

# 人工智能概论

Introduction to Artificial Intelligence

## 第三章 神经网络基础



# 通知

1. 雨课堂使用问题：后续行课均会使用雨课堂完成签到和答题，**请同学们提前测试和调试好设备（包括雨课堂、手机/电脑电量充足等）。**
  - a. 签到：签到**仅通过雨课堂进行，且不可补签。**
  - b. PPT雨课堂问题回答：答题**通过雨课堂提交，且不可补交。**

考虑到部分同学可能遇到设备问题，我们允许有少量未签到或未提交答题的情况。具体来说，**一定次数内的未签到或未提交，对应成绩的满分不受影响。**
2. 课堂主动回答登记问题只允许**当节课后到前排联系助教进行现场线下学号登记**，课后线上联系助教登记一律不处理。
3. 课后联系助教问题
  - a. 课堂教学意见欢迎课后线上联系助教和老师。
  - b. 其余签到、雨课堂题目、回答问题登记等问题课后均不做处理。**

2

## 通知

请还没组队的同学抓紧联系有空位队伍的组长完成组队

如果**9.29 (周一) 晚上18点**还没有联系好队伍加入的话，  
我们会**随机分配**到第5周后（国庆后）有空位的队伍

## 课程回顾

---

# 预测 vs 分类

4

# 课程回顾

传统机器学习任务分为两大类：**预测和分类**

- **预测任务**→连续变量（房价、温度.....）

1. **线性回归**: 用**线性模型**建立特征与目标的关系，预测连续值

预测 -> 分类

- **分类任务**→离散类别（是否为垃圾邮件.....）

2. **逻辑回归**: 先用**线性模型预测**

连续值，再利用**Sigmoid函数**将结果映射到(0,1)区间，最后划分阈值后，可用于**二分类任务**

3. **贝叶斯网络**: 基于概率理论，利用**贝叶斯定理**来**分类**数据，并假设特征间相互独立以简化计算

4. **支持向量机**: 通过寻找一个最优超平面来**分类**数据，以**最大化不同类别数据点之间的间隔**

5. **K-邻近算法**: 通过计算待分类样本与训练集中**K个最近邻样本的相似性**来直接**分类**数据

## 课程回顾

---

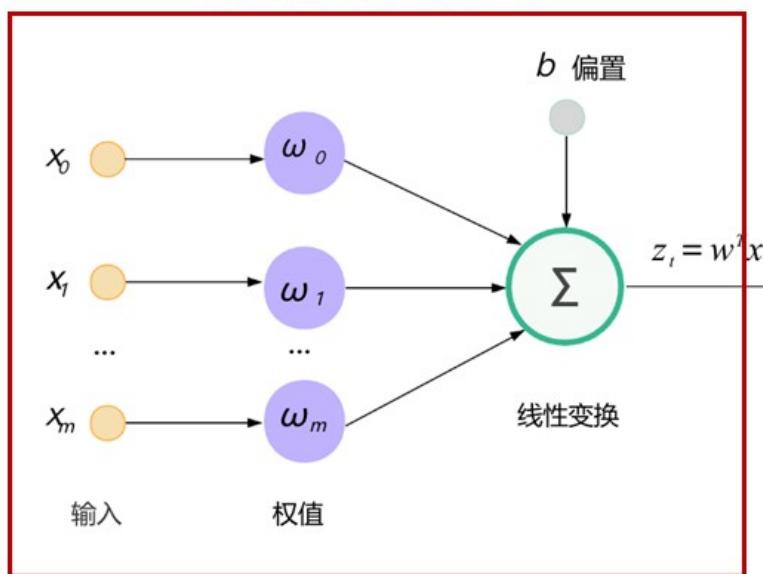
**线性回归和逻辑回归在模型结构上的不  
同点和相同点是什么？**

6

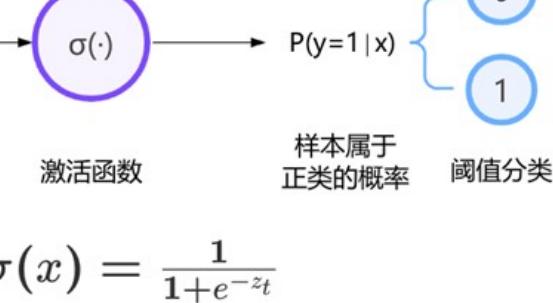
# 课程回顾

相同点：  
单层的线性模型

逻辑回归：



不同点：逻辑回归使用sigmoid激活  
函数将连输入信号映射为离散信号



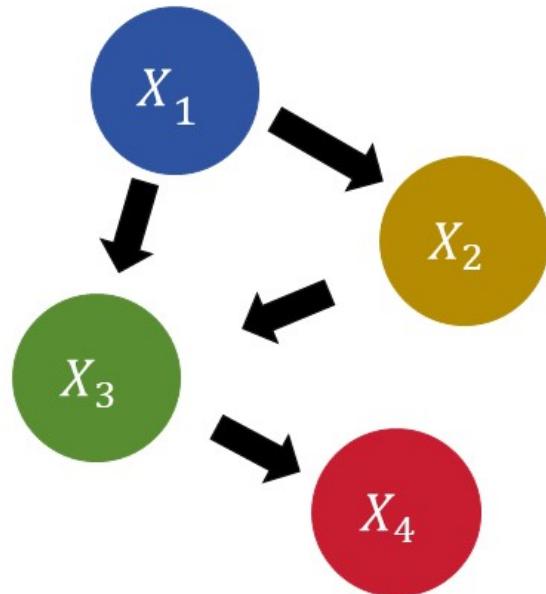
线性回归：

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b.$$

7

主观题 10分

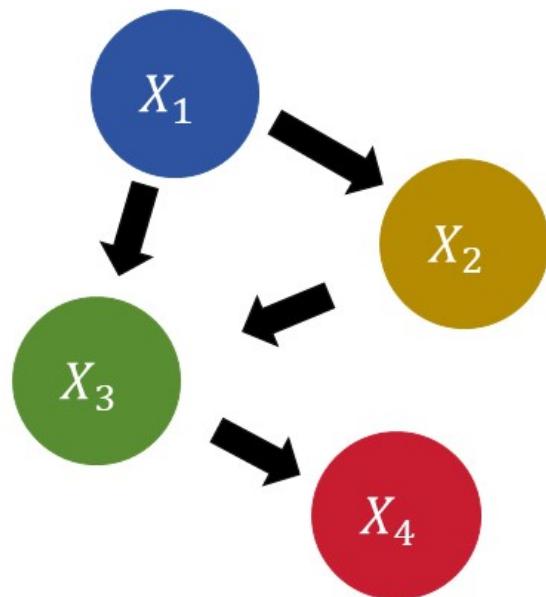
1. 根据左下图的贝叶斯网络， $x_1, x_2, x_3, x_4$ 的联合概率为？



$$p(x_1, x_2, x_3, x_4) = ? \text{ 请上传图片回答}$$

## 课堂回顾

根据左下图的贝叶斯网络， $x_1, x_2, x_3, x_4$ 的联合概率为？



$$p(x_1, x_2, x_3, x_4) =$$

$$p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_3)$$

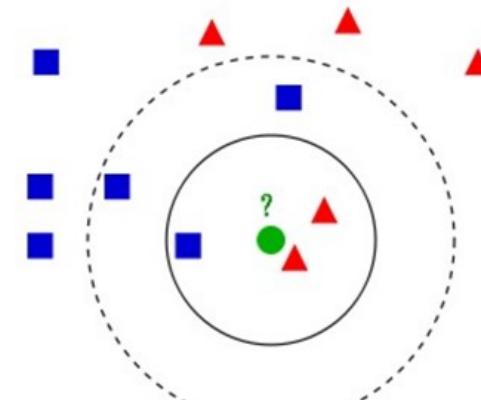
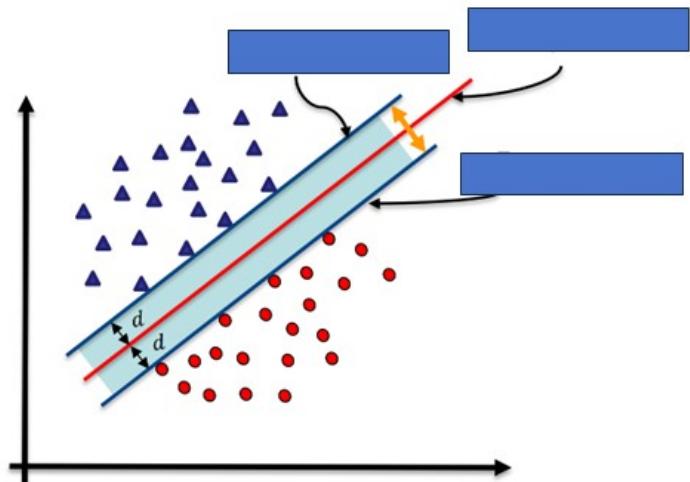
## 填空题 5分

### 2. SVM

(1) 任意超平面可以用什么线性方程来描述? [填空1]

(2) 二维空间点  $(x, y)$  到直线  $Ax + By + C = 0$  的距离是? [填空2]

3. KNN的三要素是 [填空3] , [填空4] , [填空5] 。  
(近朱者赤, 近墨者黑)



## 课堂回顾

### 支持向量机概述

任意超平面可以用下面这个线性方程来描述：

$$w^T x + b = 0$$

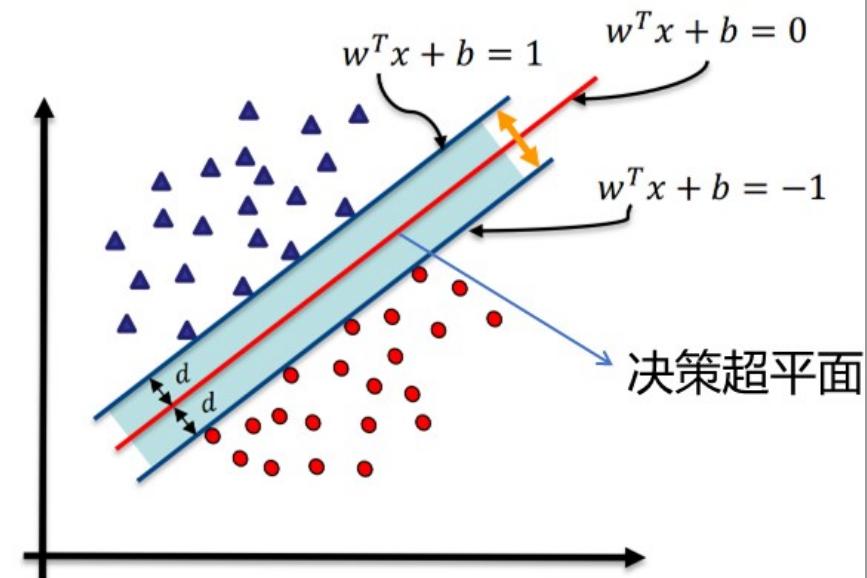
二维空间点  $(x, y)$  到直线  $Ax + By + C = 0$  的距离是：

$$\frac{|Ax + By + C|}{\sqrt{A^2 + B^2}}$$

扩展到  $n$  维空间后，

点  $x = x^{(1)}, x^{(2)}, \dots, x^{(n)}$  到超平面  $w^T x + b = 0$  的距离为： $\frac{|w^T x + b|}{\|w\|}$

其中， $\|w\| = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2}$

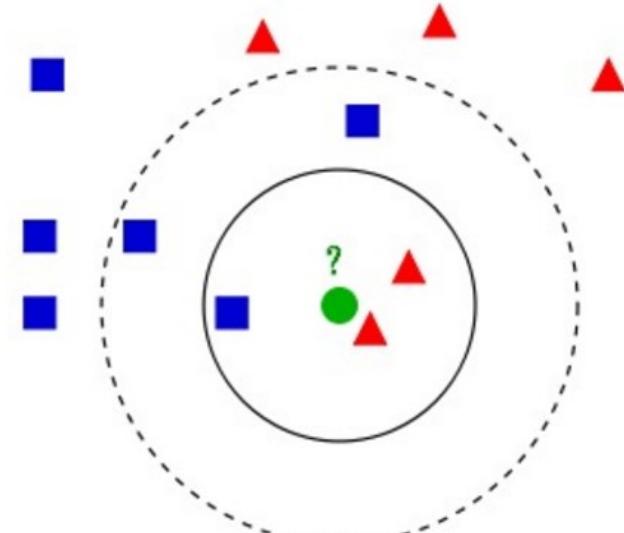


## 课堂回顾

### K-近邻算法

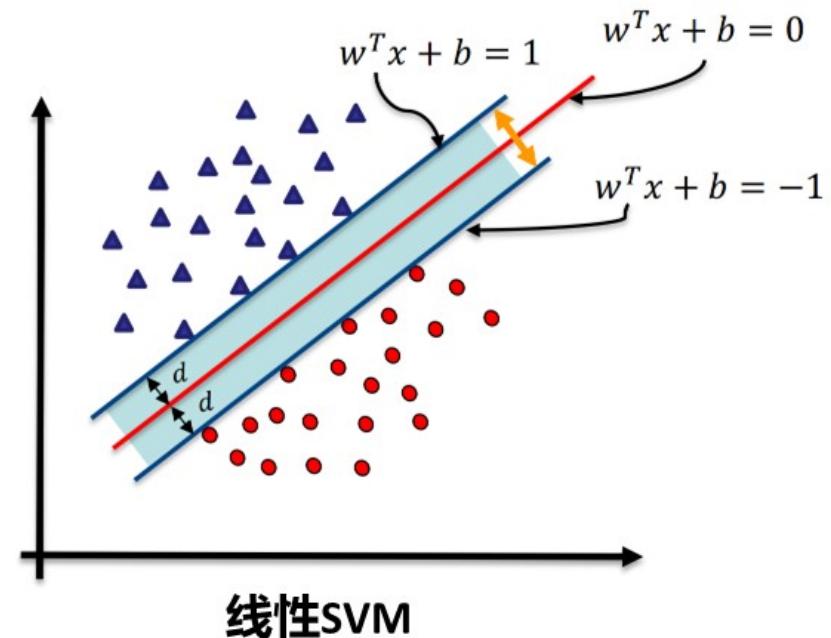
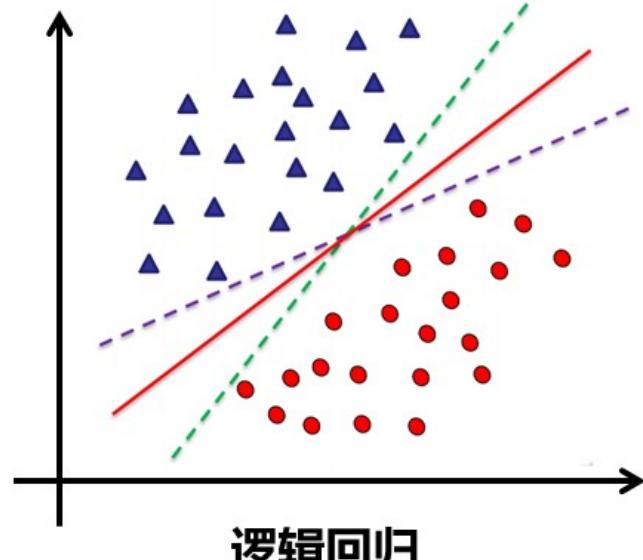
#### 三要素

- **距离度量**: 用于衡量样本之间的相似性，常用的有欧氏距离、曼哈顿距离、余弦相似度等。
- **K值选择**: 表示选取最近的 K 个邻居，K 值过小容易过拟合，过大则可能欠拟合，需通过验证选择最优值。
- **决策规则**: 决定如何根据邻居的结果做出预测，分类任务用投票法，回归任务用平均或加权平均。



