



机器学习（研究生）

# 神经网络与深度学习

Neural Networks & Deep Learning

赵凯 上海大学

<https://kaizhao.net/teaching/shu-ml-grad>

# 本讲内容索引

## 01 神经元

神经元模型 · 激活函数

## 02 多层感知机

单层网络的局限 · 多层级联与前向传播 · 万能近似 · 损失函数与训练

## 03 反向传播

计算图与链式法则 · 误差回传与参数梯度 · 常见算子 · 算法 · 自动微分

## 04 卷积神经网络

局部连接与权值共享 · 卷积、填充、步幅、池化 · 典型结构 · 残差连接

## 05 循环神经网络

序列数据 · 循环计算 · 单向与遗忘 · CNN 与 RNN 的局限

## 06 注意力机制与 Transformer

注意力：软查找 · 缩放点积与自注意力 · 多头与位置编码 · Transformer 及其变体

## 07 小结

本讲要点 · 课后作业

# 本讲内容

从单个神经元出发，到卷积网络、循环网络与 Transformer

## Part 1 神经网络

由简单单元逐层复合而成的可学习函数

神经元

多层感知机

反向传播

## Part 2 深度学习

把数据的结构写进网络结构

卷积神经网络

循环神经网络

注意力机制与 Transformer

# 符号约定

沿用前几讲的记号；新增与网络层有关的记号

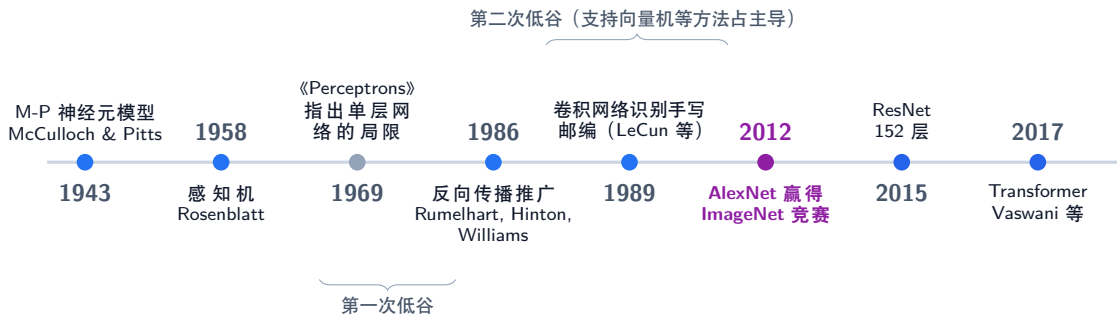
---

|     |                                                                                                                                                                                                                  |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 数据  | $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$ ; $\mathbf{x} \in \mathbb{R}^d$ 为输入, $\mathbf{y}$ 为标签 (分类时为独热向量)                                                                                                  |
| 层   | 共 $L$ 层, 第 $l$ 层有 $n_l$ 个神经元, $n_0 = d$ ; 上标 $^{(l)}$ 表示第 $l$ 层                                                                                                                                                  |
| 参数  | 权重矩阵 $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ , 偏置 $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ ; 全部参数记为 $\theta$                                                                                       |
| 前向量 | 净输入 $\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ , 激活值 $\mathbf{a}^{(l)} = \sigma(\mathbf{z}^{(l)})$ , $\mathbf{a}^{(0)} = \mathbf{x}$ , 输出 $\hat{\mathbf{y}} = \mathbf{a}^{(L)}$ |
| 损失  | 单个样本的损失 $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ ; 训练目标 $\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i)$                                                        |

---

记号  $\sigma(\cdot)$ : 激活函数, 作用于向量时逐元素计算;  $\odot$ : 逐元素乘积; 独热向量: 第  $c$  类对应的分量为 1、其余为 0 的向量。Bishop (2006) 第 5 章记净输入为  $\mathbf{a}$ 、激活值为  $\mathbf{z}$ , 与此处相反; 周志华《机器学习》第 5 章用阈值  $\theta$  ( $b = -\theta$ )。

# 神经网络的发展脉络



**要点** 2012 年 AlexNet 以来的进展依赖三个条件：大规模标注数据（ImageNet）、图形处理器（GPU）算力、训练技术（新的激活函数、Dropout、归一化、残差连接）。

# 01

# 神经元

Neurons

神经元模型 · 激活函数

# 神经元

多个输入，一个输出：对输入加权求和，再经过非线性激活

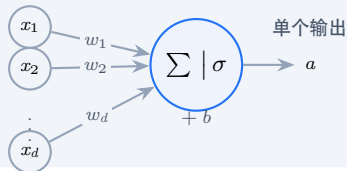
## 生物神经元

树突：多路输入



## 人工神经元

多个输入

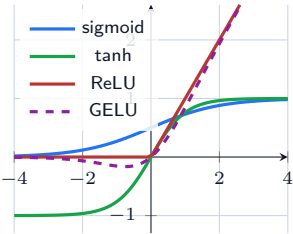


$$a = \sigma \left( \sum_{j=1}^d w_j x_j + b \right) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$$

记号  $x_j$ : 第  $j$  个输入;  $w_j$ : 连接权重;  $b$ : 偏置;  $\sigma$ : 激活函数;  $a$ : 输出 (一个数), 可同时送往下一层的多个神经元。

# 常用激活函数

激活函数为网络引入非线性；需要（几乎处处）可导，才能用梯度训练



| 名称      | $\sigma(z)$                         | $\sigma'(z)$       | 代表模型                  |
|---------|-------------------------------------|--------------------|-----------------------|
| sigmoid | $\frac{1}{1 + e^{-z}}$              | $\leq \frac{1}{4}$ | LSTM 的门控；二分类输出层       |
| tanh    | $\frac{e^z - e^{-z}}{e^z + e^{-z}}$ | $\leq 1$           | LeNet-5；RNN、LSTM 的隐状态 |
| ReLU    | $\max(0, z)$                        | 0 或 1              | AlexNet、VGG、ResNet    |
| GELU    | $z\Phi(z)$                          | 光滑                 | BERT、GPT-2、ViT        |
| SiLU    | $\frac{z}{1 + e^{-z}}$              | 光滑                 | LLaMA、PaLM (SwiGLU)   |

修正线性单元（Rectified Linear Unit, ReLU）计算简单、正半轴导数为 1，是 CNN 的默认选择；Transformer 多用 GELU 或 SiLU。

记号  $\Phi$ ：标准正态分布函数；GELU：Gaussian Error Linear Unit；SiLU：Sigmoid Linear Unit；SwiGLU：以 SiLU 为门控的前馈层；LSTM：长短期记忆网络（Long Short-Term Memory）；RNN：循环神经网络（Recurrent Neural Network）；CNN：卷积神经网络（Convolutional Neural Network）；BERT、GPT、ViT 见第 6 节；图中 GELU 用近似式  $z\sigma(1.702z)$  绘制。

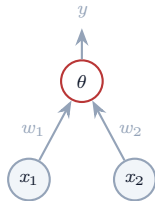
# M-P 神经元与感知机

感知机（perceptron）：两层神经元，输出层是一个 M-P 神经元（阈值逻辑单元）

$$y = f\left(\sum_{i=1}^d w_i x_i - \theta\right)$$

$$f(z) = \text{sgn}(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

$\theta$ ：阈值； $f$  为单位阶跃函数



## 实现逻辑运算

| 运算                 | 参数                                   |
|--------------------|--------------------------------------|
| 与 $x_1 \wedge x_2$ | $w_1 = w_2 = 1, \theta = 2$          |
| 或 $x_1 \vee x_2$   | $w_1 = w_2 = 1, \theta = 0.5$        |
| 非 $\neg x_1$       | $w_1 = -0.6, w_2 = 0, \theta = -0.5$ |

**问题** 这些参数是人为给定的。给定训练数据，权重  $w_i$  与阈值  $\theta$  能否自动学得？

记号 M-P 神经元：McCulloch & Pitts (1943)；感知机：Rosenblatt (1958)。例：“与”运算  $y = f(x_1 + x_2 - 2)$ ，仅当  $x_1 = x_2 = 1$  时  $y = 1$ 。此处  $\text{sgn}$  沿用书中记号，指取值  $\{0, 1\}$  的单位阶跃函数（图 5.2），与通常取值  $\{-1, 0, 1\}$  的符号函数不同；与神经元一页的偏置关系为  $b = -\theta$ 。周志华《机器学习》第 5 章 5.1–5.2 节，图 5.1–5.3，pp.97–99（与、或、非的参数取自 pp.98–99；书中输入个数记  $n$ ）。

# 感知机学习规则

把阈值  $\theta$  看作固定输入为  $-1.0$  的“哑结点”（dummy node）的连接权重  $w_{d+1}$ ：权重与阈值统一学习

对训练样例  $(x, y)$ ，若当前感知机的输出为  $\hat{y}$ ：

$$w_i \leftarrow w_i + \Delta w_i, \quad \Delta w_i = \eta(y - \hat{y})x_i$$

对阈值  $(x_{d+1} = -1)$ ：

$$\theta \leftarrow \theta - \eta(y - \hat{y})$$

- 预测正确 ( $\hat{y} = y$ )：不发生变化
- 预测错误：按错误的程度调整权重

**例：学习“与”运算**， $\eta = 0.1$

初值  $w_1 = 0.2$ ， $w_2 = -0.1$ ， $\theta = 0.3$ ；按  $(0, 0), (0, 1), (1, 0), (1, 1)$  循环输入

