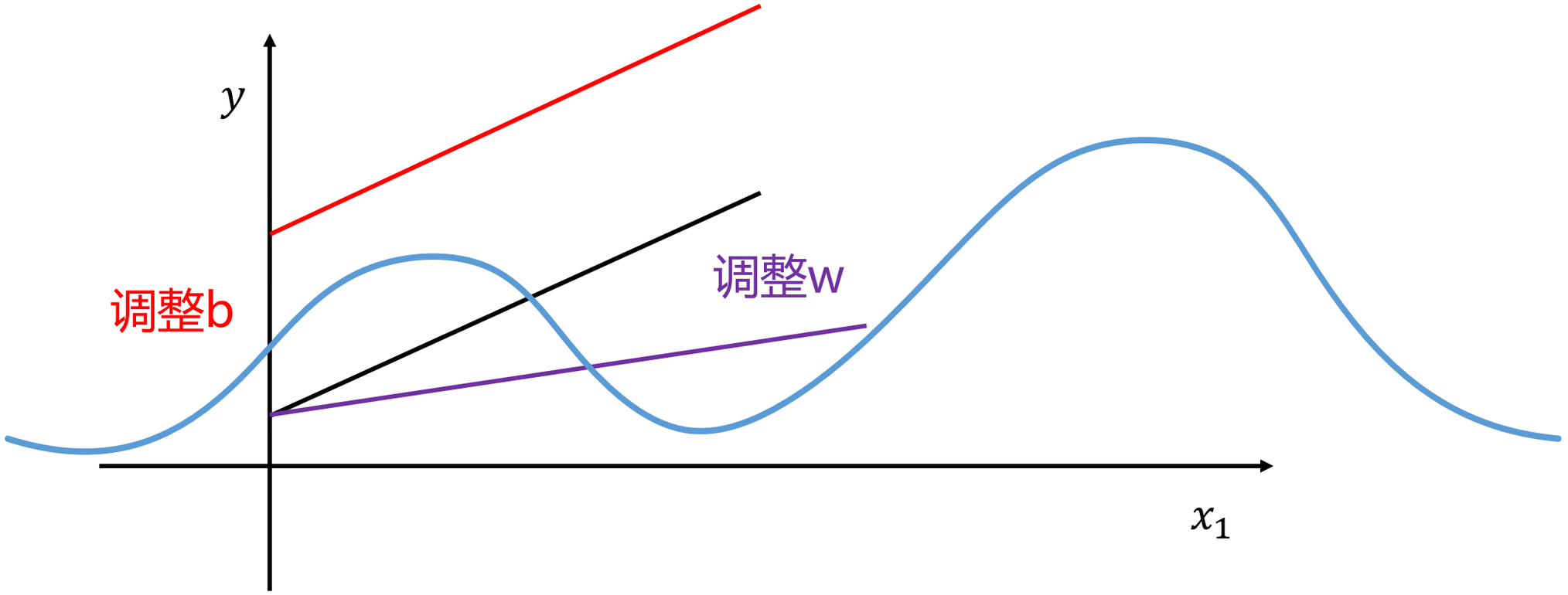
第八章

深度学习简介

线性模型的局限性

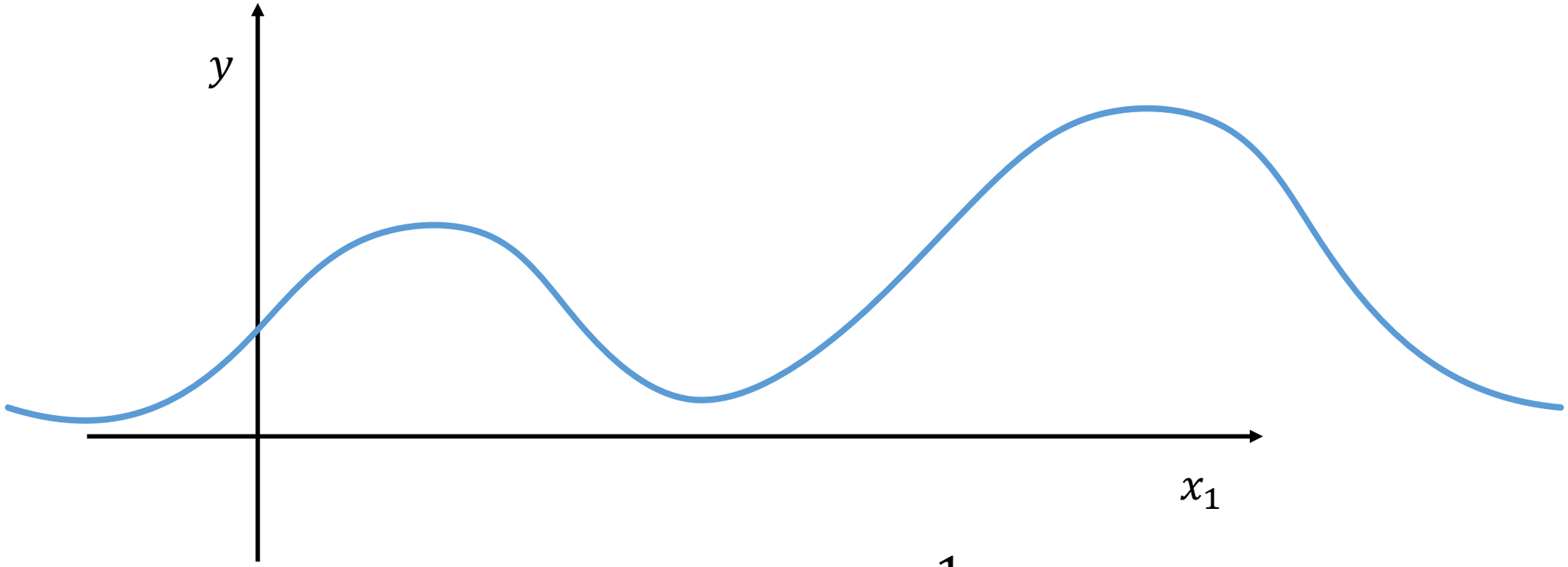
- **模型偏差 (model bias)**：模型过于简单，无法很好地拟合一些数据