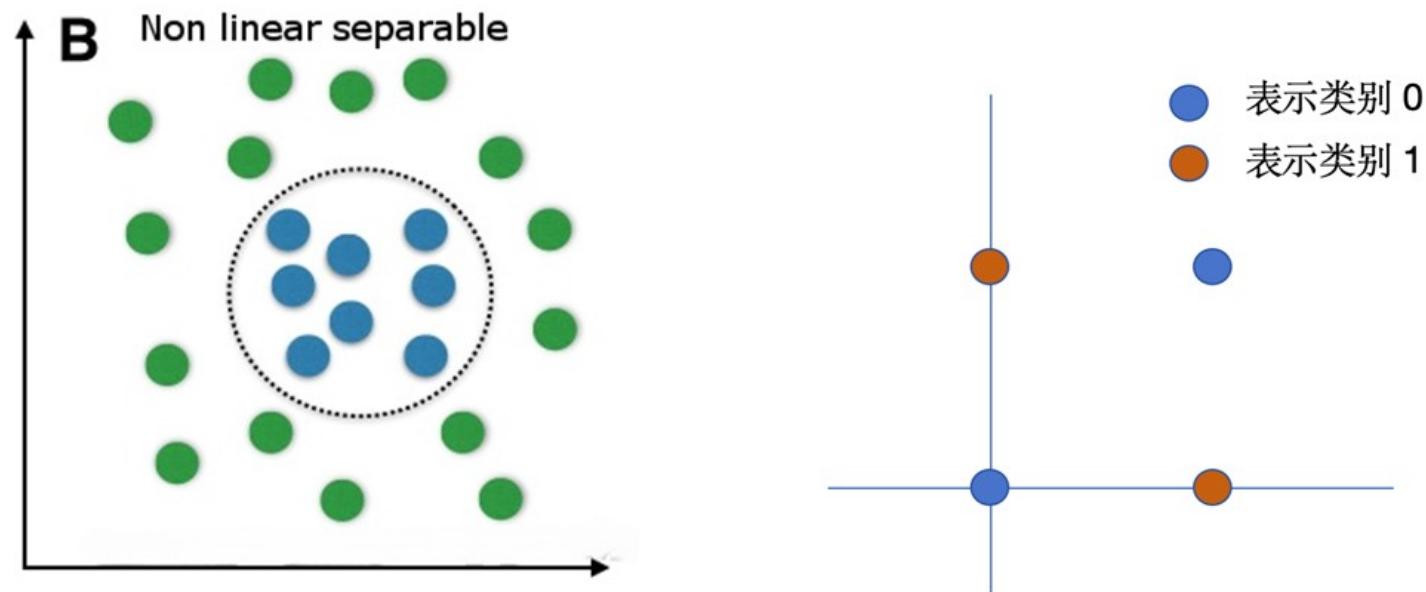
## 线性分类模型

逻辑回归以及线性 SVM，本质上都是基于线性决策边界的模型，可以解决**线性可分** (linearly separable) 问题。



## 课程回顾

### 线性不可分



无法通过直线或超平面进行划分

14

# Lecture Plan

- 神经网络基础知识
- 前馈神经网络
- 反馈神经网络
- 图神经网络



15

# Lecture Plan

- 神经网络基础知识
- 前馈神经网络
- 反馈神经网络
- 图神经网络

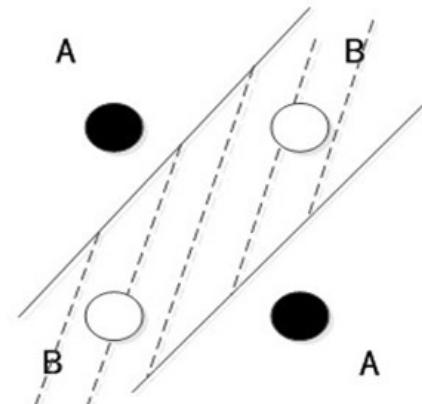
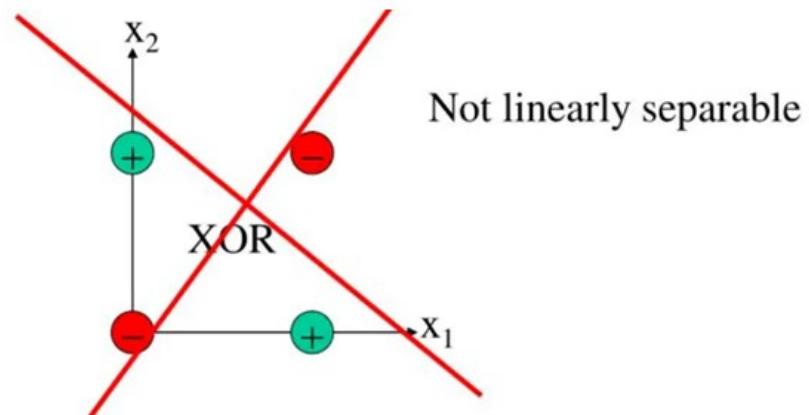


16

## 神经网络基础知识 线性不可分问题

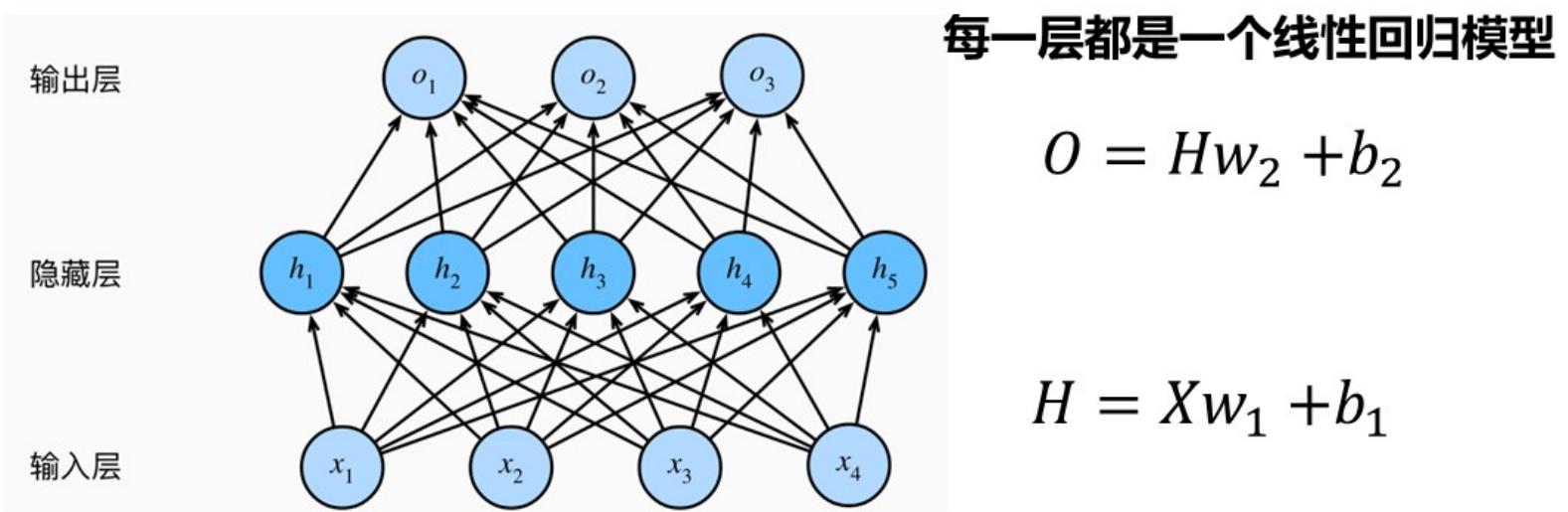
如何解决线性不可分问题?

解决方法之一：拟合多条直线



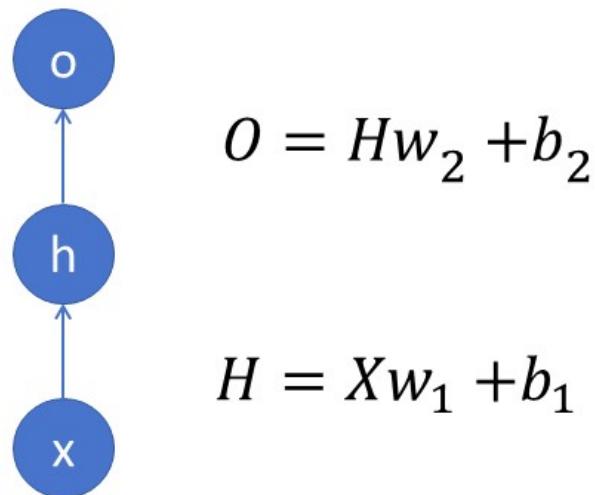
## 神经网络基础知识 多层感知机 (Multilayer Perceptron, MLP)

单层的线性结构是拟合单条直线，那多叠几层是不是就能拟合多条直线了？



## 主观题 10分

请大家写出 $o$ 对于 $x$ 的表达式  
为了方便起见，我们假设每层只有一个神经元



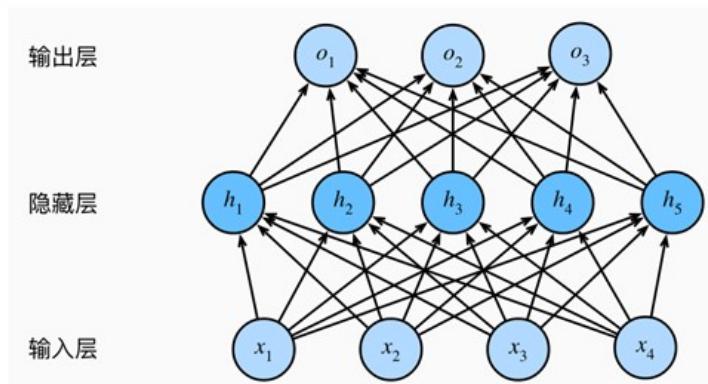
## 神经网络基础知识 多层感知机 (Multilayer Perceptron, MLP)

- 若单纯添加线性隐藏层

$$H = Xw_1 + b_1$$

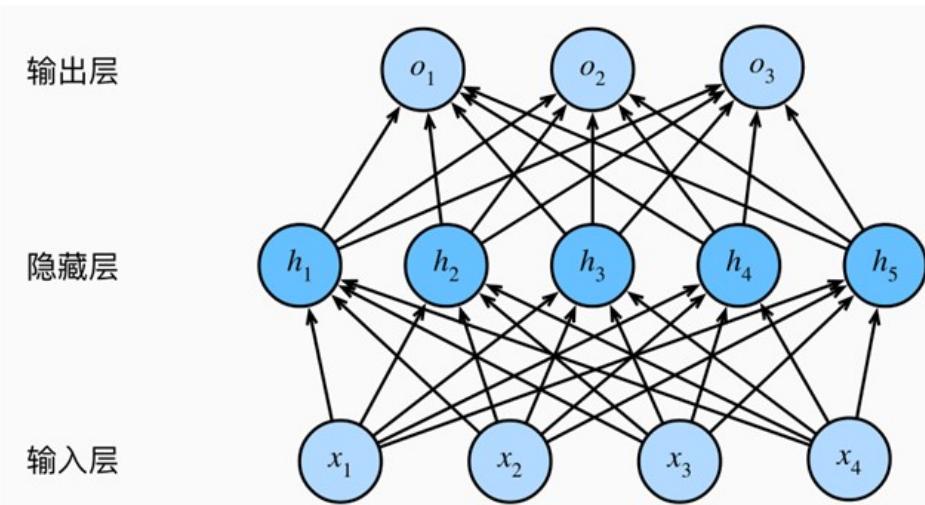
$$O = Hw_2 + b_2$$

对于任意权重值，只需合并隐藏层，便可产生具有参数  $w = w_1w_2$  和  $b = b_1w_2 + b_2$  的等价单层模型



## 神经网络基础知识 多层感知机 (Multilayer Perceptron, MLP)

- 解决方案：对每个隐藏单元应用**非线性的激活函数** (activation function)  $\sigma$ ，引入非线性因素



$$O = \sigma(Hw_2 + b_2)$$

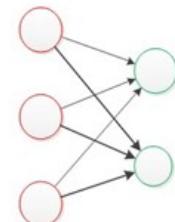
$$H = \sigma(Xw_1 + b_1)$$

Sigmoid: 逻辑回归

“如果没有激活函数，那么该网络仅能够表达线性映射，即便有再多的隐藏层，其整个网络跟单层神经网络也是等价的。”

# 神经网络基础知识 效果对比

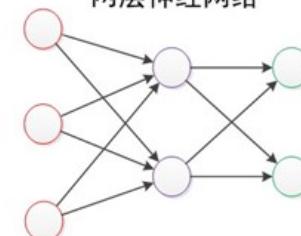
单层神经网络



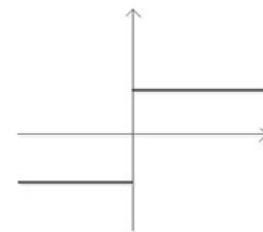
网络结构:



两层神经网络



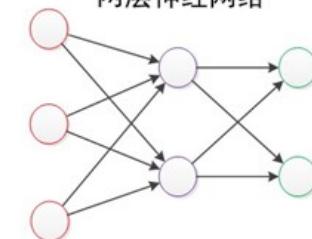
sgn (阶跃函数)



激活函数:

无或只有  
输出层有  
激活函数

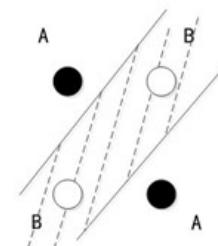
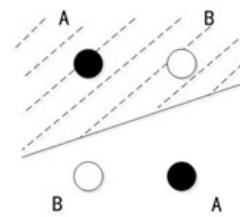
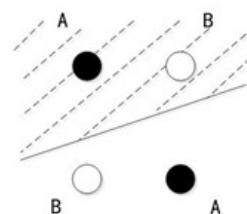
两层神经网络



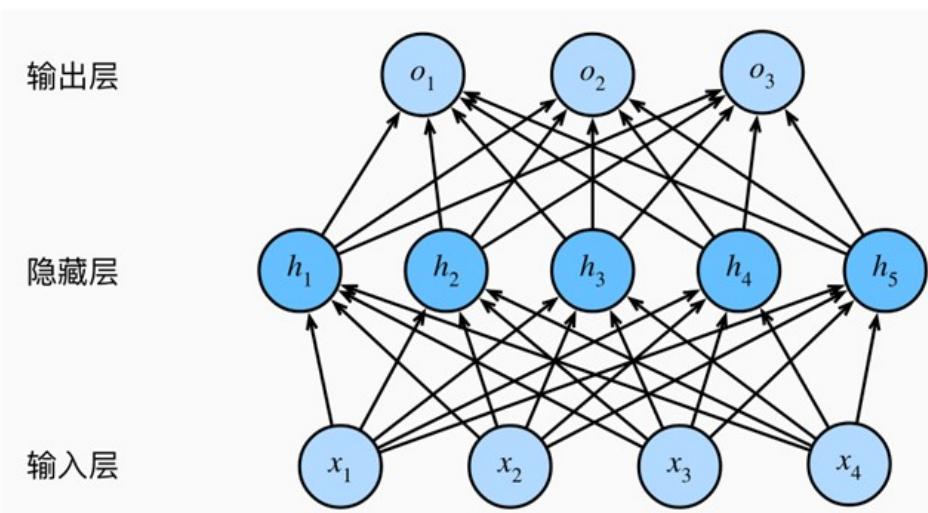
sigmoid



异或问题:



## 神经网络基础知识 多层感知机 = (线性变换+激活函数) \* n



$$O = \sigma(Hw_2 + b_2)$$

$$H = \sigma(Xw_1 + b_1)$$

## 神经网络基础知识 损失函数 (loss function)

损失函数用于**量化实际值（你写的答案）与预测值（参考答案）之间的差距**

通常选择**非负数**作为损失，数值越小表示损失越小，完美预测时的损失为0

例如，回归问题中最常用的损失函数是**平方误差函数**

当样本*i*的预测值为  $\hat{y}^{(i)}$ ，其相应的真实标签为  $y^{(i)}$  时，**平方误差**定义为：

$$l^{(i)}(\mathbf{w}, b) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$$

其中，权重向量w和偏置b是我们的**学习目标**

# 神经网络基础知识 损失函数 (loss function)

## 常用损失函数：

- 均方误差(Mean Square Error, MSE)

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

- 平均绝对误差(Mean Absolute Error, MAE)

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2}$$

- 均方根误差RMSE(Root Mean Square Error, RMSE)

$$\text{MAE}(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^m |y_i - \hat{y}_i|$$

- R方 (R Squared)

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^m (y_i - \hat{y}_i)^2}{\sum_{i=1}^m (y_i - \bar{y})^2} \quad \bar{y} = \frac{1}{m} \sum_{i=1}^m y_i$$

- 交叉熵损失 (Cross-Entropy Loss) – 二分类

$$L = -\frac{1}{m} \sum_{j=1}^m \left[ y^{(j)} \log(\hat{y}^{(j)}) + (1 - y^{(j)}) \log(1 - \hat{y}^{(j)}) \right]$$

