人工智能期末复习课详细提纲

袁肖赟，上海交通大学计算机学院人工智能研究院，yuanxiaoyun@sjtu.edu.cn

Contents

1 基础知识

这一章用于补齐后面章节会反复出现的基础概念：张量形状、参数与偏置、激活函数、softmax 和交叉熵。这里不讲训练过程中的前向传播和反向传播。

1.1 张量与形状

神经网络里的数据通常不是单个数字，而是向量、矩阵或更高维数组。复习时最重要的是看懂“形状”。

张量（Tensor）：张量可以理解为多维数组；向量是 1 维，矩阵是 2 维，图像和特征图通常是 3 维或 4 维。

常见形状记法：

数据	常见形状	含义
向量	d	有 d 个数
矩阵	$m \times n$	m 行、n 列
图像	$H \times W \times C$	高、宽、通道数
一批样本	$B \times H \times W \times C$	batch size、高、宽、通道数
token 序列	$n \times d_m$	序列长度、每个 token 的向量维度

例子 1：图像形状。一张 RGB 图像大小为 32×32，有 3 个颜色通道，因此形状可以写成 $32 \times 32 \times 3$。

例子 2：**token 序列形状**。一个句子有 $n = 4$ 个 token，每个 token 的 embedding 维度为 $d_m = 128$，则输入矩阵形状为 4×128。

1.2 参数、权重与偏置

神经网络中需要学习的数通常叫参数。最常见的参数是权重和偏置（bias）。

参数：参数是神经网络中需要通过数据学习得到的数，例如全连接层的权重矩阵 W 和偏置向量 b。

偏置：偏置是在线性变换 $Wx + b$ 中额外加上的可学习参数。

全连接层常写成：

$$y = Wx + b$$

如果输入维度为 d_{in}，输出维度为 d_{out}，则：

$$W \in R^{d_{out} \times d_{in}}, \qquad b \in R^{d_{out}}, \qquad y \in R^{d_{out}}$$

参数量：

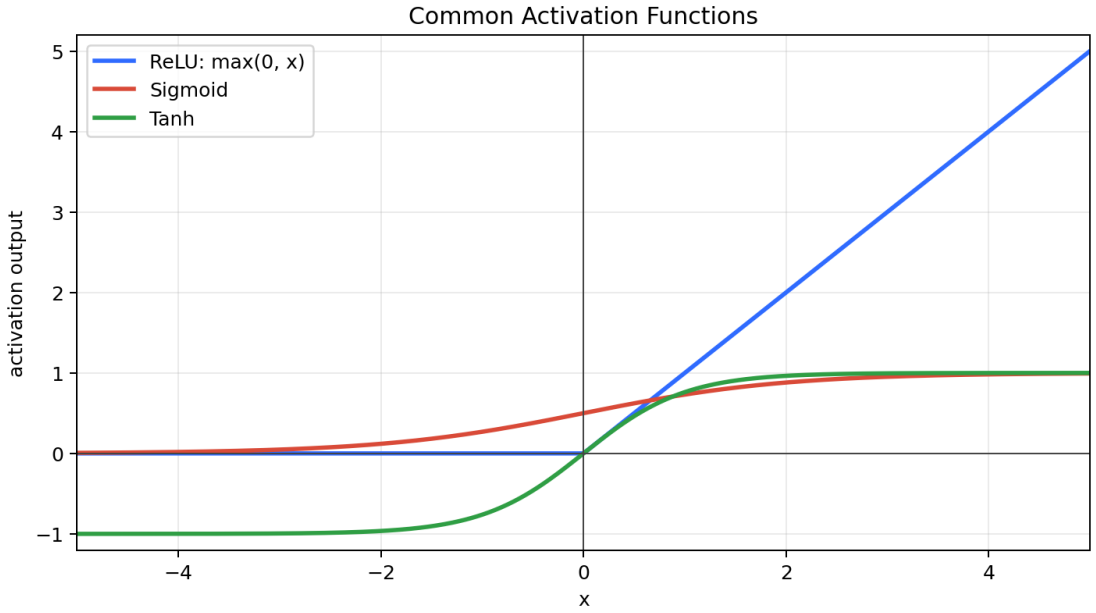
$$\#\text{params} = d_{out} \times d_{in} + d_{out}$$

其中，$d_{out} \times d_{in}$ 是权重矩阵 W 的参数量，d_{out} 是偏置向量 b 的参数量。

例子 **3**：全连接层参数量。输入维度为 4，输出维度为 3。权重参数量为 $3 \times 4 = 12$，偏置参数量为 3，总参数量为 15。

1.3 激活函数

激活函数把线性变换后的结果变成非线性输出。没有非线性激活函数，多层线性变换叠在一起仍然等价于一个线性变换。



激活函数：激活函数是作用在神经元输出上的非线性函数，用来提升神经网络表达复杂关系的能力。

激活函数本质上作用在一个标量上；当输入是向量或矩阵时，通常对每个元素逐元素应用同一个激活函数。

常见激活函数：

激活函数	公式	记忆
ReLU	$\mathrm{ReLU}(x) = \max(0, x)$	负数变 0，正数保留
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	输出在 0 到 1 之间
Tanh	$\tanh(x)$	输出在 -1 到 1 之间

例子 **4**：**ReLU** 计算。对单个数，$\mathrm{ReLU}(-2) = 0$，$\mathrm{ReLU}(3) = 3$。对向量逐元素应用：

$$\mathrm{ReLU}([-2, 0, 3]) = [0, 0, 3]$$

例子 **5**：**Sigmoid** 的用途。 Sigmoid 输出在 0 到 1 之间，适合表示比例或概率倾向。

1.4 Softmax 与概率输出

分类任务和语言模型输出时，模型通常先给每个类别或 token 一个分数，再把这些分数变成概率。

Softmax： Softmax 把一组分数转换成概率分布；输出非负，并且所有概率之和为 1。

Softmax 公式为：

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Softmax 会放大分数之间的差异：大的分数对应更大的概率，小的分数对应更小的概率，输出会比原始分数更接近 one-hot 分布。

例子 6：**Softmax 把分数变成概率**。假设两个类别的分数为 $[1, 2]$，则

$$e^1 \approx 2.72, \qquad e^2 \approx 7.39$$

所以 softmax 后的概率约为：

$$\left[\frac{2.72}{2.72+7.39}, \frac{7.39}{2.72+7.39}\right] \approx [0.27, 0.73]$$

分数更大的类别概率更大，小分数对应的类别概率更小；结果比原始分数更像 one-hot，但两个概率之和仍然为 1。

1.5 交叉熵

交叉熵常用于分类任务和 next token prediction。它的直觉是：真实类别的预测概率越高，交叉熵越小。

交叉熵：交叉熵衡量真实标签和模型预测概率之间的差距；对真实类别给出的概率越高，交叉熵越小。

二分类形式：

如果真实标签 $y_i \in \{0, 1\}$，模型预测样本 x_i 属于正类的概率为 $f_\theta(x_i)$，则二分类交叉熵常写成：

$$H(y, p) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log f_\theta(x_i) + (1 - y_i)\log(1 - f_\theta(x_i))\right]$$

其中，n 是样本数，y_i 是真实标签，$f_\theta(x_i)$ 是模型预测为正类的概率。对单个样本，可以简化为：

$$L = -\left[y \log p + (1 - y)\log(1 - p)\right]$$

当 $y = 1$ 时，$L = -\log p$；当 $y = 0$ 时，$L = -\log(1 - p)$。

例子 7：**二分类交叉熵计算**。对二分类任务，真实标签为正类，也就是 $y = 1$。比较两个预测：
预测 A 的正类概率为 0.9：

$$L = -\left[1 \cdot \log 0.9 + 0 \cdot \log(1 - 0.9)\right] = -\log 0.9 \approx 0.105$$

预测 B 的正类概率为 0.6：

$$L = -\left[1 \cdot \log 0.6 + 0 \cdot \log(1 - 0.6)\right] = -\log 0.6 \approx 0.511$$

因为真实类别是正类，0.9 比 0.6 更接近正确答案，所以第一种预测的交叉熵更小。

多分类形式：

如果第 i 个样本的真实标签用 one-hot 向量 $y_{i,c}$ 表示，模型对第 c 类的预测概率为 $p_{i,c}$，则更 general 的多分类交叉熵为：

$$H(y,p) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log p_{i,c}$$

其中，C 是类别数。因为 one-hot 里只有真实类别对应的 $y_{i,c}=1$，所以对单个样本也可以理解为“真实类别概率越大，loss 越小”。

1.6 关键记忆

- 看神经网络结构时，先看清输入和输出的形状。
- 参数是网络中需要学习的数，常见参数包括权重 W 和偏置 b。
- 激活函数提供非线性能力；ReLU 的公式是 $\max(0,x)$。
- Sigmoid 输出在 0 到 1 之间，常用于表示比例或门控。
- Softmax 把分数变成概率分布，所有概率之和为 1；大的更大、小的更小，结果更接近 one-hot。
- 交叉熵衡量预测概率和真实标签的差距；二分类常写成 $-\frac{1}{n}\sum_i[y_i\log f_\theta(x_i)+(1-y_i)\log(1-f_\theta(x_i))]$。

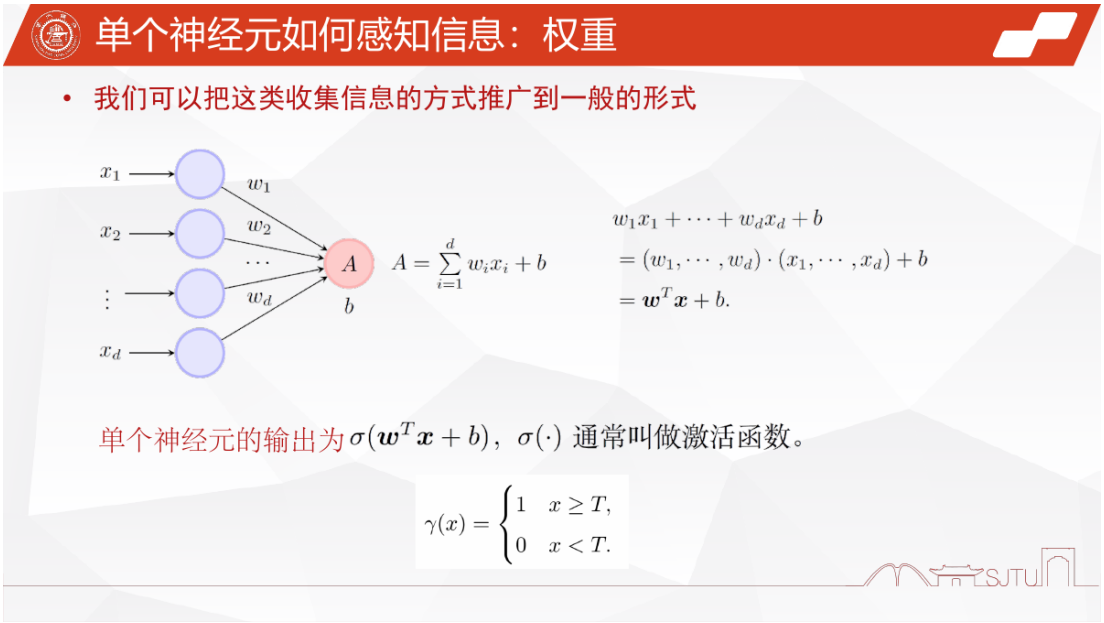
2 全连接网络与残差网络基础

这一章按 PDF 顺序复习：单个神经元 -> 全连接层 -> 多层神经网络 -> 深层网络训练困难 -> 残差连接。第 11-12 节中，全连接网络只保留理解后续章节所需的基础；本章主要围绕残差网络为什么出现、残差连接怎么写、它解决什么问题。

2.1 单个神经元与权重

页码：p4-p6

单个神经元可以理解为“收集多个输入，按重要性加权，再经过激活函数输出”。权重越大，说明对应输入对输出影响越大。



神经元：神经元把输入按权重加权求和，加上偏置，再通过激活函数得到输出。

单个神经元公式：

$$z = w^T x + b, \qquad y = \sigma(z)$$

其中，x 是输入向量，w 是权重向量，b 是偏置，σ 是激活函数，y 是神经元输出。

例子 **1**：单个神经元计算。设 $x = [2, 1]^T$，$w = [0.5, -1]^T$，$b = 0.2$，则

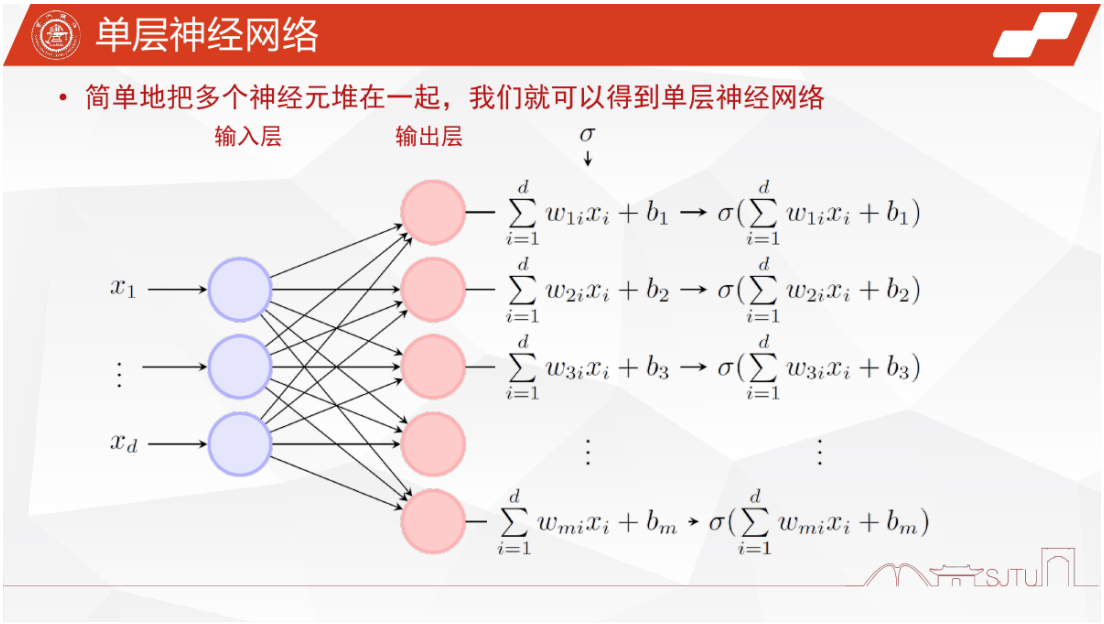
$$z = 0.5 \times 2 + (-1) \times 1 + 0.2 = 0.2$$

神经元最后输出为 $y = \sigma(0.2)$。如果 σ 是 ReLU，则 $y = \max(0, 0.2) = 0.2$。

2.2 全连接层

页码：p7-p8

把多个神经元放在同一层，并让每个输出神经元都连接到所有输入，就得到全连接层。全连接层的核心是矩阵乘法。



全连接层：全连接层中，每个输出神经元都接收上一层所有输入；它可以写成矩阵形式 $y = \sigma(Wx + b)$。

全连接层公式：

$$y = \sigma(Wx + b)$$

如果输入维度是 d_{in}，输出维度是 d_{out}，则

$$x \in R^{d_{in}}, \qquad W \in R^{d_{out} \times d_{in}}, \qquad b \in R^{d_{out}}, \qquad y \in R^{d_{out}}$$

例子 2：全连接层可以看成多个并行神经元。设输入 $x = [1, 2]^T$，一层有两个输出神经元：

$$W = \begin{bmatrix} 1 & 0 \\ 0.5 & -1 \end{bmatrix}, \qquad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

则

$$Wx + b = \begin{bmatrix} 1 \\ -0.5 \end{bmatrix}$$

如果激活函数是 ReLU，输出为 $[1, 0]^T$。矩阵 W 的每一行可以看成一个输出神经元的权重。

2.3 多层神经网络与激活函数

页码：p10-p18

多层神经网络是在全连接层基础上继续堆叠隐藏层。只做线性变换不够，因为很多任务本身是非线性的；加入非线性激活函数后，网络才能表达更复杂的函数。

多层神经网络 MLP： MLP 由多层全连接层和非线性激活函数堆叠而成，常见形式是“线性变换 + 激活函数”重复多次。

一层隐藏层的常见形式：

$$h = \sigma(W_1 x + b_1), \qquad y = W_2 h + b_2$$

把 h 代入后，可以写成：

$$y = W_2 \sigma(W_1 x + b_1) + b_2$$

三层网络（两层隐藏层）的常见形式：

$$h_1 = \sigma(W_1 x + b_1), \qquad h_2 = \sigma(W_2 h_1 + b_2), \qquad y = W_3 h_2 + b_3$$

展开后可以写成：

$$y = W_3 \sigma\left(W_2 \sigma(W_1 x + b_1) + b_2\right) + b_3$$

更深的 MLP 本质上就是不断重复“线性变换 + 非线性激活”。其中，h_1, h_2 是隐藏层表示；激活函数让网络具有非线性表达能力。

例子 3：为什么需要激活函数。如果连续两层都只是线性变换，那么

$$W_2(W_1 x + b_1) + b_2 = (W_2 W_1)x + (W_2 b_1 + b_2)$$

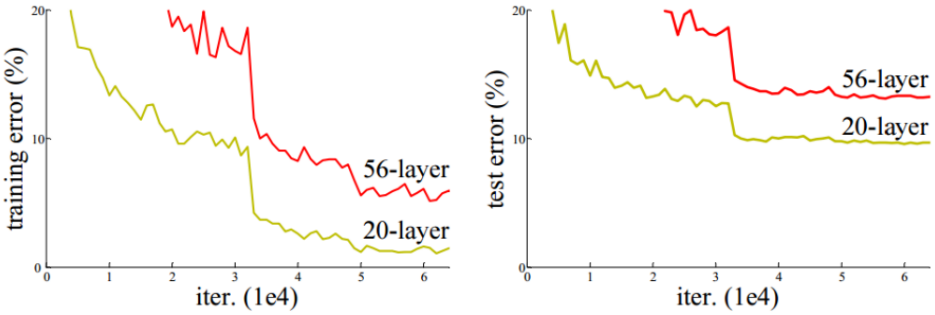
整体仍然只是一个线性变换。加入非线性激活函数后，多层网络才不容易退化成单层线性模型。

2.4 深层网络训练困难

页码：p19-p21

网络变深后，理论表达能力更强，但实际训练不一定更好。普通深层网络可能因为链式反向传播太长而出现梯度消失或梯度爆炸，使优化变困难，最后表现为退化问题：深层网络反而可能不如浅层网络好。

退化问题



利用20层和56层的普通网络对于CIFAR-10数据集的训练误差（左图）和测试误差（右图）。可见**越深的网络会产生越大的训练误差和测试误差。**

[1] He, Kaiming , et al. "Deep Residual Learning for Image Recognition." IEEE (2016).

退化问题：当网络加深后，训练误差和测试误差没有变好，反而可能变差，这说明深层普通网络更难优化。

Sigmoid 函数在输入很大或很小时会饱和，梯度接近 0。深层网络反向传播要经过很多层，梯度可能不断变小，导致前面层的参数很难更新。

例子 4：梯度消失的直观理解。如果每一层反向传播时梯度都乘以 0.5，经过 10 层后大约变成 $0.5^{10} \approx 0.001$。梯度太小，前面层就学得很慢。

2.5 残差连接

页码：p22-p23

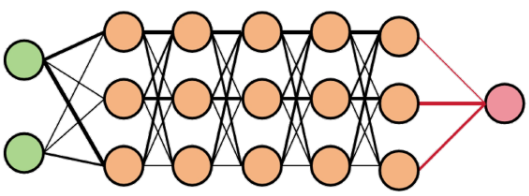
残差网络的核心做法是在普通网络层旁边加入一条 shortcut，让输出同时包含新学习到的变化和原始输入信息。

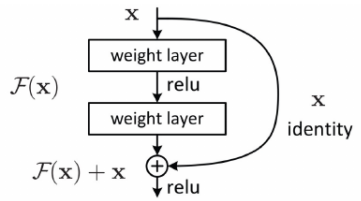
残差神经网络

问题：随着网络层数的增加，训练深层网络越来越困难

- 梯度爆炸和梯度消失问题
- 退化问题：网络深度增加时，性能出现饱和甚至下降的现象
  - 额外增加的层可能并未学习到新的特征表示，而只是近似恒等映射
  - 网络越深,参数量越大,过拟合风险也随之增加，导致泛化性能下降

解决方案：加入残差连接：$F_{l+1} = \sigma(W_l F_l + b_l) + F_l$





[1] He, Kaiming , et al. "Deep Residual Learning for Image Recognition." IEEE (2016).