我们需要一个更加“有弹性”的模型！

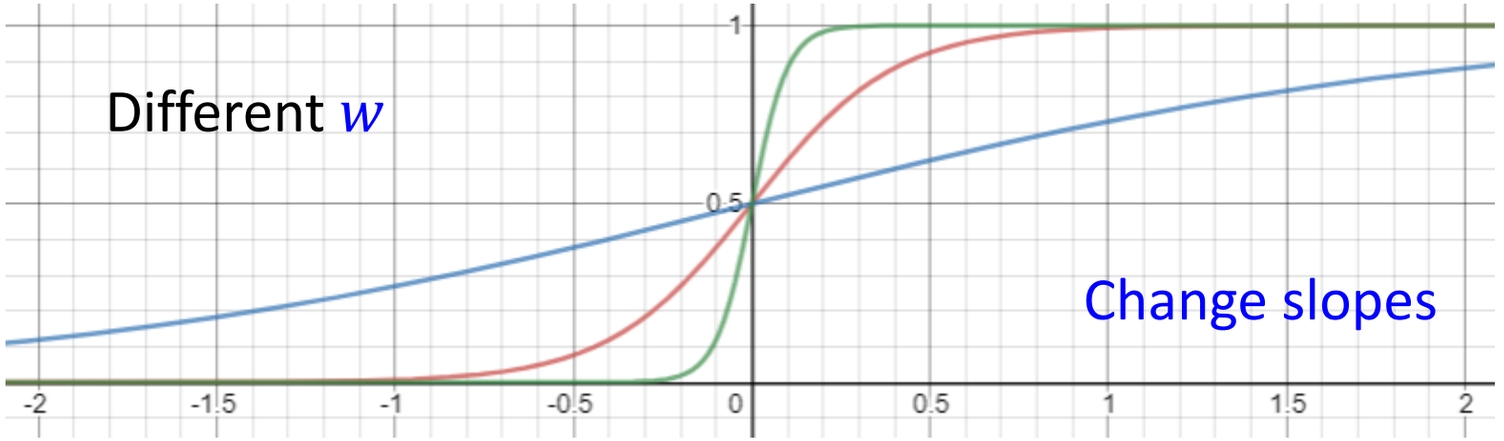
非线性模型

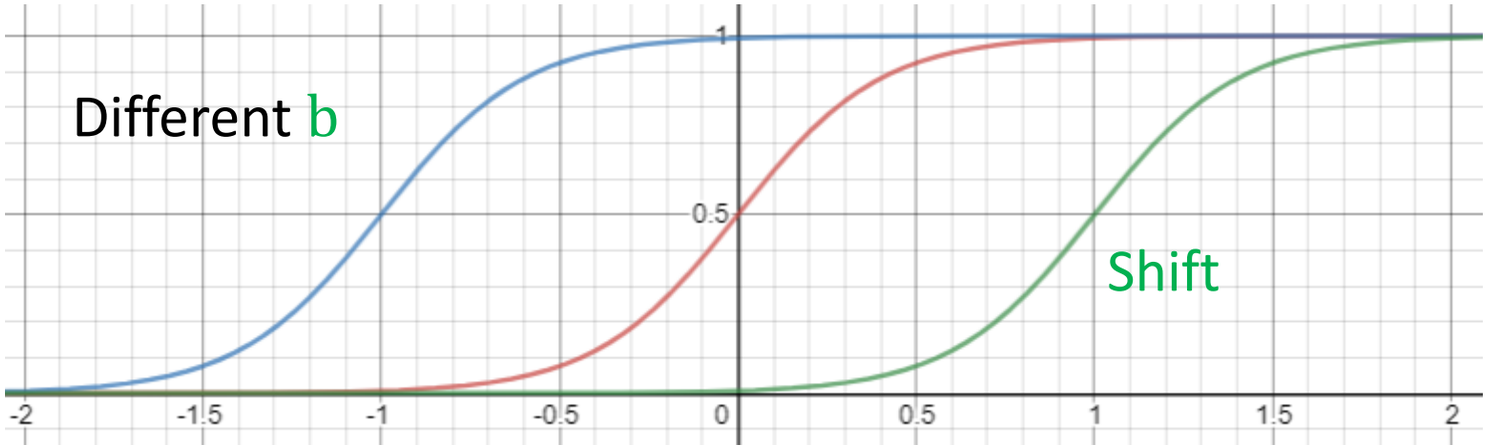
- **方法**：将原曲线分解成多个曲线的叠加

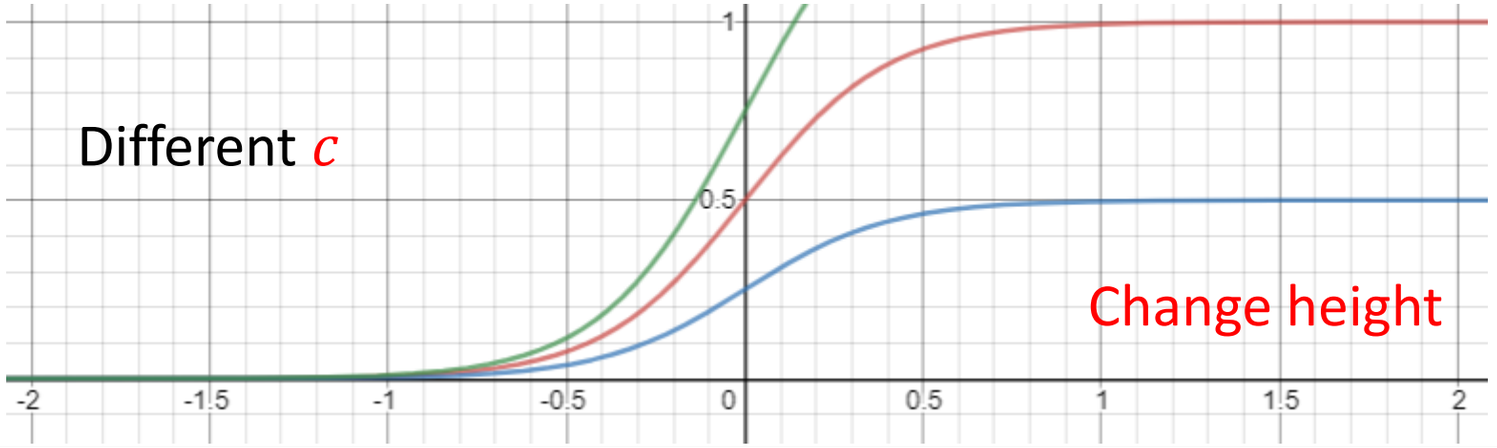


Sigmoid function: $y = c \cdot \frac{1}{1+e^{-(b+wx_1)}}$

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


Different w
Change slopes
1
0.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Different b
Shift
1
0.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Different c
Change height
1
0.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

非线性模型

线性模型：$y = b + wx_1$

非线性模型：$y = b + \sum_i c_i \, sigmoid(b_i + w_i x_1)$

多个特征的情况：$y = b + \sum_j w_j x_j$

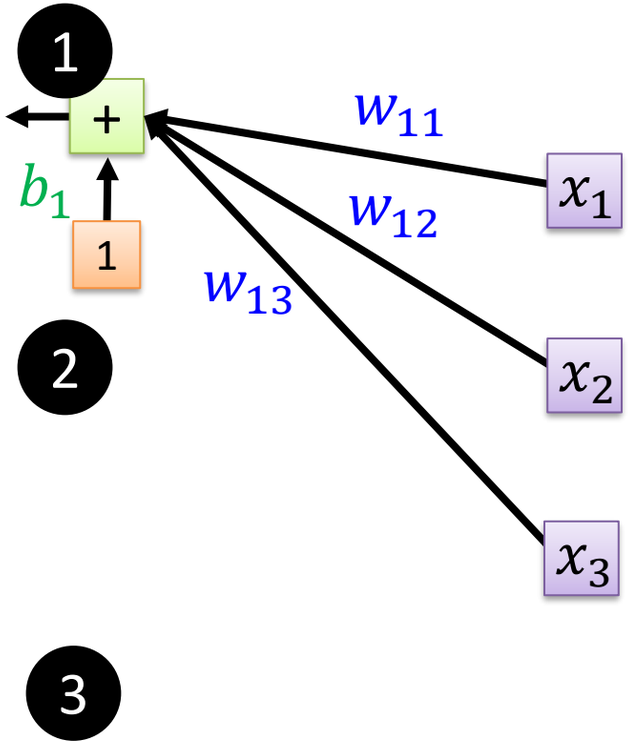
$$y = b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right)$$

