人工智能数学原理与算法

课程互助手册

AI 互助文档项目组

主要编写：周少承　陈熙麟　史睿铭　操朗峰

2026 年 3 月 10 日

目录

总序 v

第一部分　理论部分 1

理论部分序言 2

第一章　概述 3
1.1 课程基本信息 . . . 3
1.1.1 课程定位与学习建议 . . . 3
1.1.2 课程内容框架 . . . 3
1.1.3 授课内容安排 . . . 4
1.2 人工智能简介 . . . 4
1.2.1 人工智能的主要子领域 . . . 4
1.2.2 人工智能的主要流派 . . . 5
1.2.3 人工智能历史 . . . 5

第二章　机器学习基础与有监督学习 6
2.1 机器学习介绍 . . . 6
2.1.1 机器学习的定义 . . . 6
2.1.2 相关名词逻辑关系 . . . 7
2.1.3 机器学习基础应用 . . . 7
2.1.4 机器学习相关基础概念 . . . 7
2.1.5 机器学习的分类 . . . 9
2.2 线性回归 . . . 11
2.2.1 假设类 . . . 12
2.2.2 损失函数 . . . 12
2.2.3 优化算法 . . . 14
2.3 逻辑回归 . . . 19
2.3.1 假设类 . . . 19
2.3.2 逻辑回归的损失函数与梯度优化 . . . 21
2.3.3 多分类问题 . . . 22
2.3.4 过拟合 . . . 24
2.3.5 如何评估分类模型的好坏 . . . 26

第三章　神经网络基础 29

3.1 前馈神经网络 . . . 29
3.1.1 神经网络基本介绍 . . . 29
3.1.2 感知机与全连接神经网络 . . . 32
3.1.3 万能近似定理 . . . 35
3.1.4 卷积神经网络 . . . 36
3.2 神经网络优化 . . . 41
3.2.1 反向传播算法 . . . 41
3.2.2 其他内容简介 . . . 44
3.3 深度神经网络 . . . 48
3.3.1 训练策略一：残差连接 . . . 49
3.3.2 训练策略二：归一化 . . . 50
3.3.3 训练策略三：初始化 . . . 54

第四章 图神经网络 **55**
4.1 图学习基础 . . . 55
4.1.1 图学习简介 . . . 55
4.1.2 图的表示 . . . 56
4.2 图表征学习 . . . 60
4.2.1 节点嵌入 . . . 61
4.2.2 图嵌入 . . . 62
4.2.3 下游任务 . . . 63
4.3 图神经网络 . . . 64
4.3.1 图神经网络的建模 . . . 64
4.3.2 图神经网络的优势 . . . 65
4.3.3 训练流程与数据划分 . . . 65

第五章 Transformer **69**
5.1 注意力与自注意力机制 . . . 69
5.1.1 循环神经网络（RNN） . . . 69
5.1.2 注意力机制 . . . 71
5.1.3 自注意力机制 . . . 75
5.2 Transformer 的编码器与解码器 . . . 79
5.2.1 Transformer 的编码器 . . . 79
5.2.2 Transformer 的解码器 . . . 86
5.3 Transformer 的典型应用与跨模态能力 . . . 91

第六章 自监督学习 **94**
6.1 自监督学习概述 . . . 94
6.1.1 基本概念与范式 . . . 94
6.1.2 自监督学习的框架 . . . 96
6.2 word2vec 和 BERT 模型 . . . 99
6.2.1 word2vec . . . 99
6.2.2 BERT 模型 . . . 101

6.3 自回归语言建模 . . . 104
6.3.1 n 元语法语言模型 . . . 106
6.3.2 神经网络语言模型 . . . 109
6.3.3 GPT 系列 . . . 110
6.4 大语言模型简介 . . . 110
6.4.1 语言模型发展历程 . . . 110
6.4.2 能力特点 . . . 111
6.4.3 构建方法 . . . 112