残差连接：残差连接把某一层的输入 x 直接加到变换结果 $F(x)$ 上，使模块输出变为 $F(x)+x$。

残差连接的基本形式：

$$y = F(x) + x$$

课件中也写成：

$$F_{l+1} = \sigma(W_l F_l + b_l) + F_l$$

其中，F_l 是第 l 层输入特征，$\sigma(W_l F_l + b_l)$ 是这一层学习到的变换，$+F_l$ 表示把输入信息直接加回来。

例子 5：残差输出计算。如果输入 $x = [1, 2]^T$，网络学习到的变化是 $F(x) = [0.2, -0.1]^T$，则残差模块输出为

$$F(x) + x = [0.2, -0.1]^T + [1, 2]^T = [1.2, 1.9]^T$$

残差连接的意义不是让网络“少学一点”，而是让网络更容易学习恒等映射。如果某些额外层暂时学不到有用特征，它们可以让 $F(x)$ 接近 0，此时输出接近 x，网络不会因为加深而明显变差。

例子 6：恒等映射。如果一个残差模块暂时没有学到有效变化，令 $F(x) \approx 0$，则输出 $F(x) + x \approx x$。这说明深层网络至少可以比较容易地保留原来的信息。

2.6 关键记忆

- 单个神经元的基本计算是 $y = \sigma(w^T x + b)$，其中 σ 是激活函数。p4-p6
- 全连接层可以看成多个并行神经元，也可以写成 $y = \sigma(Wx + b)$；如果输入维度为 d_{in}、输出维度为 d_{out}，则 $W \in R^{d_{out} \times d_{in}}$。p7-p8
- 多层 MLP 本质上是不断重复“线性变换 + 非线性激活”；如果多层网络只有线性变换，整体仍然等价于一个线性变换。p10-p18
- 深层普通网络可能遇到梯度消失、梯度爆炸和退化问题；网络更深不一定更容易训练。p19-p21
- 残差连接的基本形式是 $F(x) + x$，它帮助信息和梯度在深层网络中传播。p22-p23
- ResNet 通过残差连接保留输入信息，让深层网络更容易优化，从而缓解退化问题。p22-p23

2.7 思考题

1. 填空：单个神经元常见计算形式是 $z = w^T x + b$，其中 b 叫做 _____。答案：偏置（bias）。
2. 判断：如果多层网络中每一层都只有线性变换，那么多层叠加后整体仍然等价于一个线性变换。答案：正确。
3. 计算：输入维度为 5、输出维度为 2 的全连接层，如果包含偏置，总参数量是多少？答案：$2 \times 5 + 2 = 12$。
4. 填空：残差连接的基本形式可以写成 $F(x) +$ ____。答案：x。
5. 判断：残差连接本身的核心作用是让信息和梯度更容易在深层网络中传播。答案：正确。

3 卷积神经网络 CNN

这一章按 PDF 顺序复习：图像特性 -> 类视皮层启发 -> 卷积计算 -> padding -> stride -> 多通道卷积 -> 池化 -> CNN 结构。页码均指本章 PDF 页码。

3.1 图像数据的特征：关联性

页码：p3-p5

图像有空间结构。相邻像素通常更相关，距离越远，关联通常越弱。课件中强调：真实图像主要表现为局部关联，但长程关联也不能完全忽略，因此更接近幂级数衰减，而不是单一的快速指数衰减。

图像关联性：图像数据具有局部关联性，并且整体上更接近幂级数衰减；也就是说，近处像素关联强，远处像素关联弱但不一定迅速消失。CNN 的局部连接设计正是为了利用这种空间关联。

如果两个像素点完全独立、互不影响，那么它们的相关函数或互信息量为 0。这个结论常用于判断“两个位置是否还有统计关联”。

模型	形式	记忆
指数衰减	$y = e^{-kx}$	下降较快
幂级数衰减	$y = \frac{1}{x^k}$	下降较慢，长程关联更明显

幂级数衰减可以看作许多指数衰减成分的叠加：

$$\int_0^\infty e^{-\alpha t}\, d\alpha = \frac{1}{t}$$

3.2 图像数据的特征：不变性

页码：p6

图像分类希望模型识别“物体是什么”，而不是死记每个像素的绝对位置。因此，物体发生小幅平移、旋转、伸缩或光照变化时，类别通常不应改变。

图像不变性：图像通常具有平移不变性、旋转不变性和伸缩不变性；这些变化不应轻易改变图像类别。

不是置换不变：图像通常不具有置换不变性；如果把像素顺序完全打乱，空间结构会被破坏，识别结果也会受到影响。

3.3 大脑如何处理图像：类视皮层设计

页码：p7-p10

大脑视觉系统处理图像时，不是一次性理解整张图，而是先在局部区域检测简单特征，再逐步汇总成更复杂的信息。这个部分主要用于理解 CNN 的设计动机。

类视皮层设计：CNN 借鉴了“先局部、后整体”的图像处理思路，前面层提取边缘、方向等简单局部特征，后面层逐步组合成更复杂的表示。

课件中的对应关系可以这样记：

视觉皮层概念	作用	CNN 中的对应
简单细胞	对位置、边缘和方向敏感；不同简单细胞有不同感受野	卷积层：用局部卷积核提取边缘、方向等局部特征
复杂细胞	整合多个简单细胞的输入	后续汇总模块，尤其是全连接层：把局部特征组合成更整体的表示

因此，CNN 的设计思路可以理解为：卷积层先像简单细胞一样提取局部特征，再通过池化、全连接等模块逐步汇总信息。

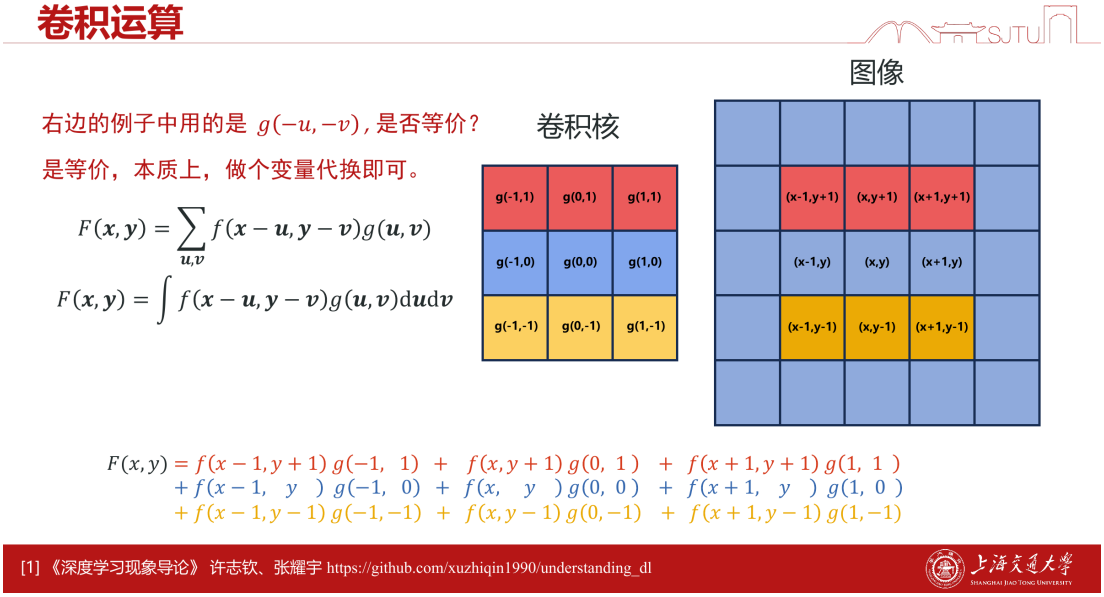
3.4 卷积运算

页码：p11-p13

卷积核是一个小窗口，例如 3 x 3。它在图像上滑动，每次覆盖一个局部区域，把局部区域和卷积核对应位置相乘后求和。

> **卷积运算**：卷积核在输入图像或特征图上滑动，每次对局部窗口做对应位置相乘再求和，得到输出特征图中的一个值。

卷积核中的权重通常是随机初始化后通过训练学习得到的，并不是固定不变的手工模板。卷积层的偏置（bias）也是可学习参数；每个输出通道通常对应一个偏置标量。



例子 **1**：单通道卷积得到多个输出位置。输入为 4×4，卷积核为 3×3，不加 padding，stride 为 1。
输入 X：

$$\begin{bmatrix}1 & 2 & 3 & 4\\ 5 & 6 & 7 & 8\\ 9 & 10 & 11 & 12\\ 13 & 14 & 15 & 16\end{bmatrix}$$

卷积核 K：

$$\begin{bmatrix}1 & 0 & 2\\ 0 & -1 & 1\\ 2 & 1 & 0\end{bmatrix}$$

输出大小为 2×2，因为 3×3 窗口在 4×4 输入上有四个可放置位置。每个位置都做“对应元素相乘，再求和”：

$$\begin{aligned}Y_{1,1} &= 1\times1+2\times0+3\times2+5\times0+6\times(-1)\\ &\quad+7\times1+9\times2+10\times1+11\times0=36\end{aligned}$$

$$\begin{aligned}Y_{1,2} &= 2\times1+3\times0+4\times2+6\times0+7\times(-1)\\ &\quad+8\times1+10\times2+11\times1+12\times0=42\end{aligned}$$

$$\begin{aligned}Y_{2,1} &= 5\times1+6\times0+7\times2+9\times0+10\times(-1)\\ &\quad+11\times1+13\times2+14\times1+15\times0=60\end{aligned}$$

$$\begin{aligned}Y_{2,2} &= 6\times1+7\times0+8\times2+10\times0+11\times(-1)\\ &\quad+12\times1+14\times2+15\times1+16\times0=66\end{aligned}$$

所以输出特征图为

$$\begin{bmatrix}36 & 42\\ 60 & 66\end{bmatrix}$$

卷积核继续向右或向下滑动，就得到下一个位置的输出。所有位置的输出组成新的特征图。

模型	连接方式	对图像结构的利用
MLP	常把图像展平成向量，全连接	容易弱化二维空间结构
CNN	局部连接，卷积核滑动	直接利用局部关联

3.5 Zero Padding

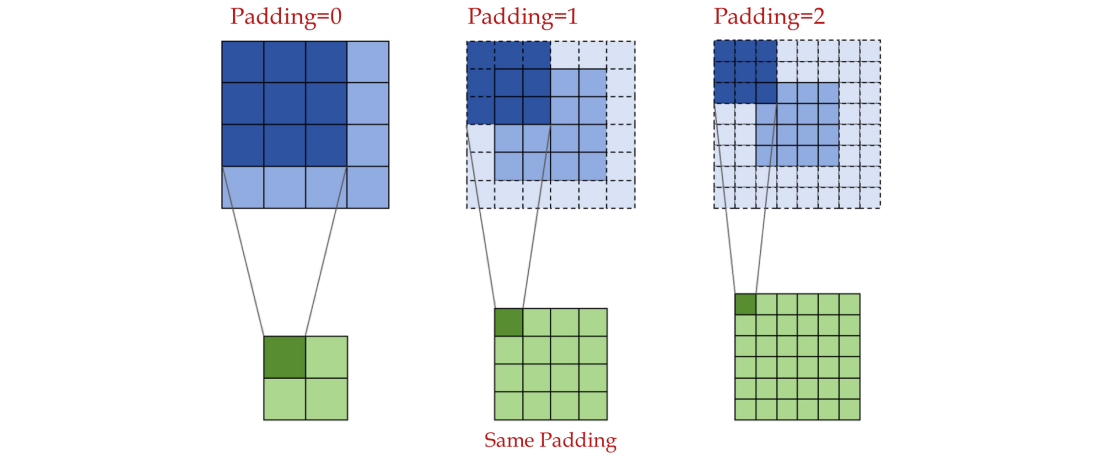
页码：p14

Padding 是在图像边缘补值，常见是补 0。作用是控制卷积后图像大小，避免图像尺寸过快变小。

Padding： Padding 是在输入边缘补值，常见是补 0，用来控制卷积后的空间尺寸，并让边缘位置也能参与更多卷积计算。

Same Padding 指通过合适的 padding 让输出空间尺寸和输入空间尺寸相同。填充本身只是补值，不会引入可学习参数。

Zero Padding———解决图像过卷积后大小改变的问题



例子 **2**：**Zero padding**。原图像为 3×3：

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

padding 为 1 后变成 5×5：

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

使用 `3 x 3` 卷积核、stride=1 时：

例子 **3**：**padding 对输出大小的影响**。使用 '3 x 3' 卷积核、stride=1 时，不加 padding：$3 \times 3 \to 1 \times 1$；加 padding 为 1：$3 \times 3 \to 3 \times 3$。

3.6 Stride

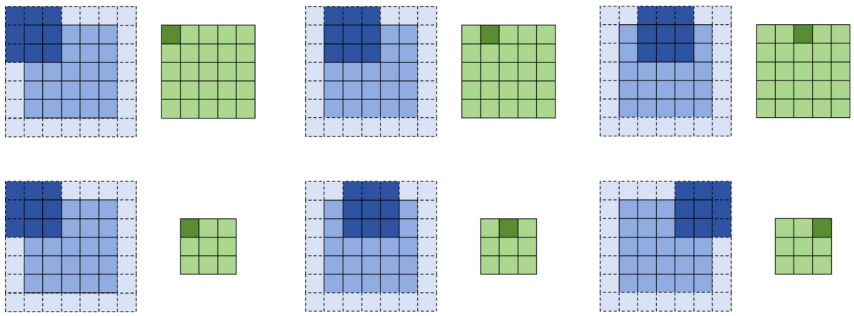
页码：p15

Stride 是卷积核滑动步长。课件用 `stride(i, j)` 表示横向跨 `i` 步、纵向跨 `j` 步。

Stride： Stride 是卷积核每次滑动的步长。stride 越大，卷积核采样位置越少，输出空间尺寸通常越小。

跨步 stride(i, j)

横向跨i步，纵向跨j步。图中第一行为stride(1,1)，第二行为stride(2,2)



[1]《深度学习现象导论》 许志钦、张耀宇 https://github.com/xuzhiqin1990/understanding_dl

例子 **4**：对照 **stride** 图理解滑动步长。上图第一行是 stride$(1,1)$：卷积核每次向右或向下移动 1 格，所以相邻窗口高度重叠，输出位置较多。
上图第二行是 stride$(2,2)$：卷积核每次向右或向下移动 2 格，中间会跳过一些位置，所以输出位置减少，空间尺寸更小。

3.7 多通道卷积

页码：p16

RGB 图像有 3 个输入通道。多通道卷积中，一个完整卷积核需要覆盖所有输入通道。

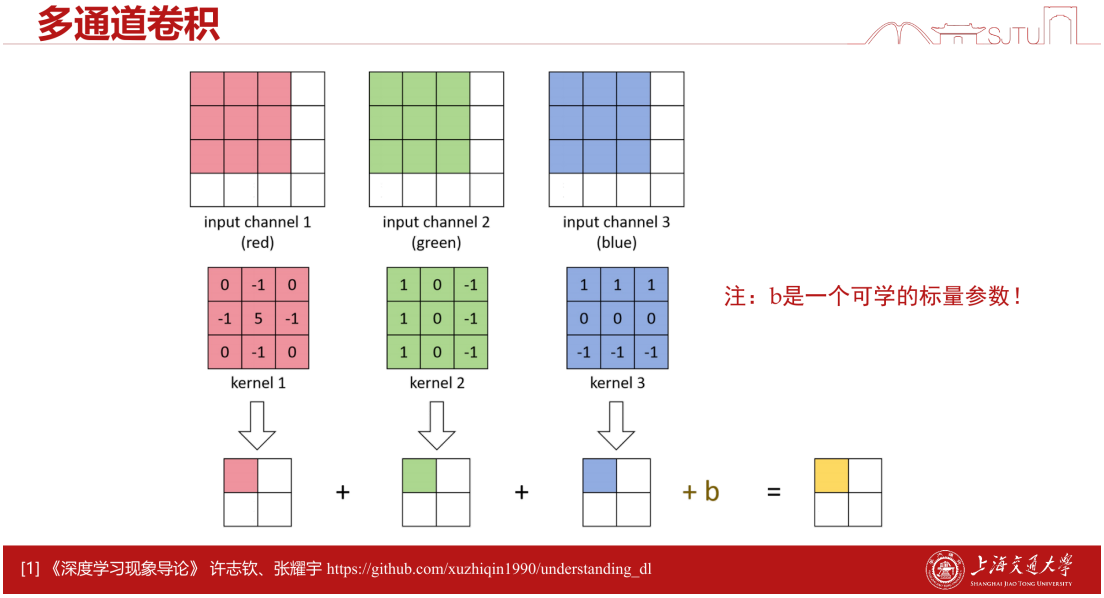
多通道卷积：对于有 C_{in} 个输入通道的数据，一个完整卷积核的深度也必须覆盖 C_{in} 个通道；卷积核个数通常记为 C_{out}，也就是输出通道数。

单个输出通道的计算：

下面公式中的 $*$ 表示单通道卷积操作：先在同一个输入通道内做局部窗口乘加，再把所有输入通道的结果相加。

$$Y_j = \sum_{c=1}^{C_{\text{in}}} X_c * K_{j,c} + b_j$$

其中，X_c 是第 c 个输入通道，$K_{j,c}$ 是第 j 个卷积核在第 c 个输入通道上的部分，b_j 是第 j 个输出通道的偏置 (bias)。对 $j = 1, \ldots, C_{\text{out}}$ 都计算一次，就得到 C_{out} 个输出通道。



例子 **5**：**多通道卷积**。RGB 输入图像有 3 个通道。R 通道局部区域与卷积核 R 部分做一次单通道卷积，G 通道局部区域与卷积核 G 部分做一次单通道卷积，B 通道局部区域与卷积核 B 部分做一次单通道卷积；三个通道结果相加，再加偏置（bias），就是这个卷积核在当前位置的输出。

例子 **6**：**通道数**。输入为 $32 \times 32 \times 3$，一个完整卷积核大小为 $5 \times 5 \times 3$。如果卷积核个数为 6，则输出空间大小另算，输出通道数为 6。

3.8 卷积层尺寸计算总结

页码：p14-p16

卷积层的计算围绕三件事：空间大小怎么变，通道数怎么变，参数量怎么算。

卷积层输出形状：输入为 $H \times W \times C_{\text{in}}$，卷积核个数为 C_{out} 时，输出形状为 $H_{\text{out}} \times W_{\text{out}} \times C_{\text{out}}$；输出通道数由卷积核个数决定。

输入、卷积核和输出形状：

$$\text{输入：} H \times W \times C_{\text{in}}$$

$$\text{一个卷积核：} K \times K \times C_{\text{in}}, \qquad \text{卷积核个数：} C_{\text{out}}$$

$$\text{输出：} H_{\text{out}} \times W_{\text{out}} \times C_{\text{out}}$$

H, W 是输入高和宽；C_{in} 是输入通道数；K 是卷积核大小；C_{out} 是卷积核个数，也就是输出通道数。

输出空间尺寸公式：

$$H_{\text{out}} = \left\lfloor \frac{H + 2P - K}{S} \right\rfloor + 1$$

$$W_{\text{out}} = \left\lfloor \frac{W + 2P - K}{S} \right\rfloor + 1$$

其中，P 是 padding 大小，S 是 stride 大小，$\lfloor \cdot \rfloor$ 表示向下取整。

尺寸影响：

参数	增大后通常会怎样	直观理解
padding	输出空间尺寸变大	边缘补了一圈，能滑动的位置更多
stride	输出空间尺寸变小	卷积核跳得更远，采样位置更少
卷积核大小	输出空间尺寸变小	窗口更大，可放置的位置更少
卷积核个数	输出通道数变多	一个卷积核产生一个输出通道

参数量计算：

卷积层通常有偏置（bias）；每个输出通道对应 1 个偏置。

$$\#\text{params} = K \times K \times C_{\text{in}} \times C_{\text{out}}$$

考虑偏置（bias）：

$$\#\text{params} = K \times K \times C_{\text{in}} \times C_{\text{out}} + C_{\text{out}}$$

例子 **7**：**参数量计算**。输入通道数 $C_{\text{in}} = 3$，卷积核大小 $K = 5$，卷积核个数 $C_{\text{out}} = 6$。不考虑偏置时，参数量为 $5 \times 5 \times 3 \times 6 = 450$；考虑偏置时，参数量为 $5 \times 5 \times 3 \times 6 + 6 = 456$。

例子 **8**：**padding 保持大小**。输入为 $32 \times 32 \times 3$，使用 3×3 卷积核，卷积核个数为 16，padding 为 1，stride 为 1。

$$H_{\text{out}} = \frac{32 + 2 \times 1 - 3}{1} + 1 = 32$$

因此输出为 $32 \times 32 \times 16$。

例子 **9**：**stride 让尺寸减小**。输入为 $32 \times 32 \times 3$，使用 3×3 卷积核，卷积核个数为 16，padding 为 1，stride 为 2。

$$H_{\text{out}} = \left\lfloor \frac{32 + 2 \times 1 - 3}{2} \right\rfloor + 1 = 16$$