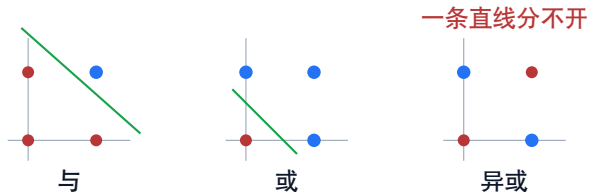
| 样例     | $y$ | $\hat{y}$ | 更新后 $(w_1, w_2, \theta)$ |
|--------|-----|-----------|--------------------------|
| (1, 1) | 1   | 0         | (0.3, 0.0, 0.2)          |
| (1, 0) | 0   | 1         | (0.2, 0.0, 0.3)          |
| (1, 1) | 1   | 0         | (0.3, 0.1, 0.2)          |
| (1, 0) | 0   | 1         | (0.2, 0.1, 0.3)          |

此后四个样例全部判对，学习结束；结果为  $y = f(0.2x_1 + 0.1x_2 - 0.3)$

记号  $\eta \in (0, 1)$ ：学习率，常设为 0.1；表中只列出判错、发生更新的样例；最终在 (1, 1) 处  $w^\top x - \theta = 0$ ，按  $\text{sgn}(0) = 1$  判为 1。  
 例：第一行  $\hat{y} = \text{sgn}(0.2 - 0.1 - 0.3) = 0$ ， $\Delta w_1 = 0.1 \times (1 - 0) \times 1 = 0.1$ 。学习规则：周志华《机器学习》第 5 章 5.2 节，式 (5.1)–(5.2)，p.99；数值迭代为补充示例（非教材例题）。

# 感知机的局限：线性可分

感知机只有一层功能神经元，学习能力非常有限



蓝点输出 1，红点输出 0；绿线为划分超平面

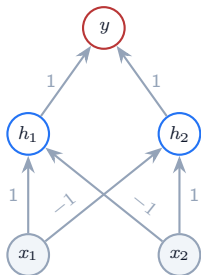
- 与、或、非问题都是**线性可分**的：学习过程一定收敛
- 线性不可分时，学习过程**振荡**， $w$ 难以稳定
- 感知机甚至不能解决**异或**问题 (Minsky & Papert, 1969)

**要点** 解决非线性可分问题需要**多层功能神经元**：加入隐层，即下一节的多层感知机。

记号 线性可分 (linearly separable): 存在一个线性超平面能将两类模式分开。周志华《机器学习》第 5 章 5.2 节，图 5.4，pp.99–100。

## 例：两层感知机解决异或问题

加入一层隐层（两个功能神经元），三个神经元的阈值均为 0.5



$$h_1 = f(x_1 - x_2 - 0.5), \quad h_2 = f(-x_1 + x_2 - 0.5)$$
$$y = f(h_1 + h_2 - 0.5)$$

| $x_1$ | $x_2$ | $h_1$ | $h_2$ | $y$ |
|-------|-------|-------|-------|-----|
| 0     | 0     | 0     | 0     | 0   |
| 0     | 1     | 0     | 1     | 1   |
| 1     | 0     | 1     | 0     | 1   |
| 1     | 1     | 0     | 0     | 0   |

**要点** 隐层把输入映射为新的表示  $(h_1, h_2)$ ，在新表示上异或变为线性可分。

记号  $f$ : 阶跃函数。例题：周志华《机器学习》第 5 章 5.2 节，图 5.5，p.100。

# 02

## 多层感知机

Multilayer Perceptrons

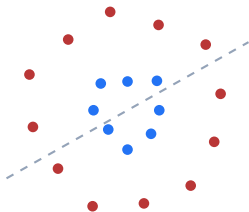
单层网络的局限 · 多层级联与前向传播 · 万能近似 · 损失函数与训练

# 单层网络的局限

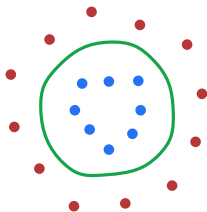
一层神经元： $\hat{y} = \sigma(Wx + b)$ ，表示能力有限

单层：直线边界

多层：弯曲边界



无论如何放置都分不开



正确分开

- 每个输出只是输入的加权和再经一次激活
- 分类：决策边界  $w^T x + b = 0$  是超平面
- 回归：只能拟合线性函数（或其单调变换）
- 图像、语音、文本中，输入与输出的关系高度非线性

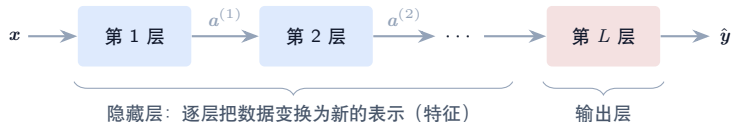
**要点** 单层模型过于简单，无法拟合复杂函数；需要把多层神经元级联起来。

记号  $W, b$ ：该层的权重矩阵与偏置； $\sigma$ ：激活函数；图中蓝、红两类样本分别分布在内圈与外圈。

# 多层级联

把简单的层逐层复合，得到复杂的函数

$$f(\mathbf{x}) = f^{(L)}\left(\dots f^{(2)}\left(f^{(1)}(\mathbf{x})\right)\right), \quad f^{(l)}(\mathbf{a}) = \sigma(\mathbf{W}^{(l)}\mathbf{a} + \mathbf{b}^{(l)})$$



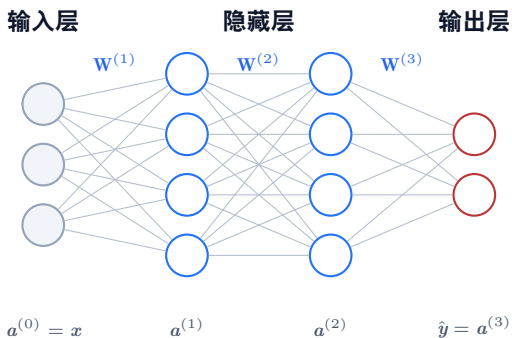
- 每一层只做一次简单变换；复合后可表示复杂函数
- 输出层在最后一层表示上做线性分类或回归

**要点** 隐藏层的表示从数据中学习得到；核方法中的特征映射  $\phi$  则是人为选定的。

记号  $f^{(l)}$ : 第  $l$  层的映射;  $a^{(l)}$ : 第  $l$  层的输出（激活值）;  $L$ : 层数;  $\phi$ : 核方法中的特征映射（第 3 次课）。

# 多层感知机的结构

多层感知机（Multilayer Perceptron, MLP），亦称多层前馈神经网络：相邻层全连接，同层无连接，无跨层连接



- 图中  $L = 3$ ：两个隐藏层、一个输出层；输入层不计入层数
- 宽度  $n_0 = 3, n_1 = n_2 = 4, n_3 = 2$
- 第  $l$  层参数： $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}, b^{(l)} \in \mathbb{R}^{n_l}$
- 参数总数  $\sum_l (n_l n_{l-1} + n_l)$ ；本图为  $16 + 20 + 10 = 46$

**深度学习中的“深”** 指隐藏层的层数多。层数越多，可表示的特征层次越丰富。

记号  $a^{(l)}$ ：第  $l$  层的激活值； $W^{(l)}, b^{(l)}$ ：第  $l$  层的权重与偏置； $n_l$ ：第  $l$  层的神经元个数； $L$ ：层数。

# 前向传播

从输入层逐层计算到输出层；隐藏层是“仿射变换 + 逐元素非线性”，输出层的非线性按任务选择

## 单个样本

$$\mathbf{a}^{(0)} = \mathbf{x}$$

对  $l = 1, \dots, L$ :

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

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

输出  $\hat{y} = \mathbf{a}^{(L)}$

## 小批量 ( $B$ 个样本按行堆叠)

$$\mathbf{A}^{(0)} = \mathbf{X} \in \mathbb{R}^{B \times d}$$

$$\mathbf{Z}^{(l)} = \mathbf{A}^{(l-1)} (\mathbf{W}^{(l)})^\top + \mathbf{1} (\mathbf{b}^{(l)})^\top$$

$$\mathbf{A}^{(l)} = \sigma^{(l)}(\mathbf{Z}^{(l)}) \in \mathbb{R}^{B \times n_l}$$

一次矩阵乘法处理整批样本，适合 GPU 并行。

例：MNIST 手写数字， $28 \times 28 = 784$  维输入，一个 256 单元的隐藏层，10 类输出。参数量  $784 \times 256 + 256 + 256 \times 10 + 10 = 203,530$ 。

记号  $\mathbf{z}^{(l)}$ ：净输入； $\sigma^{(l)}$ ：第  $l$  层的激活函数（输出层常与隐藏层不同）； $\mathbf{X}$ ：小批量输入矩阵； $B$ ：批大小； $\mathbf{1}$ ：全 1 列向量。MNIST 参数量为补充示例（非教材例题）。

# 非线性激活的必要性

去掉激活函数，多层网络退化为一个线性模型

$$\mathbf{W}^{(2)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} = \underbrace{\mathbf{W}^{(2)}\mathbf{W}^{(1)}}_{\mathbf{W}'}\mathbf{x} + \underbrace{\mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}}_{\mathbf{b}'}$$