神经网络 (neural network)

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

j: 特征的下标
i: sigmoid的下标

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



神经网络 (neural network)

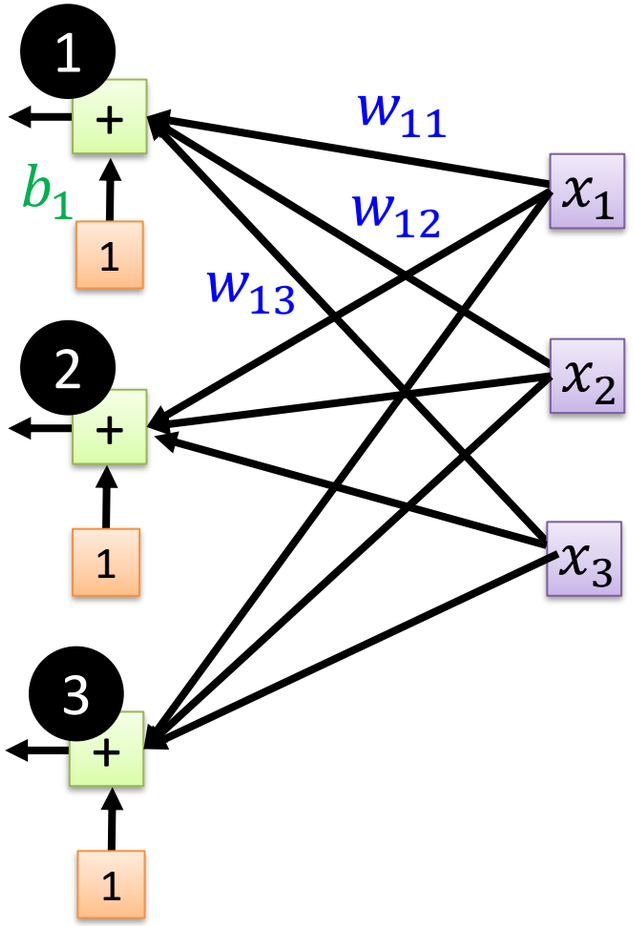
$$y = b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right)$$

j: 特征的下标
i: sigmoid的下标

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

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

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



神经网络 (neural network)

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

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

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

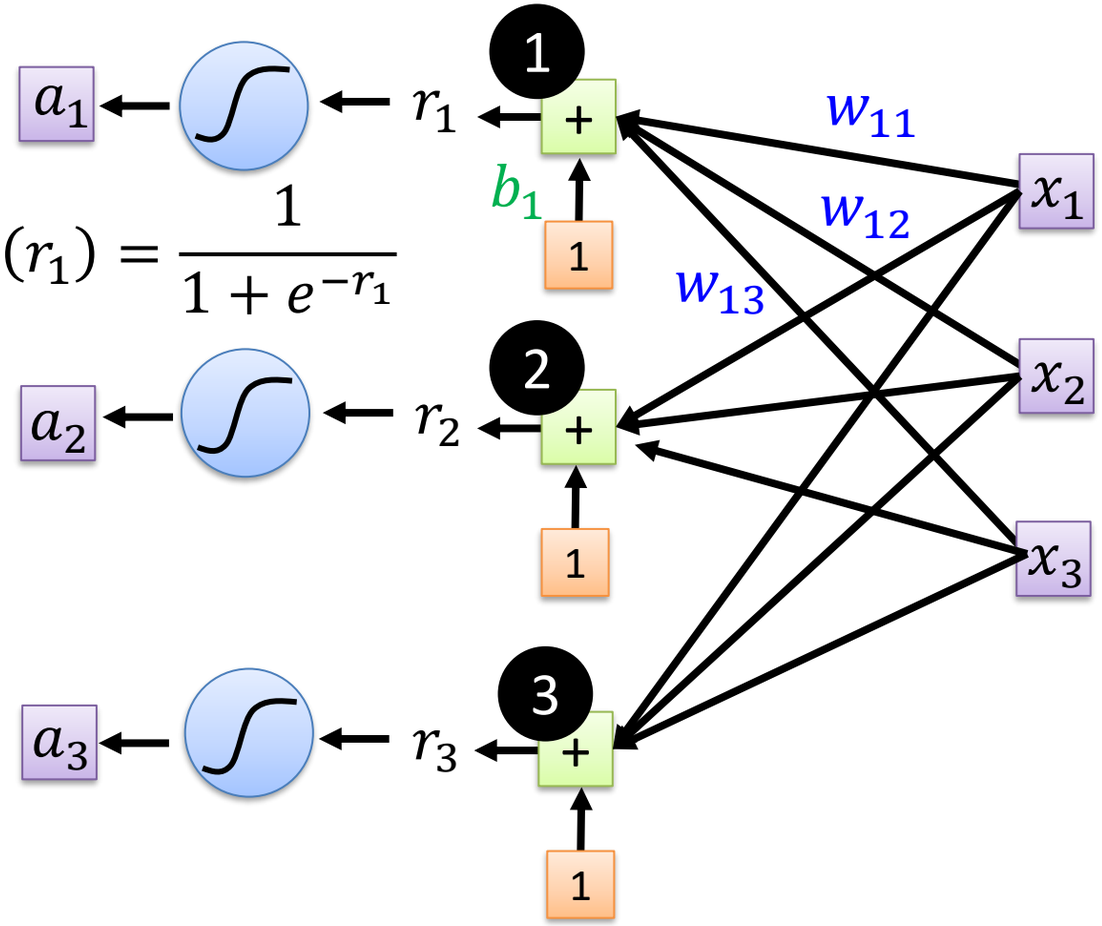
$$\boldsymbol{r} = \boldsymbol{b} + W \boldsymbol{x}$$

神经网络 (neural network)