因此输出为 $16 \times 16 \times 16$。

3.9 池化 Pooling

页码：p18

池化层用于对局部区域做汇总，常见方式是最大池化和平均池化。池化通常用于降低空间尺寸、减少计算量，一般没有可学习参数，也通常不改变通道数。

池化 Pooling：池化是在局部窗口内做汇总操作。最大池化取窗口最大值，保留最强响应；平均池化取窗口平均值，保留整体平均水平。

例子 **10**：**最大池化和平均池化**。对局部区域

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

做最大池化时，输出为 $\max(1,3,2,4)=4$；做平均池化时，输出为 $\frac{1+3+2+4}{4}=2.5$。

例子 **11**：**池化尺寸**。输入为 $32\times32\times16$，最大池化窗口为 2×2，stride 为 2，输出为 $16\times16\times16$。

全局平均池化 GAP：GAP 对每个通道在整个空间范围内取平均，把每个通道压成 1 个数。它改变空间尺寸，但不改变通道数；接在全连接层前通常可以大幅减少参数量。

3.10 CNN 整体结构与层的组合

页码：p20-p22

p20 给出了一个典型 CNN 结构：

输入 `RGB` 三通道图像
-> 卷积 + `ReLU` + 池化
-> 卷积 + `ReLU` + 池化
-> 展平
-> 全连接
-> `Softmax` 输出类别概率

CNN 结构组合：CNN 没有唯一固定的网络格式；卷积层、激活函数、池化层、全连接层等可以按任务需要组合。

例子 **12**：**经典 LeNet 结构**。输入图像 → 卷积 → 池化 → 卷积 → 池化 → 展平 → 全连接 → 输出。

只要前一层输出形状能作为后一层输入，卷积、激活、池化、展平、全连接等层就可以灵活搭配。不同 CNN 的差别主要体现在层数、顺序、卷积核个数和是否下采样。

3.11 关键记忆

- 图像具有局部关联性和一定不变性；图像关联性更接近幂级数衰减，近处关联强，远处关联弱但不一定迅速消失。p3-p6
- 图像不具有置换不变性；完全打乱像素顺序会破坏空间结构。p6
- 两个像素点完全独立时，相关函数或互信息量为 0。p3-p5
- 指数衰减下降较快，幂级数衰减下降较慢；幂级数衰减可以看作许多指数衰减成分的叠加。p3-p5
- 类视皮层设计启发 CNN：先提取局部简单特征，再逐步整合成复杂信息。p7-p10
- 卷积的核心是局部窗口加权求和；卷积核和偏置（bias）都是可训练参数。p11-p13
- Padding 在边缘补值，用来控制输出空间大小；Same Padding 可保持空间尺寸不变。p14
- Stride 是卷积核滑动步长；stride 越大，输出空间尺寸通常越小。p15
- 多通道卷积中，一个完整卷积核覆盖所有输入通道；一个卷积核产生一个输出通道。p16
- 卷积输出空间尺寸看 H, W, K, P, S；输出通道数看卷积核个数 C_{out}。p14-p16
- 卷积参数量看 $K, C_{\text{in}}, C_{\text{out}}$；有偏置（bias）时再加 C_{out}，不要乘 H_{out} 和 W_{out}。p14-p16
- 池化汇总局部信息，降低空间尺寸，通常没有可学习参数，也通常不改变通道数；GAP 把每个通道压成 1 个数，能减少后续全连接层参数量。p18
- CNN 没有唯一固定格式；卷积、激活、池化、展平、全连接、softmax 等模块可以灵活组合，但相邻层的输入输出尺寸必须匹配。p20-p22

3.12 思考题

1. 判断：池化操作一定只能由专门的池化层实现，不能用卷积实现类似效果。答案：错误。例如使用带 stride 的卷积可以在提取特征的同时降低空间尺寸，起到类似下采样的作用。
2. 判断：池化操作可以和卷积操作在同一层的设计中部分合并，例如用 stride 大于 1 的卷积完成下采样。答案：正确。
3. 填空：`1 x 1` 卷积核不改变单个位置的空间邻域大小，但可以改变 _____ 数。答案：通道。
4. 计算：输入为 `32 x 32 x 3`，卷积核为 `3 x 3`，padding=1，stride=1，卷积核个数为 16，输出尺寸是 _____。答案：`32 x 32 x 16`。
5. 计算：一个卷积层中 $K = 5$，$C_{\text{in}} = 3$，$C_{\text{out}} = 6$，且每个卷积核有偏置（bias），参数量是 _____。

   答案：`5 x 5 x 3 x 6 + 6 = 456`。

4 循环神经网络 RNN 与 LSTM

这一章按 PDF 顺序复习：语言任务特点 -> 数据准备 -> 词元化 -> one-hot 与 embedding -> next token prediction -> RNN 设计 -> BPTT -> LSTM。本章帮助学生理解：文本是可变长度序列，RNN 为什么适合处理序列，以及 LSTM 的门控机制解决了什么问题。

4.1 自然语言处理 NLP 的任务特点

页码：p3-p5

NLP 的目标是让计算机理解、生成和处理自然语言。语言有层次结构、语法、上下文相关性和长程依赖；更重要的是，文本天然是**可变长度的离散符号序列**。

自然语言处理 NLP： NLP 是让计算机理解、生成和处理人类自然语言的技术方向。

文本和图像的一个关键区别是：文本序列长度通常不固定，不同句子的 token 数可能不同。

例子：**图像与文本输入**。图像通常可以统一 resize 成固定大小，例如 $32 \times 32 \times 3$；文本句子却长短不一，例如“我喜欢 AI”和“我非常喜欢人工智能这门课”的 token 数不同。

语言数据的几个特点：

特点	含义	例子
离散符号序列	文本由词、子词、字符等符号组成	“我 / 喜欢 / AI”
顺序重要	词语顺序会影响含义	“我打他”和“他打我”
上下文相关	当前词含义依赖上下文	“上海的交通大学”和“上海的交通”
长程依赖	较远位置的信息可能影响后面理解	前文主语影响后文指代

MLP、CNN 与 RNN 的区别：

模型	输入特点	是否天然适合可变长度序列
MLP	常需要固定长度向量	不适合
CNN	擅长局部窗口特征	不天然适合
RNN	逐 token 处理序列	适合

RNN 不要求一次性输入固定长度向量，而是按时间步读入 token，并把历史压缩到隐藏状态中。

RNN 处理可变长度序列的核心： RNN 按时间步逐个处理输入 token，并在不同时间步共享同一组参数；同一个 RNN 单元可以重复使用任意多次，因此可以处理不同长度的序列。

4.2 数据准备

页码：p6-p7

数据准备是语言模型训练的基础。文本进入模型前通常需要清洗和处理。

数据预处理：数据预处理把原始文本变成模型可用的数据，通常包括收集、过滤、去重、词元化和向量化。

步骤	作用
数据收集	获得足够数量的文本
质量过滤	去掉质量很差、无意义或噪声很大的文本
敏感内容过滤	减少有毒内容、隐私信息等风险
数据去重	减少重复样本，避免模型过度记忆重复文本
词元化	把文本切成 token
向量化	把 token 变成神经网络能处理的数值向量

数据量增加通常有助于提升性能；数据质量高可以节约算力、增强稳定性、减少错误输出；大量重复数据可能让模型过度记忆重复模式，反而降低上下文处理能力。

4.3 分词、one-hot 与 embedding

页码：p8-p11

文本不能直接进入神经网络，通常要先切成 token，再变成向量。

文本 `->` `tokenization` `->` token 序列 `->` `one-hot` 或 `embedding` 向量

Tokenization： Tokenization 是把原始文本切分成 token 序列的过程；token 可以是词、子词、字符或特殊符号。

常见词元化方法：

下面这些方法只需要知道是常见切分方式，不需要背具体算法细节。

方法	记忆
基于规则	按空格、标点、词典等规则切分
BPE	从基本字符开始，逐步合并高频相邻片段
WordPiece	常见子词切分方法
Unigram	用统计模型选择合适的子词切分

例子 1：**Tokenization**。“我喜欢人工智能”可以切成词级 token：[我, 喜欢, 人工智能]；也可以切成子词级 token：[我, 喜欢, 人工, 智能]。

常见特殊符号：

下面这些符号只是常见约定，用来帮助读图和理解输入格式，不需要逐个背诵。

特殊符号	常见含义
`CLS`	句首或分类标记

特殊符号	常见含义
SEP	句尾或分隔标记
MASK	掩码标记
PAD	填充标记，用于补齐长度
UNK	未登录词或未知 token

Token 还需要变成数值向量，常见表示方式是 one-hot 和 embedding。

One-hot： One-hot 是用词表长度的稀疏向量表示一个 token，只有该 token 对应位置为 1，其余位置为 0。

Embedding： Embedding 是把 token 映射成低维稠密向量的表示方法，通常由可训练的 embedding 矩阵得到。

表示方式	特点	记忆
one-hot	高维、稀疏	不表达词义相似性
embedding	低维、稠密、可训练	可以学习语义关系

例子 **2**：**One-hot 与 embedding**。设词表为“我、喜欢、人工智能、这门课”，token “喜欢”的 one-hot 可以写成 $[0, 1, 0, 0]$，维度等于词表大小 4。如果 embedding 维度设为 3，它会通过可训练矩阵映射为一个三维向量，例如 $[0.12, -0.38, 0.51]$。

Embedding 向量由 one-hot 向量乘上一个可训练矩阵得到。设词表大小为 N，embedding 维度为 d：

$$\varepsilon_i \in R^N, \qquad W_{\text{emb}} \in R^{d \times N}$$

$$x_i = W_{\text{emb}} \varepsilon_i, \qquad x_i \in R^d$$

其中，ε_i 是第 i 个 token 的 one-hot 向量，W_{emb} 是可训练的 embedding 矩阵，x_i 是该 token 的 embedding 向量。

4.4 Next Token Prediction

页码：p13-p16

Next Token Prediction 是根据已有前文预测下一个 token。它是现代语言模型常用的训练框架。

Next Token Prediction：给定前文 token 序列，模型预测下一个 token；训练时每个位置的目标通常是输入序列中下一个位置的 token。

Next Token Prediction 是一种自监督训练方式：训练标签直接来自原始文本中的“下一个 token”，不需要额外人工标注。

Next Token Prediction框架

语言模型的训练普遍采用Next Token Prediction框架，即输出序列的每一个token为输入序列的下一个token

假设输入为： $X^{\text{in}} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$

模型的预测输出为： $\hat{X}^{\text{out}} = (\hat{\boldsymbol{x}}_2, \hat{\boldsymbol{x}}_3, ..., \hat{\boldsymbol{x}}_{n+1})$

根据Next Token Prediction，我们期望它的输出为: $X^{\text{out}} = (\boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_{n+1})$

于是可以对其计算交叉熵损失:

$$\mathcal{L}(X, \hat{X}) = -\frac{1}{n}\sum_{i=2}^{n+1}\sum_{j=1}^{d}\boldsymbol{x}_{ij}\log\hat{\boldsymbol{x}}_{ij}$$

训练时，每个位置预测下一个 token；推理时，模型生成一个 token 后，把它接到上下文后继续生成。图中的 X^{in} 是输入序列，$\hat{X}^{out}$ 是预测分布，X^{out} 是输入序列整体向后移动一位后的目标序列。推理可用贪婪解码，也可按概率采样；生成到终止符（如 SEP）时停止。

训练时常用交叉熵损失，真实 token 的概率越高，loss 越小。图中给出的交叉熵损失可以写成：

$$\mathcal{L}(X, \hat{X}) = -\frac{1}{n}\sum_{i=2}^{n+1}\sum_{j=1}^{d}x_{ij}\log\hat{x}_{ij}$$

其中，x_{ij} 表示真实下一个 token 的 one-hot 标签，$\hat{x}_{ij}$ 表示模型预测的概率。

4.5 RNN 的设计

页码：p17-p18

RNN 是循环神经网络，适合处理序列数据。它的核心设计是：当前隐藏状态不仅看当前输入，也看上一时刻隐藏状态。

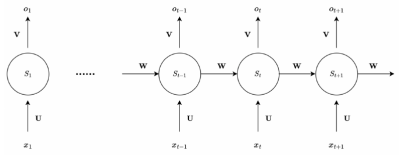


图 3.19: 循环神经网络的基本结构

如图3.19所示，这是一个基本的循环神经网络单元，下面我们详细介绍一下这个单元是如何工作的。我们的输入为时序序列 $x_1, x_2, \cdots, x_{t-1}, x_t, x_{t+1}, \cdots, x_n$，并通过公式(3.15),(3.16)和(3.17)的方式计算得到对应的输出序列 $o_1, o_2, \cdots, o_{t-1}, o_t, o_{t+1}, \cdots, o_n$。

$$S_0 = 0, \tag{3.15}$$

$$S_t = f(\mathbf{U} \cdot x_t + \mathbf{W} \cdot S_{t-1} + b), \quad t = 1, 2, 3, \cdots, n, \tag{3.16}$$

$$o_t = g(\mathbf{V} \cdot S_t) \tag{3.17}$$

其中 f 是激活函数，通常使用 Tanh 函数，$g = \text{Softmax}(x)$ 是 softmax 函数。S_t 是状态记忆，$\mathbf{U}$，$\mathbf{W}$，$\mathbf{V}$ 和 b 分别是输入权重矩阵、循环权重矩阵、输出权重矩阵和偏置项。

隐藏状态： RNN 的隐藏状态 S_t 是模型在第 t 步保存的历史信息摘要，它由当前输入 x_t 和上一时刻隐藏状态 S_{t-1} 共同决定。

RNN 的隐藏状态可以理解为“目前为止读过内容的压缩记忆”。序列长度不同也没关系，因为 RNN 可以重复使用同一个计算单元，按时间步逐个处理输入。

RNN 的基本公式：

$$S_0 = 0$$

$$S_t = f(Ux_t + WS_{t-1} + b), \qquad t = 1, 2, \dots, n$$

$$o_t = g(VS_t + b_y)$$

其中：

设输入维度为 d_x，隐藏状态维度为 d_h，输出维度为 d_y：

符号	含义	形状
x_t	第 t 个输入向量	R^{d_x}
S_t	第 t 步隐藏状态	R^{d_h}
S_{t-1}	上一时刻隐藏状态	R^{d_h}
o_t	第 t 步输出	R^{d_y}
U	输入到隐藏状态的权重矩阵	$R^{d_h \times d_x}$
W	隐藏状态到隐藏状态的权重矩阵	$R^{d_h \times d_h}$
V	隐藏状态到输出的权重矩阵	$R^{d_y \times d_h}$
b	隐藏状态偏置项	R^{d_h}
b_y	输出偏置项	R^{d_y}

f 常用 tanh，g 在分类或预测 token 时常用 softmax。

RNN 参数量计算：

$$\#\text{params} = d_h d_x + d_h d_h + d_y d_h + d_h + d_y$$

其中，$d_h d_x$ 来自 U，$d_h d_h$ 来自 W，$d_y d_h$ 来自 V，d_h 来自隐藏状态偏置 b，d_y 来自输出偏置 b_y。

RNN 参数量例子：若输入维度 $d_x = 4$，隐藏维度 $d_h = 3$，输出维度 $d_y = 2$，则

$$\#\text{params} = 3 \times 4 + 3 \times 3 + 2 \times 3 + 3 + 2 = 32$$

RNN 公式中的三件事：

1. S_t 同时依赖当前输入 x_t 和历史状态 S_{t-1}。
2. 参数 U, W, V 在不同时间步共享。
3. 输入多少个 token，就重复多少次同样的 RNN 单元，所以可以处理可变长度序列。

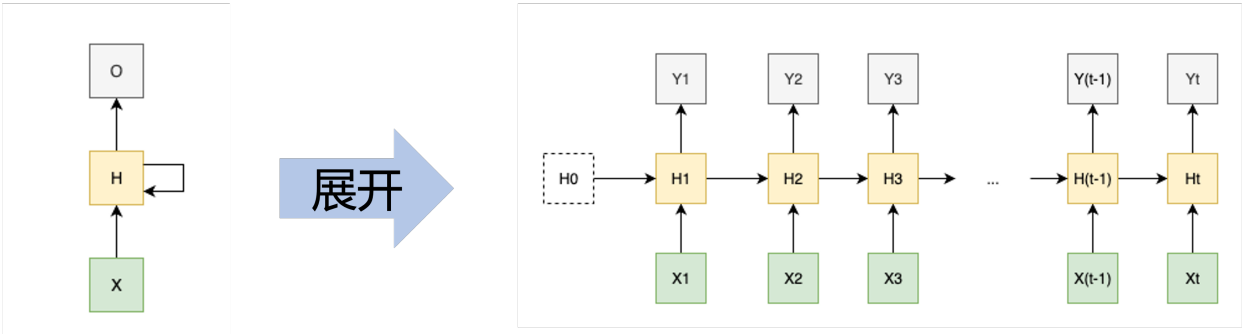
Encoder-Decoder RNN： Encoder-Decoder RNN 先把输入序列编码成上下文表示，再逐步解码生成输出序列，可以处理输入输出长度不同的任务，例如机器翻译。这部分理解思想即可。

4.6 BPTT 与长序列训练

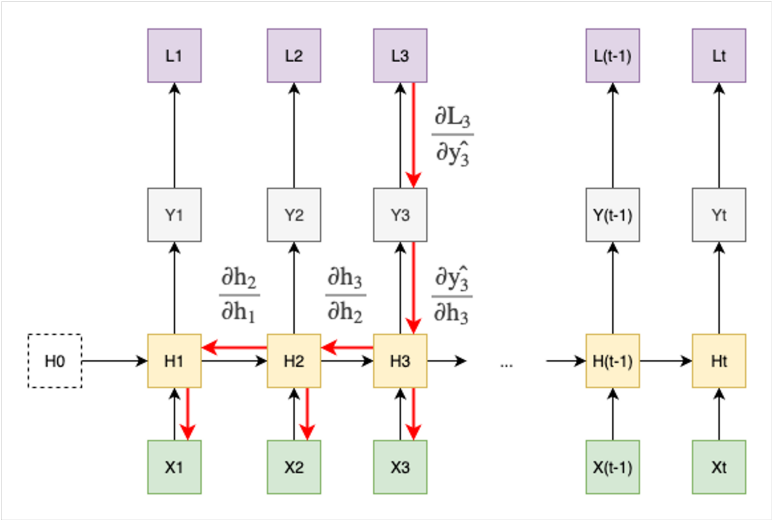
页码：p20

BPTT 是 BackPropagation Through Time，意思是随时间反向传播。训练 RNN 时，误差要沿多个时间步向前传播回去。

BPTT： BPTT 是随时间反向传播，把 RNN 按时间步展开后，沿时间方向反向计算梯度。



BPTT



图中上半部分表示 RNN 按时间步展开；下半部分的红色箭头表示误差沿展开后的时间方向反向传播，计算梯度并用梯度下降更新参数。

长序列训练时，完整 BPTT 需要展开并回传很多时间步，计算和显存开销都很大。实际训练中常用截断 BPTT，只向前回看有限步数，而不是对无限长历史完整反传。

只记结论：BPTT 在很长时间步上反向传播时，可能出现梯度消失或梯度爆炸；这里不展开原因。

普通 RNN 对长程依赖的建模能力有限。下一节的 LSTM 可以理解为在 RNN 中加入记忆状态和门控机制，用来更好地保留长期信息。

4.7 LSTM

页码：p21-p25

LSTM 是 Long Short-Term Memory，长短期记忆网络。它在 RNN 基础上引入记忆状态，并通过门控机制控制信息流动。

长短期记忆网络（Long short-term memory, LSTM）

基本原则：1) 任何信息使用前先做线性变换,权重可学
2) 控制比例的称为“门”，激活用Sigmoid
3) 提取信息的运算，激活用Tanh

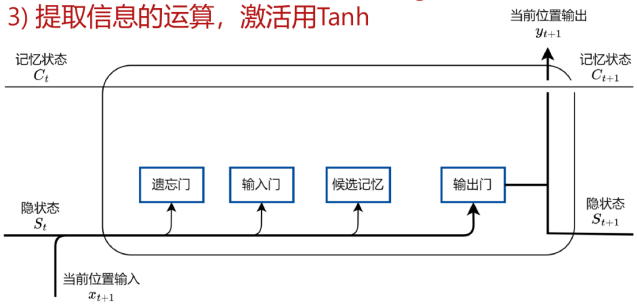


图 3.21: 单个门控神经元主要部件和并行隐藏状态示意图

门控神经元-遗忘门

保留的比例 $f_t = \sigma(W_{Sf}S_t + W_{xf}x_{t+1} + b_f)$

门控神经元-输入门

写入的比例 $i_t = \sigma(W_{Si}S_t + W_{xi}x_{t+1} + b_i),$

新的知识 $\tilde{C}_{t+1} = \tanh(W_{SC}S_t + W_{xC}x_{t+1} + b_C)$

新的记忆状态 $C_{t+1} = f_t * C_t + i_t * \tilde{C}_{t+1},$

门控神经元-输出门

输出的比例 $o_t = \sigma(W_{So}S_t + W_{xo}x_{t+1} + b_o),$

输出 $y_{t+1} = S_{t+1} = o_t * \tanh(C_{t+1}).$

为什么能称为“遗忘门”、“输入门”、“输出门”、“知识”？
这是一种形象类比，没有严格证明。

LSTM：LSTM 是一种改进的 RNN，它通过记忆状态和门控机制控制信息的保留、写入和输出，用来缓解普通 RNN 难以保留长期信息的问题。

结构	作用
记忆状态	保存较长期的信息
遗忘门	控制保留多少旧记忆
输入门	控制写入多少新信息
输出门	控制输出多少记忆信息

LSTM 三条设计原则：

1. 信息使用前先做可学习的线性变换。
2. 控制比例的“门”用 sigmoid，输出在 0 到 1 之间。
3. 提取候选信息常用 tanh。

门控机制：门控机制用 0 到 1 之间的比例控制信息流动：遗忘门控制保留多少旧记忆，输入门控制写入多少新信息，输出门控制输出多少记忆信息。这里理解概念即可，不需要记复杂公式。

