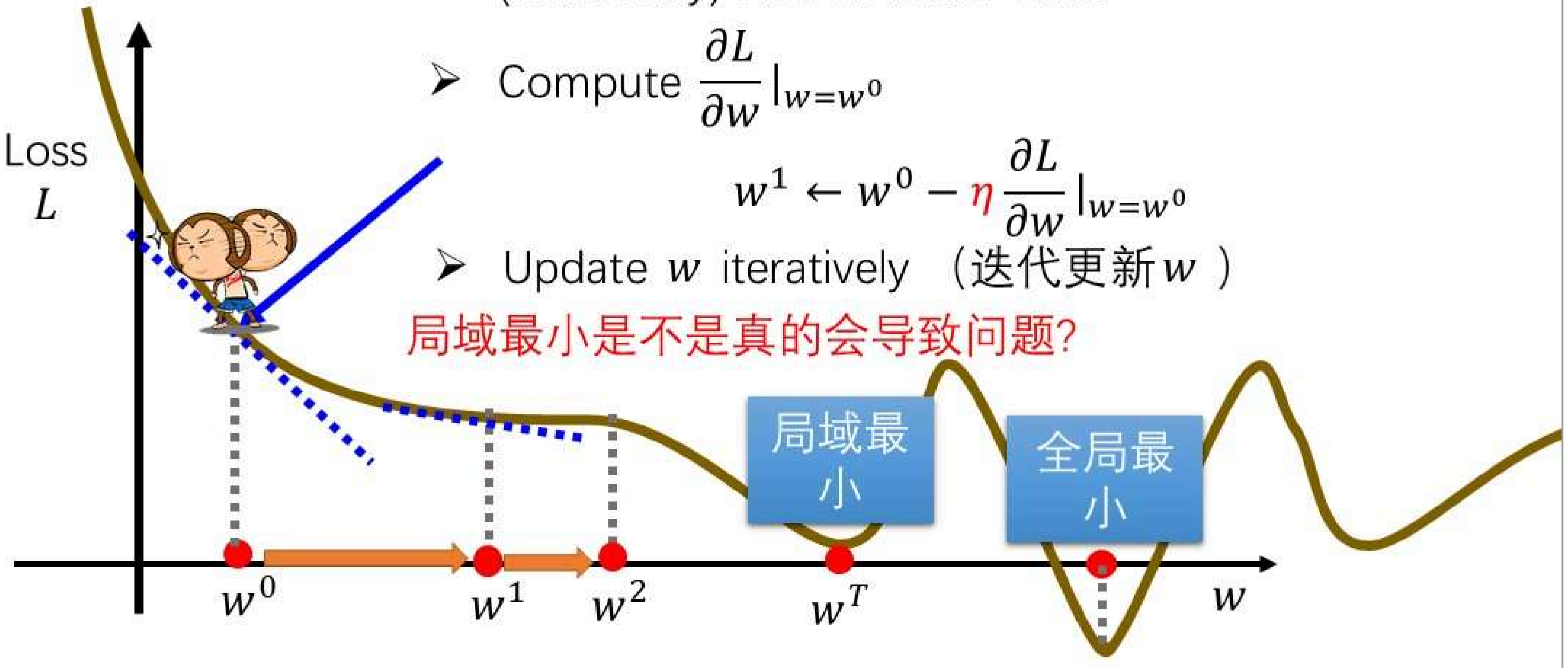
Gradient Descent

- (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 （迭代更新 w ）

局域最小是不是真的会导致问题?



深度学习的过程-优化

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

- 随机初始化w^0, b^0
- 计算

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

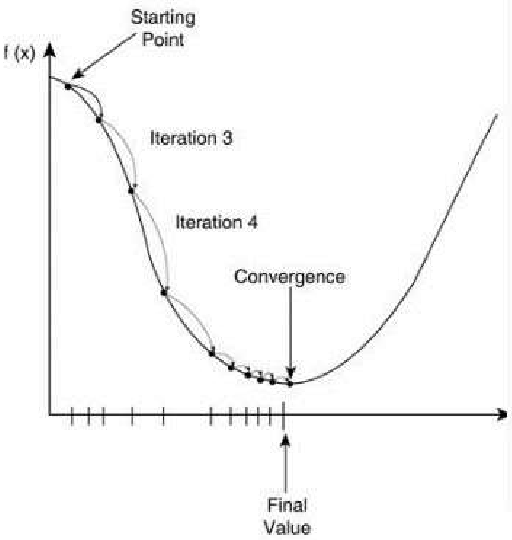
$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0}$$

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

Can be done in one line in most deep learning frameworks（一行代码解决）

- Update（更新） w and b interatively（迭代）

梯度下降法（Gradient Descent ）



1. 给定待优化连续可微函数 $L(\Theta)$、学习率 η 以及一组初始值 $L_0 = (\theta_{01}, \theta_{02}, \cdots, \theta_{0l},)$
2. 计算待优化函数梯度： $\nabla L(\Theta_0)$
3. 更新迭代公式： $\Theta^{0+1} = \Theta_0 - \eta\nabla L(\Theta_0)$
4. 计算 Θ^{0+1} 处函数梯度 $\nabla L(\Theta_{0+1})$
5. 计算梯度向量的模来判断算法是否收敛： $\|\nabla L(\Theta)\| \leqslant \varepsilon$
6. 若收敛，算法停止，否则根据迭代公式继续迭代