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

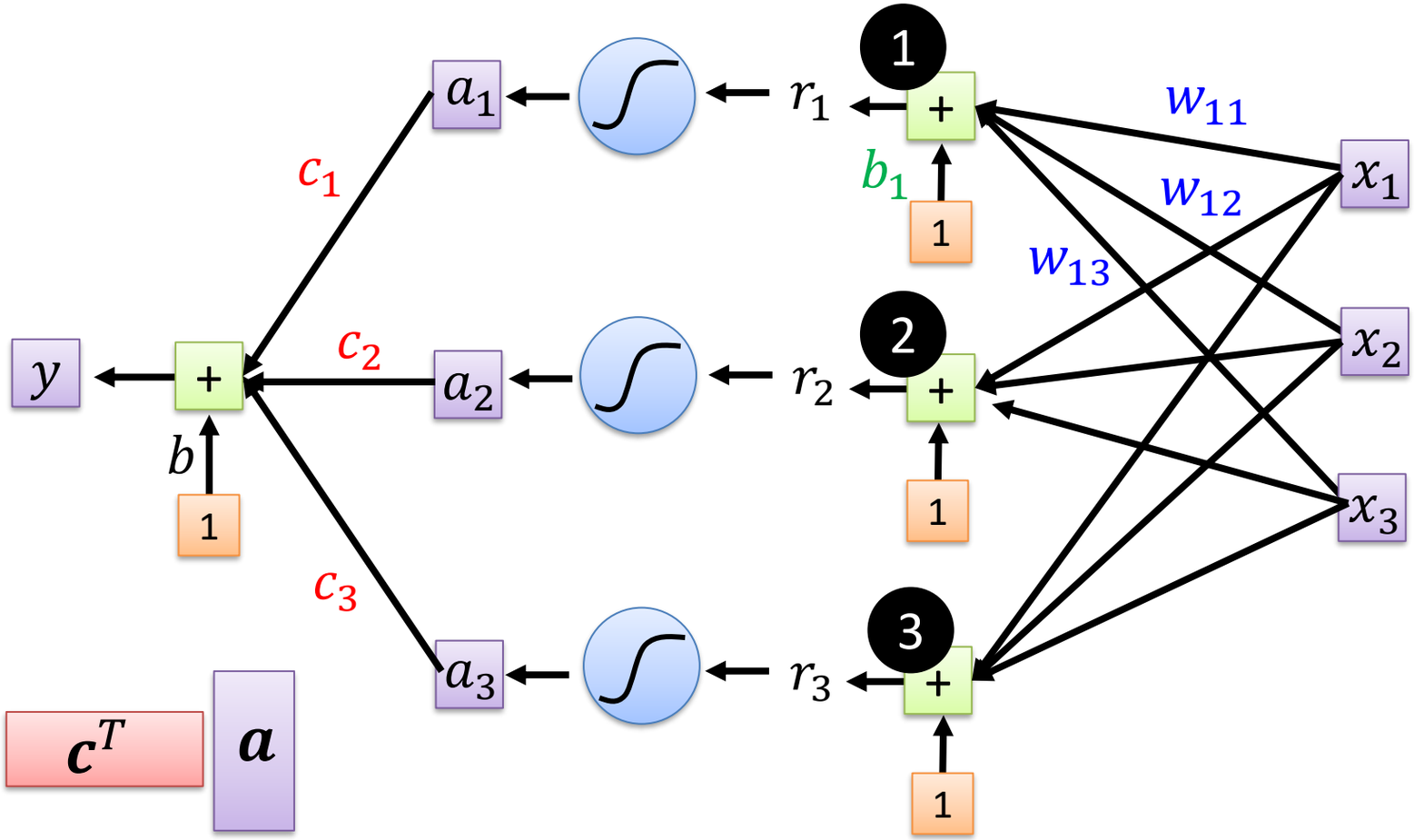
$$a_1 = sigmoid(r_1) = \frac{1}{1 + e^{-r_1}}$$

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



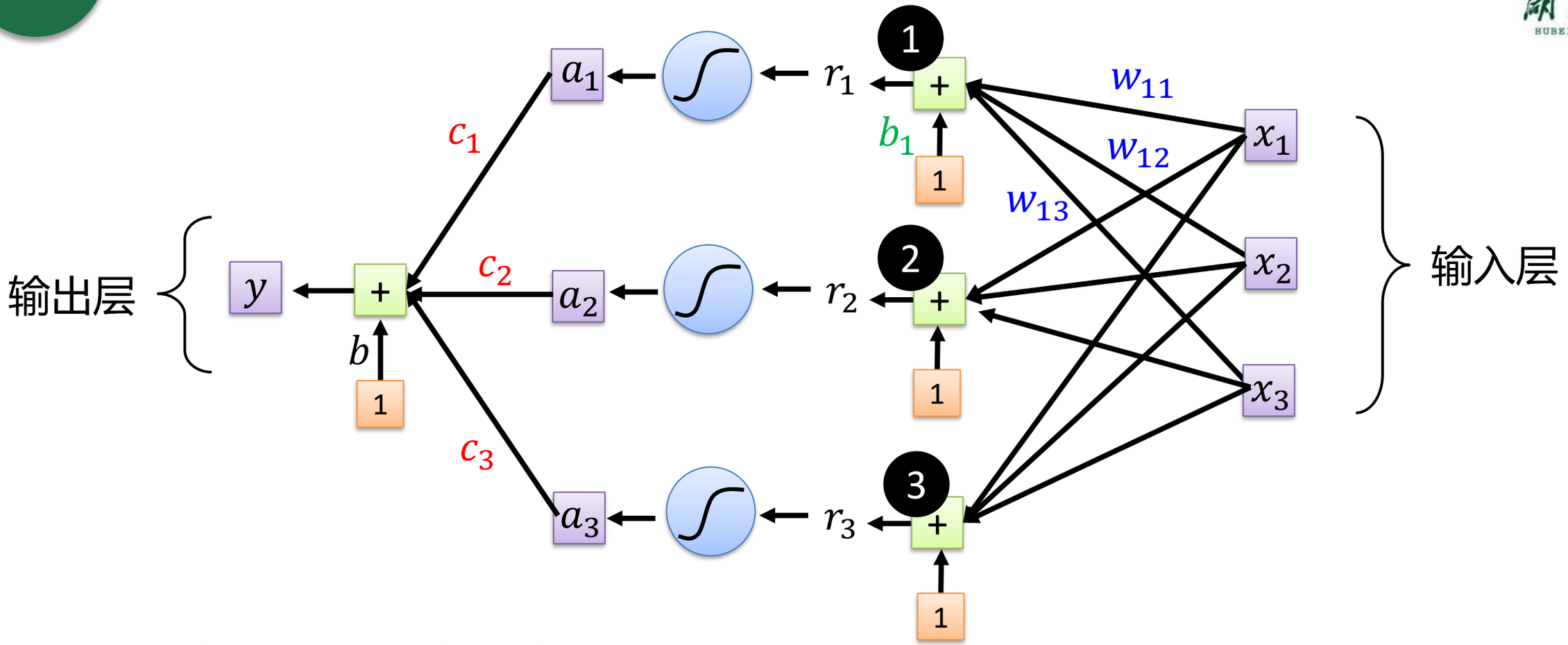
神经网络 (neural network)