普通 RNN 与 LSTM 的区别可以概括为：

对比	普通 RNN	LSTM
保存历史信息	主要依靠隐藏状态 S_t	额外引入记忆状态 C_t
信息控制	缺少显式门控	用遗忘门、输入门、输出门控制信息比例
长程依赖	长序列中长期信息更难保留	通过记忆状态和门控机制缓解长程依赖
很长序列训练	常配合截断 BPTT	仍可配合截断 BPTT，但更容易保留长期信息

截断 BPTT 解决的是“序列太长，不能无限展开反传”的训练开销问题；LSTM 解决的是“普通 RNN 难以保留长期信息”的建模问题。

4.8 关键记忆

- NLP 处理的是离散符号序列，词语顺序和上下文都很重要。
- 文本数据通常是可变长度的；MLP/CNN 不天然适合记忆历史信息，RNN 通过隐藏状态逐步处理序列。
- 文本进入神经网络前通常要经过数据清洗、tokenization 和向量化。
- Tokenization 是把文本切成 token；常见词元化方法和特殊符号只需理解用途，不需要作为背诵清单。
- One-hot 高维稀疏，本身不表达词义相似性；embedding 由 one-hot 乘可训练矩阵得到，低维稠密，可以通过训练学习语义关系。
- Next Token Prediction 是根据前文预测下一个 token，是自监督训练框架，标签来自文本中的下一个 token。
- RNN 的当前隐藏状态依赖当前输入和上一时刻隐藏状态，公式为 $S_t = f(Ux_t + WS_{t-1} + b)$；参数量计算要包含 U, W, V, b, b_y。
- BPTT 是随时间反向传播；长序列完整反传开销大，常用截断 BPTT；很长时间步上可能出现梯度消失或梯度爆炸，只需记结论。p20
- LSTM 通过记忆状态和门控机制控制信息保留、写入和输出，用来缓解普通 RNN 难以保留长期信息的问题。

4.9 思考题

1. 判断：文本数据通常是可变长度序列，因此普通 MLP 不天然适合直接处理任意长度文本。答案：正确。
2. 填空：把文本切成 token 的过程叫做 _____。答案：tokenization 或词元化。
3. 填空：RNN 的基本公式中，当前隐藏状态可以写成 $S_t = f(Ux_t + W$ ____ $+ b)$。答案：S_{t-1}。
4. 判断：RNN 可以处理可变长度序列，一个重要原因是同一个 RNN 单元可以在不同时间步重复使用。答案：正确。
5. 选择：LSTM 中控制保留多少旧记忆的是：A. 输入门 B. 遗忘门 C. 输出门 D. Softmax。答案：B。
6. 计算：若 RNN 的输入维度 $d_x = 4$，隐藏维度 $d_h = 3$，输出维度 $d_y = 2$，参数量是多少？答案：$3 \times 4 + 3 \times 3 + 2 \times 3 + 3 + 2 = 32$。

5 Transformer

这一章按 PDF 顺序复习：为什么需要 Transformer -> next token prediction 回顾 -> 位置编码 -> 注意力机制与 Q/K/V -> causal mask -> Transformer block -> 输出层。本章围绕 Transformer 如何用注意力机制建模 token 之间的关系，以及为什么它比 RNN 更适合并行计算。

5.1 为什么需要 Transformer

页码：p3-p9

Transformer 出现前，序列任务常用 RNN/LSTM。RNN/LSTM 能处理序列，但有两个明显限制：长程依赖不容易保留，并且计算通常要按时间步顺序进行，不方便并行。

Attention is all you need!

Why Transformer?

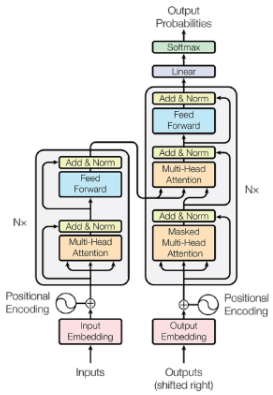
Transformer [1] 诞生之前，在自然语言处理中大都采用基于RNN的编码器-解码器（Encoder-Decoder）结构来完成序列翻译。

RNN（LSTM）的不足：

- 难以捕获长期依赖关系（梯度消失和梯度爆炸仍存在）

Transformer的优势：

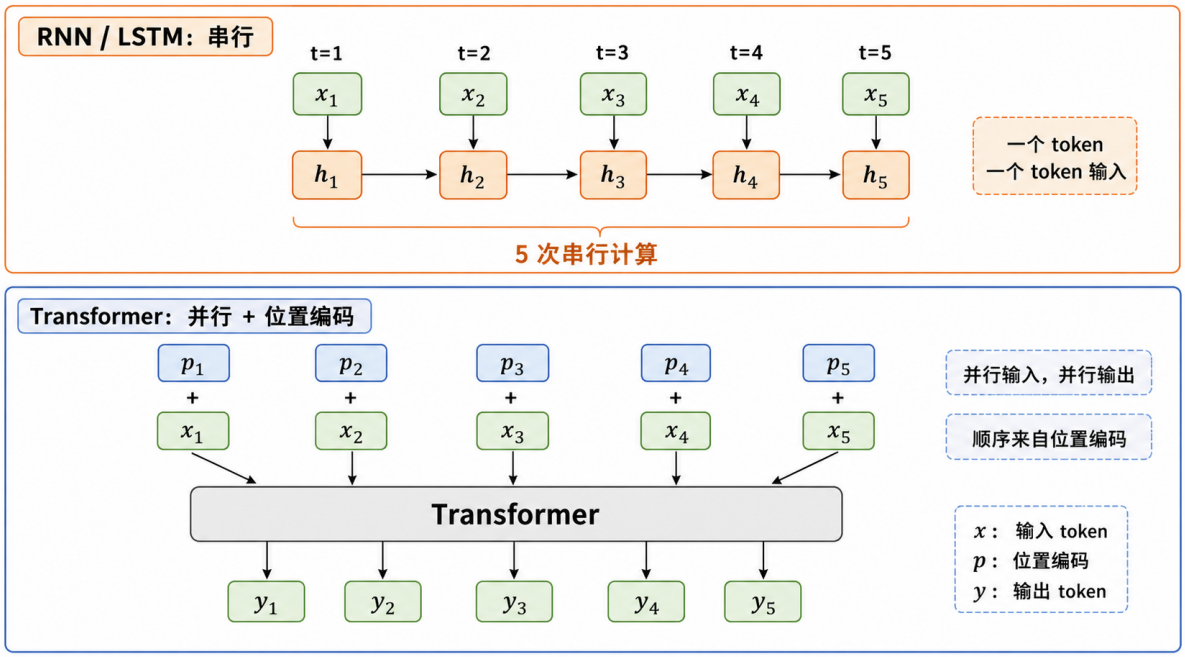
- 更好地捕捉全局语义关系（多头注意力机制）
- 更好地理解上下文（位置编码）
- 更快的计算效率！（结构使得高效的并行计算成为可能）



[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

Transformer： Transformer 是以注意力机制为核心的序列模型，能够直接建模 token 之间的关系，更容易捕捉全局上下文，也更适合并行计算。

Transformer 的直观思想是：当前 token 更新自己时，直接计算“应该关注哪些 token”，再按关注程度汇总信息。原始 Transformer 用于机器翻译，采用 encoder-decoder 结构；现在常见生成式大语言模型多使用 decoder-only 结构。



图中两点最重要：RNN/LSTM 按时间步串行传递信息，5 个 token 通常要做 5 次串行计算；Transformer 可把整段 token 同时送入模型，并行得到输出，但 token 向量本身没有顺序概念，因此必须加入位置编码。x_i 是输入 token，p_i 是位置编码，y_i 是输出 token。

模型	处理序列的方式	主要特点
RNN/LSTM	按时间步顺序读入 token	有隐藏状态，但并行不方便，长程信息传递更困难
Transformer	直接计算 token 之间的关系	注意力机制更适合全局关系和并行计算

例子 1：注意力的直观类比。如果要估计某位同学缺失的化学成绩，可以先找“和他各科表现相似”的同学，再参考这些同学的化学成绩。注意力机制做的事情也类似：先计算相关性，再按相关性汇总信息。

5.2 Next Token Prediction 回顾

页码：p10-p13

Next token prediction 已经在“循环神经网络 RNN 与 LSTM”章的 4.4 节讲过：给定前文 token，模型预测下一个 token，并常用交叉熵损失。Transformer 这一章不重复展开，后面只保留这个联系：decoder 做生成任务时必须保证当前位置不能看到未来 token，因此需要 causal mask。

Next Token Prediction 回顾：生成任务中，每个位置预测下一个 token；decoder 训练和生成时不能提前看到未来 token。

5.3 Embedding 与位置编码

页码：p15-p18

Embedding 已经在“循环神经网络 RNN 与 LSTM”章的 4.3 节讲过：token 先变成 one-hot，再通过可训练 embedding 矩阵映射成低维稠密向量。Embedding 矩阵通常随机初始化，并在训练过程中持续更新。本节讨论位置编码。

Transformer 会把一组 token 同时送入模型。Embedding 只表示“这个 token 是什么”，不表示“它在第几个位置”；位置编码用于补上顺序信息。

嵌入 Embedding + 位置编码

$$X^{\text{in}} \in \mathbb{R}^{n \times d}$$

$$X^{\text{em}} = X^{\text{in}} W^{\text{em}} \in \mathbb{R}^{n \times d_m}$$

$$W^{\text{em}} \in \mathbb{R}^{d \times d_m}$$ 随机初始化，可训练

$$X^{\text{pos}} \in \mathbb{R}^{n \times d_m}$$

$$X^{(1)} = X^{\text{em}} + X^{\text{pos}}$$

X^{pos}: 可以是可学的参数，也可以是针对绝对位置或者相对位置的不可学习量。

位置编码：位置编码为模型提供 token 的位置信息；输入表示通常由 token embedding 和位置编码相加得到。位置编码不改变词表大小。

输入表示：

$$X^{(1)} = X^{emb} + X^{pos}$$

其中，$X^{emb} \in R^{n \times d_m}$ 是 token embedding，$X^{pos} \in R^{n \times d_m}$ 是位置编码，因此 $X^{(1)} \in R^{n \times d_m}$。$n$ 是序列长度，d_m 是每个 token 的向量维度。

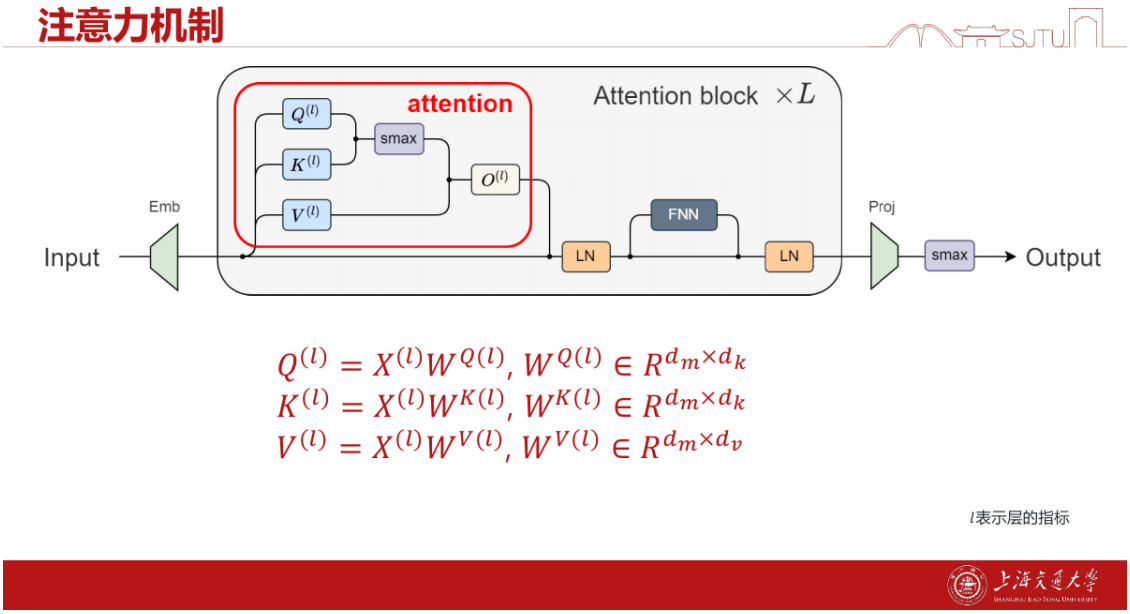
> **例子 3**：为什么需要位置编码。“我 爱 小 猫”和“猫 爱 小 我”包含相同 token，但顺序不同，含义也不同。位置编码就是告诉 Transformer：这些 token 分别出现在第几个位置。

复杂位置编码属于阅读材料，不展开推导；记住输入表示通常是 embedding 加位置编码。

5.4 注意力机制与 Q、K、V

页码：p21-p31

注意力主要计算 token 之间的相关性，本身不负责记录“谁在第几个位置”，所以输入 Transformer 前要先把位置编码加进去。



> **Q、K、V**：Q 是 Query，表示当前 token 发出的查询；K 是 Key，表示每个 token 可被匹配的键；V 是 Value，表示真正被加权汇总的信息内容。

总公式：缩放点积注意力

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

四步理解：

1. 线性映射得到 Q, K, V。在 self-attention 中，输入是 $X^{(l)} \in R^{n \times d_m}$，其中 n 是 token 数量，d_m 是每个 token 的隐藏向量维度。Q, K, V 不是新的输入数据，而是由同一个 $X^{(l)}$ 通过三个可训练线性映射得到：

$$\begin{aligned} Q^{(l)} &= X^{(l)}W^{Q(l)}, \quad W^{Q(l)} \in R^{d_m \times d_k} \\ K^{(l)} &= X^{(l)}W^{K(l)}, \quad W^{K(l)} \in R^{d_m \times d_k} \\ V^{(l)} &= X^{(l)}W^{V(l)}, \quad W^{V(l)} \in R^{d_m \times d_v} \end{aligned}$$

输出为 $Q^{(l)}, K^{(l)} \in R^{n\times d_k}$，$V^{(l)} \in R^{n\times d_v}$。这里 K 和 V 的第一维相同，表示每个 key 都对应一个 value；Q 和 K 的最后一维必须相同，才能做点积；V 的最后一维可以不同。课件中学习的是 self-attention，因此 Q, K, V 都来自同一段序列，第一维都是 n。

扩展说明：在更一般的 attention，尤其是 cross-attention 中，Query 的个数可以和 Key/Value 的个数不同，也就是 Q 的第一维可以不同；但 K 和 V 仍然要一一配对，所以二者第一维相同。

2. 用 Q 和 K 计算关联强度。输入是 $Q^{(l)}, K^{(l)} \in R^{n\times d_k}$：

$$A^{(l)} = \frac{Q^{(l)}(K^{(l)})^T}{\sqrt{d_k}}, \qquad A^{(l)} \in R^{n\times n}$$

A_{ij} 表示第 i 个 token 对第 j 个 token 的关注分数；除以 $\sqrt{d_k}$ 是为了让训练更稳定。

3. 对关联强度做 softmax 得到注意力权重。输入是 $A^{(l)} \in R^{n\times n}$：

$$\text{Attn}^{(l)} = \text{softmax}(\text{mask}(A^{(l)})), \qquad \text{Attn}^{(l)} \in R^{n\times n}$$

mask 是可选遮挡步骤，生成任务里的 causal mask 见下一节；如果不需要遮挡，可直接对 $A^{(l)}$ 做 softmax。

4. 用注意力权重对 V 加权求和，再用 W^O 调整维度。输入是 $\text{Attn}^{(l)} \in R^{n\times n}$，$V^{(l)} \in R^{n\times d_v}$：

$$X^{qkv(l)} = \text{Attn}^{(l)} V^{(l)}, \qquad X^{qkv(l)} \in R^{n\times d_v}$$

再通过输出矩阵 $W^{O(l)}$ 映射回模型维度：

$$X^{pr(l)} = X^{qkv(l)} W^{O(l)}, \qquad W^{O(l)} \in R^{d_v\times d_m}$$

最终输出 $X^{pr(l)} \in R^{n\times d_m}$。$W^O$ 的作用是把 $n \times d_v$ 变回 $n \times d_m$，这样才能和原输入 $X^{(l)}$ 做残差相加。

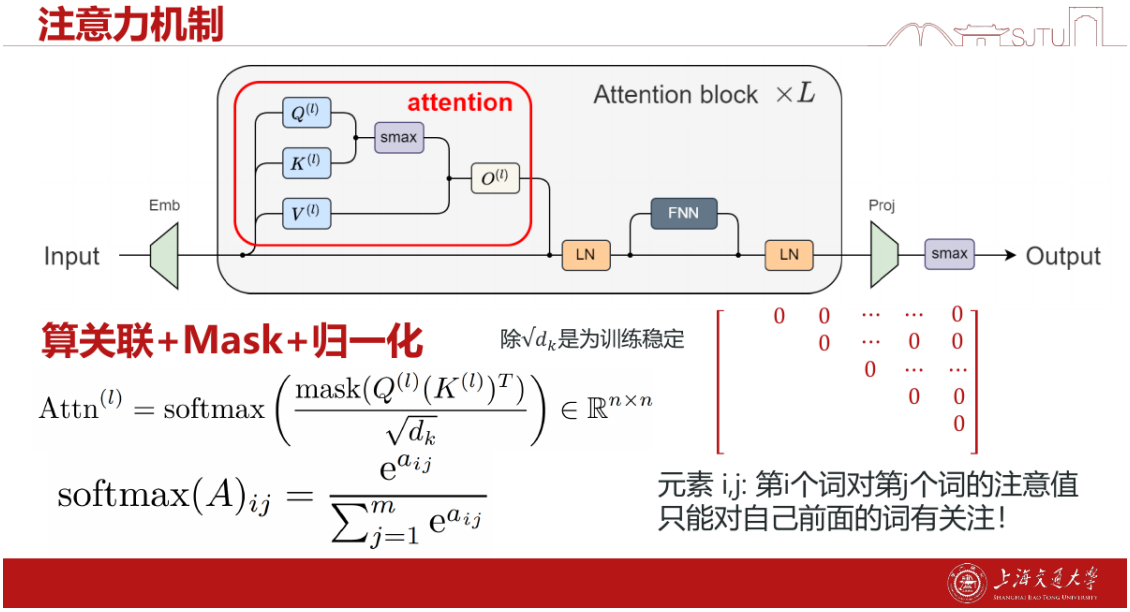
例子 **4**：**注意力加权求和**。如果某个 token 对三个位置的注意力权重是 $[0.5, 0.3, 0.2]$，三个 value 向量的第二个分量分别是 $2, 4, 1$，则输出向量的第二个分量为

$$0.5 \times 2 + 0.3 \times 4 + 0.2 \times 1 = 2.4$$

5.5 Causal Mask

页码：p27-p29

在生成任务中，预测当前位置时不能看到未来 token，否则模型就相当于提前看到了答案。Causal mask 用来遮住未来位置。



Causal Mask： Causal mask 会把未来位置的注意力分数设为很小的值，使 softmax 后这些位置的权重接近 0，从而保证预测时不能看未来 token。

对第 i 个 token 来说，causal mask 只允许它关注第 1 到第 i 个 token。这样就保证 next token prediction 的因果性。