- 交叉熵损失 (Cross-Entropy Loss) – 多分类

$$L = -\frac{1}{m} \sum_{j=1}^m \sum_{i=1}^c y_{ji} \log(\hat{y}_{ji})$$

主观题 10分

你在高中时，是怎么把一个知识点从不熟到完全掌握的？

## 神经网络基础知识 多层感知机 (Multilayer Perceptron, MLP)

你在高中时，是怎么把一个知识点从不熟到完全掌握的？

1. 刷题

2. 查漏补缺

# 神经网络基础知识 训练

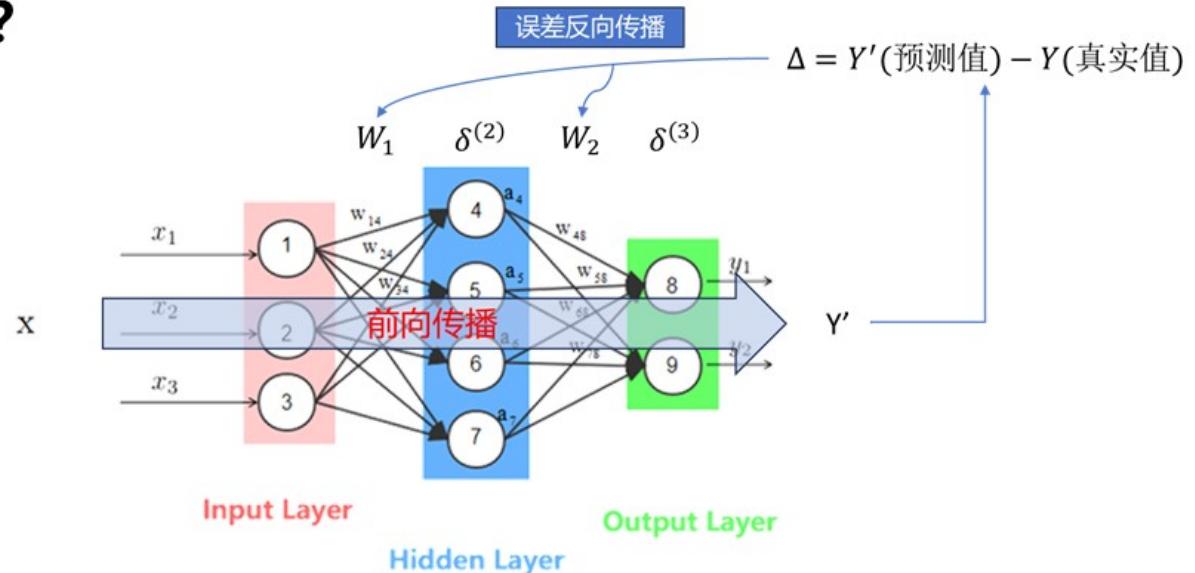
## 如何训练一个多层次感知机? —— 更新参数 (w和b)

1. 数据前向传播 (刷题)

2. 误差反向传播 (查漏补缺)

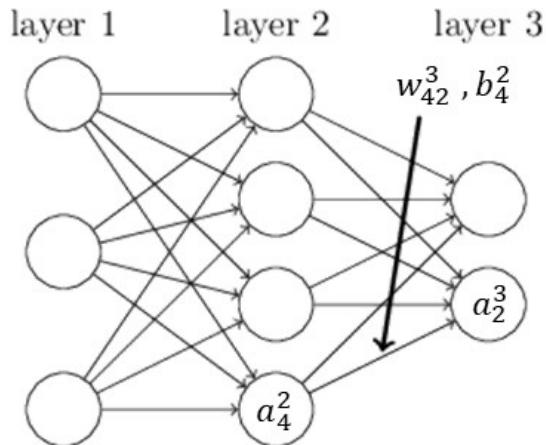
1) 通过损失函数计算误差

2) 通过反向传播计算权重参数的梯度, 用优化算法更新权重



# 神经网络基础知识 训练-前向传播 (forward propagation)

按顺序（从输入层到输出层）计算和存储神经网络中每层的结果。



记号	含义
$L$	神经网络的层数
$w_{jk}^l$	$l-1$ 层第 $k$ 个神经元到第 $l$ 层第 $j$ 个神经元的权重
$b_j^l$	第 $l$ 层第 $j$ 个神经元的偏置
$z_j^l$	第 $l$ 层第 $j$ 个神经元输入激活函数前的输出
$a_j^l$	第 $l$ 层第 $j$ 个神经元经过激活函数后的输出

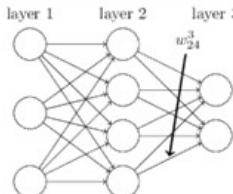
前馈计算：

$$x = \mathbf{a}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{a}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \dots \rightarrow \mathbf{a}^{(L-1)} \rightarrow \mathbf{z}^{(L)} \rightarrow \mathbf{a}^{(L)} = \phi(x; \mathbf{W}, \mathbf{b})$$

# 神经网络基础知识 训练-前向传播 (forward propagation)

## 代数法

对于Layer 2的输出  $a_1^{(2)}, a_2^{(2)}, a_3^{(2)}$ ,



$$a_1^{(2)} = \sigma(z_1^{(2)}) = \sigma(w_{11}^{(2)}x_1 + w_{12}^{(2)}x_2 + w_{13}^{(2)}x_3 + b_1^{(2)})$$

$$a_2^{(2)} = \sigma(z_2^{(2)}) = \sigma(w_{21}^{(2)}x_1 + w_{22}^{(2)}x_2 + w_{23}^{(2)}x_3 + b_2^{(2)})$$

$$a_3^{(2)} = \sigma(z_3^{(2)}) = \sigma(w_{31}^{(2)}x_1 + w_{32}^{(2)}x_2 + w_{33}^{(2)}x_3 + b_3^{(2)})$$

对于Layer 3的输出  $a_1^{(3)}$ ,

$$a_1^{(3)} = \sigma(z_1^{(3)}) = \sigma(w_{11}^{(3)}a_1^{(2)} + w_{12}^{(3)}a_2^{(2)} + w_{13}^{(3)}a_3^{(2)} + b_1^{(3)})$$

$$a_2^{(3)} = \sigma(z_2^{(3)}) = \sigma(w_{21}^{(3)}a_1^{(2)} + w_{22}^{(3)}a_2^{(2)} + w_{23}^{(3)}a_3^{(2)} + b_2^{(3)})$$

## 矩阵法

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$$

$$a^{(l)} = \sigma(z^{(l)})$$

其中  $\sigma$  为激活函数，如 Sigmoid, ReLU 等。

# 神经网络基础知识 训练-反向传播 (backward propagation)

## 如何训练一个多层感知机? —— 更新参数 (w和b)

### 误差反向传播

- 1) 通过**损失函数**计算误差
- 2) 通过**反向传播**计算权重参数的梯度，用**优化算法**更新权重

## 神经网络基础知识 训练-损失函数 (loss function)

**如何根据损失函数的结果（答题情况）中挖掘我们参数需要修改的信息（知识点的更新）？**

## 神经网络基础知识 训练-损失函数 (loss function)

以均方误差为例  $MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$

假设现在是很简单的线性回归  $y = wx$

假设平面内有一个点  $(x_1, y_1)$ ，则其对应的误差为：

$$e_1 = (y_1 - w * x_1)^2$$

写成关于w的表达式：

$$e_1 = (w * x_1 - y_1)^2$$

$$e_1 = w^2 * x_1^2 - 2(w * x_1 * y_1) + y_1^2$$

$$e_1 = x_1^2 * w^2 - 2(x_1 * y_1) * w + y_1^2$$

# 神经网络基础知识 训练-损失函数 (loss function)

**单数据损失:**

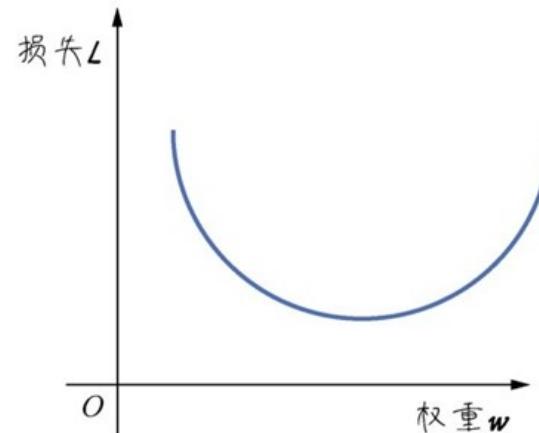
$$e_1 = x_1^2 * w^2 - 2(x_1 * y_1) * w + y_1^2$$

$$e_2 = x_2^2 * w^2 - 2(x_2 * y_2) * w + y_2^2$$

$$e_3 = x_3^2 * w^2 - 2(x_3 * y_3) * w + y_3^2$$

...

$$e_n = x_n^2 * w^2 - 2(x_n * y_n) * w + y_n^2$$



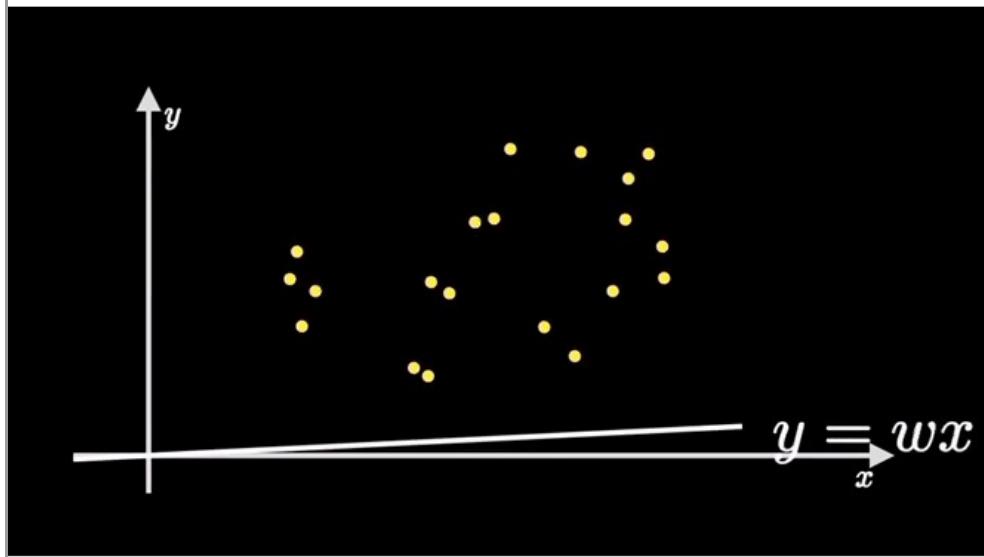
**整体损失:**

$$e = \frac{1}{n} \left( (x_1^2 + \dots + x_n^2) * w^2 + (-2 * x_1 * y_1 - \dots - 2 * x_n * y_n) * w + (y_1^2 + \dots + y_n^2) \right)$$

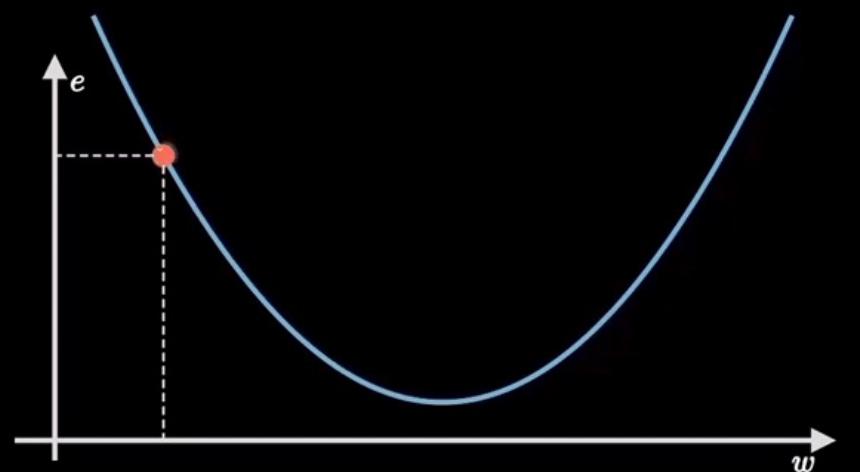
$$e = \frac{1}{n} (a * w^2 + b * w + c)$$

# 神经网络基础知识 训练-损失函数 (loss function)

预测函数 (线性模型)



整体损失 w.r.t. w



【【梯度下降】3D可视化讲解通俗易懂】

[https://www.bilibili.com/video/BV18P4y1j7uH/?share\\_source=copy\\_web&vd\\_source=ec062cb66d80e1d8984c60843ca97804](https://www.bilibili.com/video/BV18P4y1j7uH/?share_source=copy_web&vd_source=ec062cb66d80e1d8984c60843ca97804)

# 神经网络基础知识 训练-反向传播 (backward propagation)

优化算法：从代价函数中找到最优  
w，使得整体损失最低

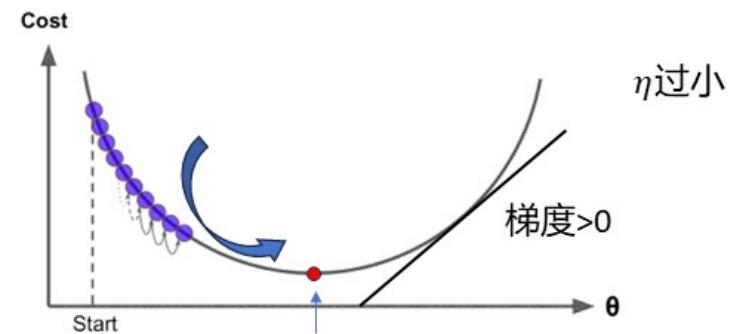
## 梯度下降

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

- $\eta$ 为学习率，控制收敛速度

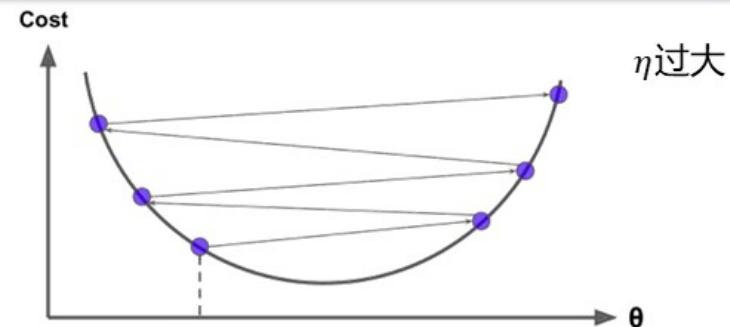
为什么用减法更新权重？

- 如果梯度 $>0$ ，则说明w越小，损失越小，所以需要减小（减去正数）
- 如果梯度 $<0$ ，则说明w越大，损失越小，所以需要增大（减去负数）



$\eta$ 大小需要控制

- 如果太小，会导致w更新很慢
- 如果太大，会导致无法找到最优解



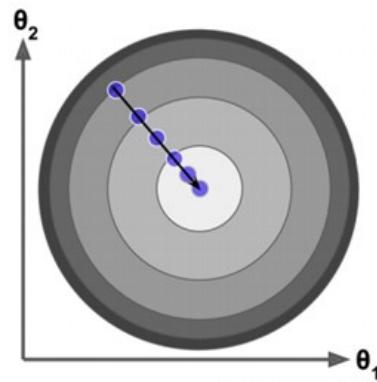
## 神经网络基础知识 训练-反向传播 (backward propagation)

Why：我们通常采用梯度下降法而不是直接用一元二次方程求解？

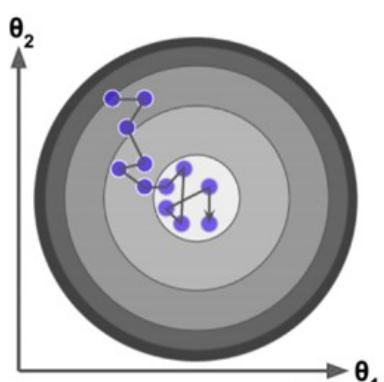
- 现在的神经网络参数量庞大且目标函数高度非线性、非凸，难以直接计算解析解；
- 相比之下，梯度下降方法**只需利用损失函数的梯度**，就能通过**迭代逐步逼近最优解**，并且可以结合小批量数据高效更新参数，因而成为深度学习中普遍采用的优化方法。

# 神经网络基础知识 训练-反向传播 (backward propagation)

## 随机梯度下降



梯度下降  
(批量梯度下降)



随机梯度下降

- 批量梯度下降 (Batch Gradient Descent , BGD) : 一次迭代是对所有样本进行计算。计算代价大，但更平稳。

$$\theta := \theta - \alpha \frac{1}{n} \sum_{i=1}^n (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

- 随机梯度下降 (Stochastic Gradient Descent , SGD) : 一次迭代在所有的样本数据中选择随机的一个实例。不平稳，且很可能无法收敛到全局最优解，但随着迭代次数的增加，误差会越来越小。

$$\theta := \theta - \alpha (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

为了平衡计算效率和稳定性，实际应用中通常采用**小批量梯度下降 (Mini-batch Gradient Descent)**，即每次迭代使用一个较小批量的样本 (**batch size**) 进行梯度更新

## 神经网络基础知识 训练-反向传播 (backward propagation)

# 梯度怎么求?

# 神经网络基础知识 训练-反向传播 (backward propagation)

## 链式法则

根据微积分中的**链式规则**，从输出层到输入层遍历网络，计算神经网络参数梯度并更新。

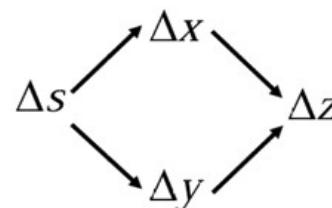
### Case 1

$$y = g(x) \quad z = h(y)$$

$$\Delta x \rightarrow \Delta y \rightarrow \Delta z \quad \frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$$