- 任意多个线性层的复合仍是一个线性层
- 加深网络不增加表示能力
- 决策边界仍是超平面

**要点** 激活函数在层与层之间插入非线性，复合后的函数才能表示弯曲的决策边界。

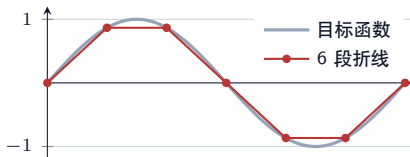
记号  $\mathbf{W}'$ ,  $\mathbf{b}'$ : 复合后等价的单层权重与偏置。

# 万能近似定理

Cybenko (1989); Hornik (1991)

只含一个隐藏层的前馈网络，隐藏层激活函数连续且非多项式（如 sigmoid、ReLU）、输出层为线性组合时，只要隐藏单元足够多，就能以任意精度逼近定义在  $\mathbb{R}^d$  有界闭集上的任意连续函数。

直观：  $f(x) = c_0 + \sum_{k=1}^n c_k \text{ReLU}(x - t_k)$  是分段线性函数（折线）；分段越密，越贴合目标曲线。



**定理没有回答的问题** 所需单元数可能随维数指数增长；定理只保证网络存在，不保证梯度下降找得到，也不保证泛化。

记号  $c_k$ : 输出层权重； $t_k$ : 第  $k$  个折点的位置（由隐藏层偏置决定）； $n$ : 隐藏单元个数。Hornik 等 (1989); Leshno 等 (1993) 给出非多项式条件；周志华《机器学习》第 5 章 5.3 节, p.105。

# 输出层与损失函数

按任务选择输出层的激活函数与对应的损失；三者都等价于极大似然估计

| 任务     | 输出层                                                         | 损失函数 $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$       |
|--------|-------------------------------------------------------------|--------------------------------------------------------|
| 回归     | 恒等: $\hat{\mathbf{y}} = \mathbf{z}^{(L)}$                   | 平方损失 $\frac{1}{2} \ \hat{\mathbf{y}} - \mathbf{y}\ ^2$ |
| 二分类    | sigmoid: $\hat{y} = \sigma(\mathbf{z}^{(L)})$               | 二元交叉熵 $-[y \ln \hat{y} + (1 - y) \ln(1 - \hat{y})]$    |
| $K$ 分类 | softmax: $\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$ | 交叉熵 $-\sum_{k=1}^K y_k \ln \hat{y}_k = -\ln \hat{y}_c$ |

**要点** 平方损失对应高斯噪声下的极大似然；交叉熵对应伯努利 / 类别分布下的极大似然。  
分类任务用交叉熵而非平方损失：sigmoid 饱和时平方损失的梯度趋于 0。

记号  $z_k$ : 输出层净输入  $\mathbf{z}^{(L)}$  的第  $k$  个分量； $c$ : 样本的真实类别； $y_k$ : 独热标签的第  $k$  个分量。

# softmax 与交叉熵：数值例子

三分类，真实类别  $c = 1$

| 类别 $k$            | 1      | 2     | 3     |
|-------------------|--------|-------|-------|
| 净输入 $z_k$         | 2.0    | 1.0   | 0.1   |
| $e^{z_k}$         | 7.389  | 2.718 | 1.105 |
| $\hat{y}_k$       | 0.659  | 0.242 | 0.099 |
| 标签 $y_k$          | 1      | 0     | 0     |
| $\hat{y}_k - y_k$ | -0.341 | 0.242 | 0.099 |

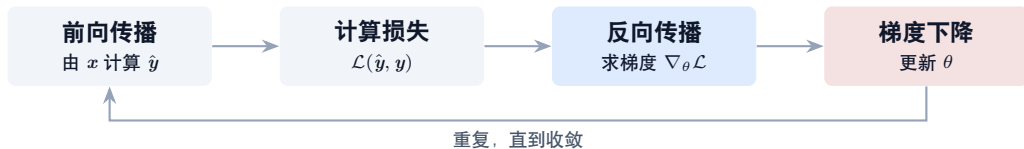
$$\mathcal{L} = -\ln \hat{y}_1 = -\ln 0.659 = 0.417$$

- softmax 把任意实数向量变成概率分布
- 预测越确信正确类别，损失越小
- 损失对净输入的梯度恰为  $\hat{y} - y$

记号  $z_k$ : 输出层第  $k$  个净输入;  $\hat{y}_k$ : softmax 输出的第  $k$  类概率;  $y_k$ : 独热标签;  $\mathcal{L}$ : 交叉熵损失。补充示例（非教材例题）。

# 训练：反向传播 + 梯度下降

目标：最小化训练集上的平均损失  $\mathcal{L}(\theta) = \frac{1}{m} \sum_i \mathcal{L}(\hat{y}_i, y_i)$



## 反向传播：求梯度

反向传播（Backpropagation）：一次前向、一次反向，求出全部参数的梯度。

## 梯度下降：更新参数

$$\theta \leftarrow \theta - \eta \frac{1}{B} \sum_{i \in \mathcal{B}} \nabla_{\theta} \mathcal{L}(\hat{y}_i, y_i)$$

每步用一个小批量估计梯度；常用优化器为 Adam。

记号  $\theta$ ：全部参数； $\eta$ ：学习率； $\mathcal{B}$ ：小批量的样本下标集合， $B = |\mathcal{B}|$ （如 32-512）；按小批量做梯度下降称为随机梯度下降（Stochastic Gradient Descent, SGD）；Adam：Adaptive Moment Estimation。

# 03

## 反向传播

Backpropagation

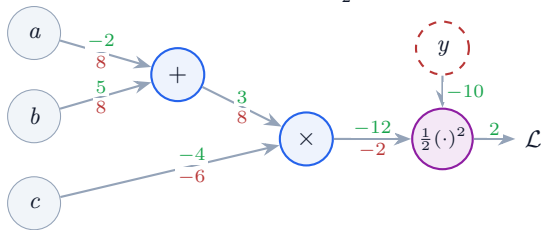
计算图与链式法则 · 误差回传与参数梯度 · 常见算子 · 算法 · 自动微分

# 链式法则与计算图

神经网络是复合函数  $\hat{y} = f^{(L)} \circ \dots \circ f^{(1)}(x)$ ；训练要求  $\nabla_{\theta} \mathcal{L}$ ：只需链式法则

**链式法则**  $\frac{d}{dx} g(f(x)) = g'(f(x)) \cdot f'(x)$ ：沿计算图从损失往回走，每个结点把上游梯度  $\times$  局部导数传给它的输入

例：  $\hat{y} = (a + b) c$ ，平方损失  $\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2$ ；取  $a = -2$ ,  $b = 5$ ,  $c = -4$ ，标签  $y = -10$



$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \hat{y}} &= \hat{y} - y = -2 \\ \frac{\partial \mathcal{L}}{\partial c} &= -2 \times 3 = -6 \\ \frac{\partial \mathcal{L}}{\partial a} &= \frac{\partial \mathcal{L}}{\partial b} = -2 \times (-4) = 8\end{aligned}$$

绿色：前向的值    红色：反向的梯度

记号 向量情形：若  $v = g(u)$ ，则  $\nabla_u \mathcal{L} = \mathbf{J}^\top \nabla_v \mathcal{L}$ ， $\mathbf{J} = \partial v / \partial u$  为雅可比矩阵。补充示例（非教材例题）。

## 逐层回传：误差与参数梯度

前向： $z^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ ,  $\mathbf{a}^{(l)} = \sigma(z^{(l)})$ 。定义第  $l$  层的误差为损失对净输入的梯度

$$\delta^{(l)} = \frac{\partial \mathcal{L}}{\partial z^{(l)}}$$

| 步骤   | 公式                                                                                                                                                                          | 来自链式法则                                                       |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| 输出层  | $\delta^{(L)} = \hat{\mathbf{y}} - \mathbf{y}$                                                                                                                              | sigmoid / softmax + 交叉熵（下式）                                  |
| 误差回传 | $\delta^{(l)} = \sigma'(z^{(l)}) \odot (\mathbf{W}^{(l+1)})^\top \delta^{(l+1)}$                                                                                            | $z^{(l)} \rightarrow \mathbf{a}^{(l)} \rightarrow z^{(l+1)}$ |
| 参数梯度 | $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^\top$ , $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$ | $z_j^{(l)} = \sum_k W_{jk}^{(l)} a_k^{(l-1)} + b_j^{(l)}$    |

$$\text{输出层: } \hat{y} = \sigma(z), \mathcal{L} = -y \ln \hat{y} - (1 - y) \ln(1 - \hat{y}) \Rightarrow \frac{\partial \mathcal{L}}{\partial z} = \left( -\frac{y}{\hat{y}} + \frac{1 - y}{1 - \hat{y}} \right) \hat{y}(1 - \hat{y}) = \hat{y} - y$$

记号  $\odot$ ：逐元素乘积。周志华《机器学习》第 5 章 5.3 节，式 (5.10)–(5.15)，pp.101–104（书中用平方误差， $g_j$ 、 $e_h$  为本页误差的相反数）；Bishop (2006) 第 5.3.1 节，式 (5.53)–(5.56)，pp.242–244（Bishop 的  $a$ 、 $z$  分别是此处的  $z$ 、 $a$ ）。