例子 5：**Causal mask 允许看哪些位置**。对序列“我 爱 小 猫”，每个位置能关注的位置如下：

当前位置	允许关注	不能关注
我	我	爱、小、猫
爱	我、爱	小、猫
小	我、爱、小	猫
猫	我、爱、小、猫	无

对应的 causal mask 可以写成一个矩阵。行表示当前位置，列表示被关注的位置；0 表示允许看，$-\infty$ 表示遮住：

$$M = \begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

计算时可以理解为先把 mask 加到注意力分数上：

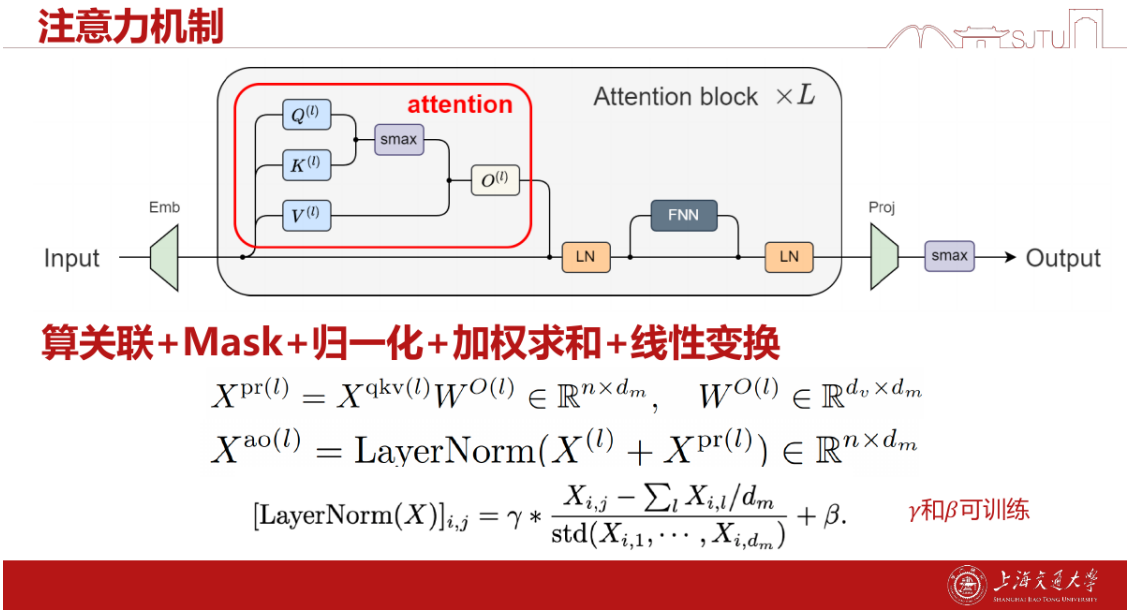
$$\mathrm{Attn} = \mathrm{softmax}(A + M)$$

因此，当模型在“我 爱”之后预测下一个 token 时，可以利用“我”和“爱”，但不能提前看到“小”或“猫”。

5.6 Transformer Block、LayerNorm 与 FNN

页码：p31-p37

Transformer block 可以理解为“先让 token 之间交流，再分别处理每个 token”。



Transformer Block：一个 Transformer block 通常包含注意力模块、残差连接、LayerNorm 和前馈全连接网络 FNN。

Block 中四个核心部件：

1. 注意力模块：负责 token 之间的信息交互。注意力输出经 W^O 后变回 $n \times d_m$，才能和输入做残差相加。
2. 残差连接：把子层输入直接加到子层输出上，基本形式是

$$y = x + F(x)$$

3. LayerNorm：逐个 token 做归一化。对 $n \times d_m$ 的矩阵，它对每一行 token 向量内部的 d_m 个数计算均值和方差，不在不同 token 之间混合。归一化后还有可训练仿射参数 γ, β。
4. FNN：逐 token 作用的小 MLP；不同位置共享同一套 FNN 参数。FNN 不负责 token 之间的信息交互，token 之间的 interaction 主要发生在 attention 里。

LayerNorm 公式示意：

$$X^{ao(l)} = \mathrm{LayerNorm}(X^{(l)} + X^{pr(l)})$$

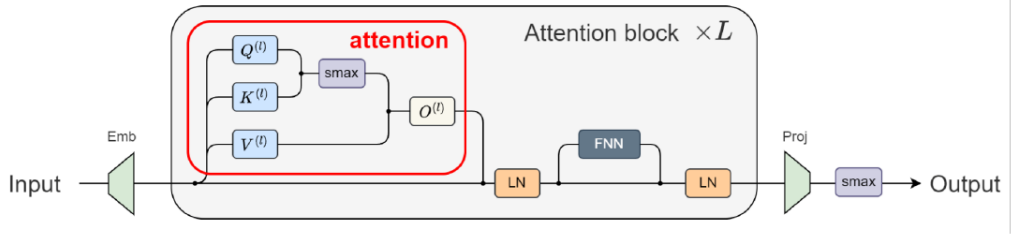
例子 6：**LayerNorm 和 FNN 都是逐 token**。如果输入有 4 个 token，每个 token 是 128 维向量，LayerNorm 会分别对 4 个 token 各自的 128 个数做归一化；FNN 也分别作用在 4 个 token 上。不同 token 之间的信息交换主要已经在注意力模块中完成。

5.7 输出投影与解码

页码：p38-p39

输出投影层把每个 token 的隐藏向量映射到词表大小，再经过 softmax 得到下一个 token 的概率分布。

输出（投影）层



$$X^{\text{out}} = X^{\text{do}(L)} W^{\text{proj}} + b^{\text{proj}} \in \mathbb{R}^{n \times d}, \quad W^{\text{proj}} \in \mathbb{R}^{d_m \times d}, \quad b^{\text{proj}} \in \mathbb{R}^{d}$$

贪婪解码 (greedy decoding) $\text{Output} = \text{argmax}(\text{softmax}(X^{\text{out}}))$

采样解码 (sampling decoding) $\text{Output} \sim \text{softmax}(X^{\text{out}})$

输出投影层：输出投影层把隐藏向量映射到词表维度，softmax 后得到每个候选 token 的概率。

输出公式：

$$X^{out} = X^{do(L)} W^{proj} + b^{proj}, \qquad W^{proj} \in R^{d_m \times d}$$

其中，d 是词表大小，所以输出的最后一维对应词表中每个 token 的分数。

语言模型训练通常用交叉熵损失衡量预测分布和真实下一个 token 之间的差距；真实 token 的预测概率越高，loss 越小。

解码方式	含义	结果特点
贪婪解码	每一步取概率最大的 token	确定性强；同一输入通常得到同一输出
采样解码	按概率分布随机抽样	有随机性；同一输入可能得到不同输出

例子 7：贪婪解码和采样解码的区别。假设下一个 token 的概率为：猫 0.70、狗 0.20、鼠 0.10。贪婪解码一定选择概率最大的“猫”；采样解码大多数时候会抽到“猫”，但也有可能抽到“狗”或“鼠”。所以采样解码更有随机性。

5.8 关键记忆

- Transformer 用注意力机制直接建模 token 之间的关系，比 RNN 更适合并行计算。p3-p9
- 原始 Transformer 是 encoder-decoder 结构；现在常见生成式大语言模型多用 decoder-only。p3-p9
- RNN/LSTM 按时间步串行处理 token；Transformer 可以并行处理 token，但需要位置编码提供顺序信息。p3-p18
- Embedding 矩阵随机初始化并可训练；位置编码提供 token 位置信息，不改变词表大小。p15-p18
- 注意力公式是 softmax$(QK^T/\sqrt{d_k})V$；Q 是查询，K 是匹配用的键，V 是被汇总的信息。p21-p31
- K 和 V 的数量必须相同；Q 和 K 的最后一维必须相同。课件重点是 self-attention：Q、K、V 来自同一段序列，所以 token 数相同，注意力权重形状通常是 $n \times n$；扩展到 cross-attention 时，Q 的数量可以和 K/V 不同。p21-p31
- Causal mask 保证预测当前位置时不能看到未来 token。p27-p29
- Block 中 attention 负责 token 交互；FNN 逐 token 作用；LayerNorm 逐 token 计算，并有可训练参数 γ, β。p31-p37
- 残差连接公式是 $y = x + F(x)$，用于保留原信息并缓解深层训练困难。p31-p37
- 输出投影层把隐藏向量映射到词表维度；贪婪解码取最大概率 token，采样解码具有随机性。p38-p39
- Transformer 整体流程：输入 token -> Embedding + 位置编码 -> 多层 block -> 输出投影 -> Softmax。p15-p39
- 多头注意力是了解项：多组注意力并行工作，不同头可以关注不同类型的信息。p21-p31

5.9 思考题

1. 判断：Transformer 使用注意力机制来建模 token 之间的关系。答案：正确。
2. 填空：Transformer 需要加入 _____，用来表示 token 在序列中的位置。答案：位置编码。
3. 选择：Q、K、V 中，用来真正汇总信息内容的是：A. Q；B. K；C. V；D. mask。答案：C。
4. 判断：Q 和 K 的最后一维必须相同，因为要计算 QK^T。答案：正确。
5. 判断：LayerNorm 是把不同 token 混在一起计算均值和方差。答案：错误。LayerNorm 是逐 token 在向量内部计算。
6. 填空：缩放点积注意力的核心公式可以写成 softmax$(QK^T/\sqrt{d_k})$____。答案：V。

6 神经网络训练现象：频率原则

这一章按 PDF 顺序复习：训练过程现象 -> 泛化谜团 -> 隐式偏好 -> 频率与傅里叶级数 -> 频率原则 -> 泛化与早停。本章围绕“神经网络通常先学低频、后学高频”这个现象，以及它为什么和泛化有关。后续多尺度网络、Fourier feature、NeRF、理论证明和激活函数频谱属于扩展/阅读材料，不纳入本提纲。

6.1 从训练现象出发

页码：p5-p8

神经网络训练过程并不是“所有部分同时学好”。对一个既有平坦区域、又有振荡区域的函数，训练时通常更容易先学好平坦、缓慢变化的部分，振荡和细节往往更晚才学好。

哪一部分先学习好

左边是振荡，右边是平坦

低频部分：变化慢、比较平滑，通常对应整体轮廓或主要趋势。

高频部分：变化快、振荡明显，通常对应细节、边缘、纹理或噪声。

例子 1：先学轮廓，再学细节。画一把梳子时，通常会先画外轮廓，再画密集的梳齿；神经网络拟合函数时也常有类似现象：平滑轮廓更容易先被学到，密集振荡更难、更晚被学到。

6.2 泛化谜团与隐式偏好

页码：p9-p11

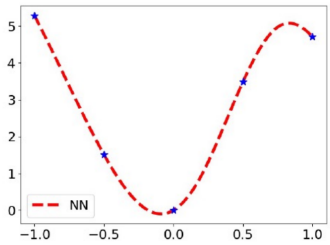
过参数化神经网络参数很多，理论上可以记住训练样本；但实际训练常能得到泛化较好的解，这就是课件中的泛化谜团。

过参数化：模型参数量相对样本数很多，模型有能力表达大量不同的函数。

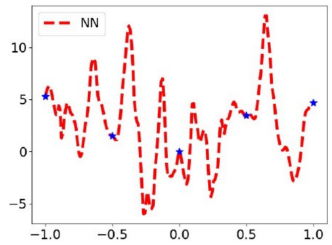
隐式偏好：当许多解都能让训练误差很低甚至为 0 时，训练过程本身会倾向于得到某些类型的解；这种偏向没有显式写在损失函数里。

选择哪种解?

蓝色训练点是完全一致的。



常用设定学到的解，
满足频率原则



参数初始值很大时学到的解，
实验观察其不满足频率原则

图中要点：同一组训练点可能对应多种拟合曲线；频率原则解释了为什么普通训练常更倾向平滑、低频的解。

6.3 频率、低频与高频

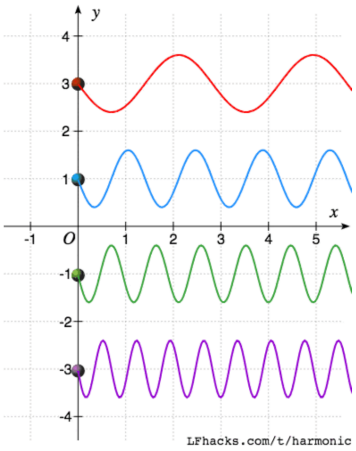
页码：p12-p16

频率用来描述函数变化的快慢。低频表示变化慢、整体趋势明显；高频表示变化快、细节多。复习时不需要推导傅里叶级数，只需要知道：复杂函数可以从“不同频率成分叠加”的角度来理解。

傅里叶级数

低频

高频



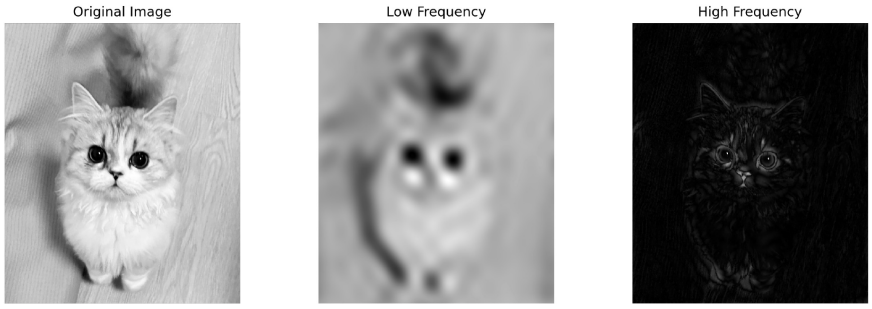
频率：频率描述函数或信号变化的快慢；变化越慢，频率越低；变化越快，频率越高。

傅里叶级数：

$$f(x) \approx a_0 + a_1 \sin x + b_1 \cos x + a_2 \sin 2x + b_2 \cos 2x + \cdots$$

可以把它理解为：复杂函数可近似看成不同频率的正弦、余弦函数叠加。a_0 表示整体平均水平，后面的项表示不同频率成分及其强度。

保留低频或高频的图像变化



在图像中，低频通常保留整体形状和大轮廓，高频通常保留边缘、纹理和细碎变化。高频并不是一定没用，但噪声也常表现为高频。

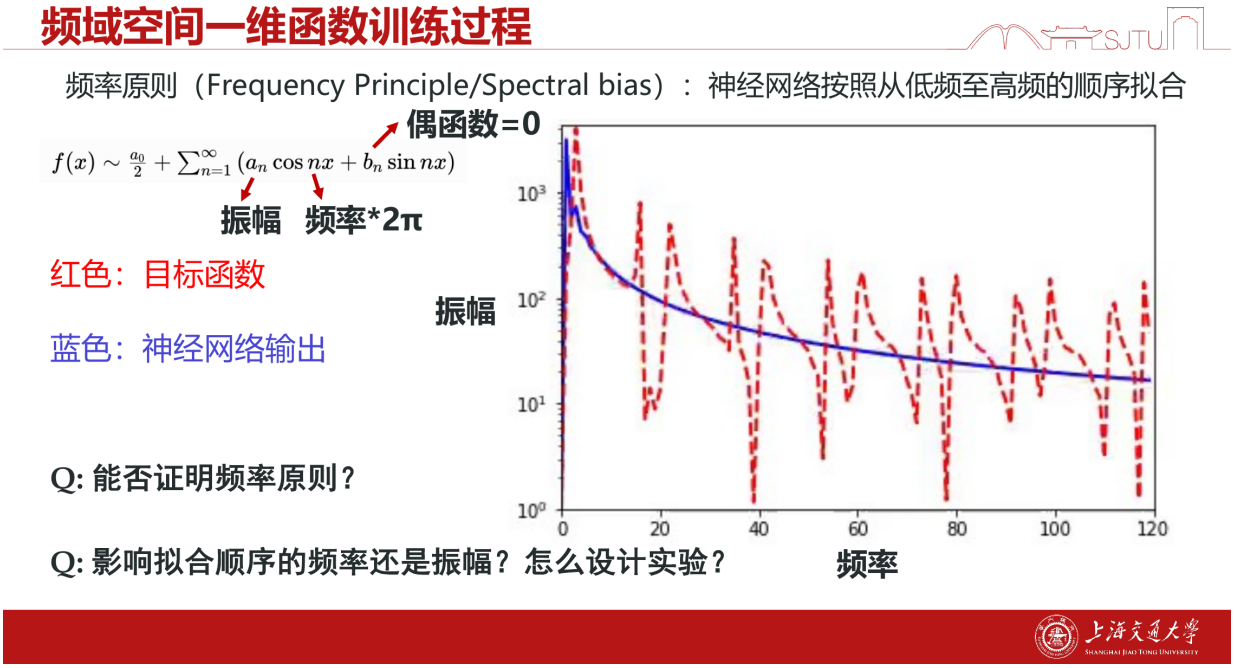
例子 3：函数中的低频和高频。 $f_1(x) = \sin x$ 变化较慢，可以看成低频；$f_2(x) = \sin 8x$ 在同样区间内振荡更多，可以看成更高频。
例子 4：图像中的低频和高频。一张人脸图像的模糊轮廓更偏低频；头发纹理、边缘和噪声点更偏高频。

6.4 频率原则

页码：p17-p22

频率原则描述的是：神经网络训练过程中，通常先拟合低频成分，再逐渐拟合高频成分。

频率原则：神经网络在训练过程中通常按照从低频到高频的顺序拟合目标函数。



注：这一页在 PPT 中是动画，建议结合原 PPT 动画理解“训练早期先学低频、后期再学高频”的动态过程。

时域：训练早期先出现整体轮廓，后期再补细节。

频域：低频误差通常先下降，高频误差通常后下降。

控制变量：若不同频率的振幅相同，仍然低频先收敛，说明拟合顺序主要由频率决定，而不只是由振幅大小决定。

例子 5：一维函数训练。对目标函数

$$f(x) = \sin x + \sin 5x$$

$\sin x$ 是低频成分，$\sin 5x$ 是更高频成分。训练时通常会先学到 $\sin x$ 对应的平滑轮廓，再逐渐学到 $\sin 5x$ 对应的振荡细节。

6.5 频率原则与泛化

页码：p23-p26

频率原则可以帮助理解泛化：很多函数都能穿过训练点，但普通训练更倾向于先得到低频、平滑的解。这样的解通常不容易把训练样本中的偶然噪声全部记住，因此更可能在测试数据上表现合理。

泛化：模型不仅在训练样本上表现好，也能在没有见过的新样本上表现好。

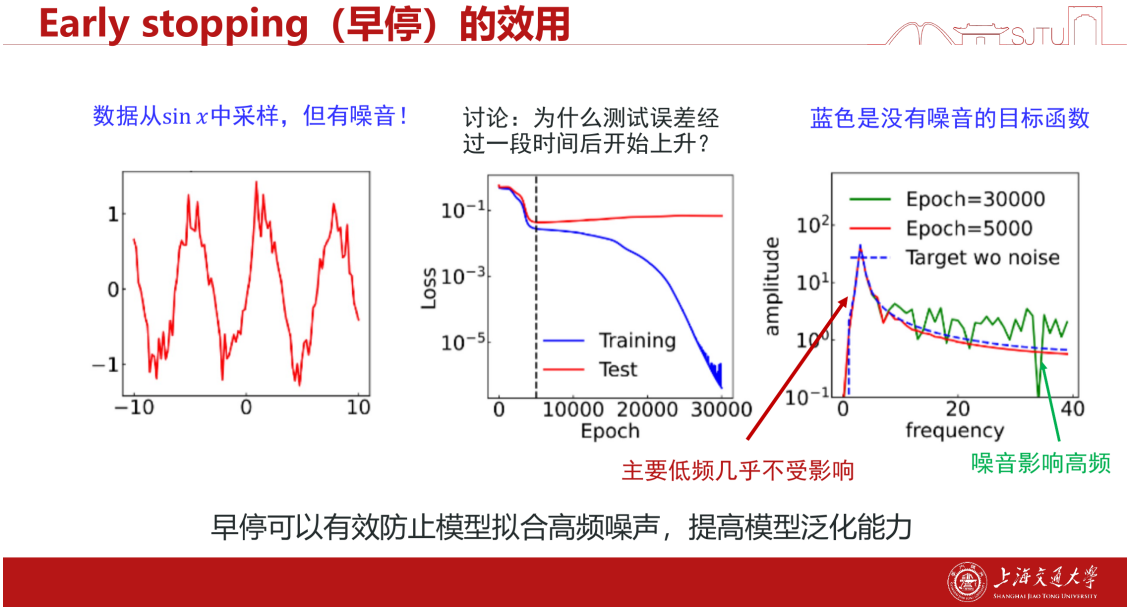
边界条件：频率原则描述的是常见训练倾向，不是说高频永远没有用，也不是说神经网络永远不会学到高频。训练足够久、模型足够强时，高频细节甚至高频噪声也可能被学到。

6.6 Early Stopping 早停

页码：p27-p28

数据中可能含有噪声，而噪声常常偏高频。由于神经网络通常先学低频信号、后学高频细节，训练太久可能开始拟合高频噪声，导致测试误差上升。早停就是在模型进一步拟合噪声前停止训练。

Early Stopping：早停是在验证误差或测试误差开始变差前后停止训练，用来减少过拟合、提升泛化能力。



例子 6：带噪声的正弦函数。目标整体趋势接近 $\sin x$，但样本中混入了抖动噪声。训练早期模型先学到 $\sin x$ 的整体趋势；如果继续训练太久，模型可能开始拟合噪声。早停可以让模型保留主要趋势，同时少学一些高频噪声。