➢ 经过迭代计算风险函数的最小值

$$\theta_{t+1} = \theta_t - \eta\frac{\partial L}{\partial \theta}$$

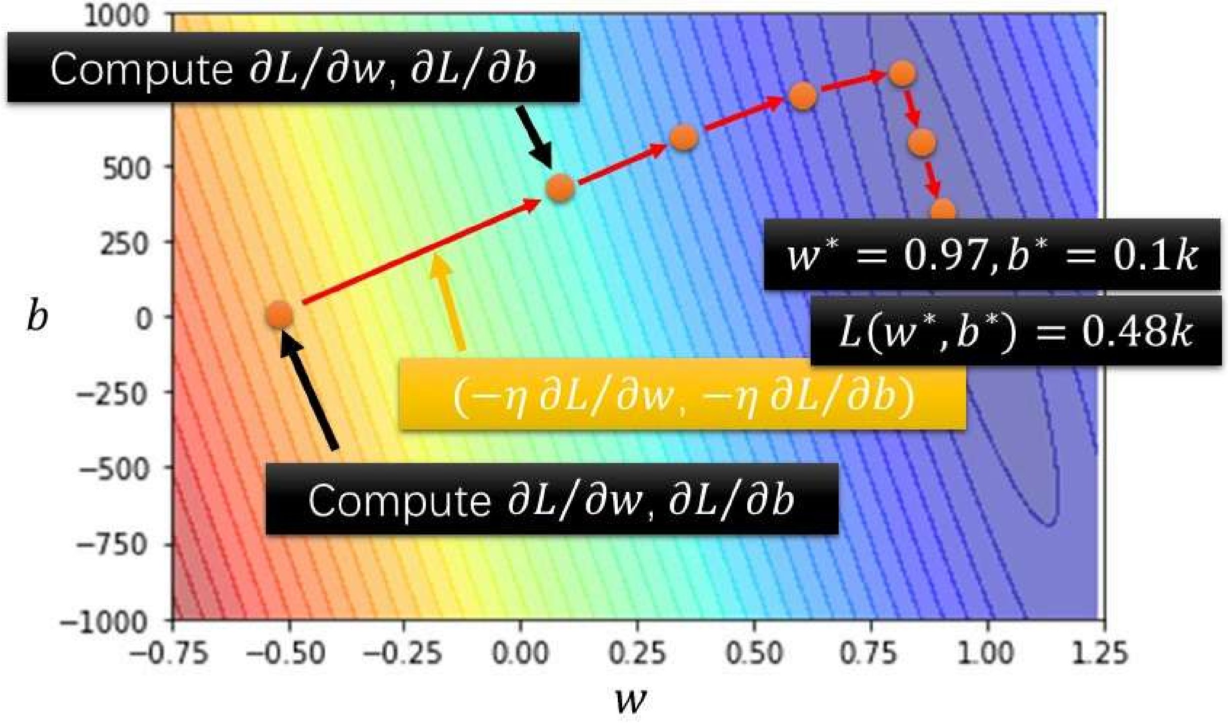
搜索中步长η也叫作**学习率**（Learning Rate）

深度学习的过程-优化

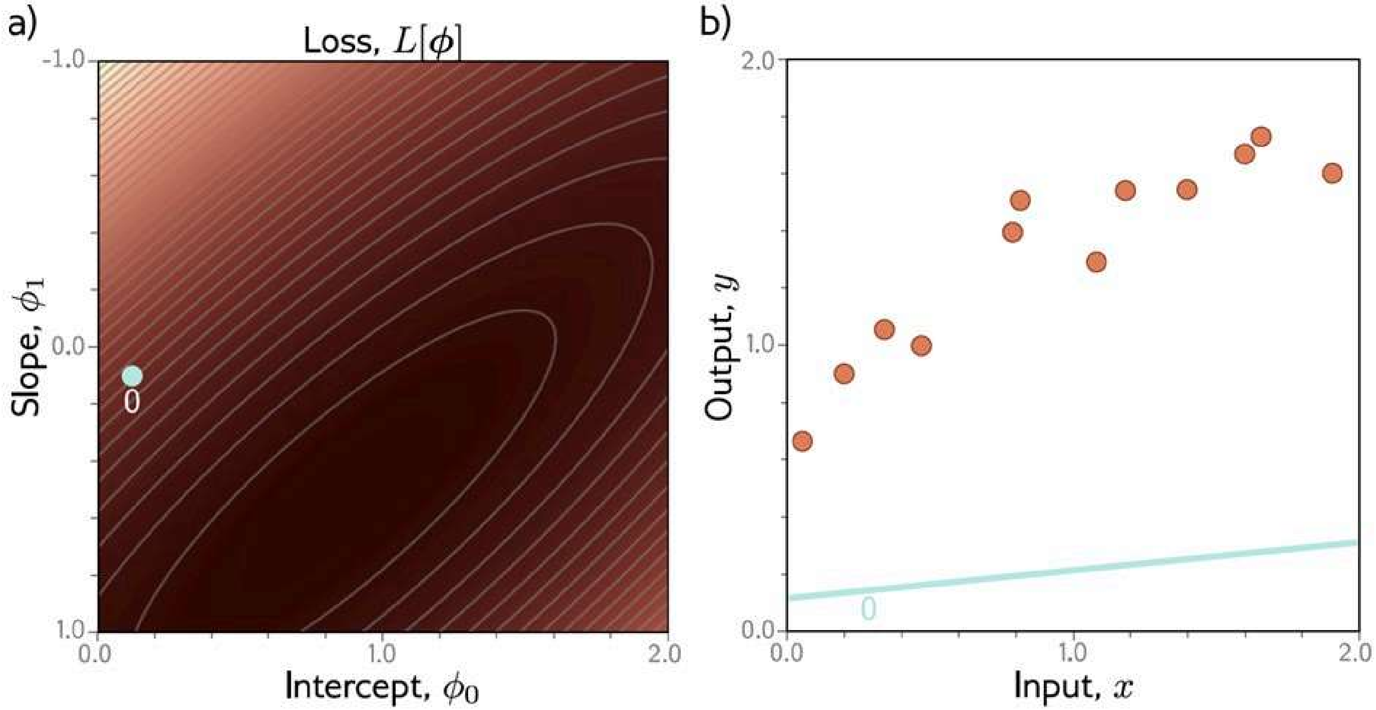
3. Optimization

模型 $y = b + wx_1$

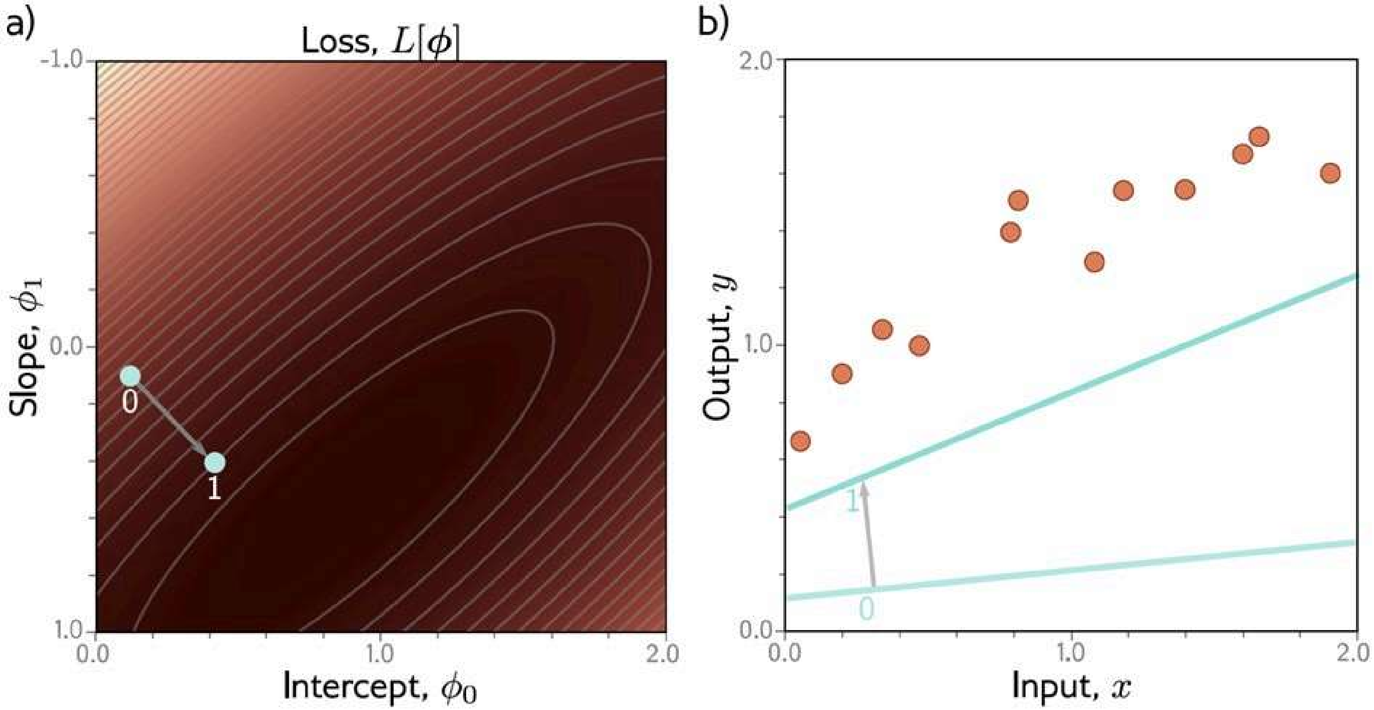
$$w^*, b^* = arg \min_{w,b} L$$



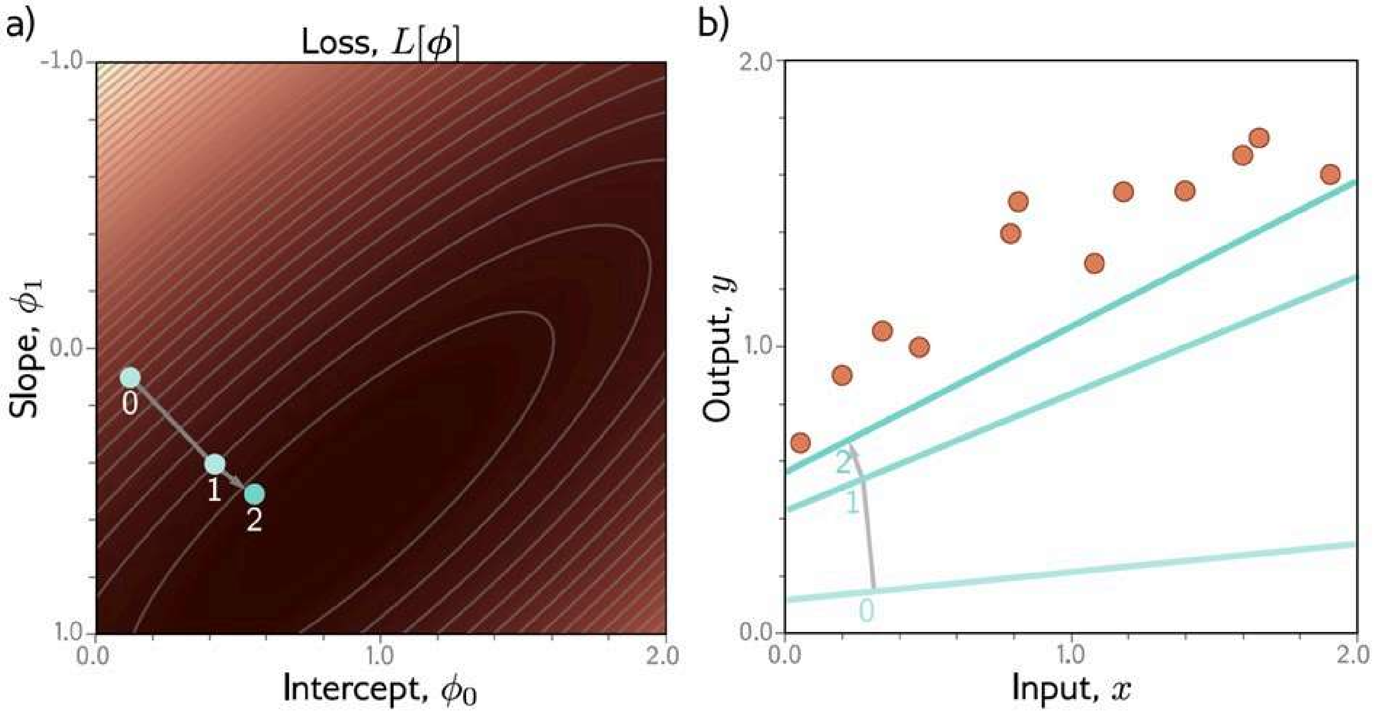
Example: 1D Linear regression training



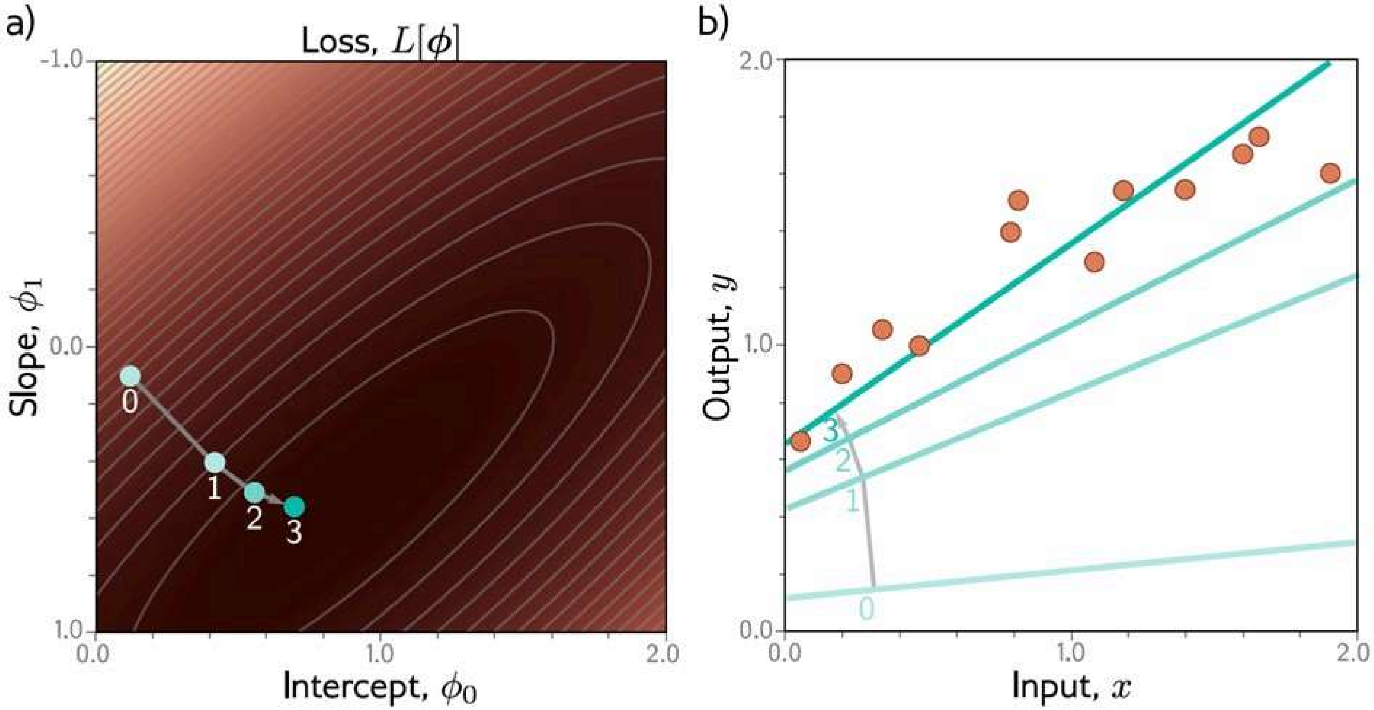
Example: 1D Linear regression training



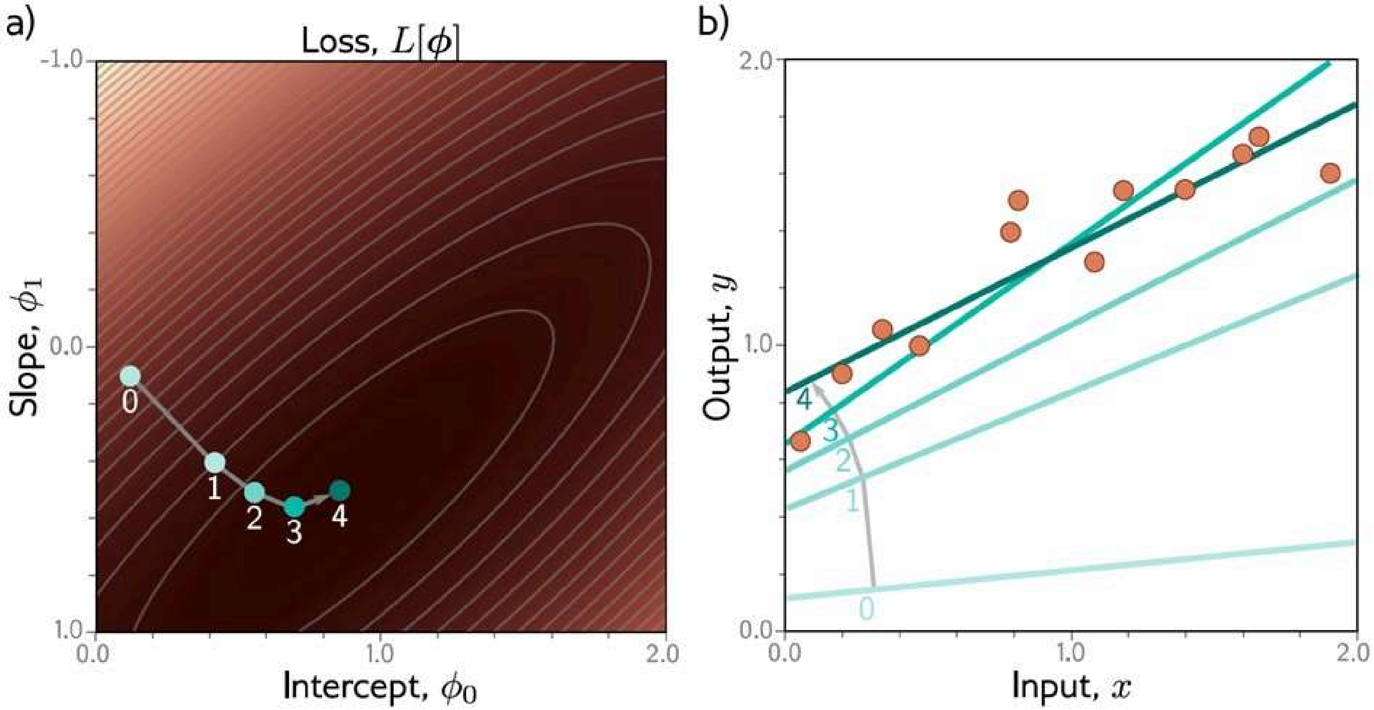
Example: 1D Linear regression training



Example: 1D Linear regression training



Example: 1D Linear regression training



This technique is known as gradient descent

深度学习的过程-构建模型-定义损失-优化