## 例：两层网络的前向与反向

两层均用 sigmoid，损失为二元交叉熵； $\mathbf{x} = (1, -1)^\top$ ， $y = 1$ ， $\mathbf{W}^{(1)} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ ， $\mathbf{b}^{(1)} = \mathbf{0}$ ， $\mathbf{w}^{(2)} = (1, -1)^\top$ ， $b^{(2)} = 0$

### 前向

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)} \mathbf{x} = (0, 2)^\top$$

$$\mathbf{a}^{(1)} = \sigma(\mathbf{z}^{(1)}) \approx (0.50, 0.88)^\top$$

$$\mathbf{z}^{(2)} = \mathbf{w}^{(2)\top} \mathbf{a}^{(1)} \approx -0.38$$

$$\hat{y} = \sigma(\mathbf{z}^{(2)}) \approx 0.41, \quad \mathcal{L} = -\ln \hat{y} \approx 0.90$$

### 反向

$$\delta^{(2)} = \hat{y} - y \approx -0.59$$

$$\begin{aligned} \boldsymbol{\delta}^{(1)} &= \sigma'(\mathbf{z}^{(1)}) \odot \mathbf{w}^{(2)} \delta^{(2)} \\ &\approx (0.25, 0.105)^\top \odot (-0.59, 0.59)^\top \\ &\approx (-0.15, 0.06)^\top \end{aligned}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(2)}} &= \delta^{(2)} \mathbf{a}^{(1)} \approx (-0.30, -0.52)^\top, \quad \frac{\partial \mathcal{L}}{\partial b^{(2)}} \approx -0.59, \\ \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}} &= \boldsymbol{\delta}^{(1)} \mathbf{x}^\top \approx \begin{pmatrix} -0.15 & 0.15 \\ 0.06 & -0.06 \end{pmatrix}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}} = \boldsymbol{\delta}^{(1)} \end{aligned}$$

记号  $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ ， $\sigma'(0) = 0.25$ ， $\sigma'(2) \approx 0.105$ ；中间结果保留全精度，显示两位小数。补充示例（非教材例题）。

# 反向传播算法

反向传播 (Backpropagation, BP): 一次前向、一次反向, 得到全部参数的梯度; 一个小批量上的一次更新

**Require:** 小批量  $\{(x_i, y_i)\}_{i \in \mathcal{B}}$ , 参数  $\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^L$ , 学习率  $\eta$

```
1: for 每个样本  $i \in \mathcal{B}$  do
2:   前向:  $\mathbf{a}^{(0)} = \mathbf{x}_i$ , 逐层算出并保存  $\mathbf{z}^{(l)}, \mathbf{a}^{(l)}$ 
3:   输出层误差:  $\delta^{(L)} = \hat{y}_i - y_i$ 
4:   for  $l = L, \dots, 1$  do
5:     累加梯度:  $\nabla_{\mathbf{W}^{(l)}} += \frac{1}{B} \delta^{(l)} (\mathbf{a}^{(l-1)})^\top$ ,  $\nabla_{\mathbf{b}^{(l)}} += \frac{1}{B} \delta^{(l)}$ 
6:     误差回传:  $\delta^{(l-1)} = \sigma'(\mathbf{z}^{(l-1)}) \odot (\mathbf{W}^{(l)})^\top \delta^{(l)}$  ( $l > 1$ )
7:   end for
8: end for
9: 梯度下降:  $\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \nabla_{\mathbf{W}^{(l)}}$ ,  $\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta \nabla_{\mathbf{b}^{(l)}}$ 
```

记号  $B = |\mathcal{B}|$ : 批大小, 梯度初始化为 0, 取批内平均; 实际实现中整批用矩阵并行计算。周志华《机器学习》第 5 章 5.3 节, 图 5.8, p.104; Werbos (1974) 最早提出, Rumelhart、Hinton、Williams (1986) 使之广为人知 (书 p.115)。

# 标准 BP 与累积 BP

BP 的目标是最小化训练集上的累积误差  $\mathcal{L} = \frac{1}{m} \sum_{k=1}^m \mathcal{L}_k$ （书中记  $E$ 、 $E_k$ ）

## 标准 BP 算法

|      |                              |
|------|------------------------------|
| 更新依据 | 单个样例的误差 $\mathcal{L}_k$      |
| 更新时机 | 每读入一个样例就更新一次                 |
| 特点   | 更新频繁，不同样例的更新可能相互“抵消”，需要更多次迭代 |

## 累积 BP 算法

|                             |
|-----------------------------|
| 累积误差 $\mathcal{L}$          |
| 读取整个训练集一遍（一轮，one epoch）后才更新 |
| 更新频率低；误差降到一定程度后下降非常缓慢       |

**要点** 两者的区别类似于随机梯度下降与标准梯度下降；训练集很大时，标准 BP 往往更快获得较好的解。

记号 累积误差逆传播（accumulated error backpropagation）。周志华《机器学习》第 5 章 5.3 节，式 (5.16)，pp.104–105。

# 自动微分

深度学习框架把反向传播推广到任意计算图：只需写前向，梯度自动得到

```
model = nn.Sequential(  
    nn.Linear(784, 256), nn.ReLU(),  
    nn.Linear(256, 10))  
opt = torch.optim.SGD(model.parameters(),  
                        lr=0.1)  
  
for X, y in loader:  
    loss = F.cross_entropy(model(X), y)  
    opt.zero_grad()  
    loss.backward()  
    opt.step()
```

PyTorch 代码：model(X) 为前向传播，backward() 为反向传播，step() 为梯度下降。

记号 `nn.Linear(784, 256)`： $\mathbf{W} \in \mathbb{R}^{256 \times 784}$  的全连接层；`cross_entropy`：softmax 与交叉熵合并计算；`lr`：学习率  $\eta$ 。

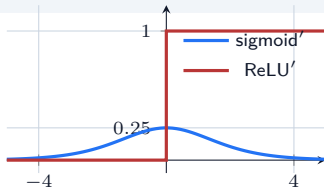
- 反向模式自动微分：每个算子提供前向计算与局部导数，框架按计算图逆序应用链式法则
- 反向的计算量约为前向的 2 倍
- 代价：前向的中间结果必须保存到反向使用，显存随批大小与层数增长

# 梯度消失与梯度爆炸

误差每回传一层，乘一次  $(\mathbf{W}^{(l+1)})^\top$  与一次  $\mathbf{D}^{(l)} = \text{diag}(\sigma'(z^{(l)}))$

$$\delta^{(1)} = \underbrace{\mathbf{D}^{(1)} (\mathbf{W}^{(2)})^\top \mathbf{D}^{(2)} (\mathbf{W}^{(3)})^\top \dots \mathbf{D}^{(L-1)} (\mathbf{W}^{(L)})^\top}_{\text{逐层连乘}} \delta^{(L)}$$

- **梯度消失**：sigmoid 的导数  $\leq \frac{1}{4}$ ；若各层  $\|\mathbf{W}^{(l)}\|_2 \leq 1$ ，经过 10 层后  $\|\delta^{(1)}\|$  至多为原来的  $0.25^{10} \approx 10^{-6}$  倍
- **梯度爆炸**：权重过大时乘积指数增长，训练发散
- 两者都源于多个因子连乘；层数越多越严重



记号  $\mathbf{D}^{(l)}$ ：以  $\sigma'(z^{(l)})$  为对角元的对角矩阵，乘以它等价于  $\odot \sigma'(z^{(l)})$ ； $\|\cdot\|_2$ ：矩阵的谱范数。

# 训练深层网络的常用技术

2010 年以来逐步积累，使训练上百层的网络成为可能

| 针对的问题     | 方法      | 做法                                                                                                                                             |
|-----------|---------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 梯度消失      | ReLU 激活 | 正半轴导数为 1                                                                                                                                       |
| 信号逐层放大或缩小 | 初始化     | Xavier: $\text{Var}(\mathbf{W}) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$ ; He: $\text{Var}(\mathbf{W}) = \frac{2}{n_{\text{in}}}$ (配合 ReLU) |
| 各层输入分布变化  | 归一化     | 批归一化 (Batch Normalization, BN), 把每层净输入标准化                                                                                                      |
| 深层网络难以优化  | 残差连接    | $y = x + F(x)$ , 见卷积神经网络一节                                                                                                                     |
| 学习率难以设定   | 自适应优化器  | Adam; 配合学习率预热与衰减                                                                                                                               |
| 过拟合       | 正则化     | 权重衰减 ( $L_2$ 正则)、早停、Dropout、数据增强                                                                                                               |

Dropout 与集成学习的联系见第 3 次课。

记号  $n_{\text{in}}, n_{\text{out}}$ : 该层的输入、输出单元数; Var: 方差。Xavier 初始化: Glorot & Bengio (2010); He 初始化: He 等 (2015); BN: Ioffe & Szegedy (2015)。

# 神经网络：小结



- 神经元 = 仿射变换 + 非线性激活
- 隐藏层学习特征变换，使问题线性可分
- 一个隐藏层即可万能近似；实践中用加深网络提高表示效率
- 反向传播 = 链式法则 + 复用中间结果
- 层数增加带来梯度消失 / 爆炸
- 全连接网络不利用输入的结构：像素的空间位置、词的先后顺序