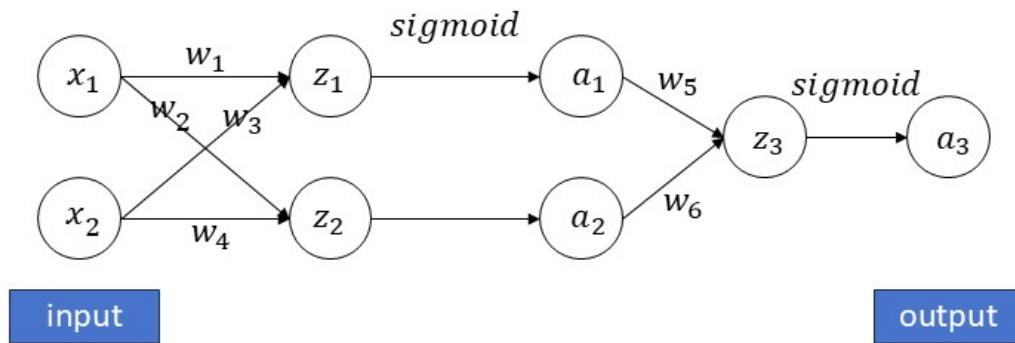
### Case 2

$$x = g(s) \quad y = h(s) \quad z = k(x, y)$$


$$\frac{dz}{ds} = \frac{\partial z}{\partial x} \frac{dx}{ds} + \frac{\partial z}{\partial y} \frac{dy}{ds}$$

# 神经网络基础知识 训练-反向传播 (backward propagation)

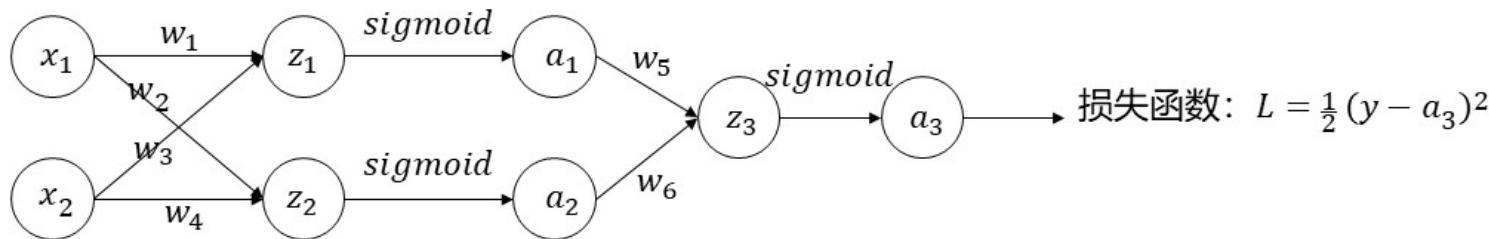
根据微积分中的**链式规则**，从输出层到输入层遍历网络，计算神经网络参数梯度并更新。



## 数据前向传播

$$\begin{cases} z_1 = w_1 \cdot x_1 + w_2 \cdot x_2 + b_1 \\ z_2 = w_3 \cdot x_1 + w_4 \cdot x_2 + b_2 \end{cases} \rightarrow \begin{cases} a_1 = \frac{1}{1 + e^{-z_1}} \\ a_2 = \frac{1}{1 + e^{-z_2}} \end{cases} \rightarrow z_3 = w_5 \cdot a_1 + w_6 \cdot a_2 + b_3 \rightarrow a_3 = \frac{1}{1 + e^{-z_3}}$$

# 神经网络基础知识 训练-反向传播 (backward propagation)



## 误差反向传播

$$w_4 \rightarrow z_2 \rightarrow a_2 \rightarrow z_3 \rightarrow a_3 \rightarrow L$$

$$\frac{\partial L}{\partial w_5} = \boxed{\frac{\partial L}{\partial a_3} \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial w_5}}$$

$$\frac{\partial L}{\partial w_3} = \boxed{\frac{\partial L}{\partial a_3} \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_3}}$$

已知  $\frac{\partial L}{\partial z_3} = \delta_3$

求  $\frac{\partial L}{\partial w_6}$  (带  $\delta_3$ ) :  $\frac{\partial L}{\partial w_6} = \frac{\partial L}{\partial a_3} \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial w_6} = \delta_3 \frac{\partial z_3}{\partial w_6}$

求  $\frac{\partial L}{\partial z_2}$  (带  $\delta_3$ ) :  $\frac{\partial L}{\partial z_2} = \frac{\partial L}{\partial a_3} \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial a_2} \frac{\partial a_2}{\partial z_2} = \delta_3 \frac{\partial z_3}{\partial a_2} \frac{\partial a_2}{\partial z_2} = \delta_2$

求  $\frac{\partial L}{\partial w_4}$  (带  $\delta_2$ ) :  $\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial a_3} \frac{\partial a_3}{\partial z_3} \frac{\partial z_3}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial w_4} = \delta_2 \frac{\partial z_2}{\partial w_4}$

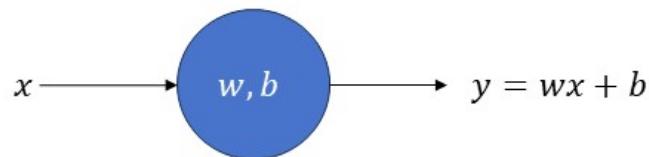
.....

红色方框部分可以复用

当层数增高，通过复用链式传导中间  
结果可以显著减少计算量

## 主观题 30分

### Case1：将梯度下降用于简单的线性回归



- 已知:  $x = 1.5, y_{gt} = 0.8$
- 线性模型函数:  $y = wx + b$
- 损失函数:  $L = \frac{1}{2}(y_{pred} - y_{gt})^2$
- 随机初始化权重:  $w_0 = 0.8, b_0 = 0.2$
- 学习率:  $\alpha = 0.1$

问题:

- 第一次前向传播的结果 ( $y_{pred}$ ) ?
- 第一次反向传播后的权重 (w 和 b) ?
- 第二次前向传播的结果 ( $y_{pred}$ ) ?

## Practice

1. 正向传播：将真实数据的 $x$ , 代入初始化好的线性模型公式

$$y = 0.8 \times 1.5 + 0.2 = 1.4 = y_0$$

2. 反向传播：计算权重 $w$ 与偏置 $b$ 的梯度并更新

$$\begin{aligned}\frac{\partial L}{\partial w} &= \frac{\partial L}{\partial y} \frac{\partial y}{\partial w} = (y - y_{gt}) x = (1.4 - 0.8) \times 1.5 = 0.9 \\ \frac{\partial L}{\partial b} &= \frac{\partial L}{\partial y} \frac{\partial y}{\partial b} = y - y_{gt} = 1.4 - 0.8 = 0.6\end{aligned}$$

更新 $w, b$ 一次为 $w_1, b_1$

$$\begin{aligned}w_1 &= w - \alpha \frac{\partial L}{\partial w} = 0.8 - 0.1 \times 0.9 = 0.71 \\ b_1 &= b - \alpha \frac{\partial L}{\partial b} = 0.2 - 0.1 \times 0.6 = 0.14\end{aligned}$$

## Practice

得到反向传播梯度后的线性模型:

$$y = 0.71x + 0.14$$

对比更新前后的模型loss

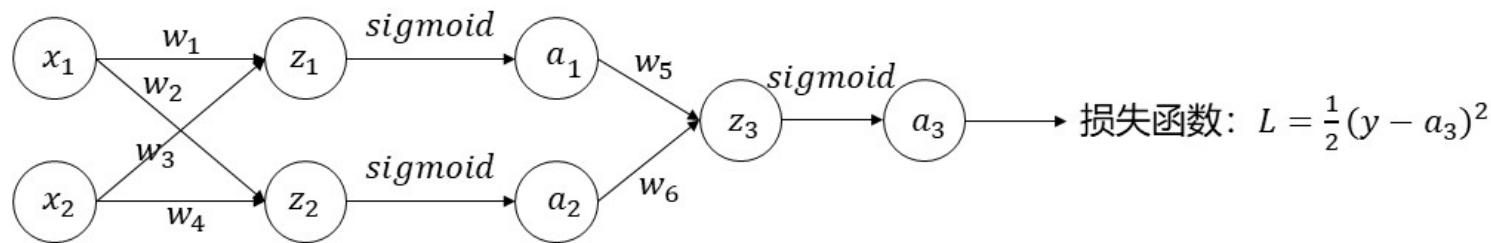
$$\begin{aligned}y &= 0.8x + 0.2 = 1.4 = y_0 \\y &= 0.71x + 0.14 = 0.71 \times 1.5 + 0.14 = 1.205 = y_1\end{aligned}$$

代入损失函数

$$\begin{aligned}L_0 &= \frac{1}{2} (y - y_{gt})^2 = \frac{1}{2} (1.4 - 0.8)^2 = 0.18 \\L_1 &= \frac{1}{2} (y - y_{gt})^2 = \frac{1}{2} (1.205 - 0.8)^2 = 0.0820125\end{aligned}$$

主观题 10分

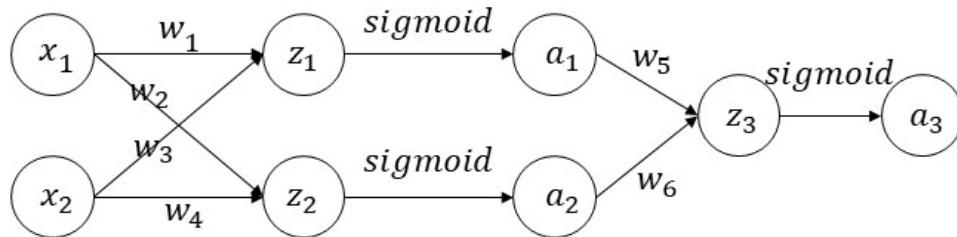
**Case2:** 反向传播



已知  $\frac{\partial L}{\partial z_1} = \delta_1$      $\frac{\partial L}{\partial z_2} = \delta_2$

求  $\frac{\partial L}{\partial w_2}$  和  $\frac{\partial L}{\partial x_2}$  (带  $\delta_1$  和  $\delta_2$ )

## Practice



$$\text{损失函数: } L = \frac{1}{2}(y - a_3)^2$$

已知  $\frac{\partial L}{\partial z_1} = \delta_1 \quad \frac{\partial L}{\partial z_2} = \delta_2$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z_2} \frac{\partial z_2}{\partial w_2} = \delta_2 \frac{\partial z_2}{\partial w_2} = \delta_2 x_1$$

$$\frac{\partial L}{\partial x_2} = \frac{\partial L}{\partial z_1} \frac{\partial z_1}{\partial x_2} + \frac{\partial L}{\partial z_2} \frac{\partial z_2}{\partial x_2} = \delta_1 w_3 + \delta_2 w_4$$

# 神经网络基础知识 激活函数

**激活函数在反向传播中同样至关重要，它直接影响梯度在网络中的传递效果。**

**激活函数的作用：**

1. 引入非线性 → 让网络可以拟合复杂函数，而不仅仅是线性分类器。
2. 控制梯度传播 → 不同函数对梯度的保留/衰减不同，直接影响训练效果。

**激活函数的性质：**

1. 连续并可导（允许少数点上不可导）的非线性函数
2. 简单 → 在深度网络中，激活函数会被调用数百万次，因此必须足够快。
3. 激活函数及其导函数的值域要在一个合适的区间内

# 神经网络基础知识 激活函数-sigmoid

Sigmoid函数最早是在逻辑回归中提到的，它作为解决二分类的问题出场。

其值域是在[0,1]之间，输出的值可以作为分类的概率。

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

优点：

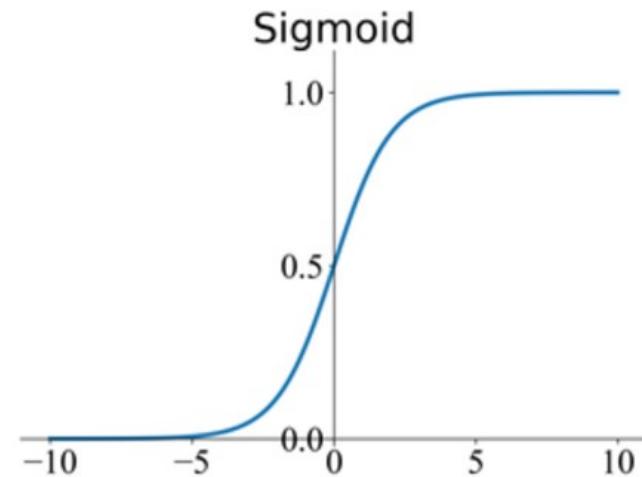
简单、非常适用二分类任务；

缺点：

1. 大输入时梯度接近 0 → 容易梯度消失。
2. 输出非零均值 → 会导致训练收敛慢。度增大，训练耗时；

应用：

现在多用于二分类输出层，不常作为隐藏层激活。



## 神经网络基础知识 激活函数-tanh

其输出值在区间 [-1, 1]，函数表达式：

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

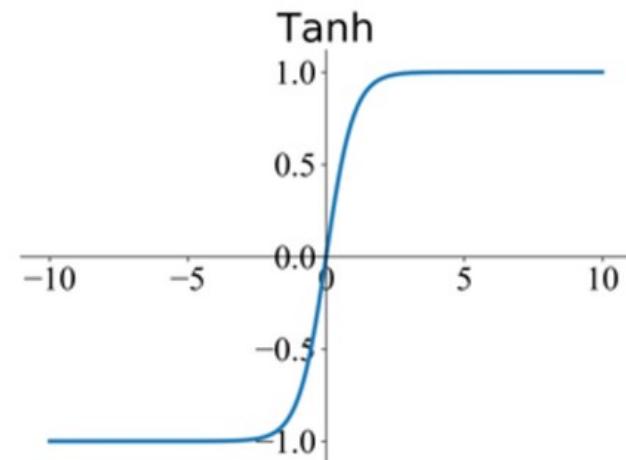
$$\tanh(x) = 2 * \text{sigmoid}(2x) - 1$$

优点：

输出在 (-1,1)，均值更居中，收敛比 Sigmoid 快。

缺点：

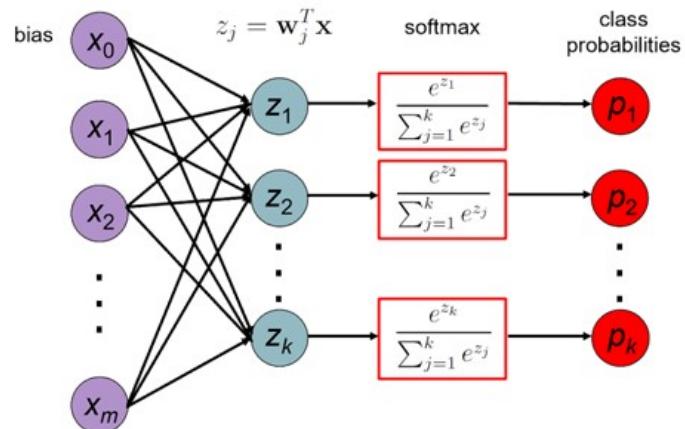
仍存在梯度消失问题。



## 神经网络基础知识 激活函数-softmax

SoftMax函数通常被用在多分类输出层中，其表达式如下：

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_j e^j}$$



- 值域在[0,1]之间
- 存在多个输出，例如是一个5分类的任务，那么SoftMax函数最终的输出是对应每个类别的概率，同时这5个类别对应的概率相加最终的结果为1。
- 多分类任务的场景下，神经网络的最后一层一般都是使用SoftMax函数来进行映射

# 神经网络基础知识 激活函数-relu

ReLU函数是目前在神经网络使用最流行的激活函数。

其函数表达式：

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

$$ReLU(x) = \max(0, x)$$

优点：

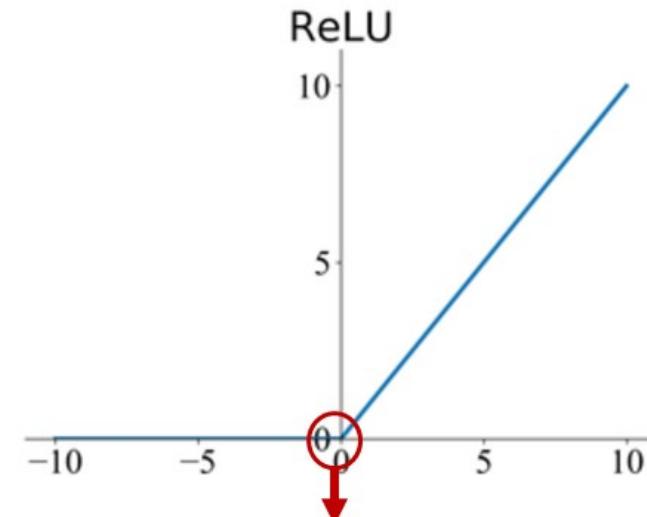
1. 计算简单！
2. 正区间梯度恒为 1 → 有效缓解梯度消失。

应用：

现代深度学习默认首选。

缺点：

“死亡Relu”问题（输入长期  $< 0$ ，梯度为 0）。



- 在数学上，ReLU 在  $x = 0$  不可导
- 但在实际的深度学习训练中，ReLU 在  $x = 0$  时可以用次梯度 (subgradient) 来处理，即直接直接选 0 或选 1 作为梯度。