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



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

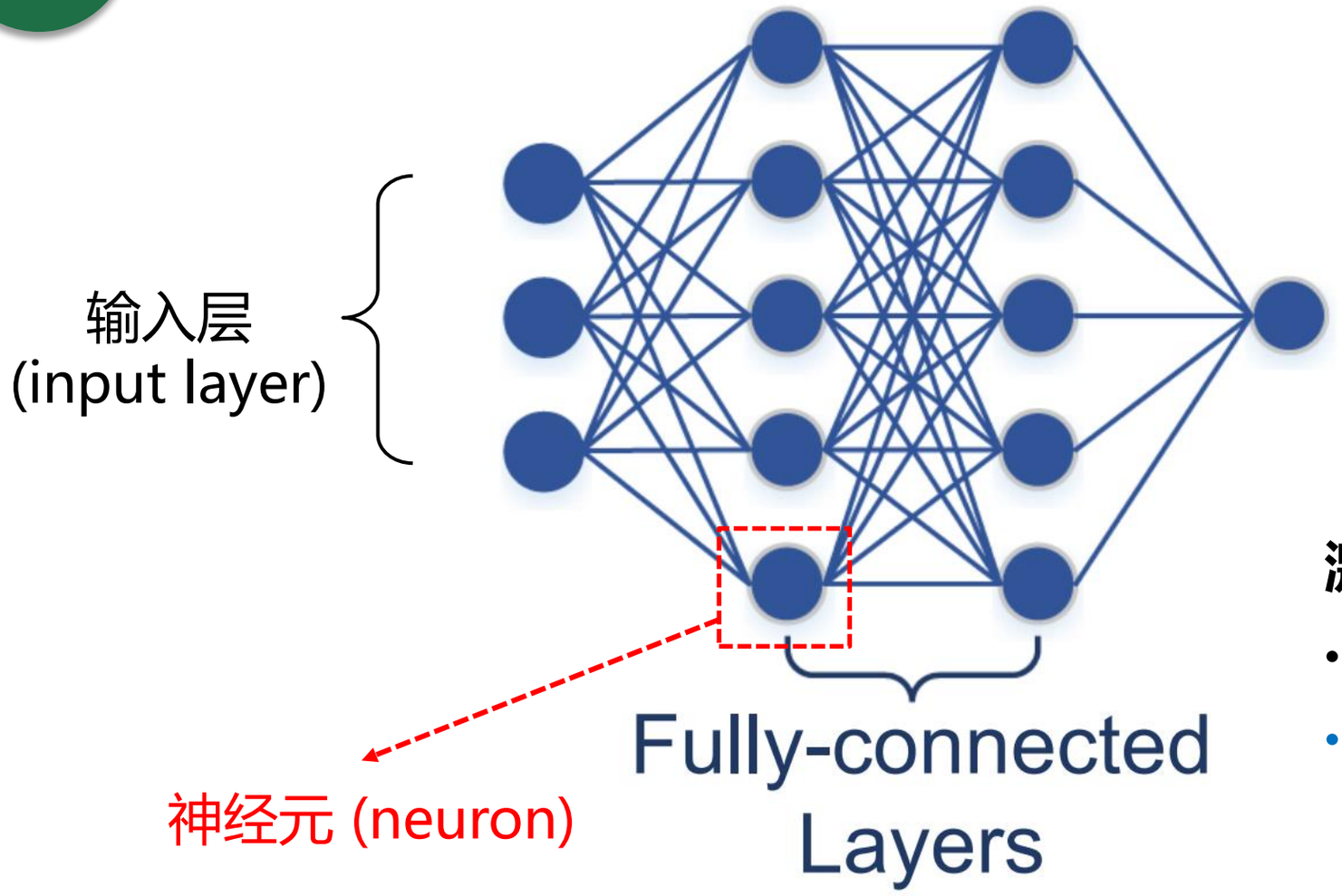
神经网络 (neural network)



包含1个全连接层的神经网络：

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

神经网络 (neural network)



激活函数 (activation function)：

- Sigmoid function
- 线性整流函数 (Rectified Linear Activation Function, ReLU)