记号  $\theta$ ：全部参数； $\mathcal{L}(\theta)$ ：训练目标；其余记号同前。

# 04

## 卷积神经网络

Convolutional Neural Networks

局部连接与权值共享 · 卷积、填充、步幅、池化 · 典型结构 · 残差连接

# 全连接网络处理图像的困难

一张  $224 \times 224$  的彩色图像有  $224 \times 224 \times 3 = 150,528$  个输入

## 参数过多

第一个隐藏层取 1000 个单元，仅这一层就有

$$150,528 \times 1000 \approx 1.5 \times 10^8$$

个权重。参数多，所需样本多，计算与存储代价高。

## 忽略空间结构

- 图像被展平为向量，相邻像素与相距很远的像素被同等对待
- 物体平移几个像素，输入向量完全改变，需重新学习

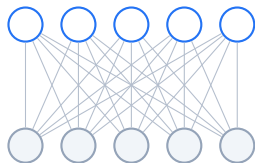
**要点** 卷积神经网络（Convolutional Neural Network, CNN）把图像的两条性质写进网络结构：**局部性与平移等变性**。

**记号**  $224 \times 224$ : ImageNet 分类中常用的输入分辨率；3: 红、绿、蓝三个颜色通道。

## 两个结构先验：局部连接与权值共享

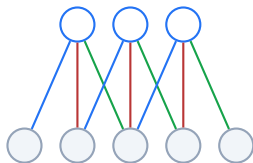
以一维输入为例：5 个输入，相邻 3 个输入决定一个输出

全连接



$5 \times 5 = 25$  个权重

卷积



只有  $w_1$ 、 $w_2$ 、 $w_3$  共 3 个权重

### 局部连接

每个输出只连接输入中的一个小窗口（感受野）

### 权值共享

所有位置使用同一组权重（卷积核）

### 平移等变

输入平移，输出随之平移相同距离

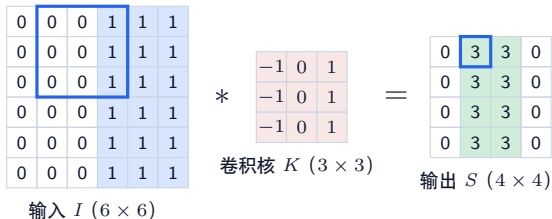
参数量从  $O(n^2)$  降到  $O(k)$ ，与输入大小无关；同一个特征检测器扫过整幅图像。

记号  $n$ ：输入长度； $k$ ：卷积核大小（图中  $k = 3$ ）；感受野：影响某个输出单元的输入区域。

# 卷积运算

卷积核在输入上滑动，每个位置做逐元素乘积再求和

$$S(i, j) = \sum_{u=0}^{k-1} \sum_{v=0}^{k-1} I(i+u, j+v) K(u, v) + b$$



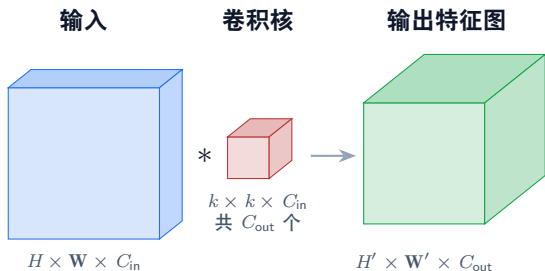
蓝框窗口：每行  $-0 + 0 + 1 = 1$ ，  
三行求和得 3

输出只在亮度由暗变亮的竖直边缘处非零：这个卷积核是一个竖直边缘检测器

记号  $I$ ：输入； $K$ ：卷积核， $k \times k$ ； $b$ ：偏置（本例取 0）； $S$ ：输出，称为特征图。深度学习中的“卷积”在数学上是互相关（未翻转卷积核），二者只差一次翻转。补充示例（非教材例题）：卷积神经网络见周志华《机器学习》第 5 章 5.6 节，pp.113–114。

## 多通道卷积

输入有  $C_{in}$  个通道，每个卷积核也有  $C_{in}$  层； $C_{out}$  个卷积核产生  $C_{out}$  个输出通道



$$\text{参数量} = (k \cdot k \cdot C_{in} + 1) \cdot C_{out}$$

例：RGB 输入 ( $C_{in} = 3$ )，64 个  $3 \times 3$  卷积核：  
 $(3 \cdot 3 \cdot 3 + 1) \times 64 = 1792$  个参数

参数量与图像尺寸  $H \times W$  无关；对比全连接层的  $1.5 \times 10^8$ 。

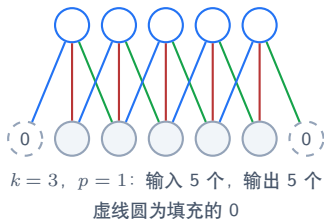
**要点** 每个输出通道是一种特征（如某个方向的边缘、某种纹理）在各位置的响应。

记号  $H, W$ ：输入的高与宽； $C_{in}, C_{out}$ ：输入、输出通道数； $k$ ：卷积核边长；+1：每个卷积核一个偏置； $H', W'$ ：输出的高与宽，见下页。补充示例（非教材例题）。

# 填充与步幅

控制输出特征图的尺寸

$$H' = \left\lfloor \frac{H + 2p - k}{s} \right\rfloor + 1$$



| $H$ | $k$ | $p$ | $s$ | $H'$ | 出处          |
|-----|-----|-----|-----|------|-------------|
| 32  | 5   | 0   | 1   | 28   | LeNet-5 第一层 |
| 224 | 3   | 1   | 1   | 224  | VGG: 尺寸不变   |
| 224 | 7   | 3   | 2   | 112  | ResNet 第一层  |

**填充**  $p$ : 在边界补  $p$  圈 0; **步幅**  $s$ : 卷积核每次移动  $s$  格。

常用组合:  $k = 3, p = 1, s = 1$  保持尺寸; 用步幅为 2 的卷积或池化把尺寸减半。

记号  $H, H'$ : 输入、输出的边长 (宽度同理);  $k$ : 卷积核边长;  $p$ : 填充圈数;  $s$ : 步幅;  $\lfloor \cdot \rfloor$ : 向下取整。

# 卷积层的参数量与计算量

输入  $H \times W \times C_{in}$ ,  $C_{out}$  个  $k \times k \times C_{in}$  的卷积核（带偏置），输出  $H' \times W' \times C_{out}$

$$\text{参数量} = (k \cdot k \cdot C_{in} + 1) \cdot C_{out}$$

$$\text{FLOPs} = 2 \cdot H' \cdot W' \cdot (k \cdot k \cdot C_{in}) \cdot C_{out}$$

输出的每个元素

- $k^2 C_{in}$  次乘法,  $k^2 C_{in} - 1$  次加法, 再加 1 次偏置: 共  $2k^2 C_{in}$  次
- 输出共  $H'W'C_{out}$  个元素, 相乘即得上式

| $C_{in} \rightarrow C_{out}$ | 参数量   | FLOPs              |
|------------------------------|-------|--------------------|
| $3 \rightarrow 64$           | 1792  | $1.73 \times 10^8$ |
| $64 \rightarrow 64$          | 36928 | $3.70 \times 10^9$ |

**要点** 参数量与特征图尺寸无关, 计算量与  $H'W'$  成正比: 卷积层参数少、计算多。

记号 浮点运算次数 (Floating-Point Operations, FLOPs); 右表:  $224 \times 224$  输入,  $k=3, p=1, s=1$ , 由上页公式  $H'=W'=224$ 。不少文献把一次乘加记作一次运算, 所报数值为上式的一半, 比较时须注意口径。补充示例 (非教材例题)。

# 池化

对每个局部窗口取最大值（或平均值），缩小特征图



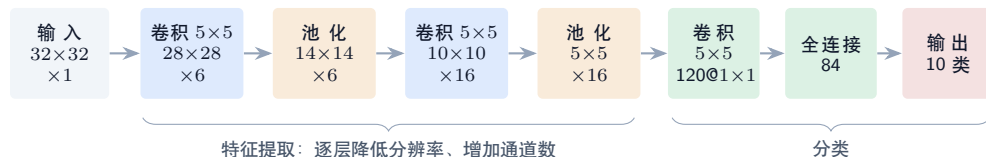
- 没有参数，每个通道单独进行
- 尺寸减半，后续层的计算量降为  $\frac{1}{4}$
- 最大响应仍在同一窗口内时，小幅平移不改变输出：对局部平移较不敏感
- 逐层扩大后续单元的感受野

现代网络中也常用步幅为 2 的卷积代替池化，最后用全局平均池化把每个通道压缩为一个数。

记号 最大池化：输出窗口内的最大值；平均池化：输出窗口内的平均值；全局平均池化：窗口取整个特征图。补充示例（非教材例题）。

# 典型结构：LeNet-5

LeCun 等（1998）：手写数字识别；“卷积—池化”交替，最后接全连接层



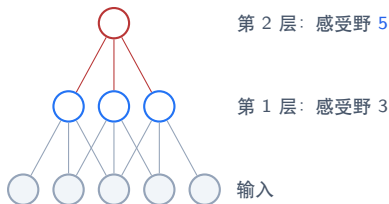
- 空间尺寸  $32 \rightarrow 28 \rightarrow 14 \rightarrow 10 \rightarrow 5$
- 通道数  $1 \rightarrow 6 \rightarrow 16$
- 全部参数约 6 万个