第二部分 实验部分 **114**

实验部分（待完成） **115**

附录 A 重要算法 **116**
A.1 全连接与卷积神经网络参数量计算 . . . 116
A.2 逻辑回归——二分类问题 . . . 117
A.2.1 预测值的计算 . . . 117
A.2.2 优化与正则化 . . . 117
A.2.3 样例 . . . 118
A.3 反向传播算法（含例题） . . . 118
A.3.1 输出层误差 . . . 119
A.3.2 隐藏层误差传播 . . . 119
A.3.3 参数梯度计算 . . . 119
A.3.4 参数更新（梯度下降） . . . 119
A.3.5 算法流程总结 . . . 119
A.3.6 示例 . . . 120
A.4 n 元语法模型 . . . 121
A.4.1 核心原理（马尔可夫假设） . . . 121
A.4.2 句子概率计算 . . . 122
A.4.3 模型评价：困惑度（Perplexity） . . . 122
A.4.4 示例 . . . 122
A.5 Transformer 编码器中的多头注意力 . . . 123
A.5.1 一、输入表示 . . . 123
A.5.2 二、线性映射 . . . 123
A.5.3 三、计算注意力分数 . . . 123
A.5.4 四、Softmax 计算注意力权重 . . . 124
A.5.5 五、计算注意力输出 . . . 124
A.5.6 六、多头注意力 . . . 124
A.5.7 Transformer 编码器多头注意力计算 . . . 124

附录 B 考点汇总 **126**

附录 C 试卷合集 **130**

C.1 25 秋（B 系列）试卷回忆版 130
C.1.1 一、判断正误并说出理由（10×5 分） 130
C.1.2 二、解答题（共五道） 130
C.2 25 春模拟卷（含答案） 131
C.2.1 模拟试卷 131
C.2.2 模拟试卷答案 141

总序

本手册用于辅导《人工智能数学原理与算法》课程的学习，力求覆盖从理论到实验，适配从日常学习到考前突击的不同需求。

笔记的编写想法得从课程的历史谈起。2025 春，学校为适应 AI 教育趋势，在少年班学院的 24 级（大一下）试点开展本门课程，作为限定选修课（少院大一学生根据初步专业规划要选的一门低年级专业课，比如《代数学基础》、《化学原理》等等），50 人班。初次的讲授形式为多位老师参与，每个老师负责一个板块，同时配备 6 位助教。但由于此类课程先例较少，临时制作的 PPT 中有诸多瑕疵。该学期结束后，学校在 24 级培养方案中加入此门课程（这引起了广泛的学生争议，详见评课社区）。大部分的学院将此课程开始在 2 春、3 秋、3 春等，其中工程科学学院强烈要求在 2 秋（即 25 秋）开课。课程组临时编写此门课程的教材，对 PPT 仅做简单修改，开设了两个班级。笔者由于一方面对相关知识有了解的欲望，一方面希望减轻后续课程负担，虽为数学专业仍提前选修该门课程。作为零基础学生，笔者全程听讲且作了一些笔记。在期末复习时，笔者结合笔记内容，利用 AI 对诸多之前感到困惑的内容重新学习，发现对之前的不少理论有所误解，以及感觉 PPT 和课本上有诸多容易对初学者造成误解之处。同时，与笔者关于此课程交流甚多的史睿铭等几位同学对于实验课大作业的完成、考前复习等方面有诸多的感触。我们想起之前一届届学长因为情怀所作的手册，从《飞跃手册》到《大雾实验不完全指北》等等，为科大学生带来了不少的帮助，遂产生了寒假编写一项类似性质的手册，不能望其项背，但也力求能带来些许帮助。

关于考核，25 春（第一学期）**总评公式严格按照 10%（作业）+10%（考勤）+40%（实验）+40%（考试）**，不过由于期末考试难度不大，以及主动选择该课程的少院同学按常理对相关知识应比较感兴趣，学习态度相对认真，优秀率达到 80%。25 秋（第二学期）**仍然遵循此公式**，考试难度、风格和第一年接近甚至略有下降，大作业评分宽松（出现诸多满分），但由于此门课程成为工院强制性的必修等原因，学习效果相对第一次较差。预计大作业的评分标准会根据考试情况做调整，如果考试分数普遍较高，可能出现大作业严格的情况。因此在不知考试难易程度的情况下请认真对待大作业。

关于课程本身，设计初衷是好的，在当下时代了解些许 AI 对各行各业多少有些用处。课程的内容方面，实则是摘取了机器学习、深度学习等课程的知识，削弱了实验和理论的难度，产生的一门广而浅、介于导论和专业课之间的课程。由于巨大的知识量、时间仓促导致设计欠佳等原因，课程可能存在入门不够简单等问题。以及，有志于学习相关知识的二、三年级学生，是否选修一些 AI 方向的专业课程，比此门课程效果更佳？一些话题需要理性地讨论而非无谓的情绪宣泄。事实上，这门课对于笔者等几位对 AI 方向感兴趣的同学起到了入门的作用，教授了机器学习和深度学习的基础知识。也许抱着一个积极而非消极的、改进而非一味批判的态度去学习，也会有诸多收获和感触。