deep Learning is so simple ……

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

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

$y = b + wx_1$

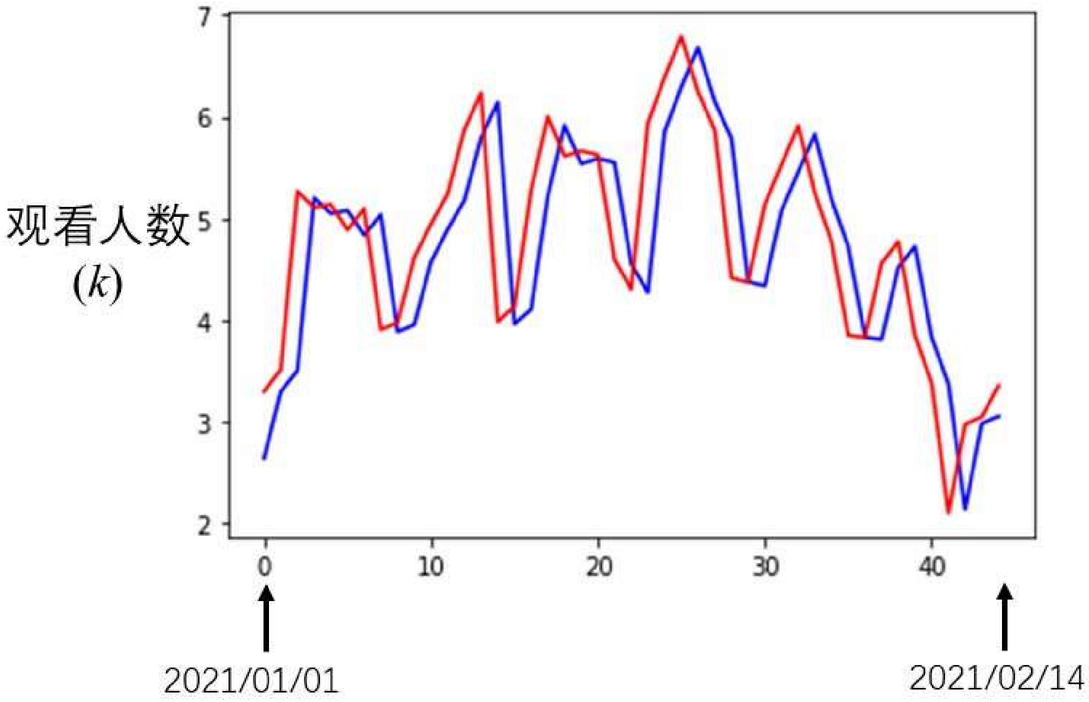
$y = 0.1k + 0.97x_1$ 得到最小的损失 $L = 0.48k$
从数据2017 – 2020 (训练数据**training data**)

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

$L' = 0.58k$

$$y = 0.1k + 0.97x_1$$

Red: 后台统计数据
blue: 模型预测数据



$$y = b + wx_1$$

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 $L' = 0.46k$

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

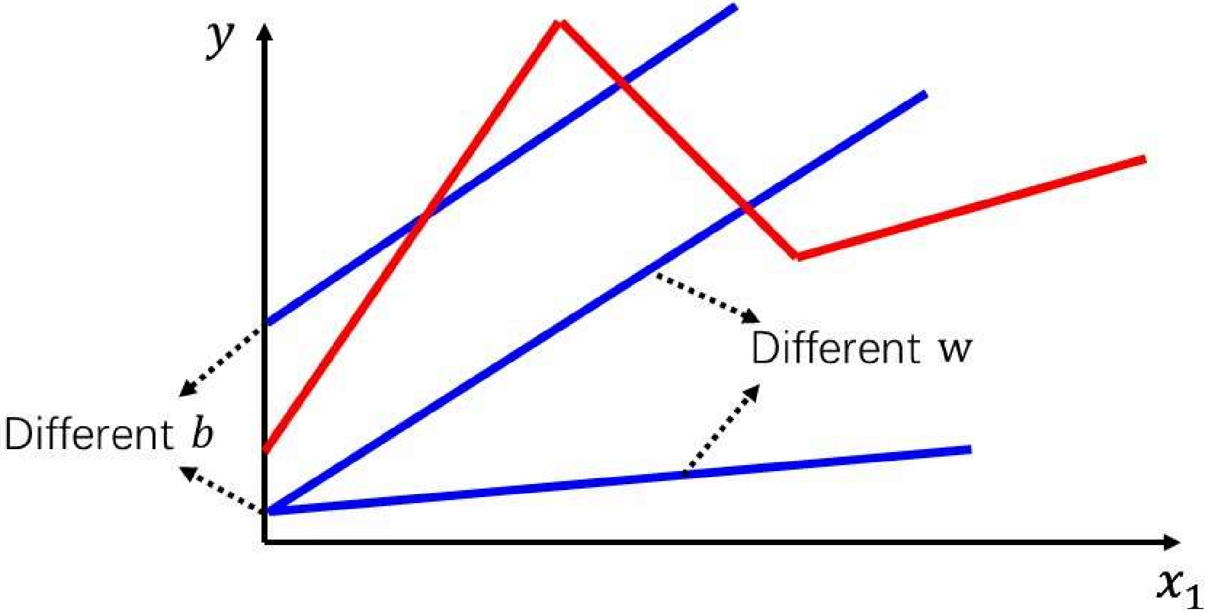
2017 - 2020 $L = 0.32k$

2021 $L' = 0.46k$

线性模型(Linear models)

深度学习非线性模型初探

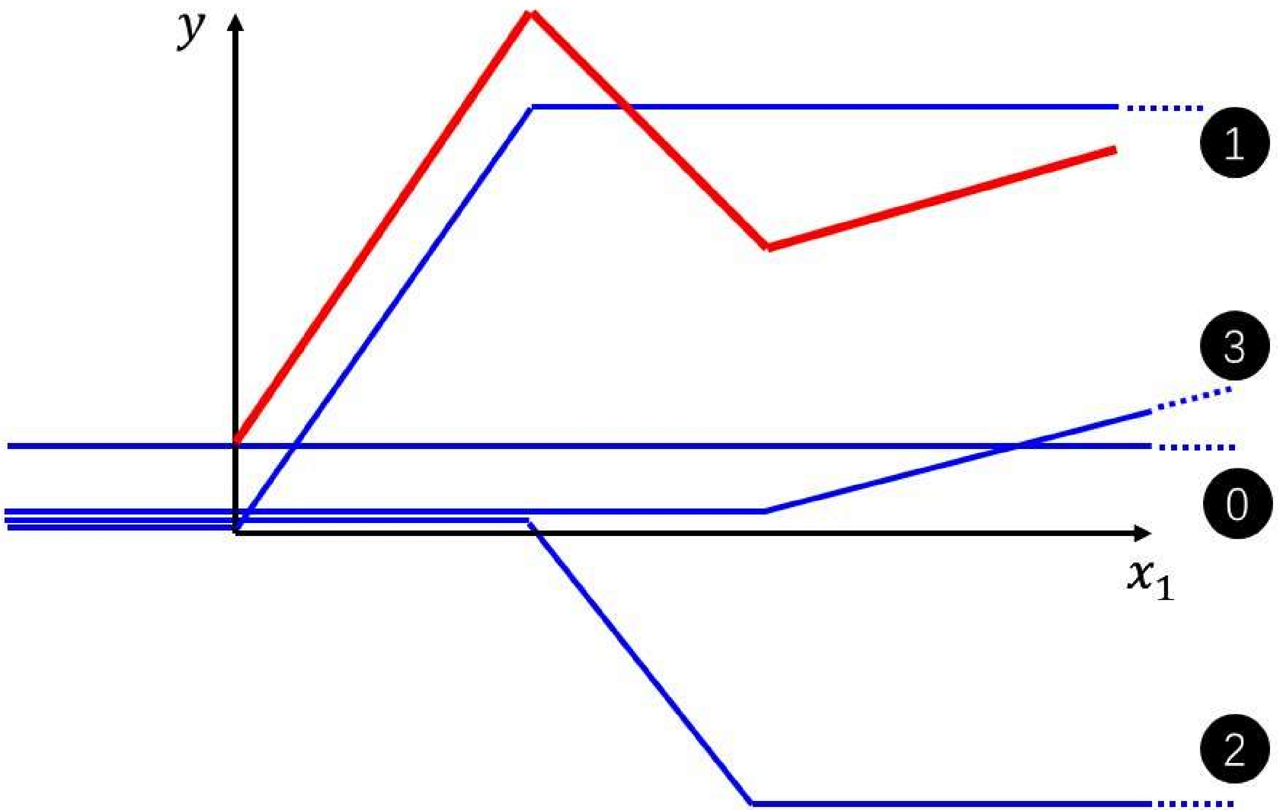
线性模型…我们需要更加复杂的模式.