**要点** 此后的 CNN 沿用同一模式，只是更深、更宽，并加入 ReLU、归一化与残差连接。

记号 120 维层在原图中是  $5 \times 5$  卷积，输入为  $5 \times 5$  时等价于全连接。周志华《机器学习》第 5 章 5.6 节，图 5.15，p.114。卷积  $5 \times 5$ ：卷积核边长 5、步幅 1、无填充；池化： $2 \times 2$ 、步幅 2； $a \times b \times c$ ：高  $\times$  宽  $\times$  通道数。

# 感受野与层级特征

堆叠小卷积核：感受野逐层扩大，特征由局部到整体



## 两层 $3 \times 3$ 代替一层 $5 \times 5$ (VGG, 2014)

|                 | 感受野          | 参数量     | 非线性层数 |
|-----------------|--------------|---------|-------|
| 一层 $5 \times 5$ | $5 \times 5$ | $25C^2$ | 1     |
| 两层 $3 \times 3$ | $5 \times 5$ | $18C^2$ | 2     |

**层级特征** (Zeiler & Fergus, 2014 的可视化)：浅层响应边缘与颜色，中层响应纹理与部件，深层响应物体整体。

图中为一维示意：每层卷积核大小为 3、步幅为 1。

记号  $C$ ：输入与输出通道数（设二者相等）；参数量忽略偏置；VGG：牛津大学视觉几何组（Visual Geometry Group）提出的网络。

# ImageNet 竞赛中的 CNN

ImageNet 大规模视觉识别挑战赛 (ILSVRC): 约 128 万张训练图像, 1000 个类别

| 模型         | 年份   | 层数  | 参数量    | 竞赛提交的 top-5 错误率   |
|------------|------|-----|--------|-------------------|
| AlexNet    | 2012 | 8   | 6000 万 | 15.3% (第二名 26.2%) |
| VGG-16     | 2014 | 16  | 1.38 亿 | 7.3%              |
| ResNet-152 | 2015 | 152 | 6000 万 | 3.57%             |

## AlexNet 的要素

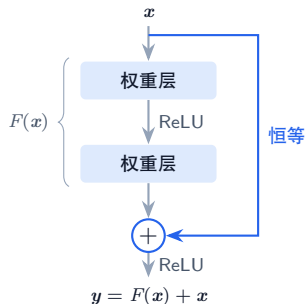
- ReLU 激活
- Dropout 正则化
- 数据增强（随机裁剪、翻转）
- 两块 GPU 并行训练

**要点** 三年间层数从 8 增加到 152, 错误率下降约四分之三。使这种加深成为可能的关键结构是残差连接。

记号 ILSVRC: ImageNet Large Scale Visual Recognition Challenge; top-5 错误率: 真实类别不在模型给出的前 5 个预测中的比例; 表中为各队竞赛提交结果 (测试集, 部分为多模型集成; 单个 ResNet-152 在验证集上的 top-5 错误率为 4.49%, He 等 2016)。

# 残差网络

残差网络（Residual Network, ResNet），He 等（2016）



**退化问题：**在 CIFAR-10 上，56 层普通网络的训练误差高于 20 层网络。这是优化困难，而非过拟合。

**残差学习：**网络只需学习残差  $F(x) = y - x$ ；令  $F = 0$  即得恒等映射：更深的网络至少能表示较浅网络的函数，通常更易优化。

**梯度直达：**对加法结点  $y = F(x) + x$ ： $\nabla_x \mathcal{L} = (\mathbf{I} + \mathbf{J}_F)^\top \nabla_y \mathcal{L}$ ，恒等项让梯度不经连乘直接传回浅层。

**要点** 残差连接是此后深度网络的标准组件，Transformer 的每个子层也使用它。

记号  $x$ ：残差块的输入； $F(x)$ ：两个权重层（卷积 + 归一化）计算的残差； $\mathbf{I}$ ：单位矩阵； $\mathbf{J}_F = \partial F / \partial x$ ：雅可比矩阵；图中加法之后另有 ReLU，此处梯度为加法结点处的值；CIFAR-10：32 × 32 彩色图像、10 类的数据集。

# 卷积神经网络：小结

适合具有网格结构的数据：图像、视频、频谱图

## 擅长

- 利用**空间相关性**：相邻像素高度相关，用**局部感受野**与权值共享建模，参数少、泛化好
- 典型任务：图像分类、目标检测、语义分割、医学影像分析

## 不擅长

- 局部感受野大小固定，远距离的两个位置要堆很多层才能交互
- 序列（文本、语音）长度可变、有先后顺序，具有**时间相关性**

**要点** 处理序列需要能逐个读入元素、记住前文的结构：循环神经网络。

# 05

## 循环神经网络

Recurrent Neural Networks

序列数据 · 循环计算 · 单向与遗忘 · CNN 与 RNN 的局限

# 序列数据与循环神经网络

循环神经网络（Recurrent Neural Network, RNN）：同一个单元沿时间重复使用

## 序列数据的特点

- 元素有先后顺序，前后相互依赖（时间相关性）：文本、语音、股价
- 长度可变：句子有长有短
- 一个元素的含义依赖上下文

### 思路

用一个隐状态  $h_t$  记录读到第  $t$  个元素为止的信息；每读入一个新元素  $x_t$ ，就用同一组参数更新一次：

$$h_t = f(h_{t-1}, x_t)$$

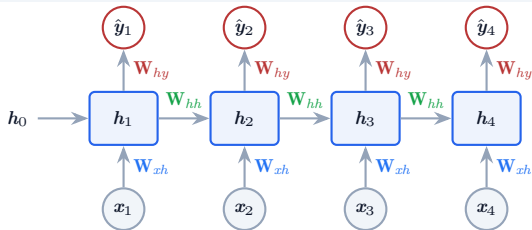


记号  $x_t$ ：第  $t$  个元素的向量表示（如一个词的嵌入向量）； $h_t$ ：第  $t$  步的隐状态； $h_0$ ：初始隐状态，通常取 0。

# RNN 的计算

每一步：上一步的隐状态与当前输入一起，经全连接和  $\tanh$  得到新的隐状态

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h), \quad \hat{y}_t = W_{hy}h_t + b_y$$



- 按时间展开后就是一个很深的前馈网络
- 所有时间步共享同一组参数  $W_{hh}, W_{xh}, W_{hy}$ ，可处理任意长度
- 输出可以每步都有（如逐词标注），也可只取最后一步（如句子分类）

记号  $W_{hh}$ ：隐状态到隐状态的权重； $W_{xh}$ ：输入到隐状态的权重； $W_{hy}$ ：隐状态到输出的权重； $b_h, b_y$ ：偏置；训练时同样用反向传播求梯度。

# RNN 的两个特点：只看前文，压缩导致遗忘

## 只能看到前面

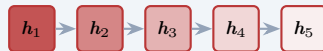
$h_t$  只由  $x_1, \dots, x_t$  决定，读到第  $t$  个词时看不到后文。



例：“他买了一个苹果，屏幕很清晰”，读到“苹果”时无法判断是水果还是手机。

## 所有历史压缩进一个 $h_t$

$h_t$  的维度固定，序列越长，早期信息被反复覆盖：



颜色深浅： $h_t$  中保留的  $x_1$  信息

远距离依赖难以学习：梯度也要穿过很多步连乘，容易消失。

记号 双向 RNN 再从后往前读一遍以获得后文；长短期记忆网络（LSTM）用门控结构缓解遗忘，但仍需逐步计算。

# CNN 与 RNN 的局限

两者各自利用一种相关性，但都难以让任意两个位置直接交互

| 模型  | 利用的相关性         | 信息如何传递           | 缺陷                                                       |
|-----|----------------|------------------|----------------------------------------------------------|
| CNN | 空间相关性：相邻像素高度相关 | 局部感受野：卷积核覆盖的小窗口  | 相距 $n$ 的位置需堆约 $n/k$ 层才能交互                                |
| RNN | 时间相关性：前后元素相互依赖 | 隐状态 $h_t$ 逐步向后传递 | (1) 只能看到前文<br>(2) 历史压缩进一个 $h_t$ ，早期信息遗忘<br>(3) 逐步计算，无法并行 |

**要点** 需要一种结构：每个位置能直接查看序列中的所有位置，并且所有位置可以并行计算。这就是注意力机制与 Transformer。

记号 CNN：卷积神经网络；RNN：循环神经网络； $k$ ：卷积核大小； $n$ ：两个位置之间的距离。

# 06

## 注意力机制与 Transformer

Attention & Transformers

注意力：软查找 · 缩放点积与自注意力 · 多头与位置编码 · Transformer 及其变体

# 长距离的上下文依赖

文本、语音、时间序列：一个元素的含义取决于序列中的其他元素

The **animal** didn't cross the street because **it** was too tired.

The animal didn't cross the **street** because **it** was too wide.