# 神经网络基础知识 激活函数-relu变体

Leaky Relu函数

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{if } x \leq 0 \end{cases}$$

$$\text{LeakyReLU}(x) = \max(0, x) + a\min(0, x)$$

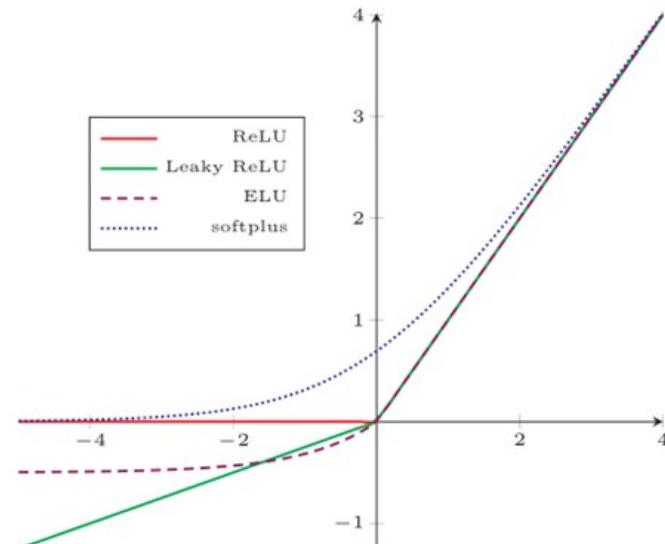
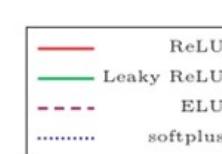
ELU函数——近似的零中心化的非线性函数

$$\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ a(e^x - 1) & \text{if } x \leq 0 \end{cases}$$

$$\text{ELU}(x) = \max(0, x) + \min(0, a(e^x - 1))$$

Softplus函数——ReLU函数的平滑版本

$$\text{Softplus}(x) = \log(1 + e^x)$$



# 神经网络基础知识 激活函数-Swish

Google团队利用**自动搜索技术（包括穷举搜索和强化学习）**发现的新激活函数，突破了人工设计的局限。

其函数表达式：

$$f(x) = x \cdot \text{sigmoid}(\beta x)$$

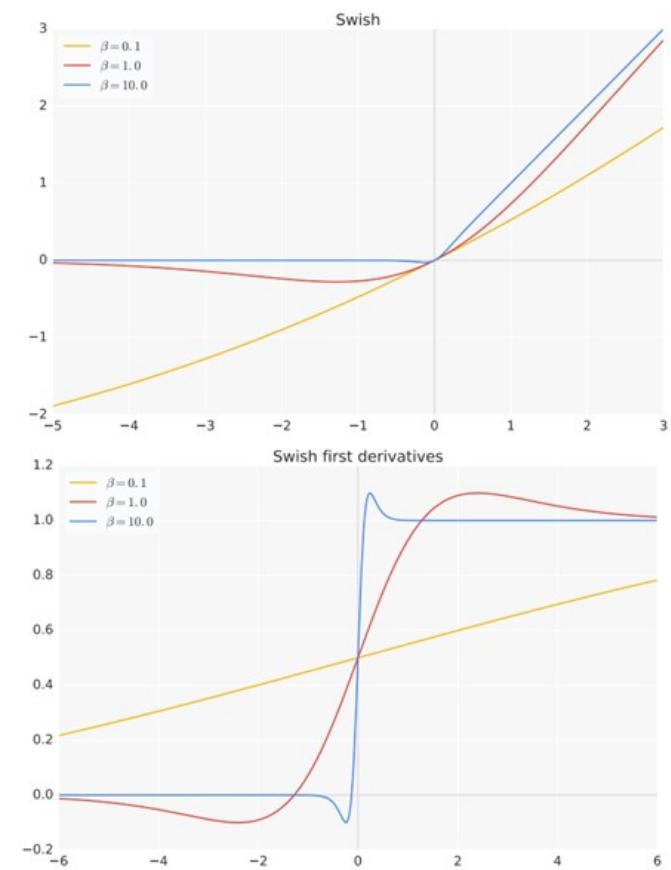
$\beta$ 是一个可调参数，常见设定是  $\beta = 1$

## 特点：

1. 与 ReLU 一样无上界，缓解梯度消失；
2. 平滑且可导，优化更稳定；
3. 非单调性使模型能捕捉更复杂的特征；

## 应用：

1. 在**40以上全连接层的效果要远优于其他激活函数**；
2. 40全连接层之内则性能差距不明显。



# 神经网络基础知识 激活函数-SwiGLU

出自 Google DeepMind 2020 年的论文《Switch Transformers》，  
后来在 LLaMA、PaLM 等大规模语言模型中被广泛采用。

其函数表达式：

$$SwiGLU(a, b) = \text{Swish}(a) \oplus (b)$$

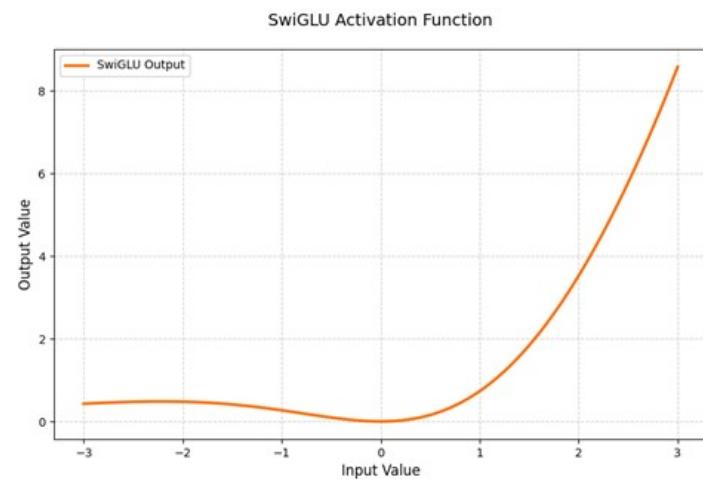
- $(a = xW_1 + b_1)$  和  $(b = xW_2 + b_2)$  是两个线性变换的输出
- $\oplus$  表示逐元素乘积 (Hadamard 积)

**特点：**

1. 门控机制：有效控制信息流动，抑制无用特征；
2. 平滑性：结合了 Swish 的“平滑非单调性”，灵活建模复杂特征；

**应用：**

相比传统 Transformer 的 FFN 层 (ReLU + 线性层，后面课程会讲解)，SwiGLU 虽引入额外线性变换，但通过调整中间层维度，可在参数量不变的情况下实现性能提升。



# 神经网络基础知识 激活函数-对比

## 输出层常用激活函数：

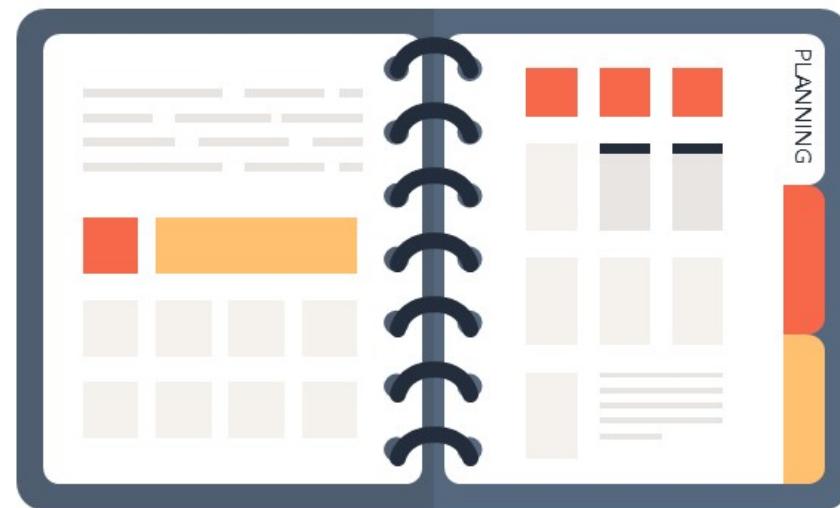
- **Sigmoid**: 常用于**二分类任务**, 将输出压缩到  $[0, 1]$ , 可以直接解释为概率。
- **Softmax**: 常用于**多分类任务**, 将多个神经元的输出归一化为概率分布, 每个类别  
的概率和为 1。

## 隐藏层常用激活函数：

- **Tanh**: 输出范围在  $[-1, 1]$ , 均值更居中, 收敛比 Sigmoid 快。
- **ReLU**: **最常用的隐藏层激活函数之一**, 计算高效、缓解梯度消失问题, 能够保持  
信息流动, 适合深层网络。
- **Leaky ReLU / ELU / Softplus**: ReLU 的改进版本。
- **Swish/SwIGLU**: 进一步改进的激活函数。

# Lecture Plan

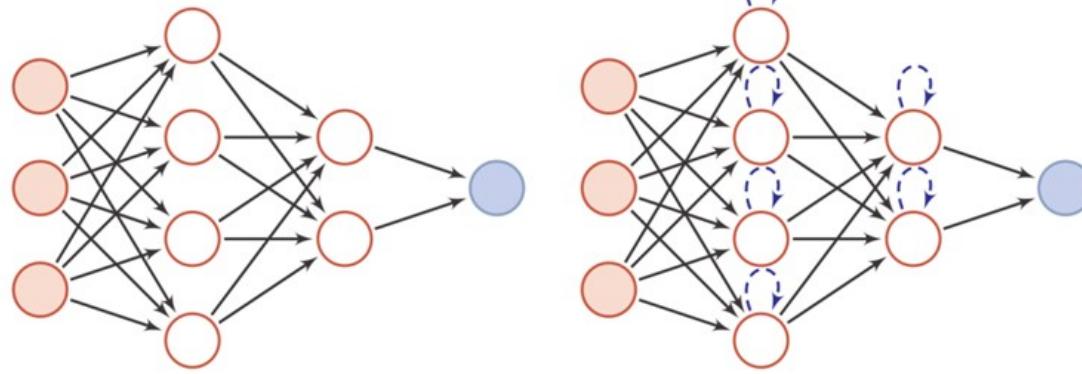
- 神经网络基础知识
- 前馈神经网络
- 反馈神经网络



57

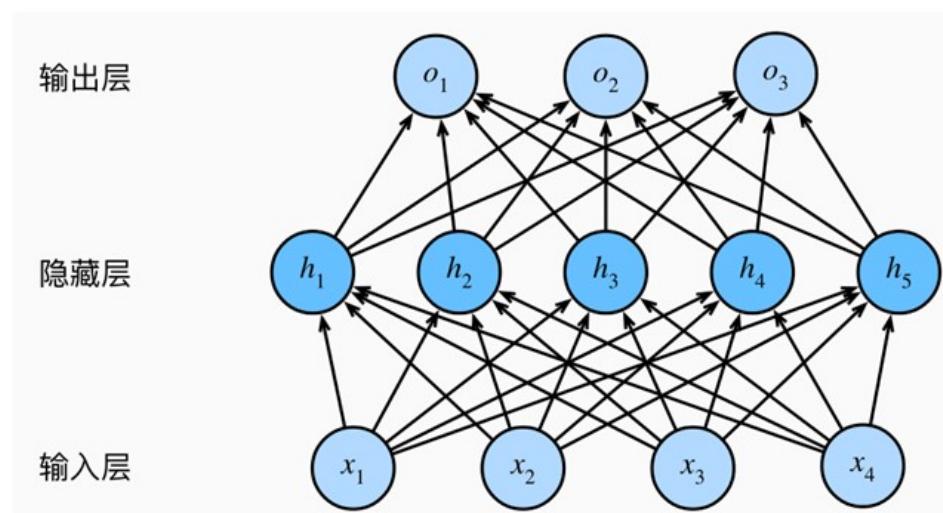
# 神经网络

神经网络由神经元模型构成，由许多神经元组成的信息处理网络具有并行分布结构。根据神经网络的**数据流动方向**，可将神经网络分为以下两类：



## 前馈神经网络 全连接神经网络

多层感知机，就是前馈网络，也是一种全连接神经网络

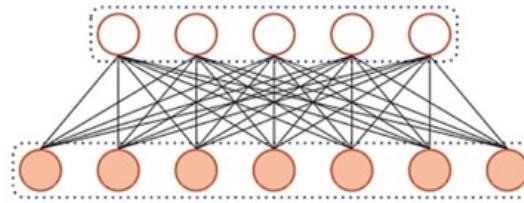


全连接层的特征是**层间全局连接**，每个神经元对前一层所有输出进行**加权组合与非线性映射**。

# 前馈神经网络 全连接神经网络的不足

## 全连接前馈神经网络不足：

1. 权重矩阵的参数非常多
2. 难以提取**局部不变性特征**



**局部不变性：**尺度缩放、平移、旋转等操作不影响其语义信息

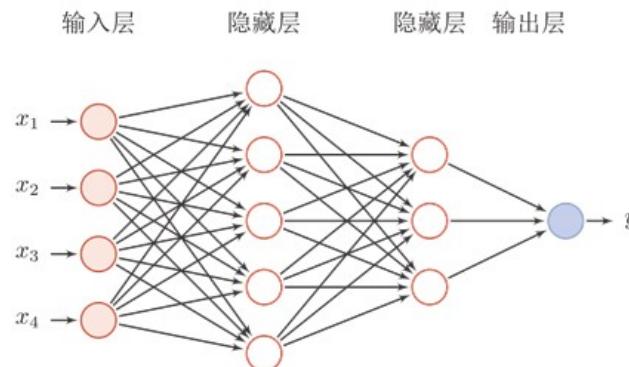
假设我们需要做图像识别，判断一张图片里面是猫还是狗：



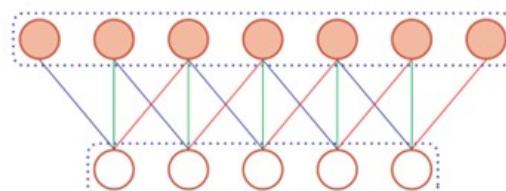
那么局部不变性就是无论这张照片里的猫或狗位于图像的哪个位置（上面还是下面，左边还是右边），训练好的神经网络都能够把它识别出来。