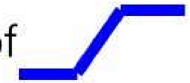
线性模型具有严重的限制. ***模型偏离***

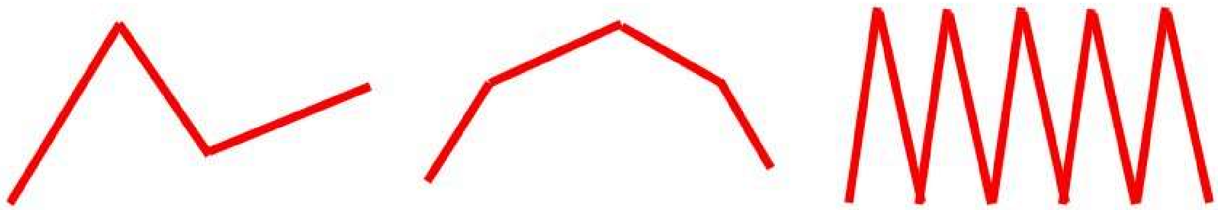
我们需要更加灵活的模型!

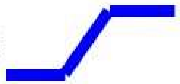
红色曲线= 常数 + 系列求和



所有分段连续线性曲线

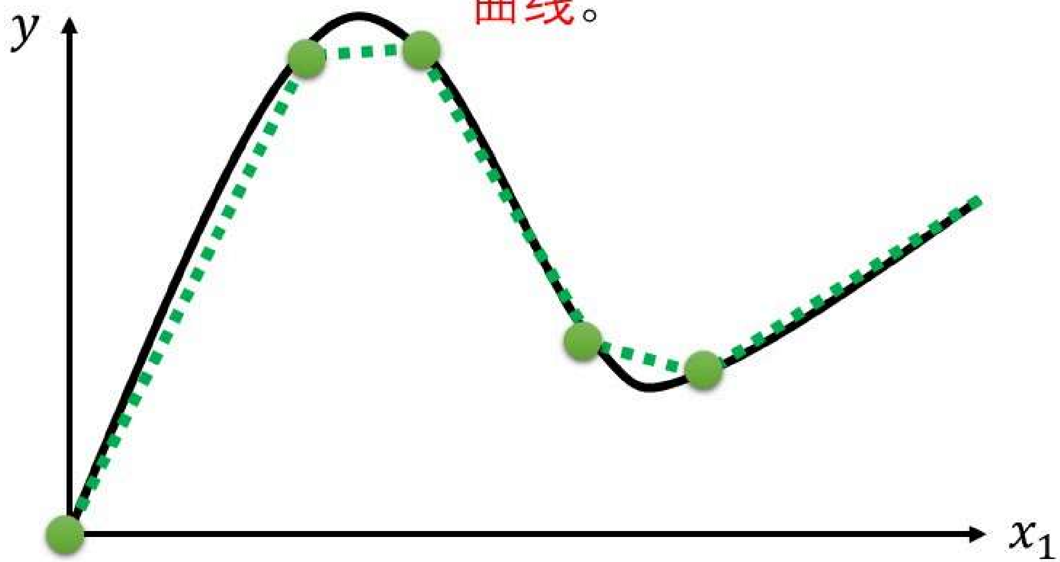
= constant + sum of a set of 



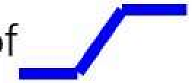
More pieces require more 

超越分段线性连续的的曲线?

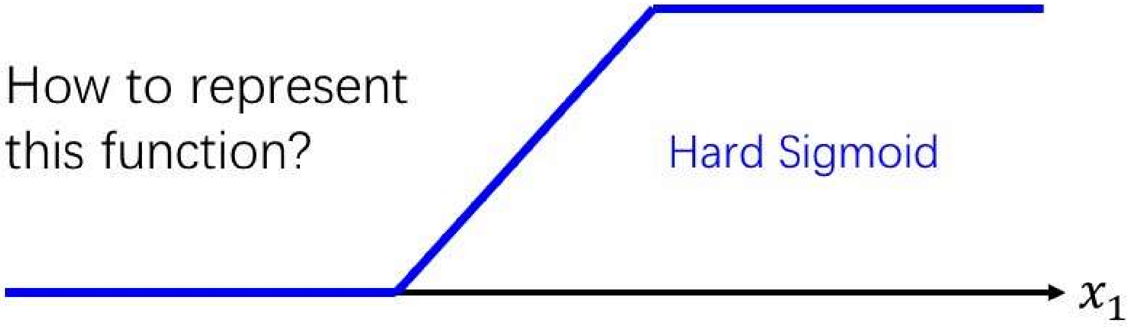
通过分段线性曲线近似连续曲线。



为了获得良好的近似值，我们需要足够的分段。

red curve = constant + sum of a set of 

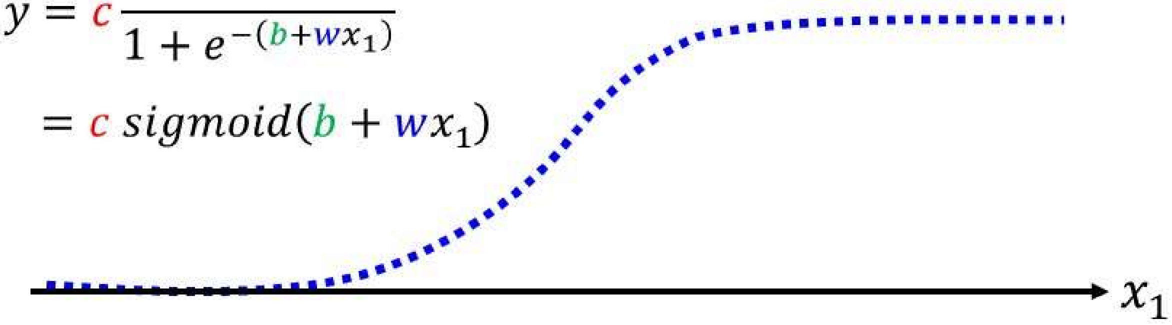
How to represent this function?