6.7 两种频率

页码：p37-p40

说“高频”时容易混淆两件事：一种是图像本身的像素频率，另一种是模型输出函数的响应频率。图像分类可以作为理解这一区别的例子。

图像频率：图像频率描述相邻像素强度变化的快慢；边缘、纹理和细节通常偏高频，平缓背景和整体轮廓通常偏低频。

响应频率：响应频率描述输入发生微小变化时，模型输出变化的敏感程度；响应频率高表示很小的输入扰动也可能引起输出明显变化。

在图像分类这个例子中，频率原则讨论的对象通常是分类函数的响应频率，而不是图像像素内部的频率。对抗噪声就是一个典型例子：图片只加入肉眼几乎看不见的小扰动，模型预测类别却发生改变，说明扰动方向上分类函数响应很敏感，具有明显高频成分。

例子 7：不要混淆两种频率。一张猫图像的毛发纹理属于图像频率里的高频；如果给图片加很小的扰动，模型输出从“猫”变成“狗”，这反映的是分类函数对该扰动方向的响应频率高。

6.8 关键记忆

- 低频表示变化慢、平滑、整体轮廓；高频表示变化快、振荡、细节或噪声。p5-p16
- 过参数化模型有能力表达很多复杂函数，但神经网络训练常能得到泛化较好的解。p9-p11
- 隐式偏好指训练过程会偏向某些解，这种偏向没有显式写在损失函数里。p10-p11
- 频率原则指神经网络通常先拟合低频，再拟合高频。p17-p22
- 频率原则可以解释为什么神经网络常先学整体规律，而不是一开始就记住高频噪声。p23-p28
- Early stopping 可以在模型进一步拟合高频噪声前停止训练，从而改善泛化。p27-p28
- 高频不是一定没用；边缘、纹理、细节可能是高频，噪声也可能是高频。p12-p16
- 两种频率要区分：图像频率描述像素变化快慢，响应频率描述模型输出对输入变化的敏感程度。p37-p40

6.9 思考题

1. 判断：频率原则指神经网络训练时通常先拟合低频成分，再拟合高频成分。答案：正确。
2. 填空：频率原则指神经网络训练时通常先拟合 _____ 频成分，再拟合 _____ 频成分。答案：低；高。
3. 选择：图像中的边缘、纹理和噪声通常更接近：A. 低频；B. 高频；C. 标签；D. 偏置。答案：B。
4. 判断：过参数化神经网络参数很多，所以一定会严重过拟合，不可能泛化好。答案：错误。
5. 判断：早停可以理解为在模型进一步拟合高频噪声前停止训练。答案：正确。
6. 判断：讨论频率时，一定只指图像像素内部的频率，与模型输出无关。答案：错误。还需要区分模型输出函数的响应频率。

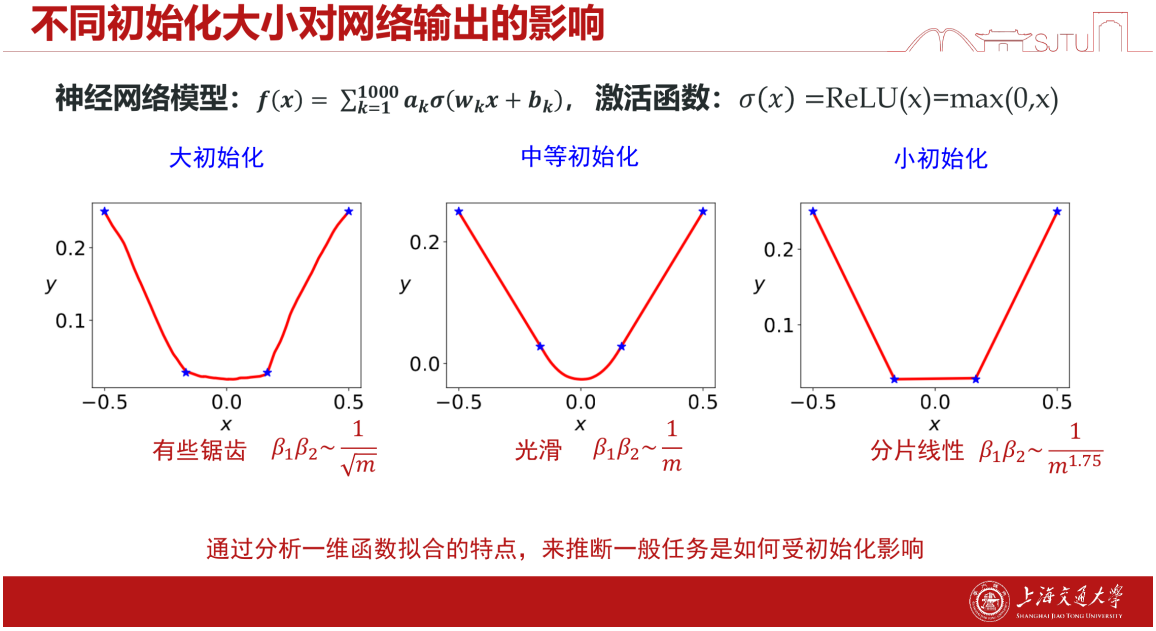
7 参数凝聚与解的平坦性

这一章按 PDF 顺序复习：初始化大小 -> ReLU 转折点 -> 参数凝聚与等效小网络 -> 平坦解与训练技巧。本章围绕“小初始化下网络可能出现凝聚”，以及“平坦解为什么通常更稳健”。

7.1 初始化大小如何影响训练

页码：p4-p8

初始化会影响初始参数位置、早期输出形态和后续优化路径。



初始化：初始化是在训练开始前给网络参数赋初值；初始化大小会影响训练速度、训练路径和最终解的形态。

初始化大小：

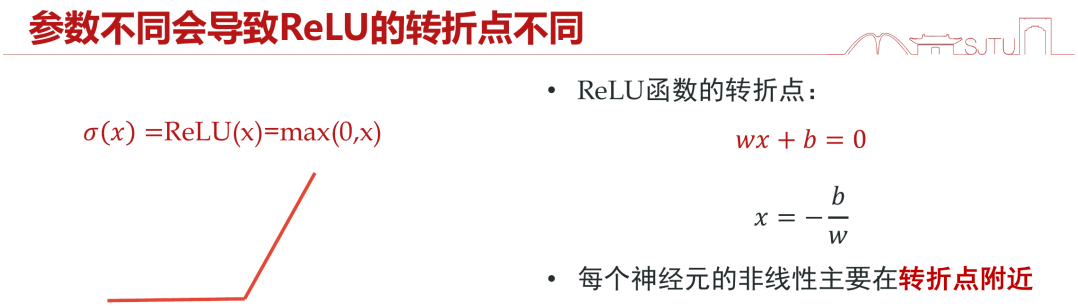
- 大初始化：初始输出更随机、更复杂，容易保留较多随机振荡。
- 中等初始化：通常是训练稳定性和表达能力之间的折中。
- 小初始化：初始输出更平滑，神经元更容易出现趋同或凝聚。
- 常见初始化方法包括 LeCun、Xavier、Kaiming，本质上都是在控制不同网络结构下的初始化尺度。
- 所有参数不能都初始化为 0，否则同层神经元完全相同，梯度更新也相同，无法有效打破对称性。

例子 **1**：**为什么不能全 0 初始化**。如果同一层两个神经元的权重和偏置（bias）都初始化成完全相同的值，它们接收相同输入、产生相同输出、得到相同梯度，训练后仍然很像，相当于没有真正训练出两个不同特征。

7.2 ReLU 神经元与转折点

页码：p9-p12

ReLU 神经元的非线性主要体现在“从关闭到打开”的转折位置。理解转折点，有助于理解后面的神经元凝聚。



ReLU 转折点：对一维输入，$\mathrm{ReLU}(wx+b)$ 的转折点满足 $wx+b=0$，即 $x=-\frac{b}{w}$。

公式需要会用：

$$z = wx + b, \qquad \mathrm{ReLU}(z) = \max(0, z)$$

$$wx + b = 0 \quad \Rightarrow \quad x = -\frac{b}{w}$$

两个 ReLU 神经元如果转折点相同、方向相同，就可能合并成一个等效神经元；如果转折点不同，一般不能简单合并。原因是：相同转折点产生同一种折线形状，只是斜率系数不同；多个这样的神经元相加后，仍然是在同一位置转折。

例如 $\mathrm{ReLU}(2x-4)$ 和 $\mathrm{ReLU}(x-2)$ 的参数不同，但转折点都在 $x=2$，且打开方向相同。因为

$$\mathrm{ReLU}(2x-4) = \mathrm{ReLU}(2(x-2)) = 2\,\mathrm{ReLU}(x-2)$$

所以

$$a_1\,\mathrm{ReLU}(2x-4)+a_2\,\mathrm{ReLU}(x-2)=(2a_1+a_2)\,\mathrm{ReLU}(x-2)$$

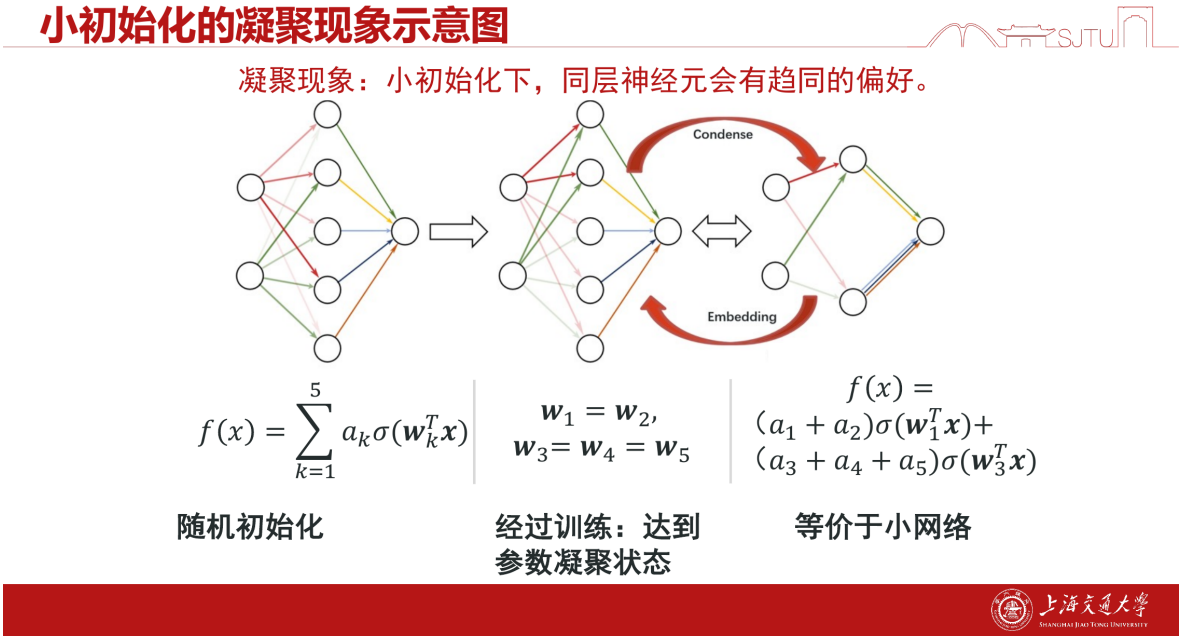
左边看起来是两个神经元，右边等价于一个神经元。相反，如果两个神经元的转折点不同，函数会在两个不同位置发生弯折，一般不能合并成一个转折点。

例子 2：转折点相同才能合并。 $\mathrm{ReLU}(2x-4)$ 和 $\mathrm{ReLU}(x-2)$ 的转折点都是 $x=2$，因此可以合并成一个等效神经元；但 $\mathrm{ReLU}(2x-4)$ 和 $\mathrm{ReLU}(x-3)$ 的转折点分别是 2 和 3，一般不能合并成一个转折点。

7.3 参数凝聚

页码：p13-p24

小初始化下，同层神经元可能逐渐变得相似，最后表现得像少数几个“有效神经元”。



参数凝聚：小初始化下，同层神经元会出现趋同偏好，多个神经元可能等效为较少的有效神经元。

参数凝聚的含义：

- 凝聚不是“网络坏了”，而是许多神经元在参数空间中靠近，表现出相似作用。
- 凝聚后，大网络在函数表达上可能等效于一个更小的网络。
- 有效参数量可以理解为等效小网络真正使用到的参数量。
- 训练过程中，模型可能先用较少有效神经元拟合主要结构，再逐渐增加有效神经元学习更复杂的部分。
- 这与频率原则可以联系起来：有效神经元变多后，网络更有能力学习更细、更高频的变化。

为什么不直接训练小网络：

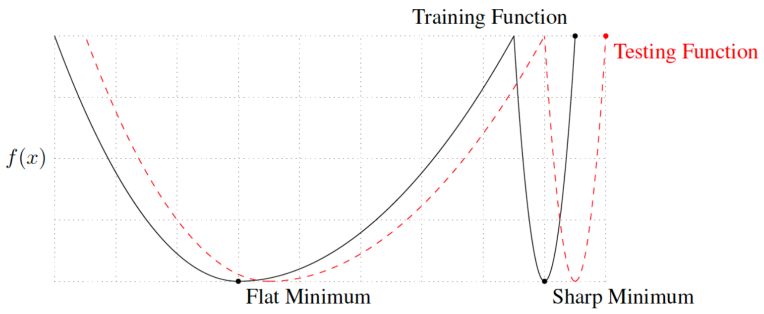
- 从表达结果看，凝聚后的大网络和等效小网络可能表达同一个函数。
- 从训练过程看，大网络参数更多、下降方向更多，通常更容易优化。
- 小网络虽然参数少，但可选择的下降方向少，训练时可能更难找到合适解。
- 这和知识蒸馏有类似思想：先训练能力更强、更容易优化的大模型，再把能力压缩或迁移到小模型；直接训练小模型未必同样容易。
- 小初始化有助于出现凝聚，但初始化过小也可能让训练变慢；默认初始化通常是实用折中。

7.4 平坦解与尖锐解

页码：p25-p30

神经网络训练不只看某一点训练误差低，还要看附近损失是否也低。附近损失也低，说明解更稳定。

平坦解与尖锐解



讨论：选择哪个解比较好？为什么？

训练集和测试集通常都会有噪音，平坦的解更鲁棒，泛化好。

Y轴为损失函数

X轴为参数

Keskar, Nitish Shirish, et al. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima." International Conference on Learning Representations. 2016.

平坦解：参数在一个小范围内扰动时，损失仍然保持较低的解。

尖锐解：参数稍微扰动，损失就明显上升的解。

平坦解与尖锐解可以这样理解：

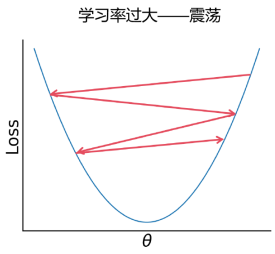
- 训练集和测试集通常都有噪声，平坦解对扰动更鲁棒。
- 平坦解通常泛化更好，但不表示训练误差一定更低。
- 尖锐解对参数变化更敏感，测试表现可能更不稳定。
- 小批量训练和合理较大的学习率都可能帮助模型避开尖锐解、靠近较平坦的解；大批量训练或过小学习率更可能停在较尖锐的区域。

7.5 提升平坦性的常见技巧

页码：p31-p33, p37

训练设置会影响模型更容易落在平坦解还是尖锐解附近。

稳定性



- 考虑一个二次函数：
- $\mathrm{L} = \frac{1}{2}\lambda\theta^2$
- 其梯度下降可以表示为：

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

- 如果我们要保证这个迭代过程不会**爆炸**：

$$|\theta_{t+1}| < |\theta_t|$$
$$|1-\eta\lambda| < 1$$
$$-1 < 1-\eta\lambda < 1$$
$$\eta\lambda < 2$$
$$\eta < \frac{2}{\lambda}$$

边缘稳定性：对二次函数附近的更新，曲率越大，稳定训练允许的学习率越小；曲率越大，说明解越尖锐；曲率越小，说明解越平坦。

有助于平坦性的常见做法：小批量训练、合理较大的学习率和 Dropout 都可能帮助模型找到更平坦、更鲁棒的解。

直观解释：

- 使用小批量训练：小批量梯度带有随机噪声。尖锐解附近“谷底”窄，轻微扰动就可能让参数跳出；平坦解附近低损失区域宽，加入噪声后仍更容易留在低损失区域。
- 使用合理较大的学习率：学习率越大，每一步更新越远。尖锐区域曲率大，大学习率容易来回震荡，难以稳定停住；平坦区域曲率小，更容易在较大学习率下保持稳定。
- 使用 Dropout：训练时随机屏蔽部分神经元，使模型不依赖单一路径，可能得到更鲁棒的解。

对图中的一维二次函数：

$$L(\theta) = \frac{1}{2}\lambda\theta^2$$

梯度下降更新为：

$$\theta_{t+1} = \theta_t - \eta\lambda\theta_t = (1-\eta\lambda)\theta_t$$

若希望参数不要来回发散，需要大致满足：

$$|1-\eta\lambda| < 1 \quad \Rightarrow \quad 0 < \eta < \frac{2}{\lambda}$$

因此，λ 越大，也就是曲率越大、解越尖锐，允许的学习率上限越小；λ 越小，也就是曲率越小、解越平坦，较大学习率也更可能稳定。

7.6 关键记忆

- 初始化大小会影响训练路径和最终解；参数不能全部初始化为 0。p4-p8
- 对一维输入，ReLU$(wx+b)$ 的转折点是 $x=-\frac{b}{w}$。p9-p12
- 参数凝聚指小初始化下，同层神经元出现趋同偏好，多个神经元可能等效为较少有效神经元。p13-p23
- 凝聚后的大网络可能像小网络，但大网络参数更多、下降方向更多，通常更容易训练；这与知识蒸馏“先大后小”的思想类似。p24
- 平坦解指参数附近一片区域损失都较低；尖锐解指参数稍微改变损失就明显上升。p25-p30
- 平坦解通常对噪声和参数扰动更鲁棒，泛化往往更好。p25-p30
- 小批量训练、合理较大的学习率、Dropout 都可能帮助找到更平坦的解。p31-p37

7.7 思考题

1. 计算：ReLU$(2x-4)$ 的转折点是 _____。

   答案：$x=2$。

2. 判断：所有参数初始化为 0 通常不是好做法，因为同层神经元会保持相同，难以打破对称性。

   答案：正确。

3. 填空：小初始化下，同层神经元可能出现趋同偏好，这种现象叫 _____。

   答案：参数凝聚。

4. 判断：凝聚后的大网络和等效小网络可能表达同一个函数，但大网络通常更容易训练。

   答案：正确。

5. 选择：平坦解通常指：A. 参数附近一片区域损失都较低；B. 参数必须全为 0；C. 训练误差一定为 1；D. 没有梯度。

   答案：A。

6. 判断：学习率越大越好，越大一定容易得到平坦解。

   答案：错误。合理较大的学习率可能有帮助，但过大会导致震荡甚至发散。

8 大模型介绍

这一章按 PDF 顺序复习：大模型定义 -> 三种架构 -> 训练流程中的关键技术 -> 大模型现象 -> 应用与推理。主要做概念辨析，不展开最新大模型技术细节。

8.1 什么是大模型

页码：p4-p7

大模型通常又叫基础模型。它的核心特点不是“某一个任务做得好”，而是经过大规模数据训练后，可以迁移或适配到多种下游任务。

基础模型：基础模型是参数量很大、经过大规模数据训练、可适应多种下游任务的模型；大语言模型是以文本 token 为主要输入输出的基础模型。

大模型可以这样理解：

- 大模型通常参数量大、训练数据规模大、训练成本高。
- 大模型可以通过提示词、微调、后训练等方式适应不同任务。
- 多模态数据也可以类似语言一样转成 token，例如图像 patch、音频片段等。
- “大模型”不等于“只能做语言任务”，但课程中默认讨论大语言模型。

8.2 三种常见架构

页码：p12-p18

大语言模型常见架构可以分成 Encoder-only、Decoder-only 和 Encoder-Decoder。三者主要比较上下文范围、是否使用 causal mask、适合什么任务。

Encoder-only：Encoder-only 可以看到前后文所有 token，通常更适合理解任务，例如分类、匹配、文本分析；代表模型是 BERT。

Decoder-only：Decoder-only 使用 causal mask，只能看到当前位置及之前的 token，通常更适合生成任务；代表模型包括 GPT、Llama、DeepSeek。

Encoder-Decoder：Encoder-Decoder 先编码输入，再逐步解码输出，适合输入和输出都需要建模的任务，例如机器翻译；代表模型包括原始 Transformer、T5、BART。

架构	能看到的信息	常见任务	代表模型
Encoder-only	双向上下文	理解任务	BERT
Decoder-only	当前 token 及之前	生成任务	GPT、Llama、DeepSeek
Encoder-Decoder	编码输入，解码输出	翻译、摘要等	Transformer、T5、BART