# 前馈神经网络

各神经元分别属于不同的层。整个网络中无反馈，信号从输入层向输出层单向传播，可用一个有向无环图表示。



从神经网络的结构特征来看，每一层的每一个节点都与上下层节点全部连接的网络也称作全连接神经网络。



隐藏层也可以采用卷积的方式进行连接，叫做卷积神经网络。

# 前馈神经网络 卷积神经网络 (convolutional neural networks, CNN)

- 一种前馈神经网络
- 受生物学上感受野 (receptive field) 的机制而提出的

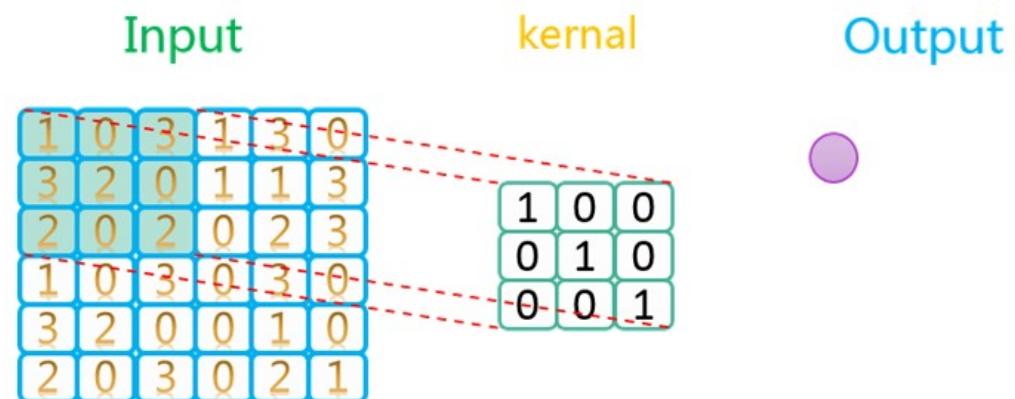
在视觉神经系统中，一个神经元的感受野是指视网膜上的特定区域，

只有这个区域内的刺激才能够激活该神经元

- 特点：

1. 局部连接

2. 权重共享



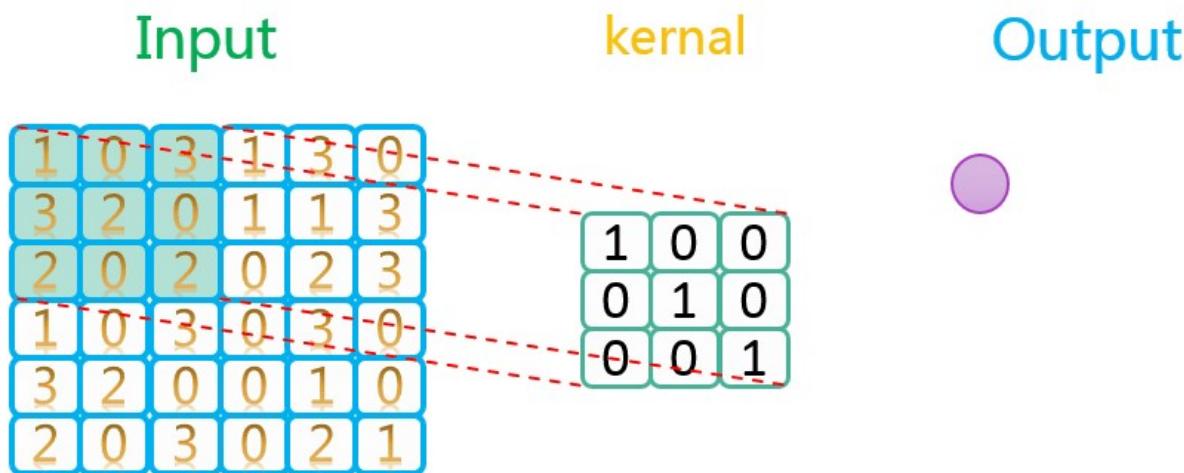
# 前馈神经网络 卷积神经网络 – 二维卷积

卷积操作步骤：

- 1.定位
- 2.对应相乘
- 3.求和输出

说人话：

不把所有神经元做线性求和了，  
只对一个个小块做线性求和



# 前馈神经网络 卷积神经网络 – 卷积核 (Kernel)

**卷积核 (Kernel) :** 一个小的权重矩阵 (比如 3x3, 5x5)，用于进行“卷积”计算。

**作用:** 每个卷积核负责从输入数据中提取一种特定的特征。

**大小:** 一般为3\*3或者5\*5。

一个卷积层通常有多个卷积核，每个卷积核会生成一个独立的输出特征图。

二维图像



原始图像

高斯卷积核

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$

=



去噪

⊗

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

=



高频信息 (边缘)

$$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

=



主对角线的边缘

滤波器

输出特征映射

# 前馈神经网络 卷积神经网络 – 卷积核 (Kernel)

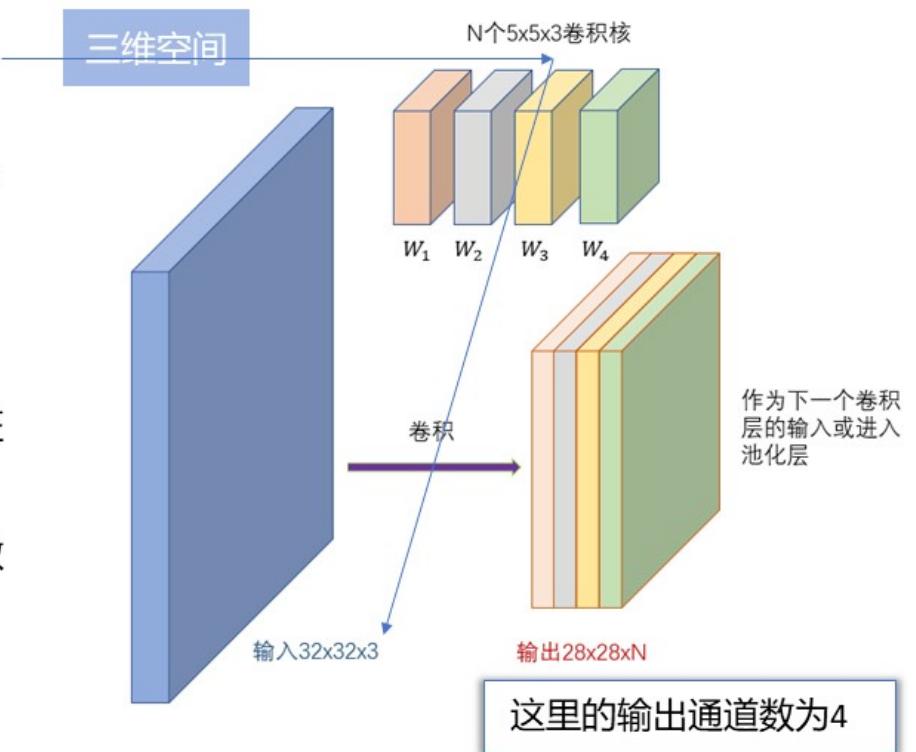
**卷积核 (Kernel) :** 一个小的权重矩阵 (比如  $3 \times 3, 5 \times 5$ )，用于进行“卷积”计算。

**深度:** 卷积核的深度必须和输入数据的深度一致。

**输出通道数:** 指定该卷积层要使用多少个卷积核，决定了该层输出特征图的深度 (或数量)。

**需要权衡:**

- 模型能力:** 通道数越多，模型能学习和表达的特征模式越丰富，拟合能力越强。
- 计算成本:** 通道数的增加会直接导致计算量与参数量的上升，需要更多的计算资源。



## 前馈神经网络 卷积神经网络 – 步长 (Stride)

**步长 (Stride) :** 卷积核在输入特征图上滑动的距离。

**核心作用:** 控制输出特征图尺寸与计算量。

**影响:**

- **小步长 (Stride=1):**

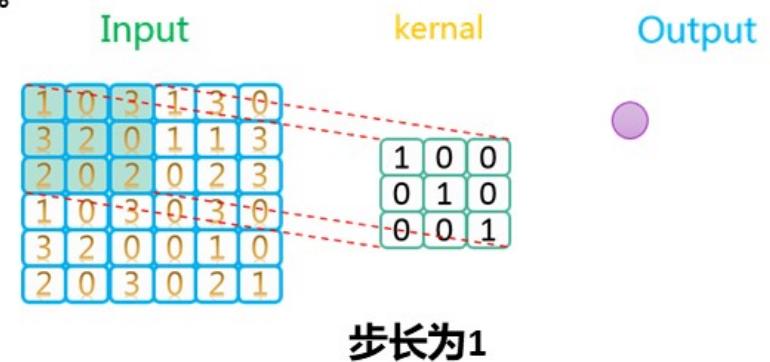
输出尺寸更大，保留信息更精细。

计算成本更高。

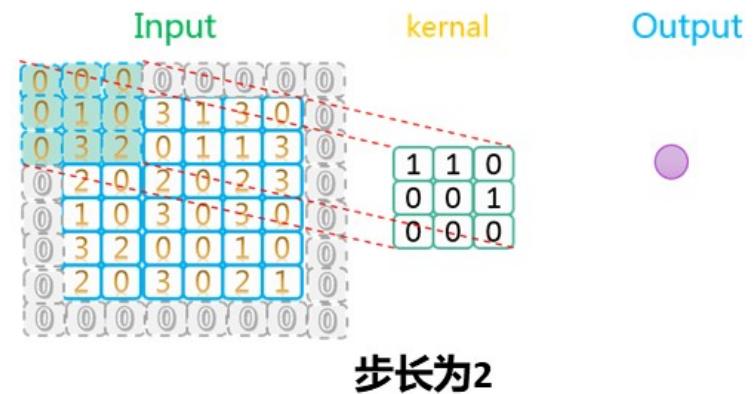
- **大步长 (Stride>1):**

输出尺寸变小，实现降采样(downsampling)。

计算更高效，但可能丢失细节。



步长为1



步长为2

## 前馈神经网络 卷积神经网络 – 填充 (Padding)

$$L_{out} = \left\lfloor \frac{L_{in} + 2P - K}{S} \right\rfloor + 1$$

**问题：**哪怕步长选择为1，**特征图的尺寸也会不断地缩小**，这将导致卷积神经网络**无法设计太深的层数**。此外，**边界元素在卷积中容易被“稀释”**，即被卷积操作次数会比再中间的元素少。

**填充 (Padding) :** 在输入特征图边界周围添加元素（通常为零）。

**作用：**

- 维持尺寸：**精准控制输出特征图尺寸，防止过快收缩。
- 保护边缘：**确保边缘元素得到充分计算，避免信息丢失。

**怎么填：**

**Padding=1：**上下左右都拓展一层，填充值通常为0

**Padding= Same (同尺寸填充) :** 填充的圈数经过精心计算，使得输出特征图的尺寸与输入尺寸完全相同。**这是最常用的模式。**

**Input**

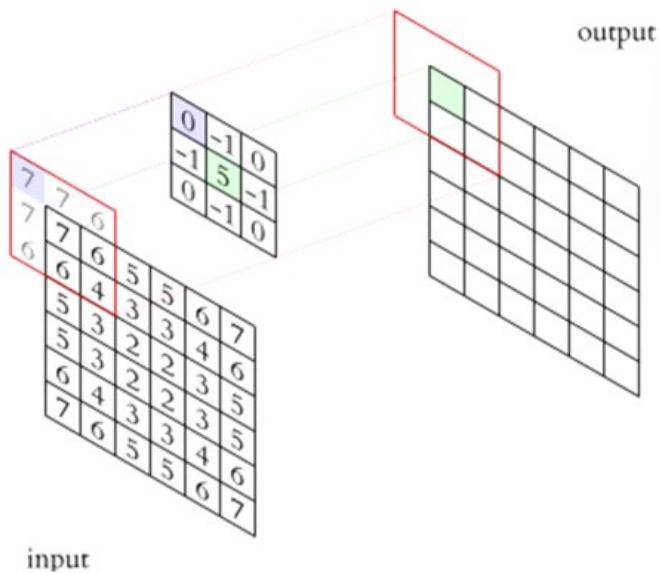
0	0	0	0	0	0	0
1	0	3	1	3	0	0
3	2	0	1	1	3	0
2	0	2	0	2	3	0
1	0	3	0	3	0	0
3	2	0	0	1	0	0
2	0	3	0	2	1	0

Padding=1：代表拓展一层  
输入图片大小：

6\*6 -> 8\*8

输出图片大小 (Stride=1)：  
4\*4 -> 6\*6

## 填空题 3分



1. 卷积核Kernel= [填空1] (填 $n \times n$ )
2. 步长Stride = [填空2] (从输出的大小和padding的大小来考虑)
3. 这一步的计算结果为 (output的绿色部分的值为) [填空3]

## 前馈神经网络 卷积神经网络 – 卷积与互相关

【注意】神经网络中的“卷积”与数学概念中的“卷积”并不一致！

- 卷积（数学概念）：需要进行卷积核翻转：

$$\begin{matrix} w_1 & w_2 & w_3 \\ x_1 & x_2 & x_3 \end{matrix} \xrightarrow{\quad} w_3 \cdot x_1 + w_2 \cdot x_2 + w_1 \cdot x_3$$

- 互相关（信号分析里的概念）：不需要进行卷积核翻转

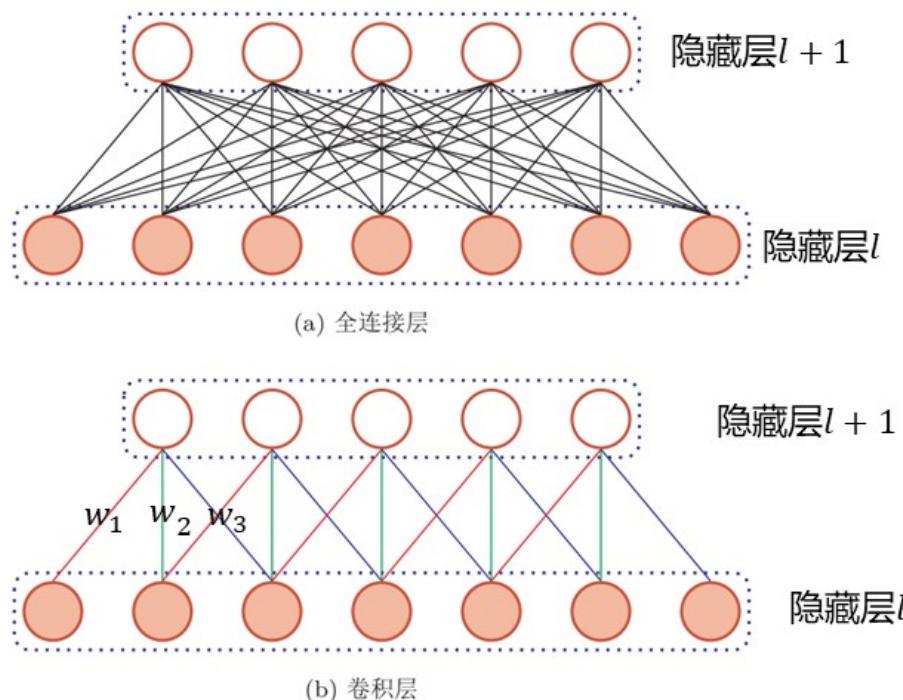
$$\begin{matrix} w_1 & w_2 & w_3 \\ x_1 & x_2 & x_3 \end{matrix} \xrightarrow{\quad} w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3$$

- 卷积神经网络中，“卷积”操作的目标是提取特征
  - 卷积核中的权重 $W$ 都是网络学习到的参数，只是下标问题，翻转是不必要的

默认情况下，神经网络里提到的卷积就是互相关

# 前馈神经网络 卷积神经网络

用卷积层代替全连接层：把层间的计算关系转换成卷积



## 优化：

1. 邻接数减少
2. 神经元参数大大较少
  - 卷积神经网络权重共享，偏置共享，只有4个参数

$$h^{l+1} = f(w * h^l) + b$$



要学习参数：

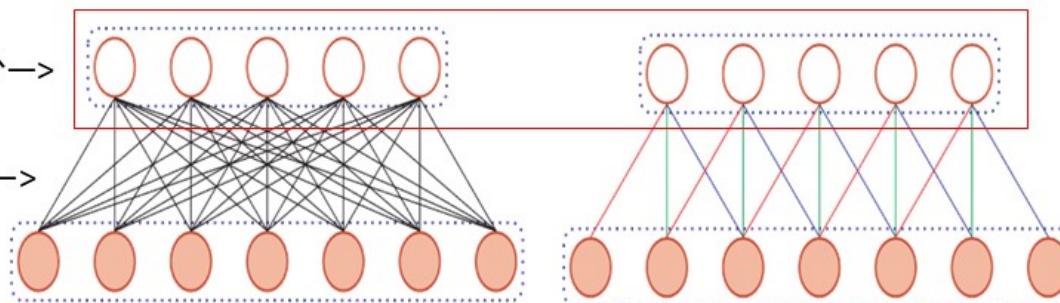
**卷积**: 一个卷积核及偏置

**全连接**: 两层节点数的乘积

## 前馈神经网络 卷积神经网络-汇聚层/池化层 (pooling layer)

卷积层虽然可以显著减少连接的个数，但是每一个特征映射的神经元个数并没有显著减少。

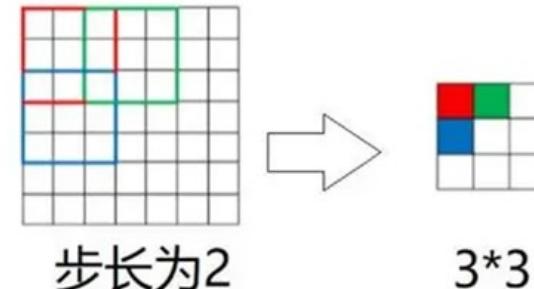
神经元个数并没有显著减少 ->



解决方法：

方式1：加大步长

方式2：引入**池化pooling**——划分区域，并取代表性的值



# 前馈神经网络 卷积神经网络-汇聚层/池化层 (pooling layer)

**池化 (Pooling) :** 在一个小窗口 (比如 $2 \times 2$ ) 里, 只选择这个窗口里最突出的那个信息, 并将其作为整个窗口的代表。

**作用:**

1. **降采样**: 减少空间分辨率 (宽和高), 从而降低计算量。
2. **特征抽象**: 保留最显著的特征 (如最大池化保留边缘/纹理的强响应), 增强模型的鲁棒性。
3. **增加局部不变性**: 小范围的输入位移不会显著影响池化结果。