同一个词 it，在两个句子中分别指代 animal 与 street；句中距离可以任意远。

**注意力机制的思路** 每个位置直接查看所有位置，按相关程度加权汇总信息；权重由数据决定，任意两个位置之间只隔一步，且所有位置可并行计算。

记号 RNN 需要逐步传递隐状态才能让 animal 与 it 交互；注意力让二者直接交互。

# 注意力：软查找

把注意力看成对一张“键—值”表的查询

## 硬查找（字典）

| 键 $k_i$ | 值 $v_i$ |
|---------|---------|
| 苹果      | 5 元     |
| 香蕉      | 3 元     |
| 橙子      | 4 元     |

查询  $q = \text{香蕉}$ ：返回键与  $q$  完全相同的那一项的值。

## 软查找（注意力）

返回所有值的加权平均，权重由查询与各个键的相似度决定：

$$\text{Attn}(q) = \sum_{i=1}^n \alpha_i v_i, \quad \alpha_i = \frac{\exp(s(q, k_i))}{\sum_j \exp(s(q, k_j))}$$

权重非负、和为 1，且对  $q, k_i$  可导，可以端到端训练。

记号  $q$ ：查询（query）； $k_i$ ：键/钥匙（key）； $v_i$ ：值（value）； $s(\cdot, \cdot)$ ：相似度打分函数； $\alpha_i$ ：注意力权重； $n$ ：表中条目数。

# 缩放点积注意力

Vaswani 等 (2017)。  $n$  个查询同时计算，写成矩阵形式

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right) V$$

|                                         |                                |
|-----------------------------------------|--------------------------------|
| $QK^{\top} \in \mathbb{R}^{n \times n}$ | 第 $(i, j)$ 元为 $q_i^{\top} k_j$ |
| softmax                                 | 按行归一化，每行和为 1                   |
| 乘以 $V$                                  | 每个查询得到值的加权平均                   |

为什么除以  $\sqrt{d_k}$

设  $q, k$  相互独立、各分量独立、均值 0、方差 1（启发式近似），则

$$\text{Var}(q^{\top} k) = \sum_{t=1}^{d_k} \text{Var}(q_t k_t) = d_k$$

$d_k$  大时点积数值大，softmax 接近独热、梯度趋于 0；除以  $\sqrt{d_k}$  使方差恢复为 1。

记号  $Q \in \mathbb{R}^{n \times d_k}$ :  $n$  个查询按行堆叠； $K \in \mathbb{R}^{n \times d_k}$ : 键； $V \in \mathbb{R}^{n \times d_v}$ : 值； $d_k$ : 查询与键的维数； $d_v$ : 值的维数； $q_i^{\top}, k_j^{\top}$ :  $Q, K$  的第  $i, j$  行 ( $q_i, k_j$  为列向量)。

# 缩放点积注意力：数值例子

一个查询、三个键值对， $d_k = 2$

|                         | $j = 1$ | $j = 2$ | $j = 3$    |
|-------------------------|---------|---------|------------|
| 键 $k_j$                 | (2, 0)  | (0, 2)  | (1, 1)     |
| 值 $v_j$                 | (1, 0)  | (0, 1)  | (0.5, 0.5) |
| $q^\top k_j$            | 4       | 0       | 2          |
| $q^\top k_j / \sqrt{2}$ | 2.83    | 0       | 1.41       |
| 权重 $\alpha_j$           | 0.768   | 0.045   | 0.187      |

查询  $q = (2, 0)$

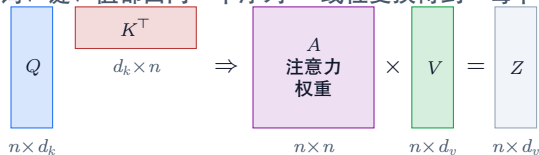
$$\sum_j \alpha_j v_j = (0.861, 0.139)$$

- 与  $q$  方向最接近的  $k_1$  获得最大权重
- 输出以  $v_1$  为主，混入少量其他值

记号  $q$ : 查询;  $k_j, v_j$ : 第  $j$  个键与值;  $\alpha_j = \exp(q^\top k_j / \sqrt{d_k}) / \sum_t \exp(q^\top k_t / \sqrt{d_k})$ ;  $d_k = 2$ ,  $\sqrt{2} \approx 1.414$ 。补充示例（非教材例题）。

# 自注意力

查询、键、值都由同一个序列  $X$  线性变换得到：每个位置向所有位置“提问”



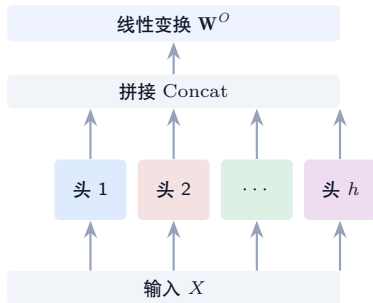
$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$
$$Z = \text{softmax}(QK^T / \sqrt{d_k}) V$$

- 第  $i$  行输出  $z_i$  是所有位置的值的加权平均：融合了上下文的新表示
- 可学习参数只有  $W^Q, W^K, W^V$ ，与序列长度  $n$  无关
- 所有位置同时计算，完全并行
- 代价：注意力矩阵  $A$  有  $n^2$  个元素，计算量  $O(n^2 d)$ （另有投影  $O(nd^2)$ ）

记号  $X \in \mathbb{R}^{n \times d}$ ：输入序列，每行是一个位置（token）的向量； $W^Q, W^K \in \mathbb{R}^{d \times d_k}$ ， $W^V \in \mathbb{R}^{d \times d_v}$ ：投影矩阵； $A \in \mathbb{R}^{n \times n}$ ：注意力权重矩阵，每行和为 1； $Z$ ：输出。

# 多头注意力

并行运行  $h$  组注意力，各自关注不同类型的关系，再拼接



$$\begin{aligned} \text{head}_i &= \text{Attention}(XW_i^Q, XW_i^K, XW_i^V) \\ \text{MultiHead}(X) &= \\ &\text{Concat}(\text{head}_1, \dots, \text{head}_h) \mathbf{W}^O \end{aligned}$$

**原始 Transformer:**  $d = 512$ ,  $h = 8$ , 每个头  $d_k = d_v = d/h = 64$ 。总计算量与单个 512 维的头相当。

**要点** 一个头只能给出一组权重；多个头可同时捕捉指代、句法、相邻位置等不同关系。

记号  $h$ : 头数;  $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ : 第  $i$  个头的投影矩阵;  $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d}$ : 输出投影;  $d$ : 模型维数 (每个 token 向量的维数)。

# 位置编码

自注意力对输入的顺序不敏感：打乱  $X$  的行，输出只是相应地打乱

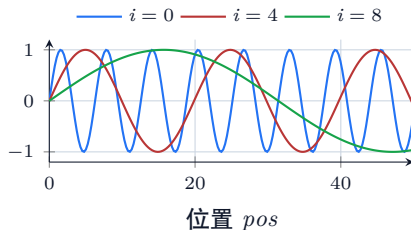
“狗咬人”与“人咬狗”在自注意力看来是同一组词。需要把位置信息加到输入上：

$$X \leftarrow X + P。$$

正弦位置编码（原始 Transformer）

$$P_{pos,2i} = \sin(pos/10000^{2i/d})$$
$$P_{pos,2i+1} = \cos(pos/10000^{2i/d})$$

不同维度对应不同频率。



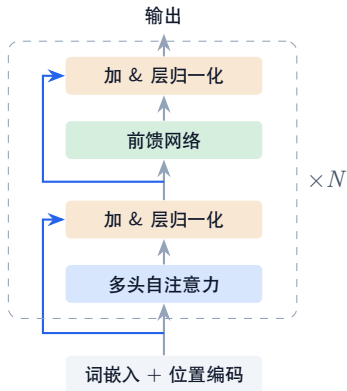
$d = 64$  时三个维度上的  $P_{pos,2i}$

其他做法：可学习的位置嵌入（BERT、GPT-2）；旋转位置编码（Rotary Position Embedding, RoPE；苏剑林等, 2021），LLaMA 等采用。

记号  $P \in \mathbb{R}^{n \times d}$ ：位置编码矩阵； $pos$ ：位置序号； $i$ ：维度序号， $2i$  与  $2i+1$  为一对； $d$ ：模型维数。

# Transformer 块

两个子层：多头自注意力在位置之间交换信息，前馈网络在每个位置内部变换特征



$$H = \text{LN}(X + \text{MultiHead}(X))$$

$$Y = \text{LN}(H + \text{FFN}(H))$$

$$\text{FFN}(\mathbf{h}) = \text{ReLU}(\mathbf{h}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$$

- 前馈网络 (Feed-Forward Network, FFN) 是逐位置作用的两层 MLP，中间维数  $d_{\text{ff}} = 4d$  (原始模型 2048)
- 层归一化 (Layer Normalization, LN) 在每个 token 的  $d$  维特征上标准化
- 残差连接使  $N$  层堆叠后易于训练；现代模型多把 LN 移到子层之前 (Pre-LN)

记号  $X, H, Y \in \mathbb{R}^{n \times d}$ : 块的输入、中间结果、输出;  $\mathbf{W}_1 \in \mathbb{R}^{d \times d_{\text{ff}}}$ ,  $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{ff}} \times d}$ ;  $N$ : 堆叠的块数 (原始模型  $N = 6$ )。