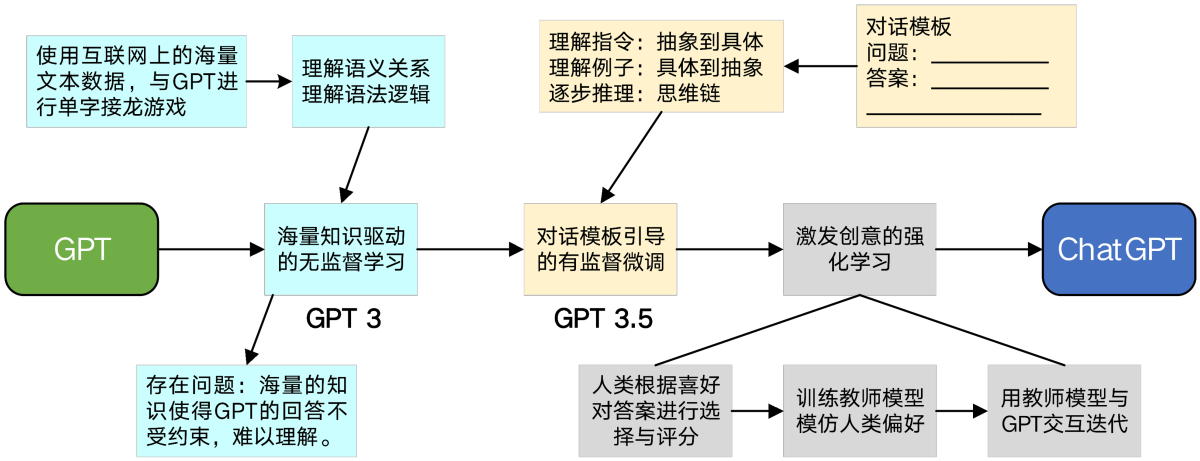
对生成任务而言：Encoder-only 因为能看到后文，通常不适合直接做“只看前文预测下一个 token”的 Next Token Prediction；当前主流生成式大语言模型多采用 Decoder-only。

例子 2：为什么生成式模型常用 **Decoder-only**。写作文时，模型要从左到右生成下一个 token，生成当前位置时不能提前看到未来答案，因此需要 causal mask 限制可见范围。

8.3 训练流程中的关键技术

页码：p20-p29

大模型通常先预训练，再通过少量关键技术适配任务或提升输出质量。这里抓住“是否更新参数”和“主要作用”即可。



这张图按三步理解：预训练学习语言规律和知识；监督微调让模型学会按指令回答；强化学习后训练用人类偏好构造奖励信号，使输出更符合人类偏好。

大模型训练流程：预训练学习基础能力；监督微调让模型按指令回答；强化学习后训练用奖励信号增强推理或对齐偏好。

适配与压缩：提示词工程通过输入引导输出，通常不更新参数；知识蒸馏让 Student model 学习 Teacher model 的能力，常用于降低部署成本。

技术对比：

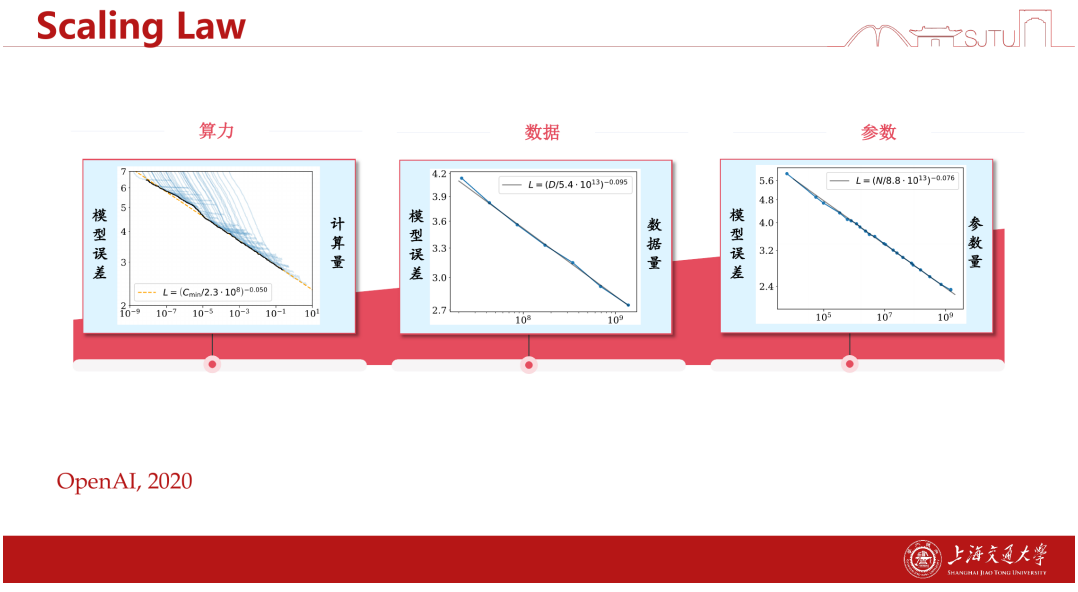
技术	是否通常更新模型参数	作用
预训练	更新	学基础能力、语言规律和知识
微调 / SFT	更新	适配特定任务或风格
强化学习后训练	更新	增强推理或对齐偏好
提示词工程	通常不更新	通过输入引导输出
知识蒸馏	训练 Student	把 Teacher 能力迁移到小模型

例子 3：提示词工程和微调的区别。给模型输入“请一步一步思考”属于提示词工程，模型参数不变；用大量数学题数据继续训练模型属于微调或后训练，模型参数会更新。

8.4 大模型中的现象

页码：p35-p45

大模型中常见现象：规模、数据、算力增加时性能通常改善；模型还可能出现上下文学习、思维链、幻觉和偏见。



Scaling law：Scaling law 描述在一定范围内，模型规模、数据量、计算量增加时，测试损失通常按照幂律关系下降。

课件中三类常见 scaling law 都是幂级数关系，可以写成：

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$$

$$L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}$$

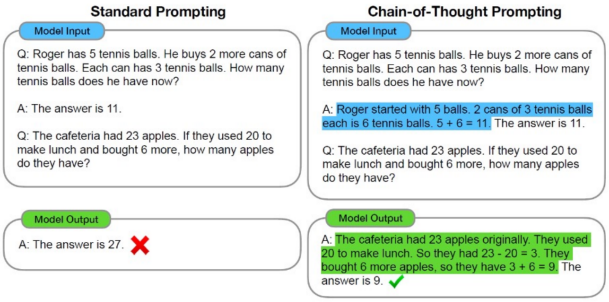
$$L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

其中，N 表示模型参数规模，D 表示数据量，C 表示计算量；$\alpha_N, \alpha_D, \alpha_C$ 是正的指数。这里的关系是幂级数下降，不是线性下降，也不是普通指数下降。

涌现与上下文学习：涌现能力指模型规模增大到一定程度后，某些能力从接近随机跃升到明显更好；上下文学习指模型不更新参数，仅依靠任务描述和少量示例适应当前任务。

思维链 Chain of Thought

- 思维链是指通过引导模型生成中间推理步骤，而不是直接给出问题答案的方法，这种方法能够显著提高模型在复杂任务中的表现。



Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. "Chain-of-thought prompting elicits reasoning in large language models." *Advances in neural information processing systems* 35 (2022): 24824-24837.

思维链：思维链是引导模型输出中间推理步骤，常用于提升复杂推理任务表现。

幻觉与偏见：幻觉是模型生成看似合理但实际错误的信息；偏见是输出中带有训练数据或模型机制造成的系统性倾向。

例子 4：上下文学习。输入“把中文翻译成英文：苹果 -> apple；猫 -> cat；学校 -> ?”，模型可以根据上下文示例回答 school，而不需要更新参数。

8.5 应用与推理

页码：p47-p96

大模型可以用于问答、摘要、翻译、代码生成、数学推理、规划等任务。推理能力常与提示方式、训练方式、模型规模和初始化有关。

大模型应用与推理：大模型能处理多类生成和推理任务，但输出可能出错；提示方式、训练方式、模型规模和初始化都会影响表现。

初始化与推理：初始化会影响模型训练路径和最终能力；课件例子中，小初始化通常更有利于模型学习可泛化的函数关系，因此推理能力更好。

在应用中：

- 大模型可以解决应用题、举一反三、生成代码，但输出不一定永远正确。
- 强化学习后训练可以增强模型推理能力，也可以用于对齐人类偏好。
- 初始化太大时模型更容易偏向记忆训练样本；小初始化更容易形成有结构的规律学习，但初始化过小也可能带来训练变慢的问题。
- RAG、MCP、Agent、Harness 等属于扩展技术，了解名称即可，不作为复杂考点。

例子 5：不要把“会生成答案”误认为“一定正确”。模型可以写出很流畅的医学建议或历史解释，但其中可能混有错误事实，这属于幻觉风险，需要外部验证。

8.6 关键记忆

- 大模型又称基础模型，通常参数量大、数据规模大，并能适应多种下游任务。p4-p7
- 大语言模型以 token 为输入输出；多模态数据也可以类似语言一样 token 化。p4-p7
- Encoder-only 看双向上下文，适合理解任务，代表 BERT。p12-p18
- Decoder-only 使用 causal mask，只看当前及之前 token，适合生成任务，代表 GPT、Llama、DeepSeek。p12-p18
- Encoder-Decoder 先编码输入再解码输出，适合翻译等输入输出都需要建模的任务。p12-p18
- 预训练学基础能力；微调/SFT 用特定数据继续训练；提示词工程通常不更新参数。p20-p29
- 强化学习后训练可用于增强推理能力，也可用于对齐人类偏好。p24-p26
- 知识蒸馏让 Student model 学习 Teacher model 的能力，常用于降低部署成本。p28-p29
- Scaling law 描述规模、数据、算力增加时测试损失通常按幂律下降。p35-p39
- 上下文学习不更新参数；思维链输出中间推理步骤；幻觉是看似合理但实际错误的生成内容。p41-p45
- 初始化会影响模型推理能力；课件例子中，小初始化通常更有利于学习可泛化规律。p66-p96

8.7 思考题

1. 判断：基础模型通常经过大规模数据训练，可以适应多种下游任务。

   答案：正确。

2. 选择：当前主流生成式大语言模型多采用：A. Encoder-only；B. Decoder-only；C. 纯 CNN；D. 纯 RNN。

   答案：B。

3. 判断：Encoder-only 因为可以看到后文，所以通常不适合直接做“只看前文”的 Next Token Prediction。

   答案：正确。

4. 填空：知识蒸馏中，提供知识的强模型通常叫 _____ model，学习知识的小模型叫 _____ model。

   答案：Teacher；Student。

5. 判断：上下文学习通常不更新模型参数，而是通过任务描述和示例让模型适应任务。

   答案：正确。

6. 判断：幻觉指模型生成看似合理但实际错误或误导的信息。

   答案：正确。

9 强化学习

这一章按 PDF 顺序复习：强化学习概念 -> 基本要素 -> 任务类型 -> 探索与利用 -> MDP、价值函数与 Bellman 方程 -> DP/MC/深度强化学习 -> 大模型中的强化学习。本章围绕概念、关系和最简单公式，不考阅读材料中的 Sarsa、Q-learning 推导。

9.1 强化学习是什么

页码：p3-p5

强化学习是决策型任务。智能体不是从标注标签中直接学习“标准答案”，而是通过与环境交互，根据奖励反馈调整行为。

强化学习：强化学习是智能体在环境中采取动作、获得奖励并转移到新状态，通过反复试错学习最大化期望回报的方法；回报通常指从当前时刻开始的累计折扣奖励。

与监督学习的核心区别：

对比	监督学习	强化学习
学习信号	标注标签	奖励反馈
任务类型	预测型任务	决策型任务
目标	预测正确标签	最大化期望回报
数据来源	固定数据集较常见	与环境交互产生数据

例子 1：游戏中的强化学习。在超级马里奥中，向右走吃到金币可能得到正奖励，撞到敌人可能得到负奖励。智能体通过反复尝试，学习哪些动作更可能带来长期高奖励。

9.2 状态、动作、奖励与回报

页码：p7-p12

强化学习最基本的循环是：智能体观察当前状态，选择动作，环境返回奖励和下一状态。

强化学习的基本要素

强化学习是一种解决控制任务（或称为决策问题）的框架，智能体从环境中获得奖励（正面或负面）作为反馈信号，通过与环境反复交互试错来学习。

强化学习的基本要素：智能体、环境、以及一个由状态、动作、奖励和下一状态组成的循环。

- **Agent（智能体）**：执行动作、学习策略的决策者。
- **Environment（环境）**：智能体交互的世界，会返回奖励和新状态。
- **State（状态）**：当前环境的描述，如“棋盘局面”或“机器人位置”。
- **Action（动作）**：智能体能做出的决策。
- **Reward（奖励）**：环境给予的反馈信号。
- **Policy（策略）**：定义在每个状态下的动作分布 $\pi(a|s)$。
- **Return（回报）**：

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

其中 γ 为折扣因子（Discount Factor）。

状态、观察、动作、奖励：状态是环境完整信息；观察是智能体实际看到的信息；动作空间是可选动作集合，可以离散或连续；奖励是环境反馈信号，可以为正或负。

如果观察包含完整状态，称为完全可观测；如果只能看到状态的一部分，称为部分可观测。

基本交互可以写成：

$$S_t \xrightarrow{A_t} R_{t+1}, S_{t+1}$$

即在状态 S_t 选择动作 A_t，环境返回奖励 R_{t+1} 和下一状态 S_{t+1}。

折扣回报：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

其中，G_t 是从 t 时刻开始的回报，γ 是折扣因子，通常 $0 \le \gamma \le 1$。γ 越接近 0，越重视眼前奖励；γ 越接近 1，越重视长期奖励。

例子 2：折扣回报计算。若未来三个奖励分别为 $1, 0, 2$，$\gamma = 0.5$，则

$$G_t = 1 + 0.5 \times 0 + 0.5^2 \times 2 = 1.5$$

9.3 任务类型

页码：p14-p16

强化学习任务可以分为情节型任务和连续型任务。

任务类型：情节型任务有起点和终止状态，一次任务形成有限序列；连续型任务没有固定终止状态，智能体需要持续交互和决策。

类型	是否有终点	例子
情节型任务	有	一局棋、一个游戏关卡、一次机械臂抓取
连续型任务	没有固定终点	股票交易、交通控制、自动驾驶

9.4 探索与利用

页码：p18-p19

强化学习的关键问题之一是平衡探索和利用。只利用可能卡在局部最优，只探索又会浪费机会。

解决强化学习问题

强化学习的关键问题——探索与利用的平衡：

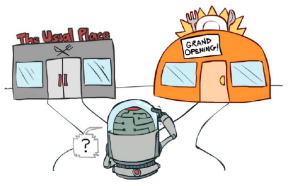
探索（Exploration）：通过尝试随机动作来探索环境，从而获取更多关于环境的信息。

利用（Exploitation）：利用已知的信息来最大化奖励。

在强化学习中，探索和利用的平衡是一个关键问题。探索的缺乏可能导致智能体遗漏环境的关键信息，无法给出最优的决策，而利用的缺乏则背离了最大化期望累计奖励的目标。

智能体控制的小老鼠周围有无限数量的小奶酪（每次获得一个），但在左上角有一个奶酪堆（一次性获得1000个）。小老鼠通过一步行动就能拿到附近的小奶酪，如果此时我们只**利用**这一信息不断取奶酪，就会遗漏左上角的巨大奖励。

选择餐厅：是每天都去同一家已知的还不错的餐厅（**利用**），还是尝试从未去过的餐厅（**探索**）？



探索与利用：探索是尝试未知动作以获得更多信息；利用是根据已有知识选择当前看起来更高回报的动作。

例子 **3**：餐厅选择。去经常吃、评价稳定的餐厅是利用；尝试一家没去过的新餐厅是探索。只利用可能错过更好的餐厅，只探索则可能长期吃不到满意的饭。

9.5 MDP、策略与价值函数

页码：p20-p24

强化学习问题常形式化为马尔可夫决策过程，也就是 MDP。它的核心是马尔可夫性：如果当前状态已经包含足够信息，就不用保留完整历史。

强化学习的模型：马尔可夫决策过程

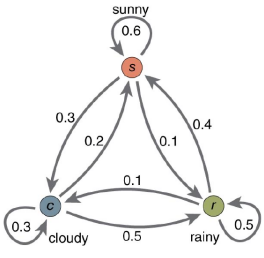
马尔可夫过程与马尔可夫性：

强化学习过程常建模成一个**马尔可夫决策过程**（Markov Decision Process, MDP），

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \ldots, S_t]$$

$$\mathbb{P}[S_{t+1}|S_t, A_t] = \mathbb{P}[S_{t+1}|S_t, A_t, S_{t-1}, A_{t-1}, \cdots]$$

马尔可夫决策过程遵循**马尔可夫性**（Markov Property），即随机过程**未来状态的概率分布仅依赖于当前状态，而与过去状态无关**。这意味着我们的智能体只需要当前状态来决定采取什么行动，而无需考虑之前的历史。



马尔可夫性：给定当前状态后，未来状态的概率分布只依赖当前状态，而不依赖更早历史；在有动作时，下一状态只依赖当前状态和当前动作。

马尔可夫性可以写成：

$$\mathbb{P}(S_{t+1} \mid S_t) = \mathbb{P}(S_{t+1} \mid S_1, \dots, S_t)$$

有动作时：

$$\mathbb{P}(S_{t+1} \mid S_t, A_t) = \mathbb{P}(S_{t+1} \mid S_t, A_t, S_{t-1}, A_{t-1}, \dots)$$

强化学习的模型：马尔可夫决策过程

强化学习问题通常被形式化为一个 马尔可夫决策过程（MDP），

包含的主要元素：

- S：状态空间（States）
- A：动作空间（Actions）
- $P(s'|s,a)$：状态转移概率（Transition Probability）
- $R(s,a,s')$：奖励函数（Reward Function）
- γ：折扣因子（Discount Factor）

$\pi(a|s)$: 在状态s, 动作的概率分布，称为策略函数

MDP： MDP 常包含状态、动作、转移概率、奖励和折扣因子；策略 $\pi(a|s)$ 表示在状态 s 下选择动作 a 的概率。

价值函数（Value Function）

- **Return（回报）**：

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

其中 γ 为折扣因子（Discount Factor）。

- 状态价值函数：

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s]$$

- 动作价值函数：

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$$

价值函数：状态价值函数 $V^{\pi}(s)$ 表示在策略 π 下从状态 s 出发的期望回报；动作价值函数 $Q^{\pi}(s,a)$ 表示从状态 s 出发、先采取动作 a、之后按策略 π 行动的期望回报。

价值函数的两个基本公式：

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s]$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a]$$

这些关系可以放在一起理解：

- 基于策略的方法：直接学习“在状态 s 下应该采取什么动作”。
- 基于价值的方法：学习“哪个状态或动作更有价值”，再据此选择动作。
- 策略评估：用当前策略估计价值函数。
- 策略改进：用估计好的价值函数更新策略。

9.6 Bellman 方程

页码：p25-p31

Bellman 方程刻画的是“当前价值”和“未来价值”的递推关系。本节关注直观含义和矩阵形式，不展开复杂推导。

Bellman 方程：MDP，不考虑动作

Bellman 方程 $V^{\pi}(s) = \sum_{s'} p(s'|s)[R(s,s') + \gamma V^{\pi}(s')]$

向量表示，定义：

$$V := \left(V^{\pi}(s_1), \cdots, V^{\pi}(s_n)\right)^T$$
$$R_s := \sum_{s'} p(s'|s) R(s,s')$$
$$R := \left(R_{s_1}, \cdots, R_{s_n}\right)^T$$

矩阵P的元素满足 $P(i,j) = p(s_j|s_i)$

Bellman 方程 $V = R + \gamma PV$

Bellman 方程： Bellman 方程把当前状态价值写成当前奖励加折扣后的未来状态价值。

直观形式：

$$\text{当前价值} = \text{当前奖励} + \gamma \times \text{未来价值}$$

矩阵形式：

$$V = R + \gamma PV$$

其中，V 是状态价值向量，R 是奖励向量，P 是状态转移矩阵，γ 是折扣因子。

例子 4：两状态价值方程。考虑两个状态 s_1, s_2。在当前策略下，转移矩阵为

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

它表示 s_1 有 100% 概率转到 s_2，s_2 有 100% 概率转到 s_1。若从 s_1 出发得到奖励 1，从 s_2 出发得到奖励 2，且 $\gamma = 0.5$，则

$$V(s_1) = 1 + 0.5V(s_2), \qquad V(s_2) = 2 + 0.5V(s_1)$$

解得 $V(s_1) = \frac{8}{3}$，$V(s_2) = \frac{10}{3}$。

9.7 DP、MC 与深度强化学习

页码：p32-p48

强化学习可以用不同方法求解。这里比较 DP、MC、深度强化学习的基本思想和区别。

方法	核心思想	优点	局限
DP	用 Bellman 方程迭代求价值	可以较精确计算	通常需要环境模型和转移概率
MC	多次实验采样取平均	简单直接，不一定需要知道转移概率	随机性大，方差可能大