本手册分为四个部分，正文一为理论部分，是基本按照课程逻辑顺序，基于原先的笔记和课件、适当利用 AI 辅助，对知识地全方位讲解，其中穿插对考试内容的理解和预测、少量拓展内

容等。主要由**周少承**（24 少院数学），其中第五章为**操朗峰**（24 工院）完成。正文二为实验部分，主要分享几个大作业的完成经验，由**史睿铭**（24 少院数学）完成。（待完成）附录一、二分别为速通文档和算法实操，是基于 25 春、秋以及模拟卷的风格做出的考前复习指南。前者的灵感来源于徐小航学长（20 少）的《原子物理 B》公式集，是从正文一中摘取出的部分可能的考点；后者为结合各章节可能考察计算大题的算法的汇编和案例实操。该部分由**陈熙麟**（24 数学）完成。附录 C 为我们收集的试卷，也期待后来者持续帮忙更新。

特别鸣谢：在 25 春（首次开课）修读课程的**陈明珠**（24 少院 AI）同学自行完成的大作业经验分享，为我们的大作业部分提供了诸多思路；**刘昊天**（24 网安）回忆了 25 秋课程的大部分试题；**沈鼎丁**（24 少院 AI）和**宣明哲**（24 少院统计）为整体设计和具体内容提供了诸多意见。另外感谢通过各渠道为本手册提供意见、资料的人员。

另外，A、B 系列课程内容差别不大，正文一中作了详细的说明。其中第七章强化学习我们没有做介绍，以及附录部分等内容需要随着考试风格的更新而更新。我们持续期待后来者为此手册的完善和迭代做出贡献。

24 少院数学　周少承

2026.3

第一部分

理论部分

理论部分序言

1. 本笔记修订自 25 秋 B 系列课程（冯福利，王文杰老师班）时笔者所做的笔记。主要内容基于老师的授课、25 版 ppt（春、秋）、以及当时的教材。

2. 据老师声称，由于编写时间仓促，25 秋版的教材上有诸多疏漏。加之笔者在上课时偶尔开小差未能完成笔记等等原因，部分笔记内容为上课期间以及期末复习时与 AI 交互，结合笔者个人理解完成。虽已反复审阅，但由于专业知识有限，难免有所纰漏。望阅读时不吝赐教。以及，可以拿对应部分的知识作为 prompt 喂给 AI 让其总结补充（学 AI 课一定要会用 AI）

3. 关于课件，25 春为多个老师分讲各个专题，25 秋版由于时间仓促，只是在原先的课件上略作修改。据老师声称，课件上出现了多处前后概念的定义上的小 bug。当然，其实并不影响对核心内容的学习，但可能会对初学者造成概念上的困扰。笔者在写作本笔记时，已经在尽力修正这些 bug。如果出现笔记与课件上不一致的，可以自行斟酌或询问 AI，如笔者有误，希望能够向我们反馈。

4. B 系列课程与 A 系列课程的区别在于对部分学时进行了删减。主要删减的内容在于第四章（图神经网络）以及第七章（强化学习）。具体见第1.1.3节。其中第四章笔者结合 A 系列 ppt 以及 AI 交互，笔记内容基本上能够覆盖 A 系列知识（当然实际上在 25 秋的 B 系列考题中，该部分内容丝毫没有考察）。而第七章内容由于授课时间仓促，只做了很简单的概念介绍，忽略了 Q-learning 等算法。综合考虑各种因素，本笔记对第七章内容介绍。望有意的后来者联系笔者补全。

5. 笔记正文中包含了一些上课中或笔者自查得到的一些拓展知识内容，正常来说不会考察。已用 高亮 文字标出。

6. 本课程的定位本身还是导论类课程（虽说个人实际体验感觉相对于导论还是太硬了），因此知识的理解难度层面其实并不大，但是内容量并不少且前后有一定关联性。又因为作业量很少，笔者没有及时复习，在第一遍学习的过程中，经常出现提到先前知识出现遗忘的现象，进而造成暂时的学习困难。对于零基础且想要认真学习本门课的同学，建议从第二章开始，每节课开始前能够抽出少量时间回顾上节课的内容，并且时刻注意把握各个知识的逻辑。

联系方式（GitHub 仓库管理同学，QQ）：

- 史睿铭：1600785006
- 周少承：1553179093
- 沈鼎丁：424939027

第一章　概述

1.1　课程基本信息

1.1.1　课程定位与学习建议

以下为笔者在 25 秋上课时按个人理解摘录了老师的部分话语：

- 本课程的目的在于帮助了解 AI 各个领域的基本知识，基本覆盖科研实践各个部分的最基础知识。
- 内容注重广度而非深度。具体细节等待需要实践时再学习。
- 在 25 秋的课件中，每节课开头有回顾内容。这是最核心的部分，可以作为复习的参考。