包含2个全连接层的神经网络

神经网络的参数

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

x 特征

需要学习的未知参数

W b

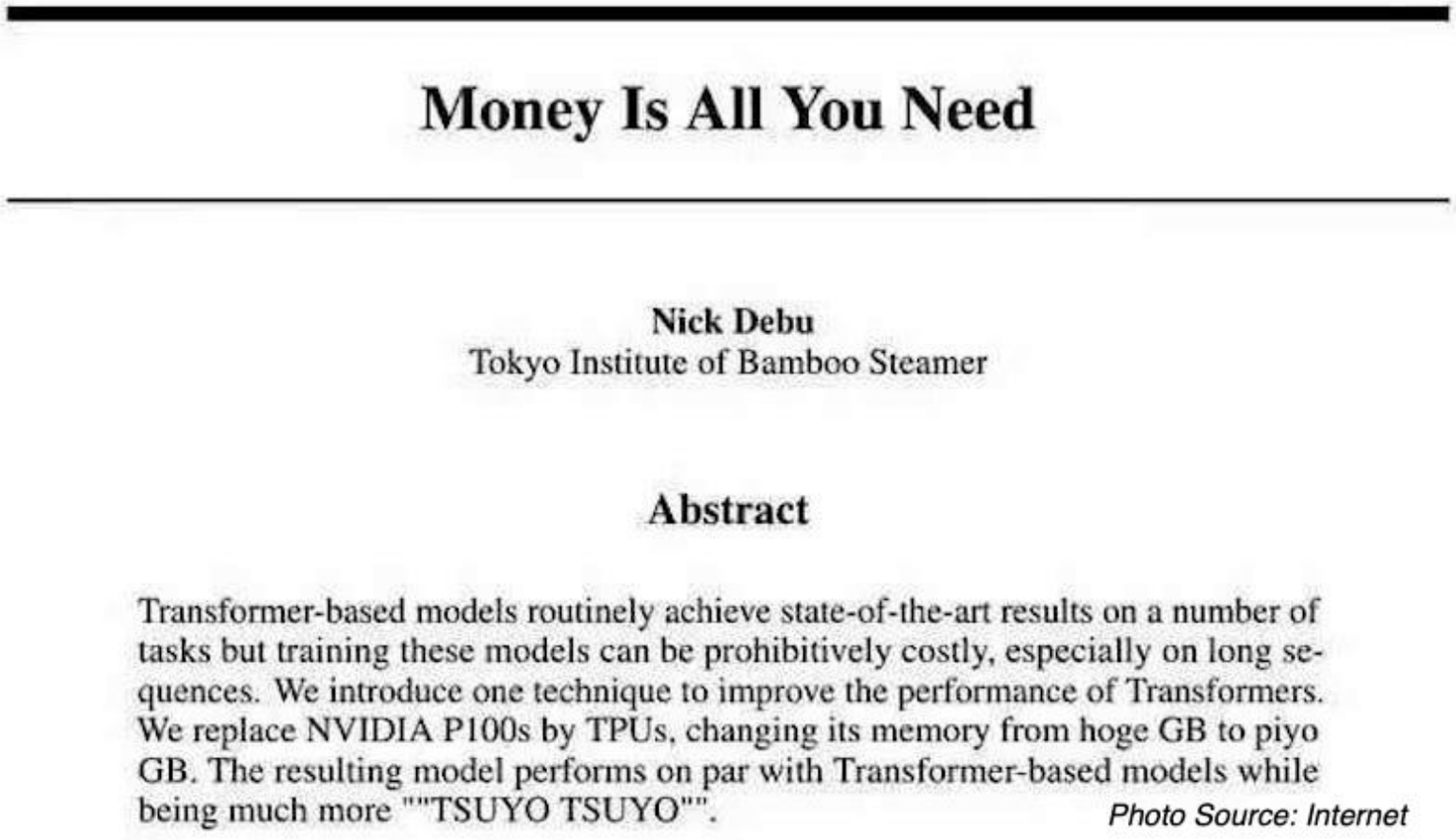
c^T b

W的所有行 (row)

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

神经网络的参数量非常大！

神经网络的参数



Money Is All You Need

Nick Debu
Tokyo Institute of Bamboo Steamer

Abstract

Transformer-based models routinely achieve state-of-the-art results on a number of tasks but training these models can be prohibitively costly, especially on long sequences. We introduce one technique to improve the performance of Transformers. We replace NVIDIA P100s by TPUs, changing its memory from hoge GB to piyo GB. The resulting model performs on par with Transformer-based models while being much more ""TSUYO TSUYO"".

Photo Source: Internet

- **大意：**基于Transformer的模型性能很好。在本文中，我们提出了一项改进Transformer性能的技术：把英伟达P100s替换成TPU，增加显存

机器学习的步骤

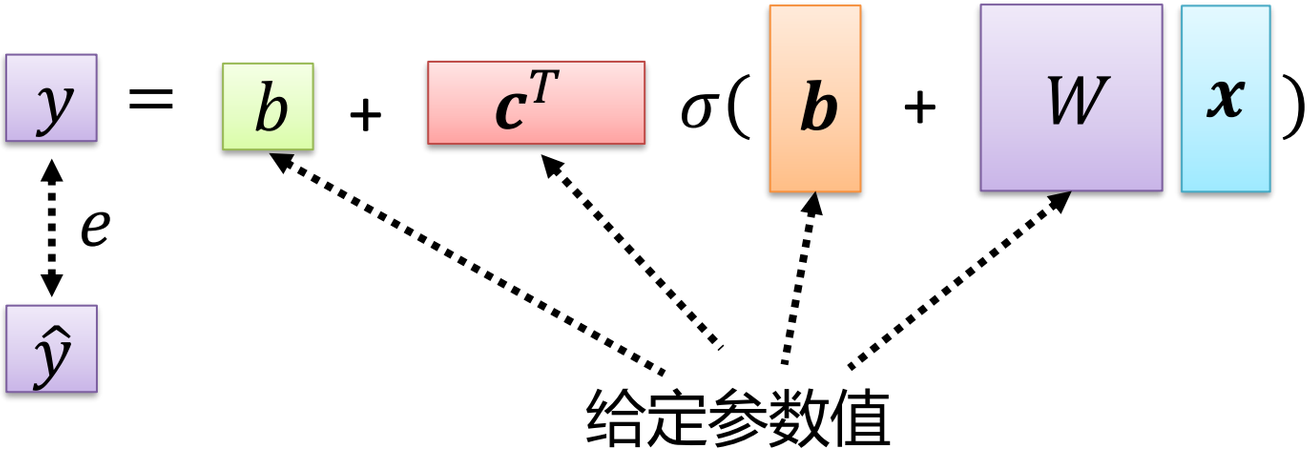
步骤一：
选择带有未知参数的函数

步骤二：
定义损失函数

步骤三：
优化

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

定义损失函数



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

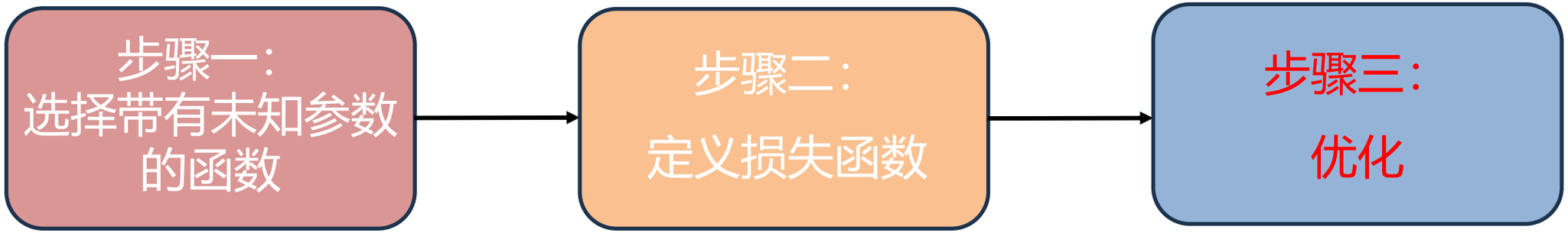
$e = |y - \hat{y}|$

平均绝对误差 (mean absolution error, MAE)

$e = (y - \hat{y})^2$

均方误差 (mean square error, MSE)