方法	核心思想	优点	局限
深度强化学习	用深度神经网络近似价值函数或策略	处理高维状态和复杂特征	训练不稳定，数据和算力需求高

深度强化学习：深度强化学习使用深度神经网络近似价值函数或策略，使强化学习能处理图像等高维输入。

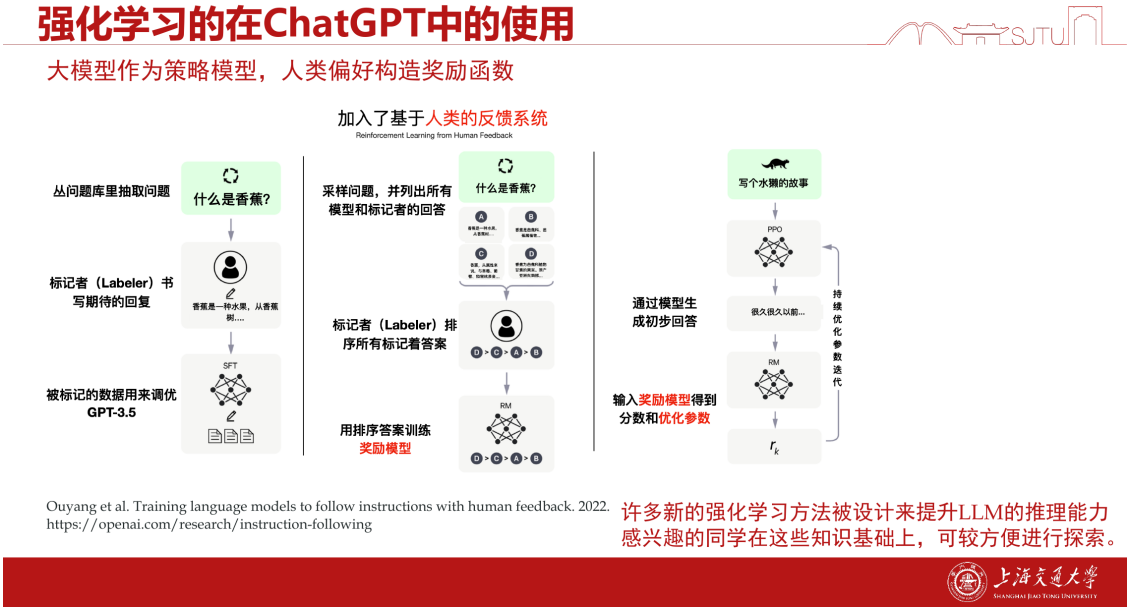
DQN： DQN 使用深度神经网络替代表格来拟合 Q 值。

阅读材料中的 TD、Sarsa、Q-learning 公式推导不纳入本提纲；DQN 和深度网络近似 Q 值有关。

9.8 强化学习在大模型中的应用

页码：p49-p52

强化学习不仅用于游戏、机器人和自动驾驶，也可用于大模型后训练。



RLHF： RLHF 是基于人类反馈的强化学习，通常用人类偏好构造奖励信号，使大模型输出更符合人类偏好和价值观。

在大模型语境中：

- 大模型可以看作策略模型，输出文本相当于选择动作。
- 人类偏好可以用来构造奖励函数，帮助模型对齐人类价值观。
- 强化学习后训练也可用于增强模型推理能力。
- 强化学习不是用来取代预训练，也不是为了直接缩短响应时间。

9.9 关键记忆

- 强化学习是决策型任务，智能体通过与环境交互和奖励反馈学习。p3-p5
- 强化学习目标是最大化期望回报，而不是匹配每一步的标注标签。p3-p5
- 状态是环境完整信息，观察是智能体实际看到的信息；观察可能只是状态的一部分。p7-p8
- 动作空间可以是离散的，也可以是连续的；奖励可以是正的，也可以是负的。p8-p12
- 折扣回报为 $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$；$\gamma$ 越大越重视长期奖励。p9-p12
- 情节型任务有终止状态，连续型任务没有固定终止状态。p14-p16
- 探索是尝试未知动作获取信息；利用是根据已有知识选择当前较优动作。p18-p19
- 马尔可夫性指未来状态分布只依赖当前状态；有动作时，下一状态只依赖当前状态和当前动作。p20-p21
- 策略 $\pi(a|s)$ 是状态到动作选择的规则；$V^\pi(s)$ 衡量从状态出发的期望回报，$Q^\pi(s,a)$ 衡量先做动作 a 后的期望回报。p21-p24
- Bellman 方程的直观含义是当前价值等于当前奖励加折扣未来价值；矩阵形式是 $V = R + \gamma PV$。p25-p31
- DP 通常需要环境模型和转移概率；MC 通过采样估计价值，随机性较大。p32-p34
- 深度强化学习用神经网络近似价值函数或策略；DQN 用神经网络拟合 Q 值。p44-p48
- RLHF 用人类偏好构造奖励信号，让大模型输出更符合人类偏好和价值观。p49-p52

9.10 思考题

1. 判断：强化学习中，智能体通过与环境交互，并根据奖励反馈学习。

   答案：正确。

2. 填空：强化学习的目标通常是最大化期望 _____。

   答案：回报。

3. 计算：未来三个奖励为 $1, 0, 2$，$\gamma = 0.5$，折扣回报为 _____。

   答案：$1 + 0.5 \times 0 + 0.5^2 \times 2 = 1.5$。

4. 选择：尝试一个不确定但可能更好的动作，属于：A. 探索；B. 利用；C. 监督学习；D. 归一化。

   答案：A。

5. 判断：马尔可夫性指给定当前状态后，未来状态分布不再依赖更早历史。

   答案：正确。

6. 填空：动作价值函数 $Q^\pi(s,a)$ 表示从状态 s 出发，先采取动作 a，之后按照策略 π 行动时的期望 _____。

   答案：回报。

7. 判断：DQN 使用深度神经网络替代表格来拟合 Q 值。

   答案：正确。

10 全部记忆点

这一章按闭卷考试来背，不做名词堆叠。每条尽量记成三件事：它用来回答什么问题、核心表述是什么、常见判断哪里容易错。

10.1 公式怎么背

- **Softmax**：用来把一组 logits 变成概率。

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

其中 z_i 是第 i 类分数，p_i 是第 i 类概率。所有 p_i 相加为 1；分数大的类概率更大，整体更接近 one-hot。

- **交叉熵**：用来衡量预测概率和真实标签的差异。二分类中，若 $y_i \in \{0,1\}$，$f_\theta(x_i)$ 是预测为正类的概率：

$$H(y,p) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log f_\theta(x_i) + (1-y_i)\log(1-f_\theta(x_i))\right]$$

更 general 的多分类形式为：

$$H(y,p) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log p_{i,c}$$

其中 C 是类别数，$y_{i,c}$ 是 one-hot 标签，$p_{i,c}$ 是第 c 类预测概率。真实类别概率越大，loss 越小。

- **残差连接**：用来保留原输入信息，缓解深层网络训练困难。

$$y = x + F(x)$$

其中 x 是输入，$F(x)$ 是残差分支学习到的变化量。

- **卷积输出尺寸**：用来计算卷积后高和宽。

$$H_{\text{out}} = \left\lfloor \frac{H+2P-K}{S} \right\rfloor + 1, \qquad W_{\text{out}} = \left\lfloor \frac{W+2P-K}{S} \right\rfloor + 1$$

其中 H, W 是输入空间尺寸，K 是卷积核大小，P 是 padding，S 是 stride。输出通道数不是由这个公式算出，而是由卷积核个数决定。

- **卷积参数量**：用来计算一个卷积层有多少可学习参数。

$$\#\text{params} = K^2 C_{\text{in}} C_{\text{out}} + C_{\text{out}}$$

其中 C_{out} 是卷积核个数，也是输出通道数；最后的 C_{out} 是 bias。参数量不乘 H_{out} 和 W_{out}。

- **多通道卷积**：用来说明一个输出通道怎样由所有输入通道共同算出。

$$Y_j = \sum_{c=1}^{C_{\text{in}}} X_c * K_{j,c} + b_j, \qquad j = 1, \ldots, C_{\text{out}}$$

这里 $*$ 表示在空间维度上做卷积运算；j 是输出通道编号，c 是输入通道编号。X_c 是第 c 个输入通道，$K_{j,c} \in R^{K\times K}$ 是第 j 个 filter 在第 c 个输入通道上的卷积核切片；完整的第 j 个 filter 形状为 $K \times K \times C_{\text{in}}$。一个 filter 产生一个输出通道；有 C_{out} 个 filter 就有 C_{out} 个输出通道。

- **RNN 状态更新**：用来说明当前隐藏状态同时依赖当前输入和历史信息。

$$S_t = f(Ux_t + WS_{t-1} + b), \qquad y_t = g(VS_t + c)$$

U, W, V, b, c 在所有时间步共享，所以序列变长时参数量不随时间步增加；同一个 RNN 单元可以重复更多步，因此可以处理变长序列。
- **Scaled Dot-Product Attention**：用来让一个 token 根据相关性汇总其他 token 的信息。

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

Q, K, V 由输入表示 X 分别乘三个可训练矩阵得到；Q 用来查询，K 用来匹配，V 是被汇总的信息。QK^T 计算关联强度，$\sqrt{d_k}$ 用来缩放，softmax 得到注意力权重，最后对 V 加权求和。

- **ReLU 转折点**：用来判断一维 ReLU 神经元在哪个输入位置发生折线变化。

$$x = -\frac{b}{w}$$

若多个神经元转折点相同，它们在函数形状上可以合并理解；不同转折点提供不同非线性位置。
- **折扣回报**：用来累计未来奖励。

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

$\gamma = 0$ 时只看眼前奖励；γ 越接近 1，越重视长期奖励。
- **Bellman 方程**：用来把当前价值和未来价值联系起来。

$$V = R + \gamma PV$$

V 是价值向量，R 是奖励向量，P 是转移矩阵，γ 是折扣因子。DP 通常需要知道转移概率。
- **Scaling law**：用来描述规模、数据、算力增加时，测试损失通常按幂律下降。

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}, \qquad L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}, \qquad L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

这里记住“幂律关系”：规模继续增大通常还能降低 loss，但收益会逐渐变小。

10.2 基础与残差网络

- **单个神经元**：基础计算是 $z = w^T x + b$，其中 w 是权重，b 是 bias；再经过激活函数得到输出。
- **参数**：神经网络中通过训练学习的量，典型是权重 W 和偏置 b；bias 不是手工常数，而是可学习参数。
- **激活函数**：用来提供非线性。没有非线性时，多层线性变换整体仍等价于一个线性变换。
- **ReLU**： $\text{ReLU}(x) = \max(0, x)$，负数变 0，正数保持不变。
- **Sigmoid**：输出范围在 0 到 1 之间，常用来表示门控开关或概率倾向。
- **Softmax 与交叉熵**： Softmax 先把分数变成概率，交叉熵再惩罚真实类别概率太小的情况。
- **深层网络困难**：深度增加后可能出现梯度消失、梯度爆炸和退化。退化指层数更深但训练效果反而变差。
- **残差连接**：让网络学习 $F(x)$ 而不是完整映射，输出写成 $x + F(x)$；它保留原信息，使深层网络更容易训练。

10.3 卷积神经网络 CNN

- **图像局部关联性**：相邻像素关系强，距离越远关系越弱；CNN 用局部卷积窗口利用这一特点。
- **幂律衰减**：可以看成许多指数衰减的叠加，因此图像中可能仍有较慢衰减的长程关联。
- **图像不变性**：图像任务常希望对平移、旋转、缩放有一定鲁棒性；但图像像素顺序不能随便打乱。
- **CNN 与 MLP 区别**： MLP 通常全连接，参数多且忽略空间局部结构；CNN 使用局部连接和参数共享，参数更少，更适合图像。
- **卷积核**：卷积核是可训练的小窗口权重，同一个卷积核在整张图上滑动，这就是参数共享。
- **输出通道数**：一个 filter 产生一个输出通道；filter 个数通常就是 C_{out}。
- **Padding**：通过补边控制输出大小和边缘信息，不引入可学习参数。
- **Stride**： stride 是滑动步长；stride 越大，输出空间尺寸越小。stride 为 2 的卷积可以实现可学习下采样，也就是在卷积提取特征的同时完成类似 pooling 的降采样。
- **池化**：池化通常没有可学习参数，用来下采样并汇总局部信息。Max pooling 取最大值，Mean pooling 取平均值。
- 1×1 **卷积**：主要用于改变通道数或进行通道间线性组合，不改变空间邻域大小。
- **CNN 结构**：卷积、激活、池化、全连接等层可以灵活搭配，不存在必须固定照抄的网络格式。

10.4 RNN 与 LSTM

- **文本数据特点**：文本是离散符号序列，长度可变，并且有顺序、上下文、语法结构和长程依赖。
- **为什么需要 RNN**： MLP/CNN 更适合固定大小输入或局部结构；RNN 按时间步处理序列，并用隐藏状态保存历史。
- **RNN 处理可变长度的核心**：同一组参数在所有时间步共享，序列变长只增加计算步数，不增加模型参数量。
- **数据质量**：数据越多通常越好，但高质量、去重、过滤偏见/隐私/有害内容的数据更稳定、更省算力。
- **Tokenization**：把文本切成 token。输入是原始文本，输出是 token 序列；常见方法有 BPE、WordPiece、Unigram。
- **One-hot 与 embedding**： One-hot 高维稀疏，只表示编号；embedding 低维稠密、可训练，能承载语义相似性。
- **Embedding 矩阵**：若词表大小为 $|\mathcal{V}|$，embedding 维度为 d，矩阵大小为 $|\mathcal{V}| \times d$；one-hot 乘这个矩阵得到对应词向量。
- **Next Token Prediction**：用前面的 token 预测下一个 token，标签来自文本本身，因此属于自监督学习；训练常用交叉熵。
- **生成过程**：模型一步生成一个 token，再把生成结果接回输入，直到生成 SEP 或达到长度限制；可用贪婪解码或采样。
- **BPTT**： RNN 训练时沿时间展开后的网络反向传播；长序列完整反传开销大，实际可用截断 BPTT 只回看有限步；很长时间步上可能出现梯度消失或梯度爆炸。
- **LSTM**： LSTM 是改进的 RNN，用记忆状态和门控缓解长程依赖问题。
- **LSTM 三个门**：遗忘门决定保留多少旧记忆，输入门决定写入多少新信息，输出门决定输出多少记忆信息。
- **LSTM 常见设计原则**：先做线性变换；sigmoid 适合做门，因为输出在 0 到 1；tanh 适合产生候选信息，因为输出可正可负。
- **Encoder-Decoder RNN**：先编码输入序列，再解码输出序列，可处理输入输出长度不同的任务。

10.5 Transformer

- **RNN/LSTM 局限**：token 需要串行处理，长依赖仍难；Transformer 允许序列中 token 并行输入、并行输出。
- **位置编码**：注意力机制本身不关心顺序，所以必须给 token 加入位置信息；位置编码不改变词表大小。
- **Embedding + Position**：embedding 表示 token 语义，position encoding 表示位置，两者相加后作为 Transformer 输入。
- **Q、K、V**：Q 是 Query，表示当前 token 要查什么；K 是 Key，表示其他 token 可被匹配的标识；V 是 Value，表示真正被汇总的信息。
- **Q/K/V 维度**：K 和 V 的数量必须相同，也就是第一维相同，因为每个 key 要对应一个 value；Q 和 K 的最后一维必须相同，才能做点积。课件重点是 self-attention，三者来自同一段序列，所以 token 数相同；扩展到 cross-attention 时，Q 的数量可以和 K/V 不同。
- **注意力四步**：先线性变换得到 Q/K/V；再用 QK 点积算关联强度；再 softmax 得到注意力权重；最后用权重对 V 加权求和。
- **Causal mask**：在生成任务中遮住未来 token，保证当前位置只能看自己和过去，不能提前看到答案。
- **LayerNorm**：以单个 token 的特征维度为单位归一化，不在 token 之间计算均值方差；有可训练的 γ, β。
- **FFN**：Transformer 中的 FFN 是逐 token 的 MLP，参数在位置间共享；token 之间的信息交互主要发生在 attention 中。
- **输出投影**：最后把隐藏向量映射到词表大小，再用 softmax 得到下一个 token 的概率。
- **解码方式**：贪婪解码每步取最大概率 token；采样解码按概率抽样，因此同一输入可能有不同输出。

10.6 训练现象：频率、凝聚、平坦性

- **过参数化问题**：神经网络参数很多，能表达复杂函数，但训练仍常得到有规律、可泛化的解；这与优化和结构带来的隐式偏置有关。
- **傅里叶级数直觉**：函数可看成不同频率成分的叠加。低频变化慢、平滑；高频变化快、振荡、细节多。
- **频率原则**：神经网络训练通常先学习低频成分，再学习高频成分。
- **控制变量理解**：判断"先低频还是先高频"时，应比较同幅值、不同频率的成分，而不是把幅值影响和频率影响混在一起。
- **两种频率**：图像频率描述像素变化快慢，响应频率描述模型输出对输入变化的敏感程度；高响应频率表示微小输入扰动就可能改变输出。
- **早停**：如果噪声或过细节信息偏高频，早停可以在模型充分学习低频主结构后，减少对高频噪声的拟合。
- **初始化影响**：初始化影响初始参数、初始函数和训练路径；全零初始化会让神经元对称，通常不可取。
- **ReLU 凝聚**：小初始化下，同层 ReLU 神经元的方向和转折点可能趋同，等效为更少的有效神经元。
- **为什么不用小网络代替**：凝聚后的大网络可近似等效小网络，但大网络的训练路径更容易优化，直接训练小网络不一定同样容易。
- **平坦解**：参数附近一片区域 loss 都低，扰动后仍稳定，通常泛化更好。
- **尖锐解**：低 loss 区域很窄，参数轻微变化 loss 就明显上升，通常更不稳定。
- **小批量与大学习率**：小批量梯度噪声有助于跳出尖锐区域；合理较大的学习率更容易得到平坦解，但学习率太大会导致震荡甚至不收敛。

10.7 大模型

- **基础模型**：用大规模数据训练、可迁移到多种任务的大模型，是后续微调和应用的基础。
- **LLM**：大语言模型以文本 token 为主要输入输出，核心训练目标常是 next token prediction。
- **Encoder-only**：双向看上下文，代表是 BERT，适合理解类任务；不适合只能看过去 token 的自回归生成。
- **Decoder-only**：使用 causal mask，只看当前位置及以前 token，代表是 GPT、Llama、DeepSeek，是当前生成模型主流结构。
- **Encoder-Decoder**：编码器处理输入，解码器生成输出，常用于翻译、摘要等输入输出都明确的任务。
- **预训练**：用海量文本学习通用语言和知识能力，代价高，是大模型能力来源之一。
- **SFT**：监督微调用标注指令数据继续训练，让模型更会按任务要求回答。
- **RL 后训练**：用奖励或人类偏好继续优化，可增强推理和对齐；它不是用来取代预训练，也不是直接缩短响应时间。
- **知识蒸馏**： Teacher 模型提供输出或行为，Student 模型模仿它，用于压缩和部署。
- **提示词工程**：改输入提示，不更新模型参数；上下文学习也是在不更新参数的情况下利用示例完成任务。
- **思维链**：让模型输出中间推理步骤，常用于复杂推理。
- **初始化与推理**：初始化会影响训练路径和最终能力；课件例子中，小初始化通常更有利于学习可泛化规律。
- **涌现**：模型规模增大到某个范围后，某些能力可能从弱到强突然显现。
- **幻觉与偏见**：幻觉是模型生成看似合理但错误的内容；偏见是输出中存在系统性倾向。

10.8 强化学习

- **强化学习**：决策型任务，智能体通过动作影响环境，根据奖励反馈学习策略，目标是最大化期望回报。
- **与监督学习区别**：监督学习依赖标注标签；强化学习依赖奖励反馈，奖励可能延迟出现。
- **状态与观察**：状态是环境完整信息；观察是智能体实际看到的信息，可能只是状态的一部分。
- **动作空间**：智能体可选动作集合，可以是离散的，也可以是连续的。
- **情节型与连续型**：情节型任务有终止状态；连续型任务没有固定终止状态。
- **探索与利用**：探索是尝试未知动作获取信息；利用是选择当前认为最优的动作。只利用可能错过长期更优选择。
- **马尔可夫性**：给定当前状态后，未来状态分布只依赖当前状态，而不依赖更早历史。
- **MDP**：常包含状态、动作、转移概率、奖励和折扣因子，是强化学习问题的标准形式化。
- **策略**： $\pi(a|s)$ 表示在状态 s 下选择动作 a 的规则；最优策略使期望回报最大。
- **价值函数**： $V^\pi(s)$ 表示从状态 s 出发的期望回报；$Q^\pi(s,a)$ 表示先采取动作 a 后的期望回报。
- **策略评估与改进**：评估是估计当前策略的价值；改进是根据价值更新策略。
- **DP 与 MC**： DP 通常需要环境模型和转移概率；MC 通过多次采样取平均，不一定需要知道转移概率，但方差较大。
- **深度强化学习**：用神经网络近似价值函数或策略；DQN 用神经网络替代表格拟合 Q 值。
- **RLHF**：用人类偏好构造奖励信号，使大模型输出更符合人类偏好和价值观。