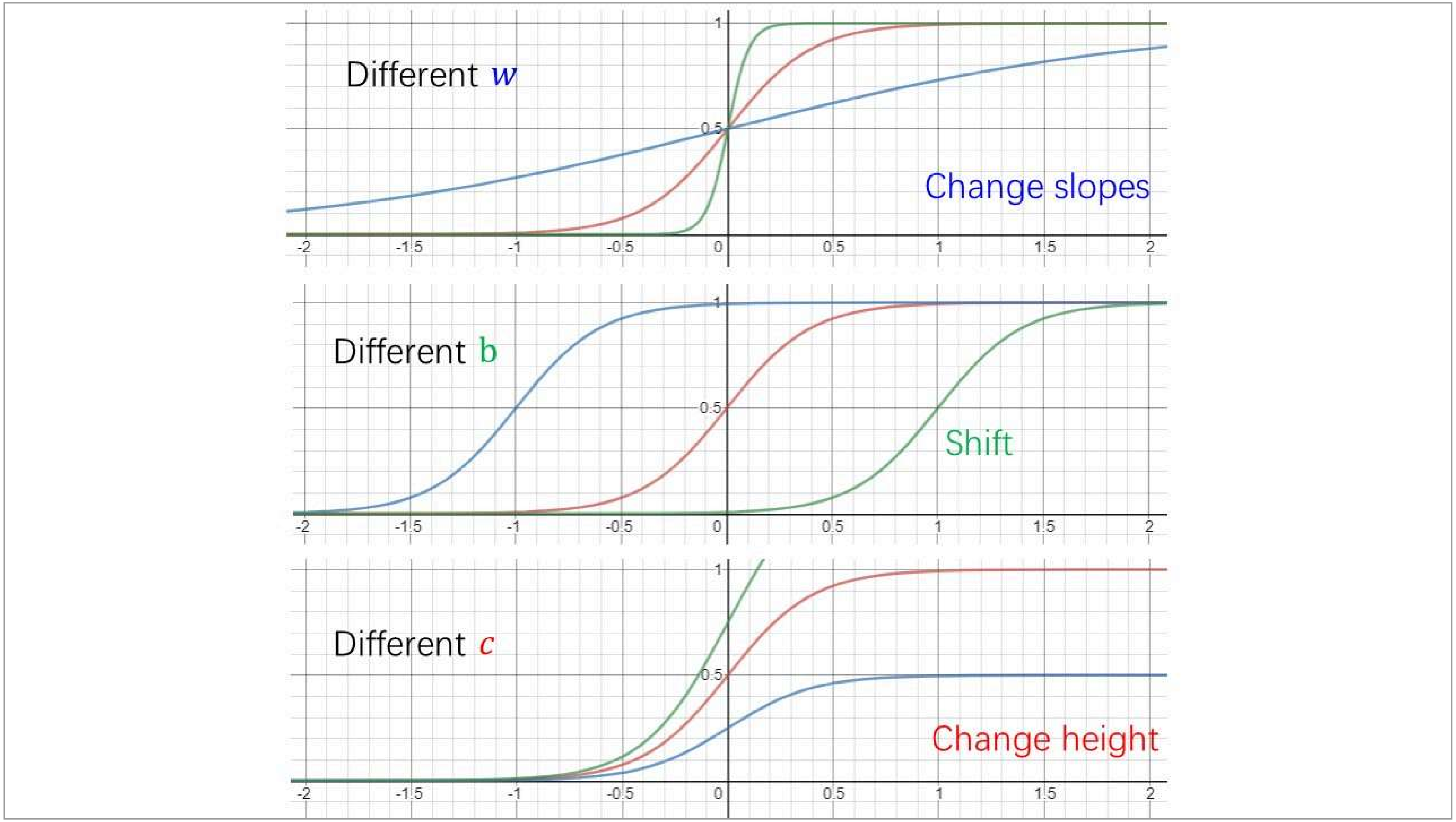


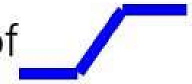
Sigmoid Function

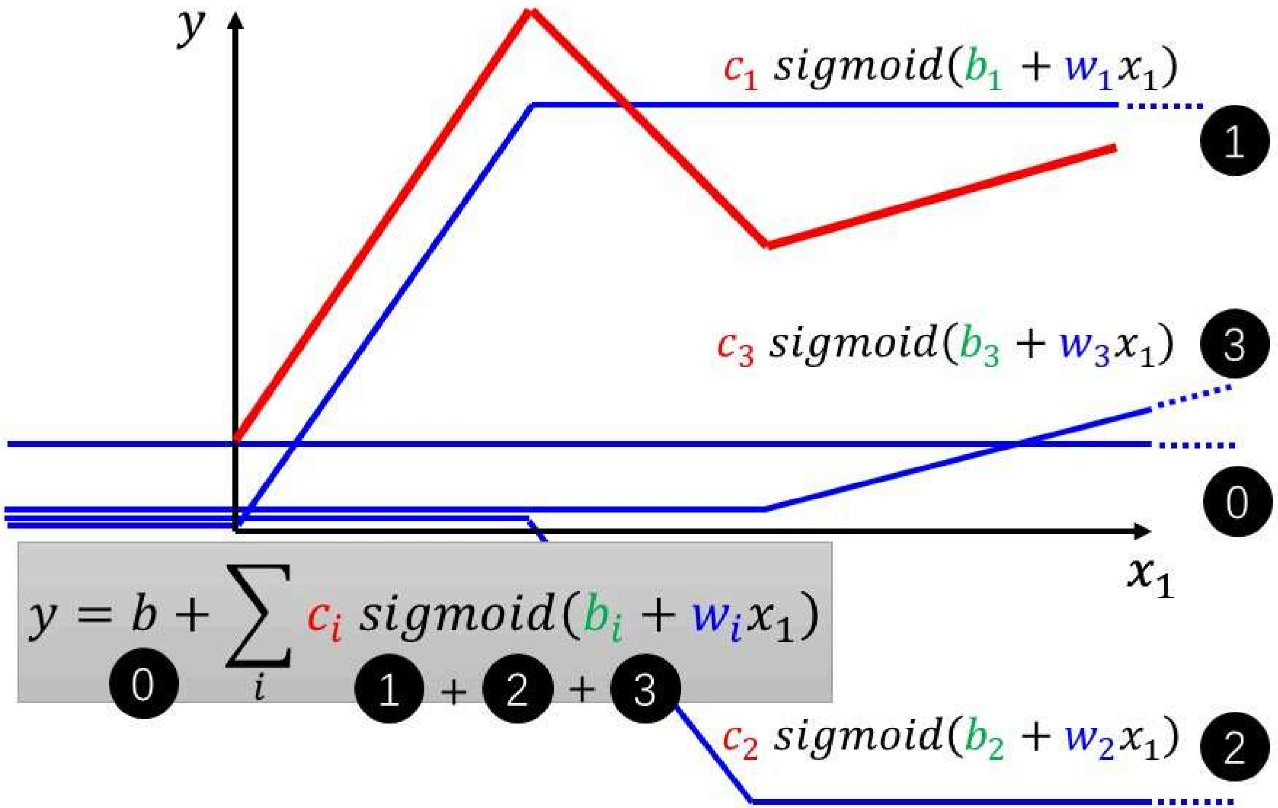
$$y = c\,\frac{1}{1+e^{-(b+wx_1)}}$$

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




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

新模型: 较多特征(More Features)

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

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

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

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

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

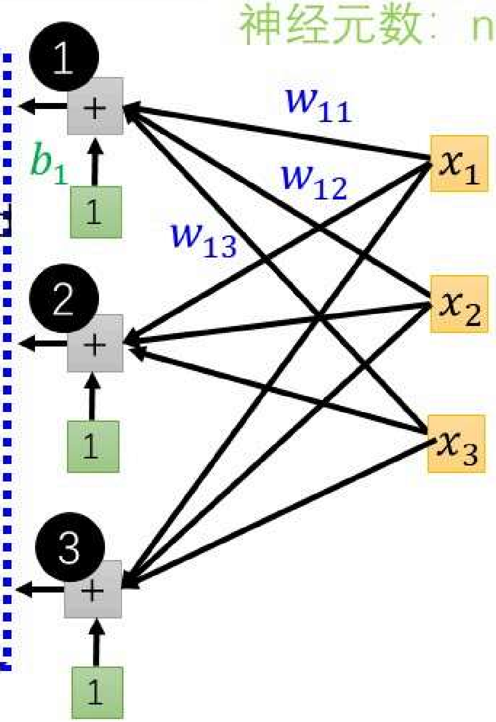
j: 1,2,3
特征数： no. of features
i: 1,2,3
神经元数： no. of sigmoid

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

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

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

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



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

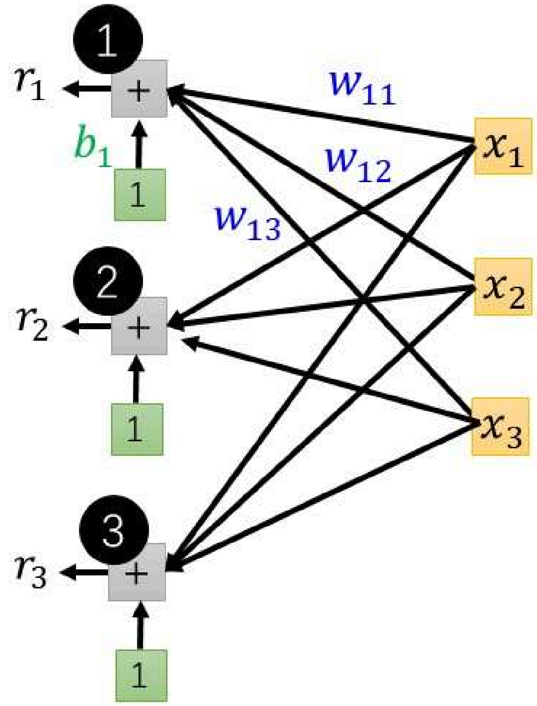
$$\boxed{\begin{aligned} r_1 &= b_1 + w_{11}x_1 + w_{12}x_2 + w_{13}x_3 \\ r_2 &= b_2 + w_{21}x_1 + w_{22}x_2 + w_{23}x_3 \\ r_3 &= b_3 + w_{31}x_1 + w_{32}x_2 + w_{33}x_3 \end{aligned}}$$

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

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

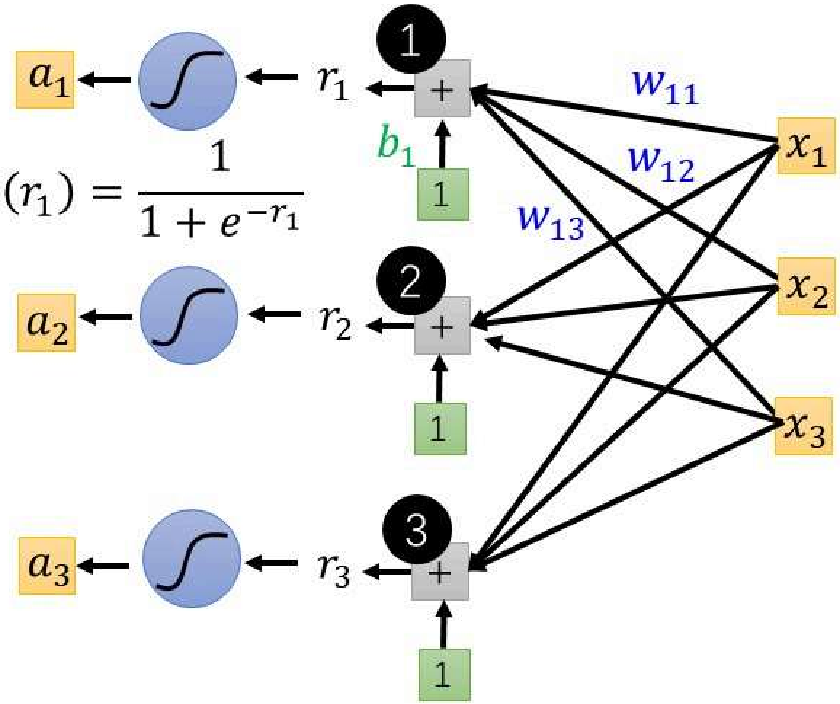
$$y = b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right) \qquad \begin{array}{l} i: 1,2,3 \\ j: 1,2,3 \end{array}$$