机器学习的步骤



神经网络的优化

所有未知参数：$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{bmatrix}$

优化目标：$\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$

✓ **梯度下降法 (gradient descent)**：随机初始化 $\boldsymbol{\theta}^0$

梯度的是一个向量，表示函数在该点处沿着该方向（此梯度的方向）变化最快，变化率最大。

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

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

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

神经网络的优化

优化目标： $\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$

✓ 随机初始化 $\boldsymbol{\theta}^0$

1. 计算梯度 $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^0)$

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

2. 计算梯度 $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^1)$

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

3. 计算梯度 $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^2)$

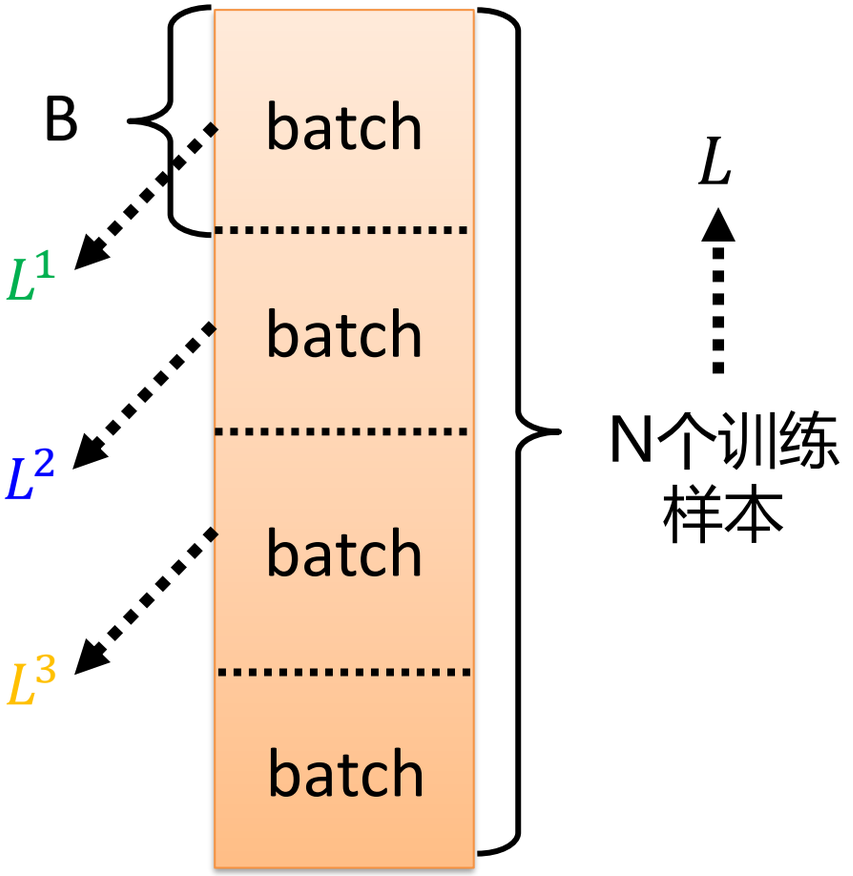
$$\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}$$

批处理 (batch training)

优化目标： $\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$

超参数 (hyperparameter)

批次 (batch)



✓ 随机初始化 $\boldsymbol{\theta}^0$

1. 计算梯度 $\boldsymbol{g} = \nabla L^1(\boldsymbol{\theta}^0)$

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

2. 计算梯度 $\boldsymbol{g} = \nabla L^2(\boldsymbol{\theta}^1)$

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

3. 计算梯度 $\boldsymbol{g} = \nabla L^3(\boldsymbol{\theta}^2)$

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

轮次 (epoch)

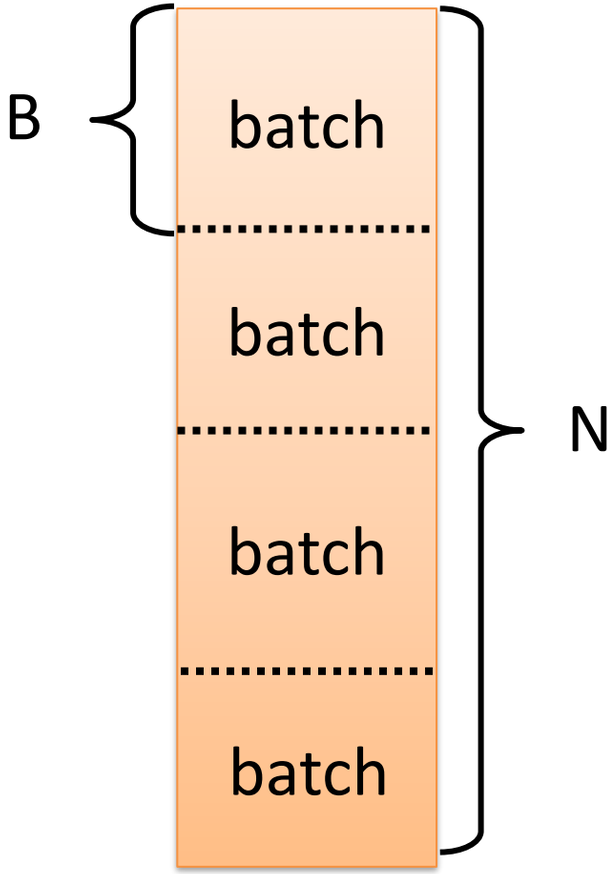
超参数 (hyperparameter)

Example 1

- 10,000 个训练样本 (N = 10,000)
- 批次大小为 10 (B = 10)

1个epoch中要更新多少次参数?

1,000 次



Example 2

- 10,000 个训练样本 (N = 10,00)
- 批次大小为 100 (B = 10)

1个epoch中要更新多少次参数?