# 因果掩码与语言模型

生成文本时，第  $t$  个位置只能看到第  $1, \dots, t$  个位置，并用于预测第  $t+1$  个 token

|          |    | 我        | 喜欢        | 机器        | 学习        |
|----------|----|----------|-----------|-----------|-----------|
| 查询（当前位置） | 我  | ✓        | $-\infty$ | $-\infty$ | $-\infty$ |
|          | 喜欢 | ✓        | ✓         | $-\infty$ | $-\infty$ |
|          | 机器 | ✓        | ✓         | ✓         | $-\infty$ |
|          | 学习 | ✓        | ✓         | ✓         | ✓         |
|          |    | 键（被看的位置） |           |           |           |

$$p(x_1, \dots, x_n) = \prod_{t=1}^n p(x_t \mid x_1, \dots, x_{t-1})$$

- softmax 之前把  $j > t$  的分数置为  $-\infty$ ，对应权重为 0
- 一次前向即可得到所有位置的下一词预测，训练完全并行
- 生成时逐个输出 token 并追加到输入

**要点** GPT 系列大语言模型：以此目标在海量文本上训练的因果 Transformer。

记号  $x_t$ : 第  $t$  个 token;  $t$ : 查询位置;  $j$ : 键位置; token: 文本切分后的基本单位（词或子词）。

# Transformer 的三种结构

原始模型（Vaswani 等, 2017）为编码器—解码器结构，用于机器翻译

| 结构      | 注意力           | 代表模型              | 典型任务       |
|---------|---------------|-------------------|------------|
| 仅编码器    | 双向自注意力        | BERT (2018)       | 文本分类、信息抽取  |
| 仅解码器    | 因果自注意力        | GPT 系列 (2018 起)   | 文本生成、大语言模型 |
| 编码器—解码器 | 双向、因果 + 交叉注意力 | 原始 Transformer、T5 | 翻译、摘要      |

**交叉注意力：**查询来自解码器，键和值来自编码器输出。解码器生成每个目标词时，按需查看源句的各个位置。

**原始结果：**WMT 2014 英德翻译 BLEU 28.4，超过此前包括集成模型在内的最好结果，训练代价更低。

**记号** BERT: Bidirectional Encoder Representations from Transformers; GPT: Generative Pre-trained Transformer; T5: Text-to-Text Transfer Transformer; BLEU: 机器翻译的自动评价指标，越高越好。

# 视觉 Transformer

视觉 Transformer (Vision Transformer, ViT), Dosovitskiy 等 (2021): 把图像切成小块, 当作 token 序列



- $224 \times 224$  图像、 $P = 16$ : 得到  $14 \times 14 = 196$  个图像块 token (加上 CLS 共 197 个)
- 每块  $16 \times 16 \times 3 = 768$  维, 投影后送入 Transformer 编码器
- 用额外的 CLS token 的输出做分类

**要点** 只用 ImageNet-1k 训练时, ViT 不如同规模的 ResNet; 在 3 亿张图像 (JFT-300M) 上预训练后超过 CNN。结构先验少, 需要更多数据。

记号  $P$ : 小块边长; CLS: 序列开头的可学习分类 token; MLP 头: 小型多层感知机; ImageNet-1k: ILSVRC 的 1000 类子集。

# 三种网络结构的比较

同一层处理长度为  $n$ 、维数为  $d$  的序列（Vaswani 等, 2017, 表 1）

| 层类型  | 每层计算量                    | 顺序操作数  | 最长路径                           | 结构先验         |
|------|--------------------------|--------|--------------------------------|--------------|
| 卷积   | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(n/k)$ （空洞卷积 $O(\log_k n)$ ） | 局部性、平移等变     |
| 循环   | $O(n \cdot d^2)$         | $O(n)$ | $O(n)$                         | 按顺序处理        |
| 自注意力 | $O(n^2 \cdot d)$         | $O(1)$ | $O(1)$                         | 几乎没有（依赖位置编码） |

**要点** 结构先验与数据量此消彼长：数据少时，先验强的 CNN 更有效；数据充足时，先验弱、并行度高的 Transformer 更有优势。这与第 2 次课的归纳偏好一致。

记号 顺序操作数：无法并行、必须依次完成的步数；最长路径：任意两个位置之间信息传递需经过的层数（或步数）； $k$ ：卷积核大小。

# 07

## 小结

Summary

本讲要点 · 课后作业

# 本讲要点

| 模型          | 基本运算        | 结构先验       | 代表                   |
|-------------|-------------|------------|----------------------|
| 多层感知机       | 全连接 + 非线性激活 | 无          | 表格数据；其他模型的组件         |
| 卷积神经网络      | 卷积 + 池化     | 空间相关性、平移等变 | LeNet、AlexNet、ResNet |
| 循环神经网络      | 循环更新隐状态     | 时间相关性、顺序处理 | LSTM                 |
| Transformer | 自注意力 + 前馈网络 | 几乎没有       | BERT、GPT、ViT         |

- 四者都由可导模块复合而成，用反向传播 + 小批量 SGD 训练
- 残差连接与归一化使深层网络可训练
- 网络从数据中学习特征，取代人工设计的特征映射
- 结构先验的强弱需要与数据量相匹配

记号 SGD：随机梯度下降；ViT：视觉 Transformer。

# 中英文术语对照

| 英文全称                         | 缩写           | 中文           |
|------------------------------|--------------|--------------|
| Multilayer Perceptron        | <b>MLP</b>   | 多层感知机        |
| Backpropagation              | <b>BP</b>    | 反向传播         |
| Rectified Linear Unit        | <b>ReLU</b>  | 修正线性单元       |
| Gaussian Error Linear Unit   | <b>GELU</b>  | 高斯误差线性单元     |
| Sigmoid Linear Unit          | <b>SiLU</b>  | sigmoid 线性单元 |
| Stochastic Gradient Descent  | <b>SGD</b>   | 随机梯度下降       |
| Adaptive Moment Estimation   | <b>Adam</b>  | 自适应矩估计       |
| Batch Normalization          | <b>BN</b>    | 批归一化         |
| Graphics Processing Unit     | <b>GPU</b>   | 图形处理器        |
| Convolutional Neural Network | <b>CNN</b>   | 卷积神经网络       |
| Floating-Point Operations    | <b>FLOPs</b> | 浮点运算次数       |
| Recurrent Neural Network     | <b>RNN</b>   | 循环神经网络       |

| 英文全称                                                    | 缩写            | 中文                     |
|---------------------------------------------------------|---------------|------------------------|
| Long Short-Term Memory                                  | <b>LSTM</b>   | 长短期记忆网络                |
| Residual Network                                        | <b>ResNet</b> | 残差网络                   |
| Visual Geometry Group                                   | <b>VGG</b>    | 视觉几何组（牛津大学）            |
| ImageNet Large Scale Visual Recognition Challenge       | <b>ILSVRC</b> | ImageNet 大规模视觉识别挑战赛    |
| Feed-Forward Network                                    | <b>FFN</b>    | 前馈网络                   |
| Layer Normalization                                     | <b>LN</b>     | 层归一化                   |
| Rotary Position Embedding                               | <b>RoPE</b>   | 旋转位置编码                 |
| Bidirectional Encoder Representations from Transformers | <b>BERT</b>   | 基于 Transformer 的双向编码表示 |
| Generative Pre-trained Transformer                      | <b>GPT</b>    | 生成式预训练 Transformer     |
| Text-to-Text Transfer Transformer                       | <b>T5</b>     | 文本到文本迁移 Transformer    |
| Vision Transformer                                      | <b>ViT</b>    | 视觉 Transformer         |

# 课后作业

反向传播手算（数值保留 3 位小数）

两层神经网络，两层均用 sigmoid 激活  $\sigma(z) = 1/(1 + e^{-z})$ ，损失为二元交叉熵：

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}, \quad \mathbf{a}^{(1)} = \sigma(\mathbf{z}^{(1)}), \quad z^{(2)} = \mathbf{w}^{(2\top)}\mathbf{a}^{(1)} + b^{(2)}, \quad \hat{y} = \sigma(z^{(2)}), \quad \mathcal{L} = -y \ln \hat{y} - (1 - y) \ln(1 - \hat{y})$$

输入、标签  $\mathbf{x} = (1, -1, 2)^\top$ ,  $y = 1$

第 1 层参数  $\mathbf{W}^{(1)} = \begin{pmatrix} 0.5 & 0 & -0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix}$ ,  $\mathbf{b}^{(1)} = (0.1, -0.1)^\top$

第 2 层参数  $\mathbf{w}^{(2)} = (1, -1)^\top$ ,  $b^{(2)} = 0.2$

1. 前向传播：求  $\mathbf{z}^{(1)}$ 、 $\mathbf{a}^{(1)}$ 、 $z^{(2)}$ 、 $\hat{y}$  及损失  $\mathcal{L}$ 。
2. 反向传播：求每一层的误差  $\delta^{(2)} = \partial \mathcal{L} / \partial z^{(2)}$ 、 $\delta^{(1)} = \partial \mathcal{L} / \partial \mathbf{z}^{(1)}$ 。
3. 求参数的梯度  $\partial \mathcal{L} / \partial \mathbf{W}^{(1)}$ 、 $\partial \mathcal{L} / \partial \mathbf{b}^{(1)}$ 、 $\partial \mathcal{L} / \partial \mathbf{w}^{(2)}$ 、 $\partial \mathcal{L} / \partial b^{(2)}$ 。
4. 用梯度下降更新参数，学习率  $\eta = 0.01$ ，求更新后的各参数。