**常见池化方式:**

- 均值池化: 取窗口内元素的均值。
- 最大池化: 取窗口内元素的最大值。

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

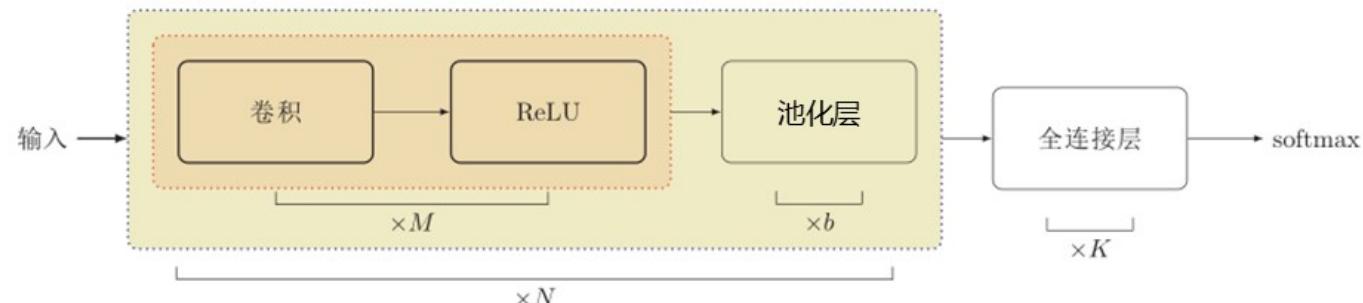
(a) Average pooling

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(b) Max Pooling

## 前馈神经网络 卷积神经网络 – 网络结构

- 卷积网络是由卷积层、池化层、全连接层交叉堆叠而成，趋向于小卷积（ $3 \times 3, 1 \times 1$ ）、大深度
- 典型结构

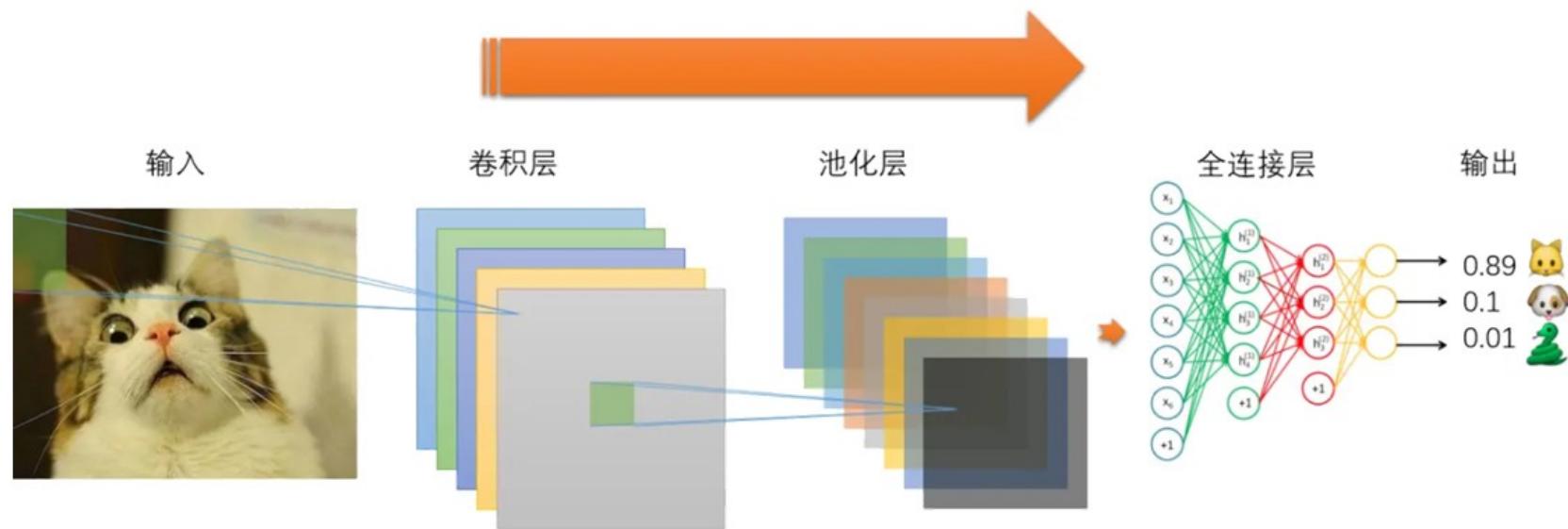


**一个卷积块：**连续M个卷积层和b个池化层  
(M通常设置为2 ~ 5, b为0或1)。

**一个卷积网络：**堆叠N个连续的卷积块，然后在接着K个全连接层 (N的取值区间比较大，比如1 ~ 100或者更大；K一般为0 ~ 2)。

# 前馈神经网络 卷积神经网络 – 网络结构

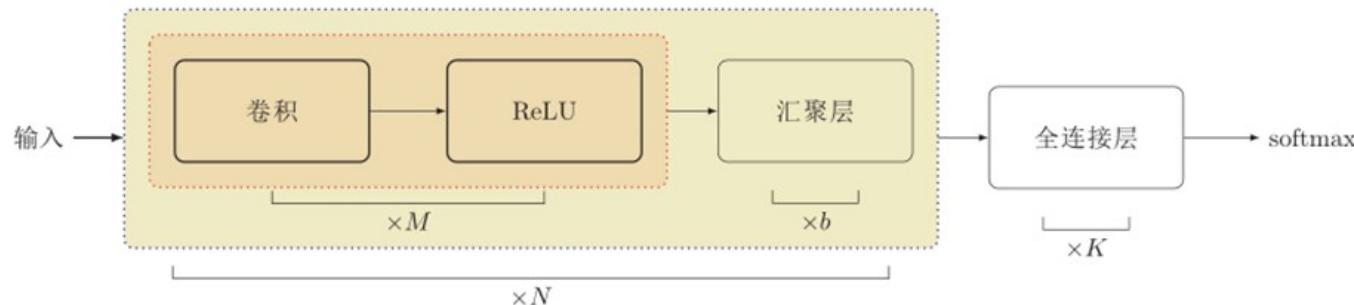
图像分类



## 主观题 10分

已知该CNN由一层卷积+一层池化+一层全连接组成

- 输入：2\*2图像  $\begin{bmatrix} [1, 2], [3, 4] \end{bmatrix}$
  - 卷积核：3\*3  $\begin{bmatrix} [1, 0, 0], [0, 1, 0], [0, 0, 1] \end{bmatrix}$ , 步长为1, 填充为1, 采用零填充, 偏置为0
  - 激活函数为ReLU
  - 池化：2\*2最大池化
  - 全连接层：输入维度为池化后的大小，输出维度为1，权重w全为1,偏置为1
- 求该CNN全连接层的输出结果：



## Practice

已知该CNN由一层卷积+一层池化+一层全连接组成

- 输入：2\*2图像  $\begin{bmatrix} 1, 2 \\ 3, 4 \end{bmatrix}$
- 卷积核：3\*3  $\begin{bmatrix} 1, 0, 0 \\ 0, 1, 0 \\ 0, 0, 1 \end{bmatrix}$ , 步长为1, 填充为1, 采用零填充, 偏置为0
- 激活函数为Relu
- 池化：2\*2最大池化
- 全连接层：输入维度为池化后的大小，输出维度为1，权重w全为1,偏置为1

求该CNN全连接层的输出结果：

**输入数据**

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

**填充后的输入**

$$X_{\text{padded}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 3 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

**卷积计算**

$$\begin{bmatrix} 5 & 2 \\ 3 & 5 \end{bmatrix}$$

**ReLU激活**

$$C_{\text{out}} = \begin{bmatrix} 5 & 2 \\ 3 & 5 \end{bmatrix}$$

**池化输出**

$$P_{\text{out}} = [5]$$

**全连接层**

$$\text{输出}=[6]$$

# Lecture Plan

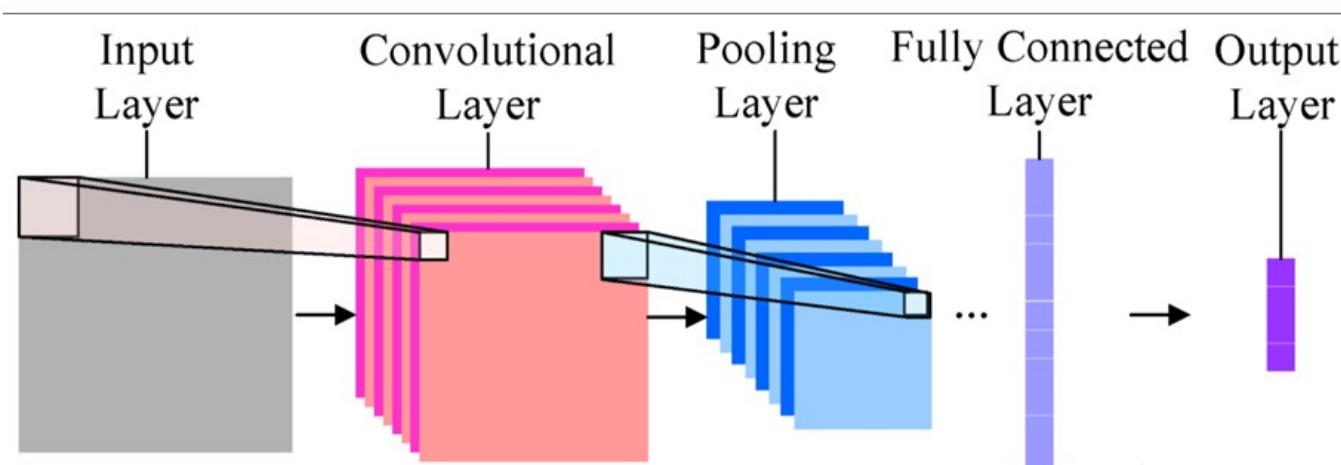
- 神经网络基础知识
- 前馈神经网络
- 反馈神经网络



78

## 反馈神经网络 前馈神经网络的不足

- 连接存在于层与层之间，每层的节点之间是**无连接的。无循环，无记忆。**
- 输入和输出的维数都是固定的，不能任意改变。**无法处理变长的序列数据。**
- 每次输入都是独立的，每次网络的输出**只依赖于当前的输入。**



## 前馈神经网络 前馈神经网络的应用

**前馈神经网络特别适合处理具有以下特点的数据：**

**□ 固定维度输入**

所有样本必须具备相同数量的特征，这直接决定了网络输入层的结构，使得模型设计得以简化。

**□ 独立同分布数据**

数据点之间不应存在时序或空间上的依赖关系，从而保证模型可以逐个样本进行学习，而无需考虑上下文信息。

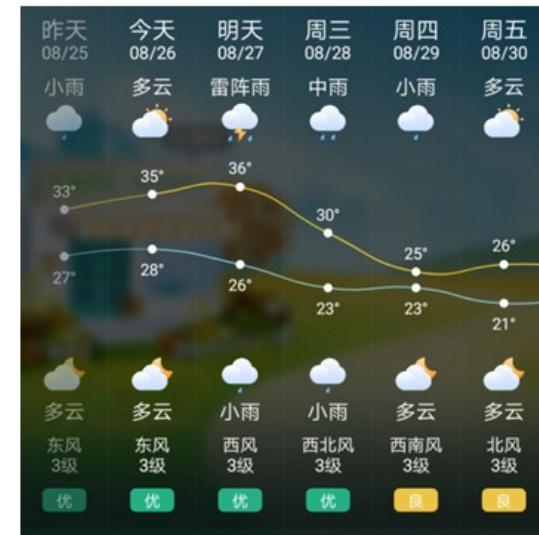
**□ 结构化的向量表示**

数据最好以结构化的向量形式呈现，例如表格数据（如CSV文件、数据库表）或将图像展平后得到的像素值向量。

# 反馈神经网络 序列数据

序列数据：文本/语音/等

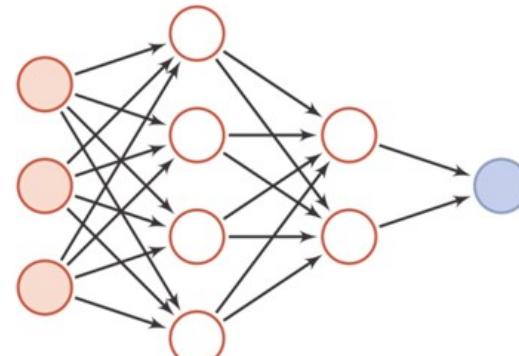
- 具有很强的时间依赖关系，而传统神经网络输出只依赖于当前的输入，无法充分利用前期信息来指导后续决策。
- 输入输出长度不固定



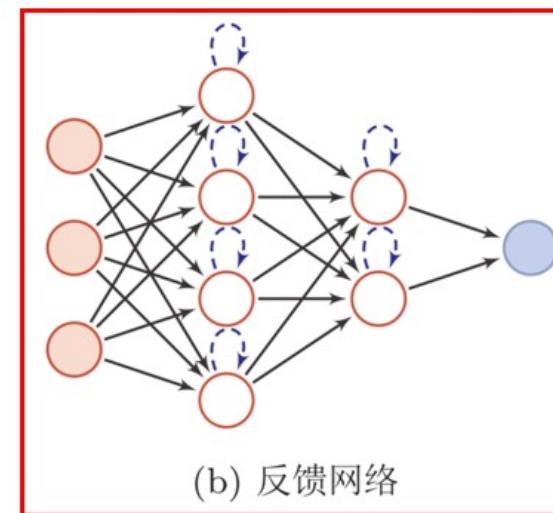
# 反馈神经网络

## ➤ 给网络增加记忆功能

- 通过使用带自反馈的神经元，能够处理**长度不一**的序列。
- 比前馈神经网络更加符合生物神经网络的结构。
- 广泛应用在语音识别、语言模型以及自然语言生成等任务上。