10 次

机器学习的步骤

步骤一：
选择带有未知参数的函数

步骤二：
定义损失函数

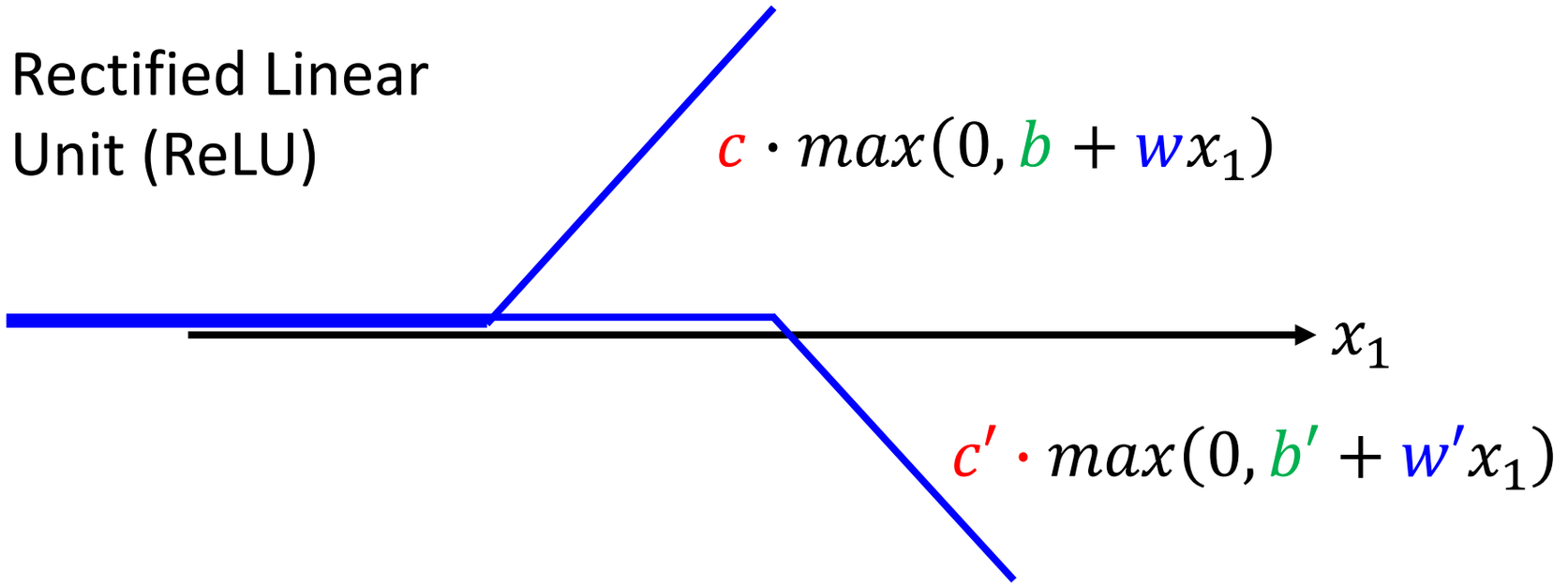
步骤三：
优化

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

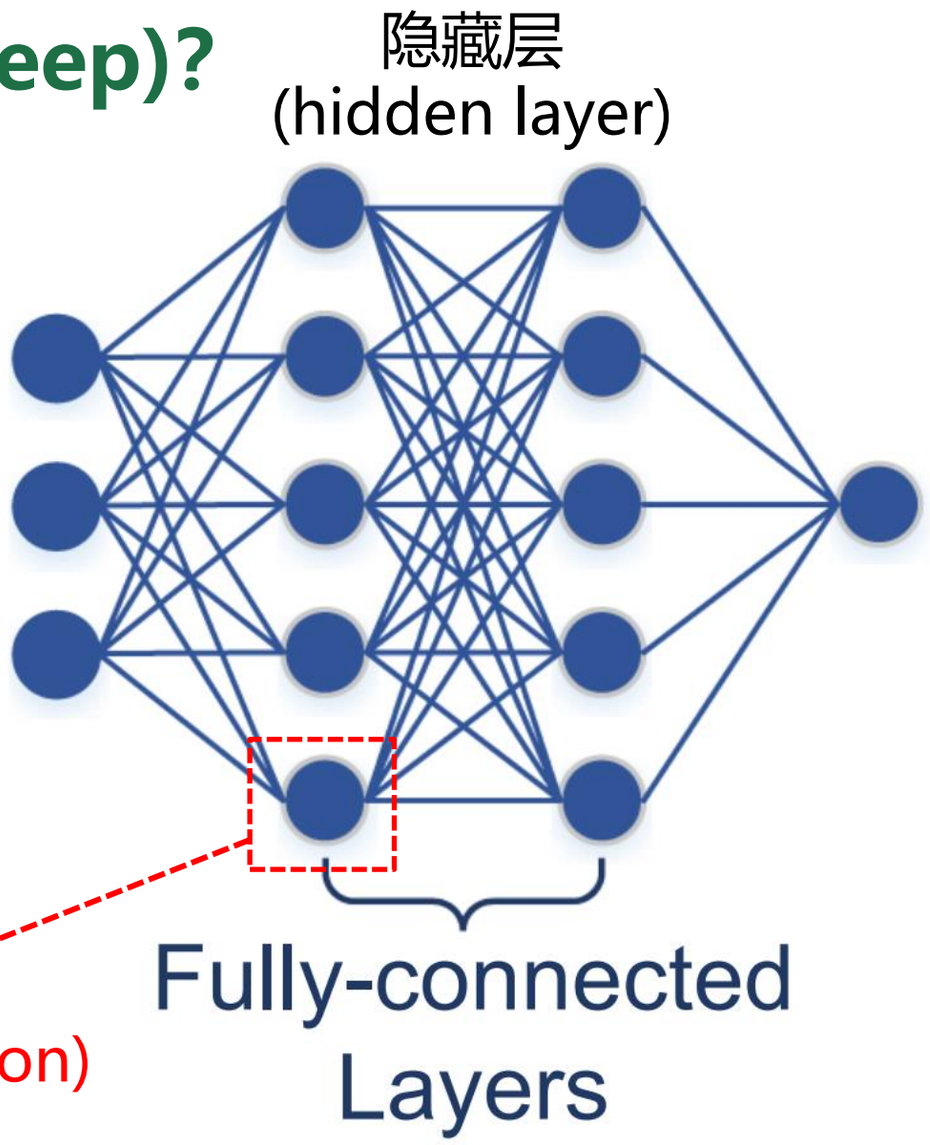
激活函数

激活函数 (activation function)：

- Sigmoid function
- 线性整流函数 (Rectified Linear Activation Function, ReLU)



深度 (deep)?



深度 (deep)：神经网络可以包含很多隐藏层 (hidden layer)

包含2个全连接层的神经网络

深度 (deep)?

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

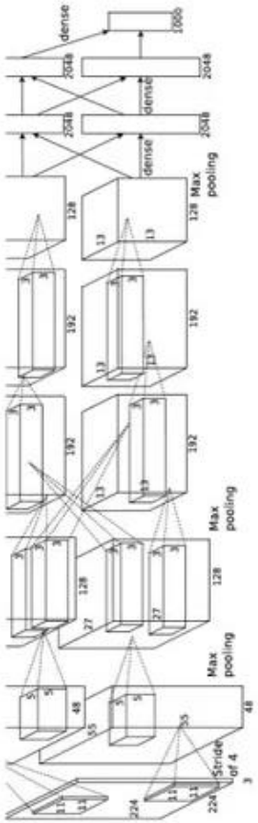
8 layers

19 layers

22 layers

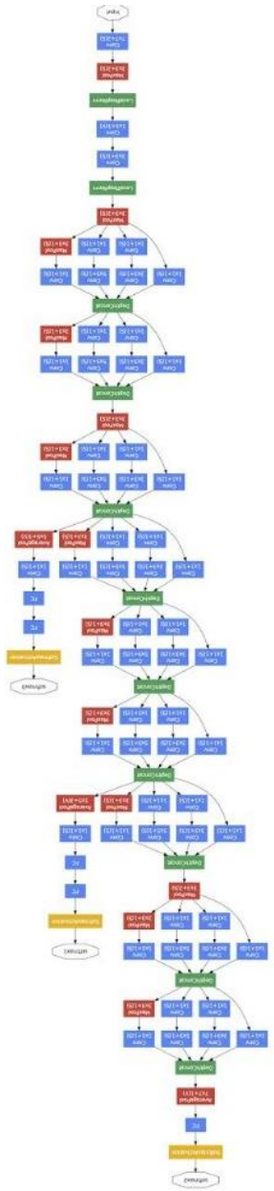
图像分类任务的错误率

16.4%



7.3%

6.7%



AlexNet (2012)

VGG (2014)

GoogleNet (2014)

小结

- 什么是机器学习 (machine learning)?
- 两种常见的函数 (function) 类型 / 任务 (task) 类型?
- 机器学习分哪三个步骤?
- 什么是线性模型 (linear model)?
- 什么是损失函数 (loss function)?
- 机器学习一般使用哪个优化算法，该优化算法的步骤是什么?
- 什么是学习率 (learning rate)?
- 什么是超参数 (hyperparameter)?
- 线性模型如何用向量 / 矩阵相乘的形式表示?

小结

- 什么是模型偏差 (model bias)?
- 什么是sigmoid function?
- 在神经网络中，什么是全连接层 / 输入层 / 输出层?
- 什么是激活函数 (activation function)?
- 什么是批次 (batch) 和轮次 (epoch)?
- 为什么叫“深度”学习?

结束语

湖北大学
HUBEI UNIVERSITY

谢 谢!