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



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

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

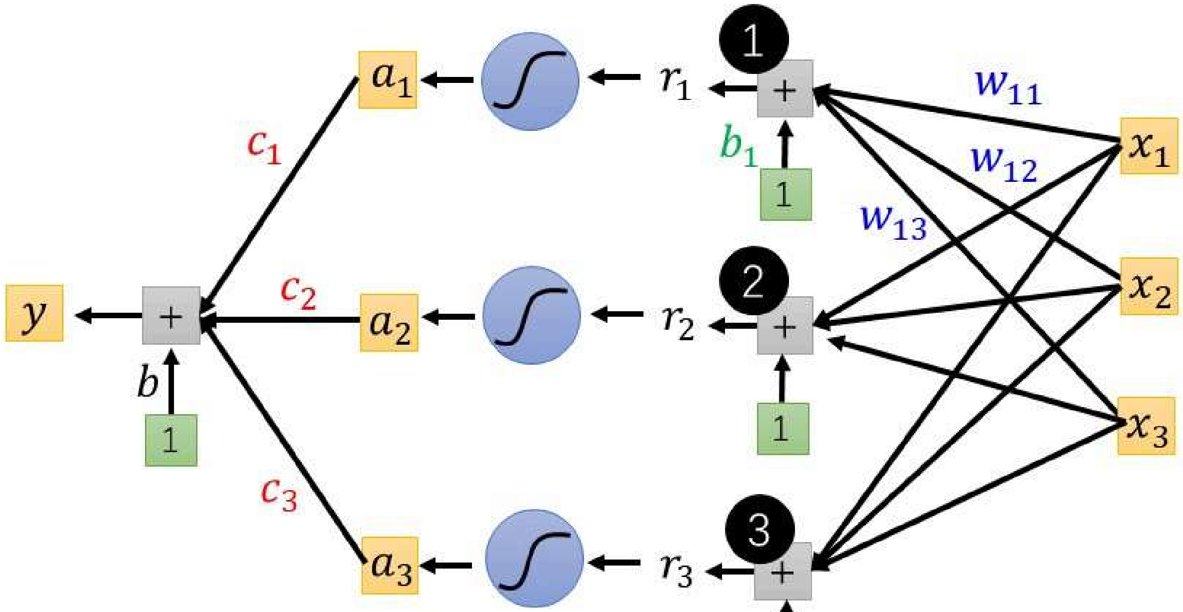


$$\boldsymbol{a} = \sigma(\boldsymbol{r})$$

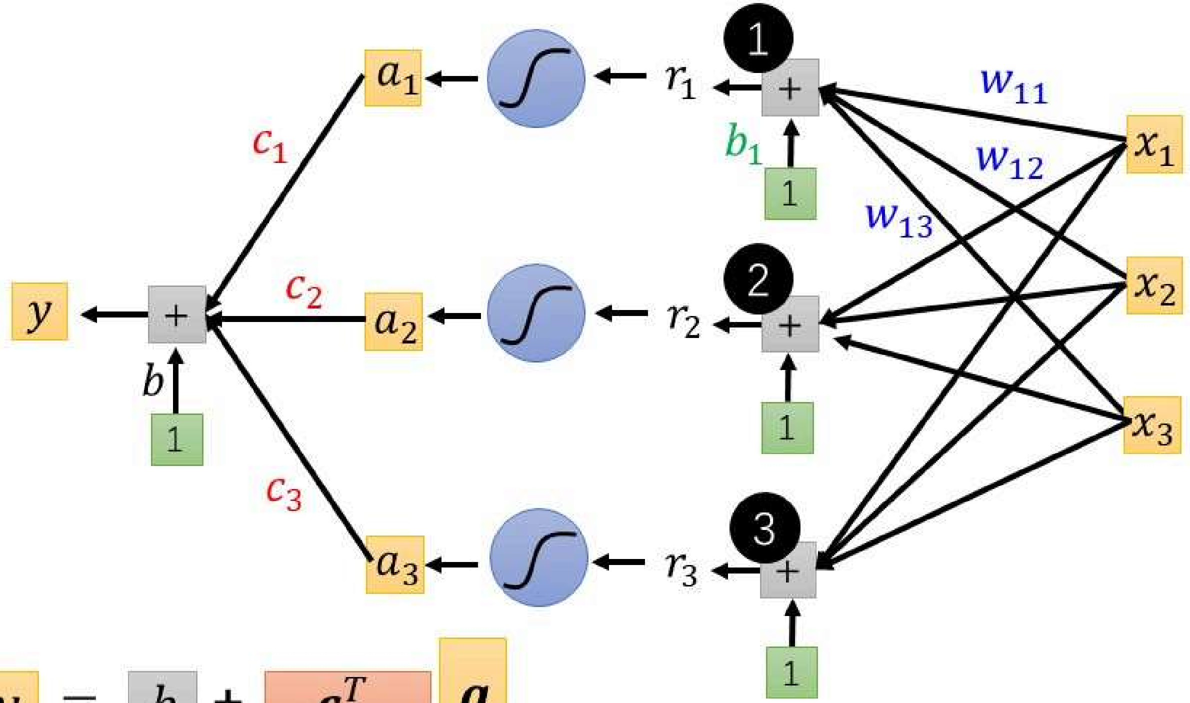
$$y = \boxed{b + \sum_i c_i \ sigmoid\left(b_i + \sum_j w_{ij} x_j\right)}$$

i: 1,2,3

j: 1,2,3

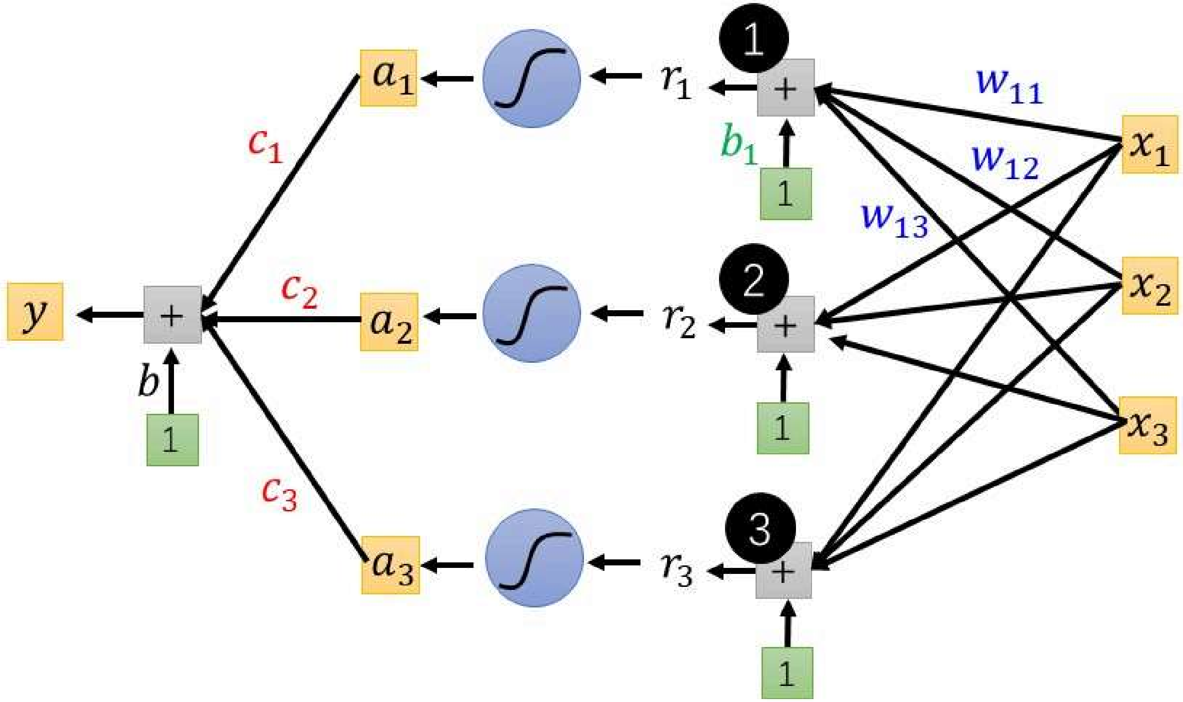


$$y = b + \boldsymbol{c}^T \boldsymbol{a}$$



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

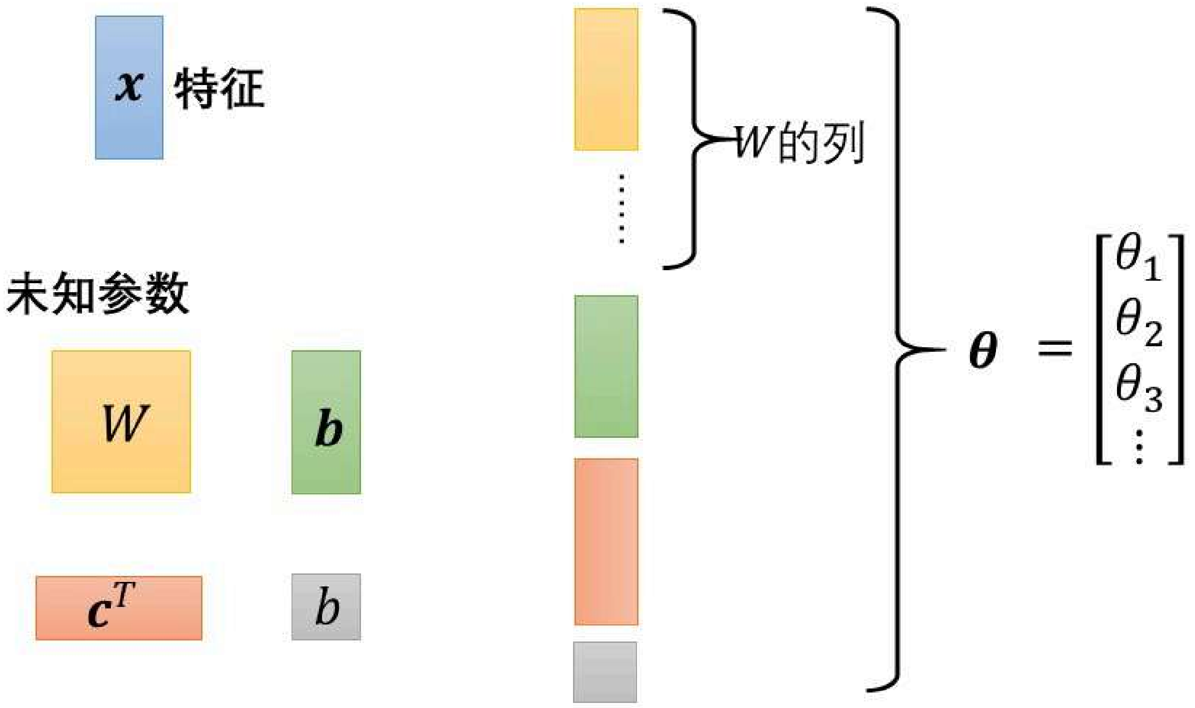
$$\boldsymbol{a} = \sigma(\boldsymbol{r}) \qquad \boldsymbol{r} = \boldsymbol{b} + W \boldsymbol{x}$$



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

带有未知参数的函数

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



回到 DL 框架

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

损失

Step 1: 定义参数模型 → Step 2: 定义损失 → Step 3: 优化

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

e

标签：Label $\hat{y}$

特征：feature

给定一套参数值

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

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

Back to DL Framework-优化

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

新模型的优化:Optimization of New Model

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

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

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

gradient

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

新模型的优化: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 $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^2)$

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

新模型的优化:Optimization of New Model

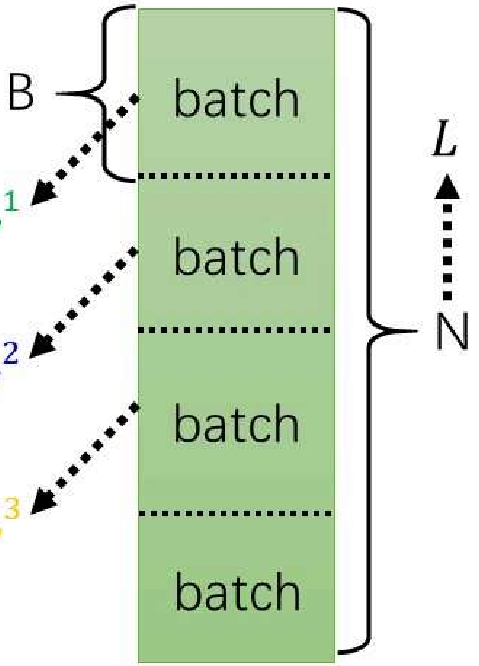
$$\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$$

- (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

新模型的优化:Optimization of New Model

例子 1

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

How many update in **1 epoch**?

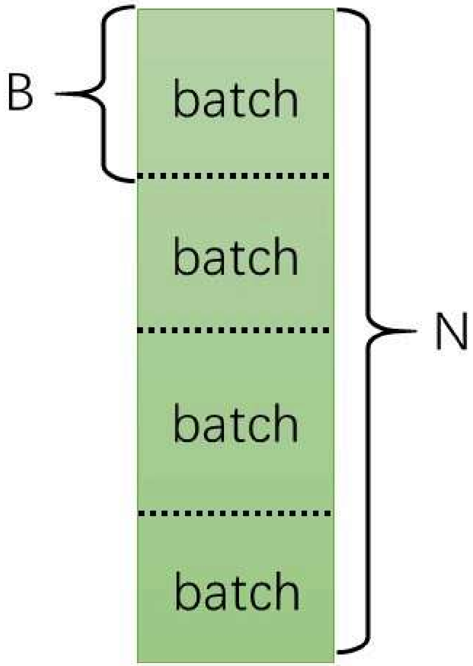
1,000 更新

例子 2

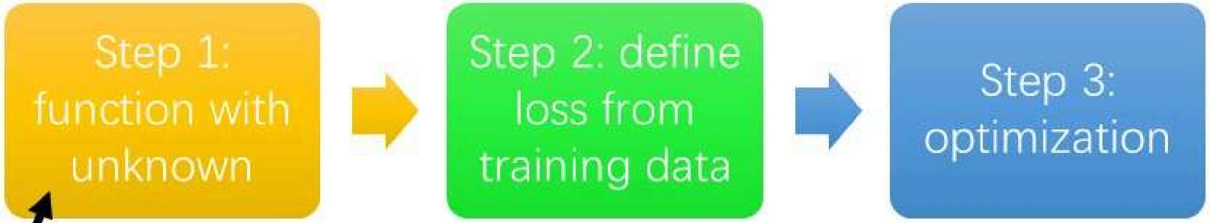
- 1,0000 examples (N = 1,0000)
- Batch size is 100 (B = 100)

How many update in **1 epoch**?