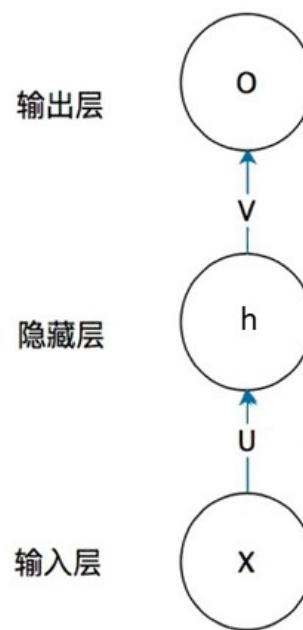
(a) 前馈网络



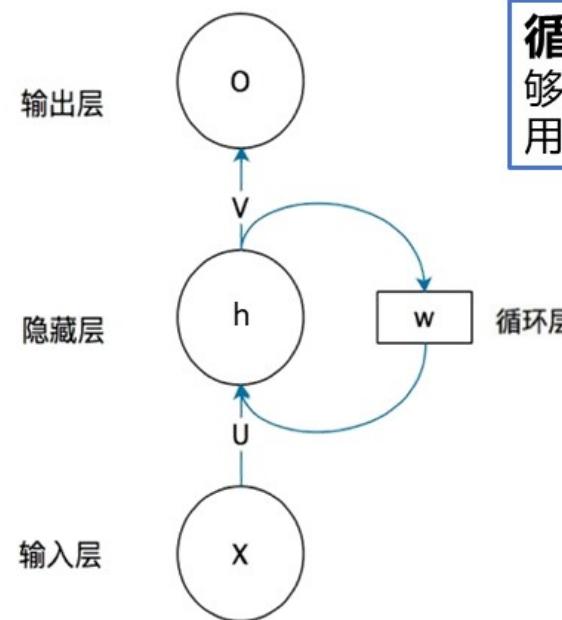
(b) 反馈网络

# 反馈神经网络 循环神经网络 (Recurrent Neural Network, RNN)

## 基本结构



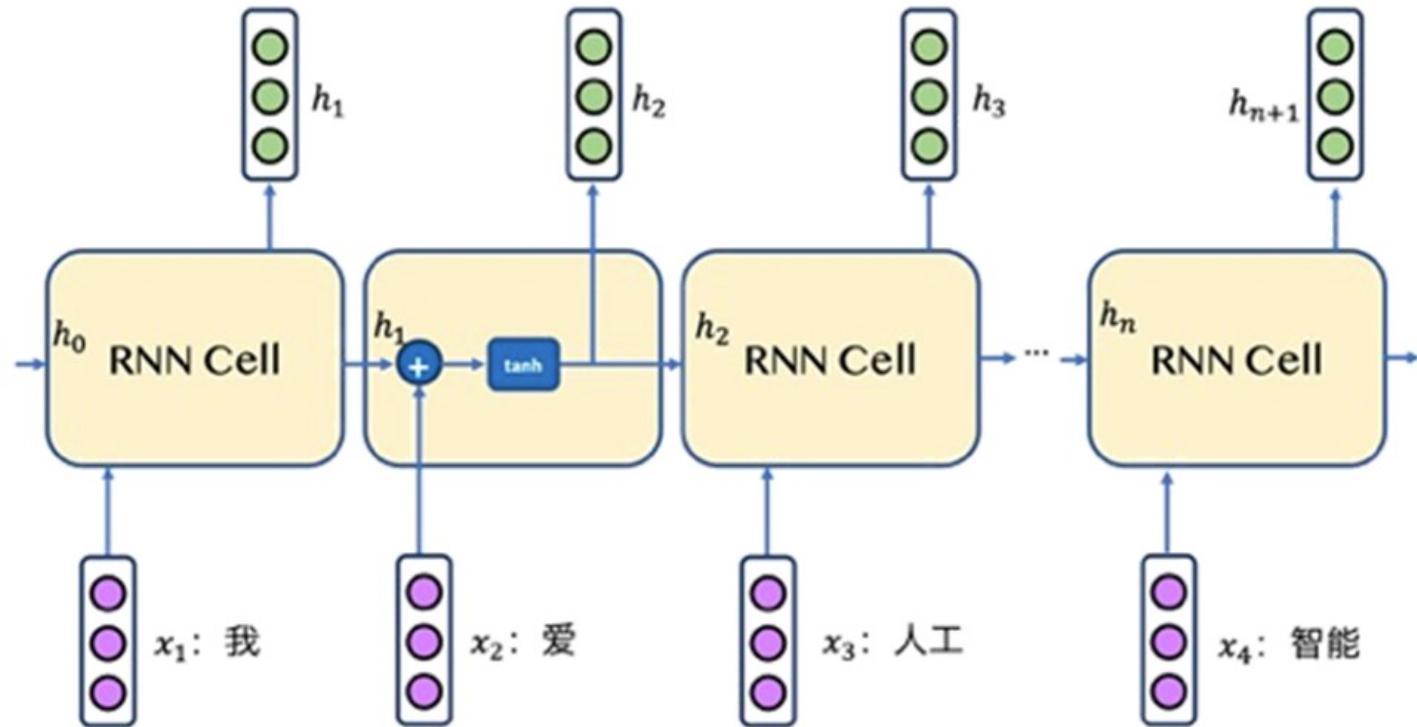
全连接神经网络



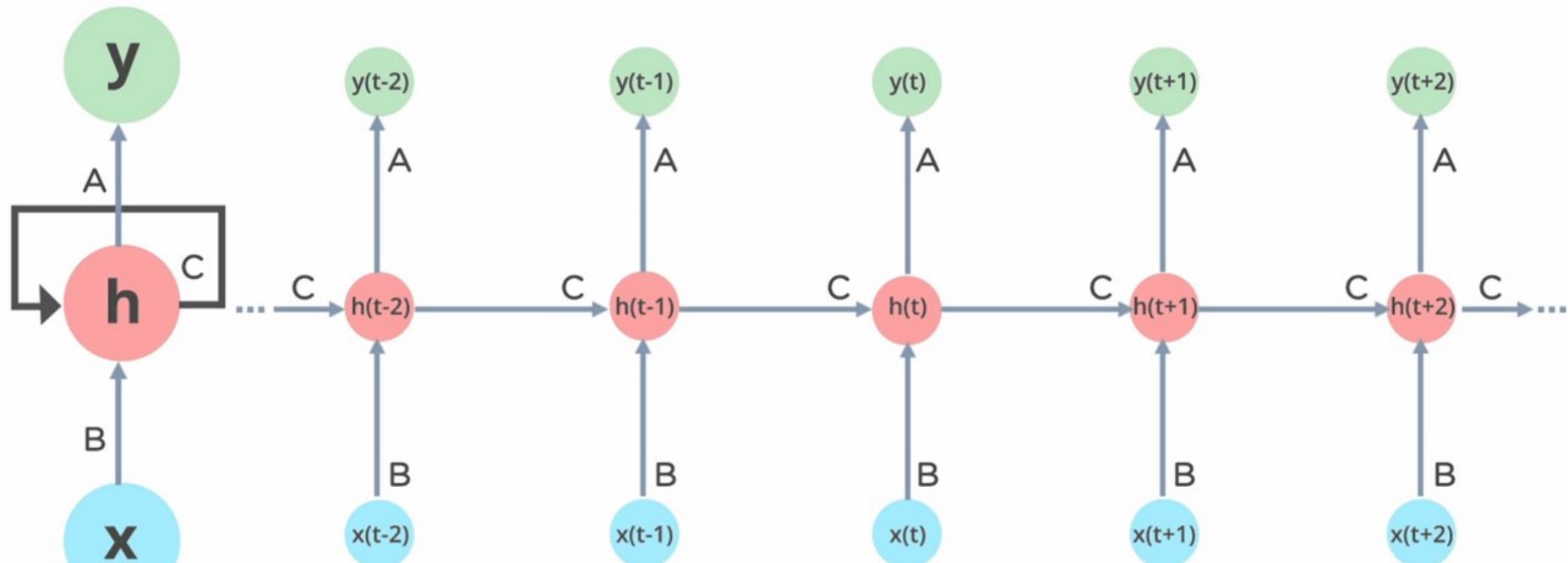
循环神经网络

循环连接，使得网络能够保留之前时间步的信息，用于当前时间步的计算。

## 反馈神经网络 循环神经网络 (Recurrent Neural Network, RNN)



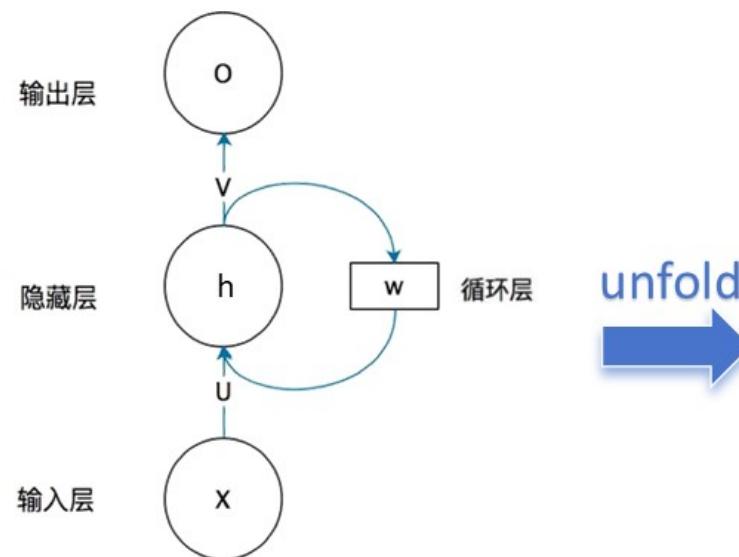
## 反馈神经网络 循环神经网络 (Recurrent Neural Network, RNN)



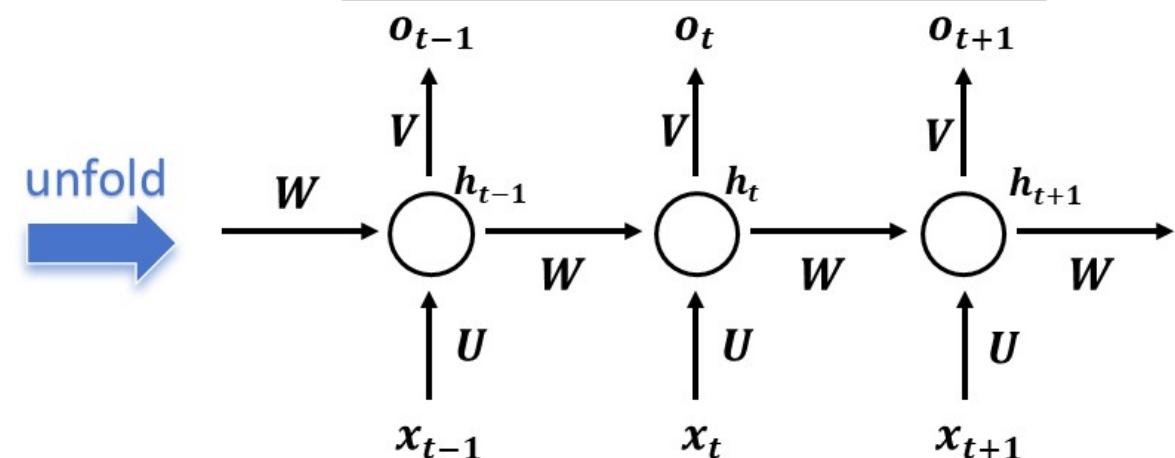
# 反馈神经网络 循环神经网络 ( Recurrent Neural Network, RNN)

- 基本结构

- ○ 表示隐藏层的一个unit (单元)



记号	含义
$x_t$	t时刻的输入
$o_t$	t时刻的输出
$h_t$	t时刻隐藏层的输出
$W, U, V$	所有单元的共享权重矩阵参数



RNN就像一个**有记忆的人**，他在阅读一句话时，会边读边记住前面看到的内容，从而理解整个句子的意思。

## 反馈神经网络 循环神经网络 – 前向传播

通用公式表示

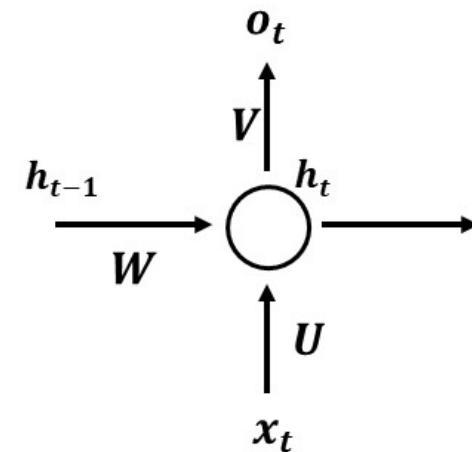
$$h_0 = 0$$

$$h_t = g_1(U \cdot x_t + W \cdot h_{t-1} + b_a)$$

$$o_t = g_2(V \cdot h_t + b_y)$$

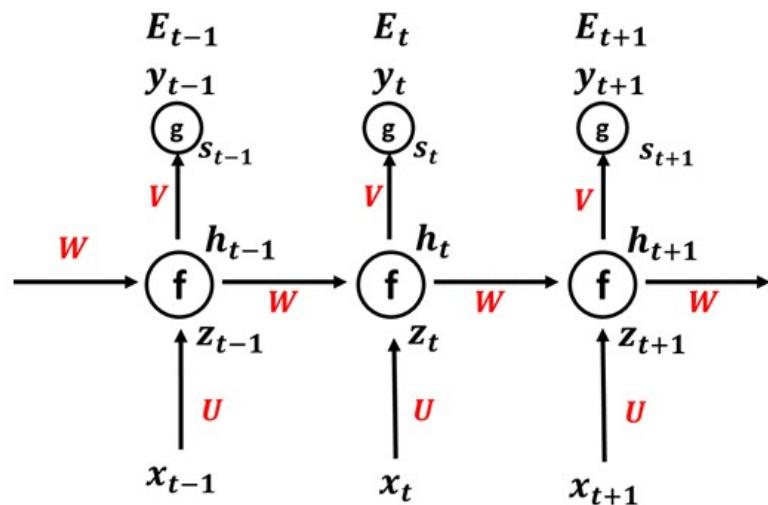
其中， $g_1, g_2$  表示激活函数：

- $g_1$  常使用tanh/relu,
- $g_2$  常用sigmoid、softmax



# 反馈神经网络 循环神经网络 – 前向传播

## 前向传播



$$\begin{cases} z_t = \mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1} \\ \mathbf{h}_t = f(z_t) \\ \mathbf{s}_t = \mathbf{V}\mathbf{h}_t \\ \mathbf{y}_t = g(s_t) \end{cases}, \quad t \in [1, 2, 3, \dots, L]$$

$$\begin{aligned} y_t &= g(\mathbf{V}\mathbf{h}_t) \\ &= g(\mathbf{V}f(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1})) \\ &= g(\mathbf{V}f(\mathbf{U}\mathbf{x}_t + \mathbf{W}(f(\mathbf{U}\mathbf{x}_{t-1} + \mathbf{W}\mathbf{h}_{t-2})))) \\ &= g(\mathbf{V}f(\mathbf{U}\mathbf{x}_t + \mathbf{W}(f(\mathbf{U}\mathbf{x}_{t-1} + \mathbf{W}f(\mathbf{U}\mathbf{x}_{t-2} + \mathbf{W}\mathbf{h}_{t-3})))))) \\ &= g(\mathbf{V}f(\mathbf{U}\mathbf{x}_t + \mathbf{W}(f(\mathbf{U}\mathbf{x}_{t-1} + \mathbf{W}f(\mathbf{U}\mathbf{x}_{t-2} + \mathbf{W}f(\mathbf{U}\mathbf{x}_{t-3} + \dots))))))) \end{aligned}$$

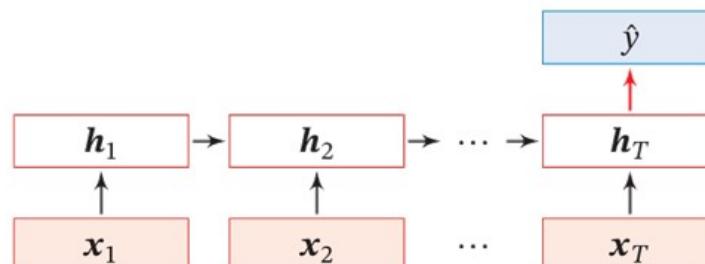
每一次的输出都与之前的输入有着千丝万缕的联系。。。

# 反馈神经网络 循环神经网络应用

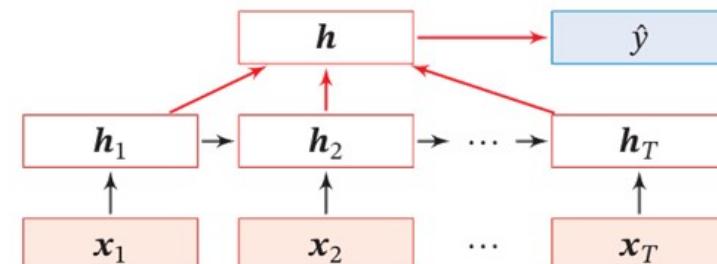
## ➤ 序列到类别

- 输入：序列
- 输出：类别

- 正常模式： $h_t$  中包含了所有的历史状态信息，将  $h_t$  放入分类器中得到类别。
- 平均采样：将所有历史信息取平均后得到  $h$ ，再输入分类器得到类别



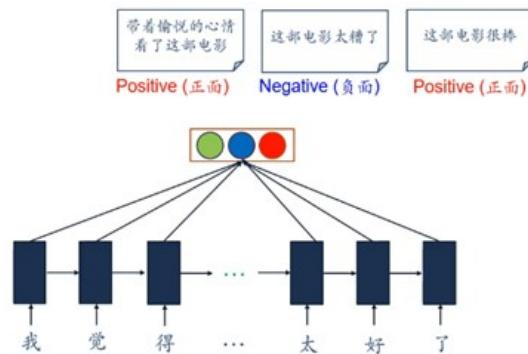
(a) 正常模式



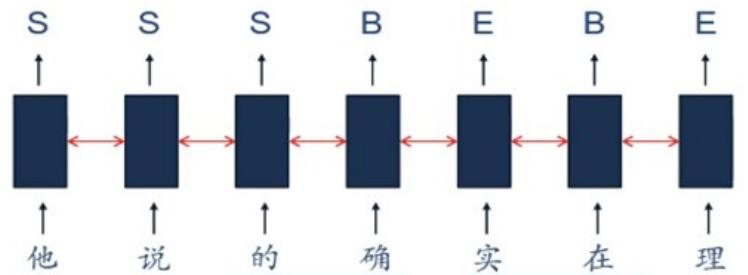
(b) 按时间进行平均采样模式

# 反馈神经网络 循环神经网络应用

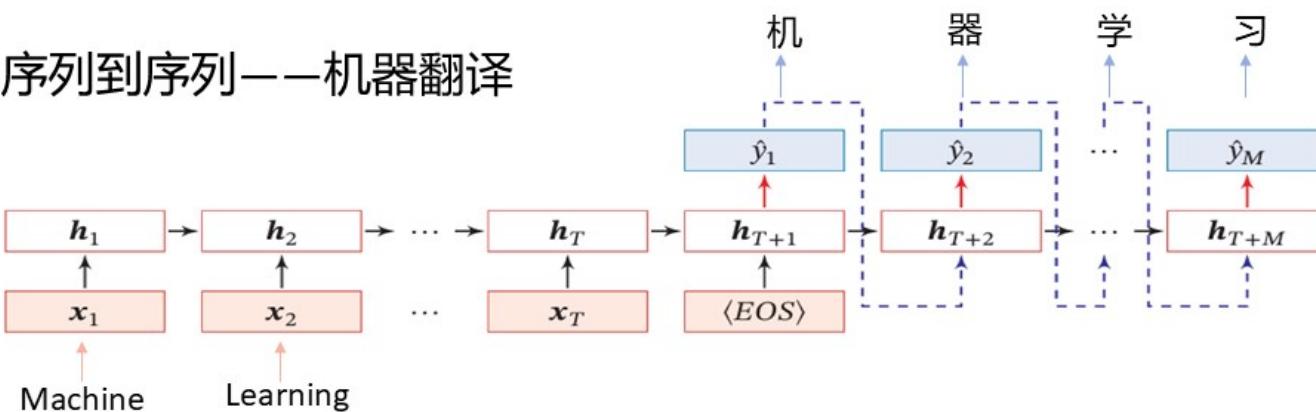
## ➤ 序列到类别——情感分类



## ➤ 同步的序列到序列——中文分词



## ➤ 异步的序列到序列——机器翻译



# 人工智能导论

Introduction to Artificial Intelligence

谢谢！



90