100updates



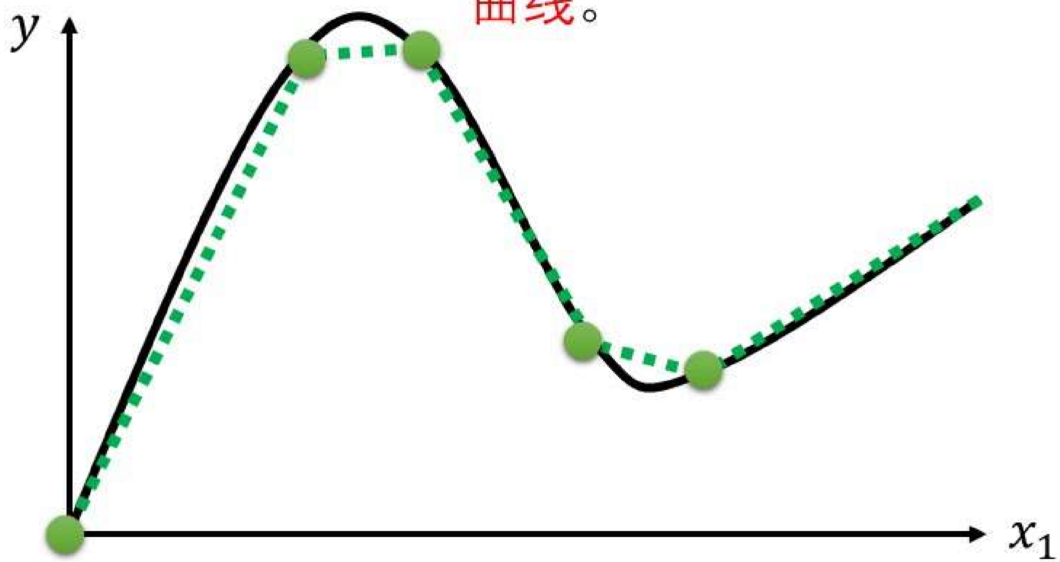
回到深度学习框架



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

很多的变形模型： More variety of models …

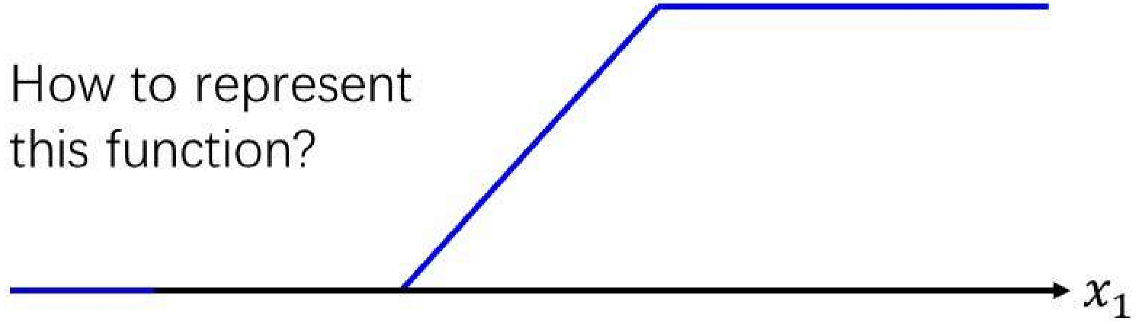
超越分段线性连续的的曲线?



为了获得良好的近似值，我们需要足够的分段。

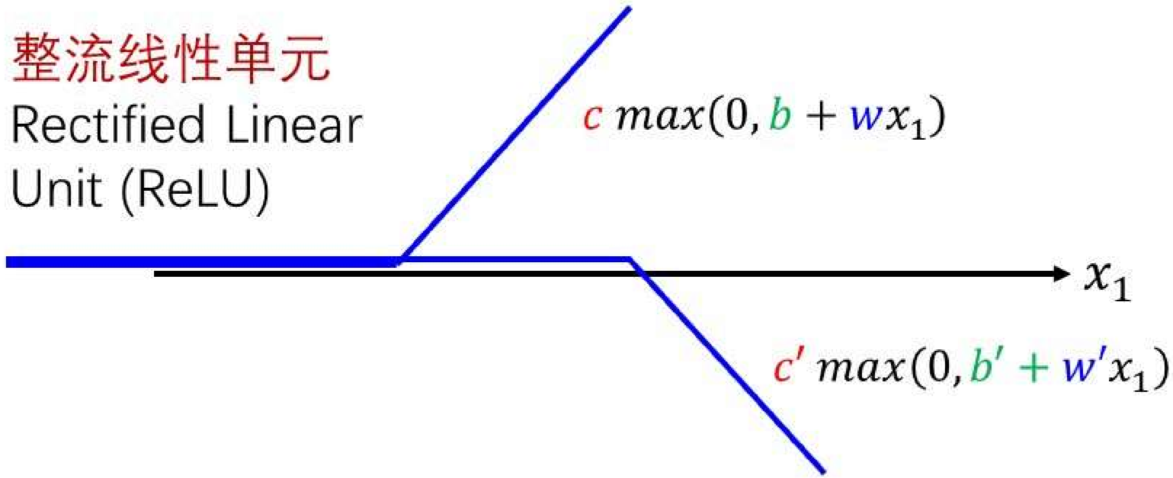
Sigmoid → ReLU

How to represent this function?



整流线性单元

Rectified Linear Unit (ReLU)



Sigmoid → ReLU

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

激活函数:

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?

实验结果:Experimental Results

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

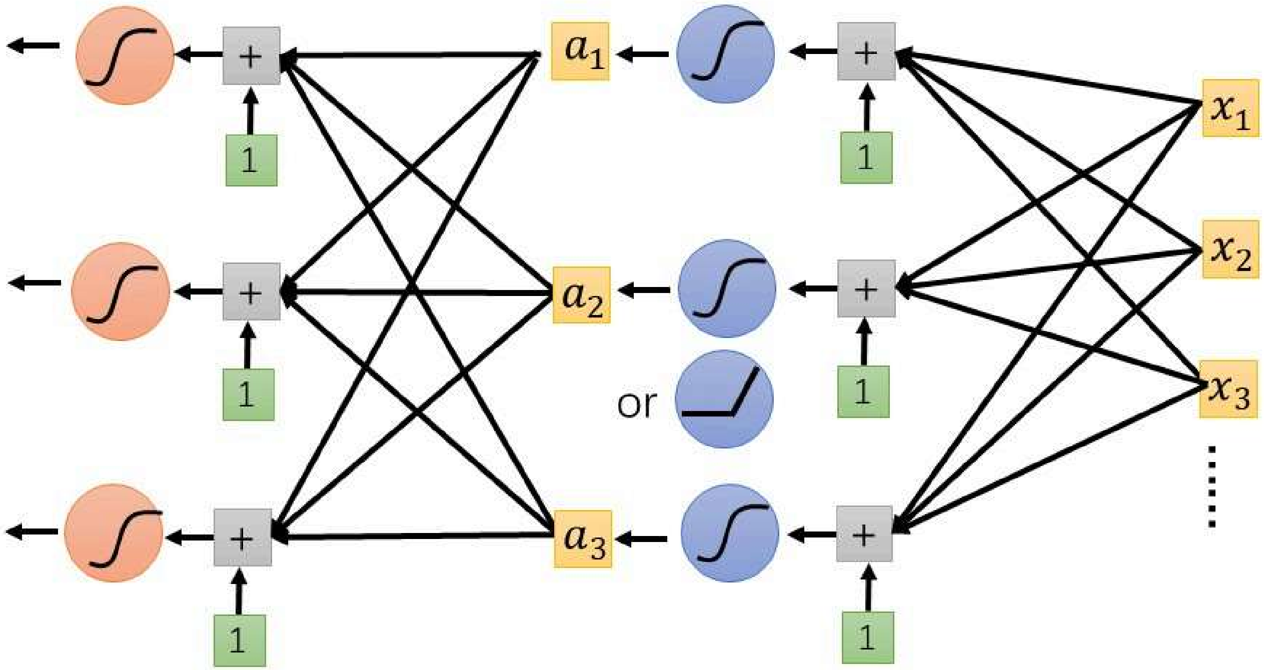
	linear
2017 – 2020	0.32k
2021	0.46k

回到深度学习框架

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 …



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

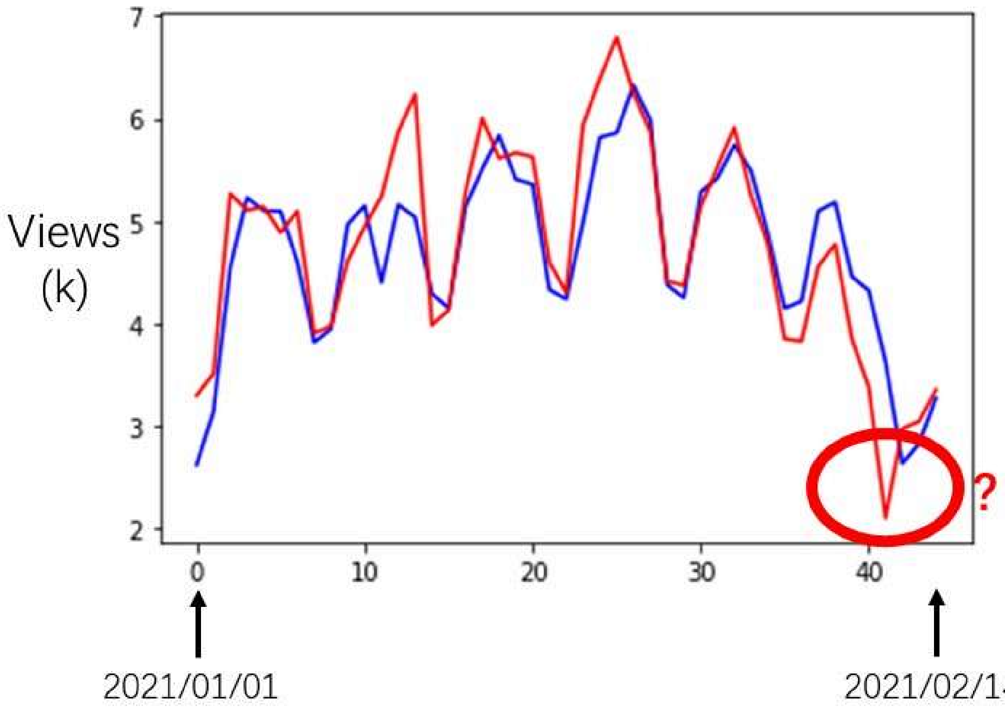
实验结果

- 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	[illegible]

3 layers

Red: 实际观看人数
blue: 预测观看人数



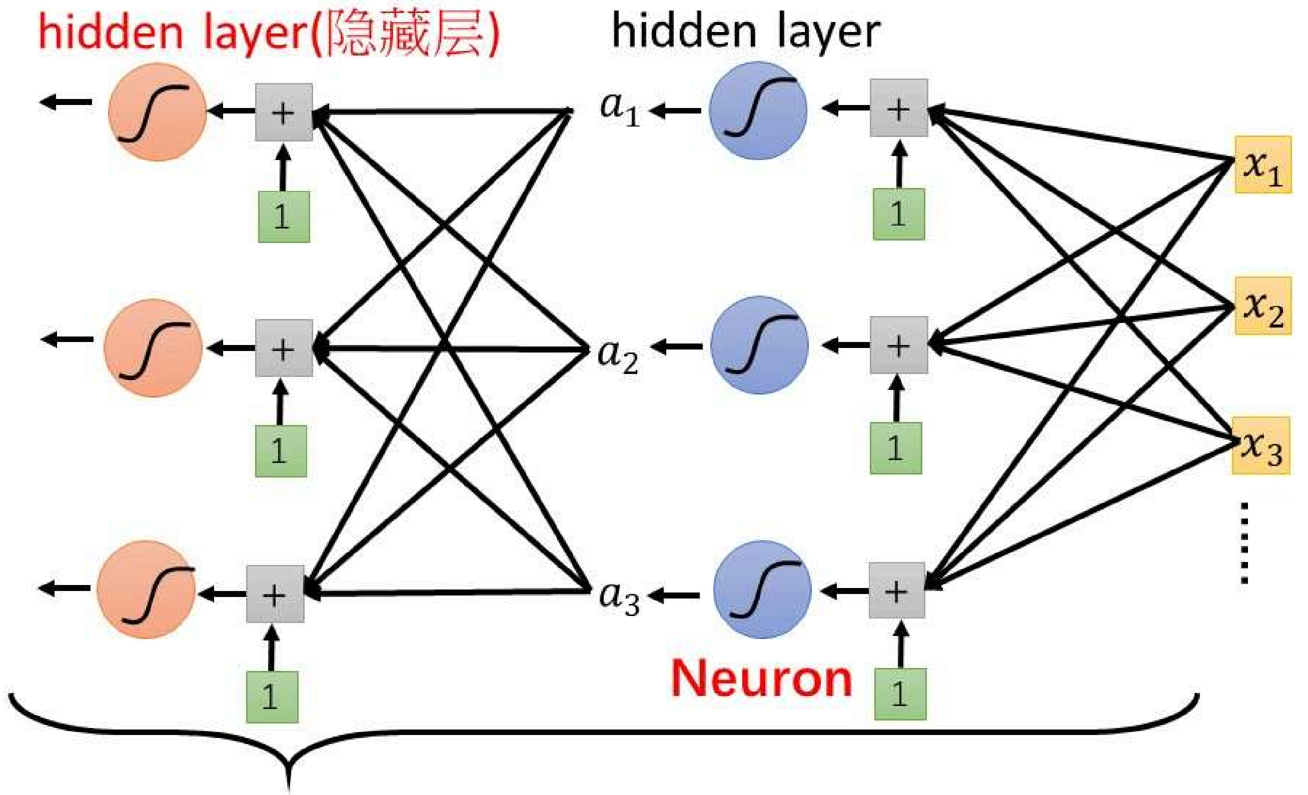
Back to DL Framework

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

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

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

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

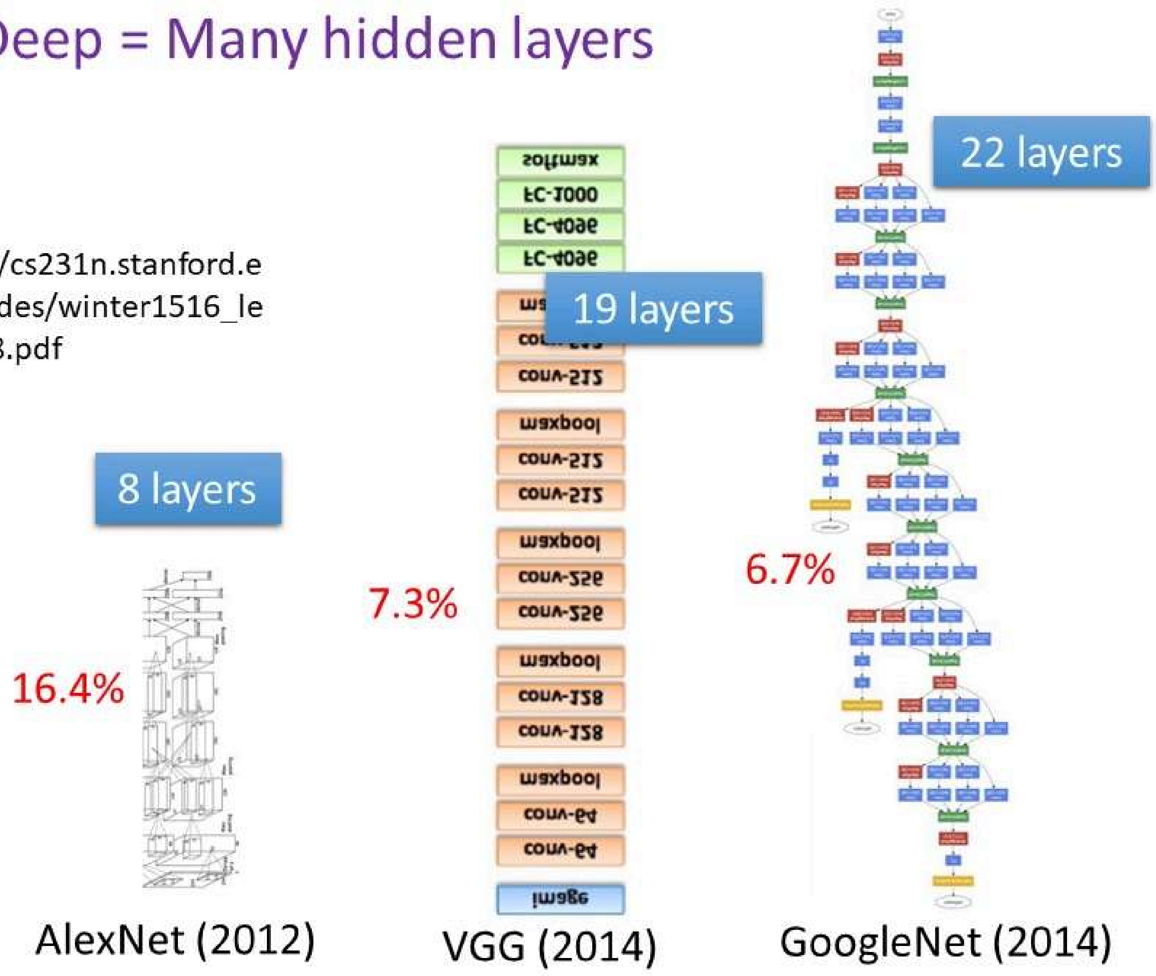


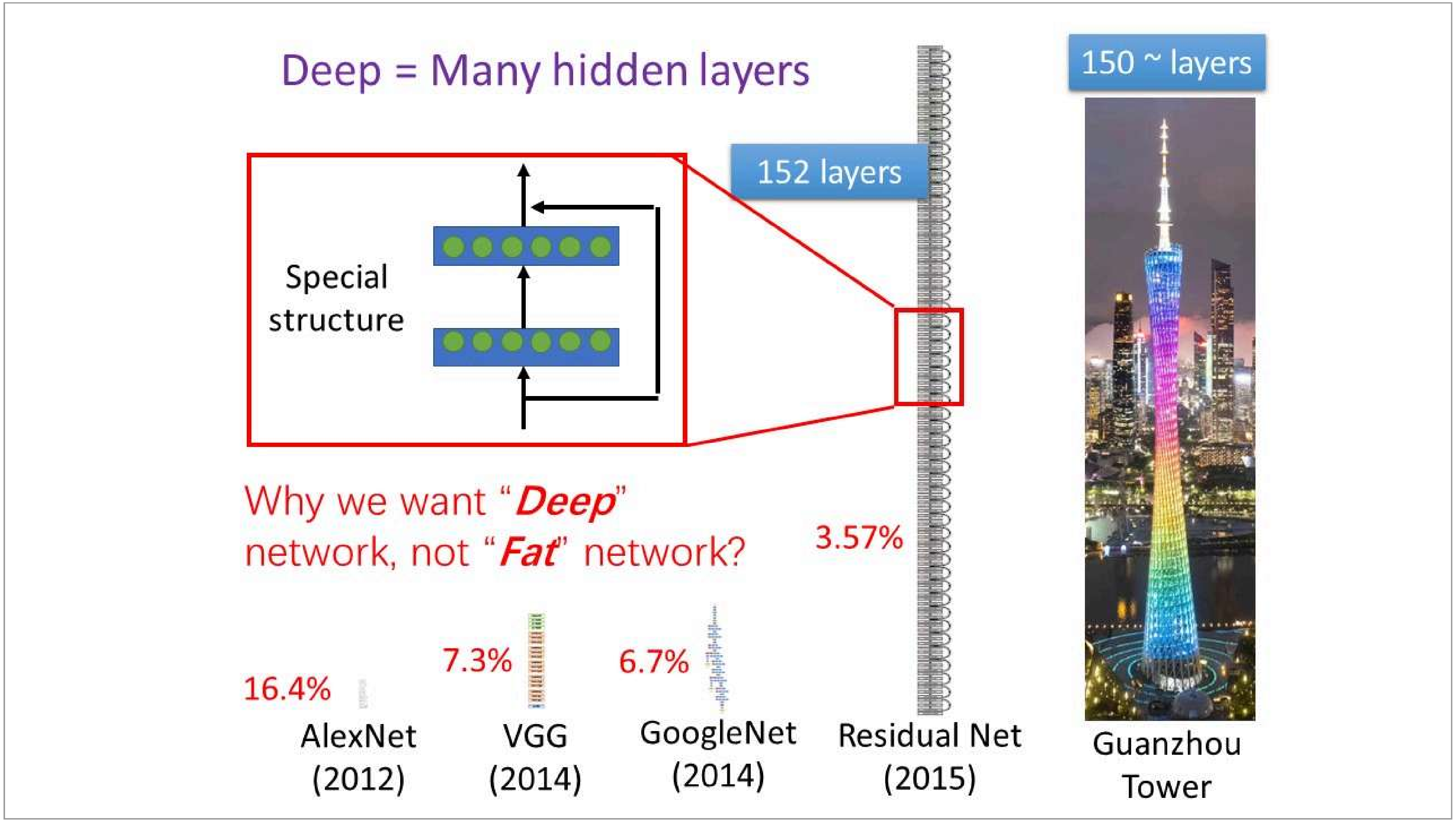
Neural Network This mimics human brains … (???)

Many layers means **Deep** ➡ Deep Learning

Deep = Many hidden layers

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





Deep = Many hidden layers
150 ~ layers
152 layers
Special structure
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)
Guanzhou Tower

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	0.44k

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. ☺