1.1.2　课程内容框架

本课程：聚焦于基于神经网络的机器学习（专深>广浅）

□ 初步了解人工智能，聚焦神经网络，掌握其数学原理、算法、数据特性与实践应用

	符号流派	连接流派	行为流派	统计流派	仿生流派	融合流派	……
自然语言处理	文法、形式语言	大模型		词频、LDA			
计算机视觉		卷积神经网络			仿生视觉		
知识表示推理	逻辑、逻辑程序、专家系统		SOAR	蒙特卡洛		知识图谱	
机器学习	决策树、归纳逻辑程序	神经网络（前向、卷积、图、注意力）		回归、支持向量机、PAC	遗传算法	知识图谱，AlphaGo	
多智能体系统	BDI				人工生命		
不确定性AI	贝叶斯网络、概率逻辑			贝叶斯网络			
行动选择	搜索、规划	强化学习	反应式系统、包容体系结构	决策论、马尔可夫决策			
……							

图 1.1：课程内容框架

标红内容为本课程主要内容。

1.1.3　授课内容安排

图 1.2: 授课内容安排

其中原图中的时间安排为 25 春第一次授课的安排（预计也就是 A 系列的授课安排）。红色字体为 25 秋 B 系列的授课安排。具体差异见序言。实验课授课安排部分见后。

1.2　人工智能简介

1.2.1　人工智能的主要子领域

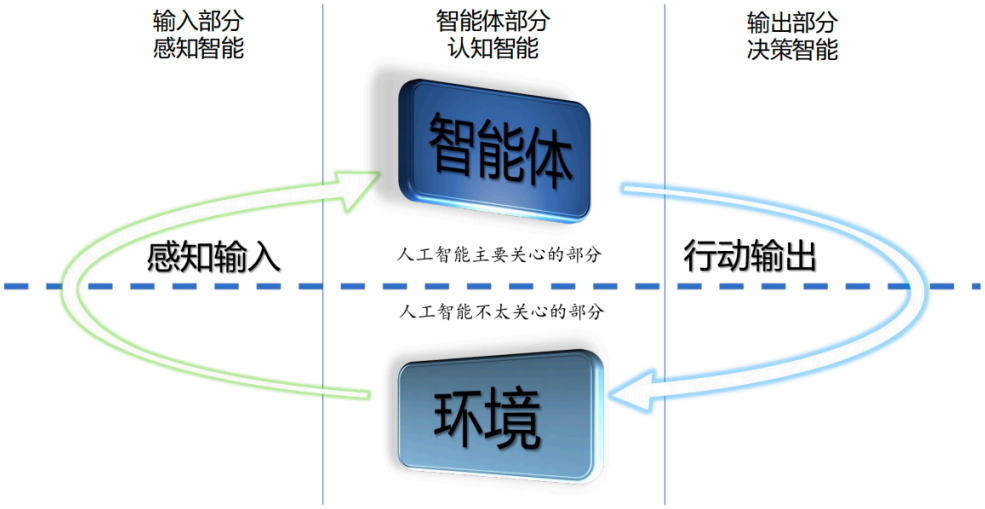


图 1.3: 人工智能的主要子领域

1. **输入部分：感知智能**。机器如何从传感器的原始输入中得到想要的数据与信息。包括计算机视觉，自然语言处理，语音识别与处理等。

2. **智能体部分：认知智能**。机器如何基于感知信息和自身模型解决问题，及如何学习更好的自身模型。包括知识表示与推理，机器学习，多智能体系统等。

3. **输出部分：决策智能**。机器如何基于自身模型，做（理性/正确的）决策，及机器如何行动。如机器人。

1.2.2 人工智能的主要流派

1. **连接流派：**模拟人脑的生物神经网络。每个神经元按照权重接受信息，集中处理后通过类似门控的装置再输出。各个神经元间连接。

2. **符号流派：**用符号与逻辑表示知识，用符号推理与搜索解决问题。也就是对输入进行形式化逻辑运算。相对古老。

3. **统计流派：**用数据方式（统计学理论与工具估算）求解。例如在建立机器翻译系统时，靠人利用语法建立语言的映射过于困难。利用统计预测对应词汇概率要高的多。

4. **行为流派：**机器模仿人的行为，用简单的反应式规则解决问题。

5. **博弈流派：**基于博弈论，主要使用多智能体决策问题。

6. **交叉/融合/统一**以上的几个流派。

1.2.3 人工智能历史

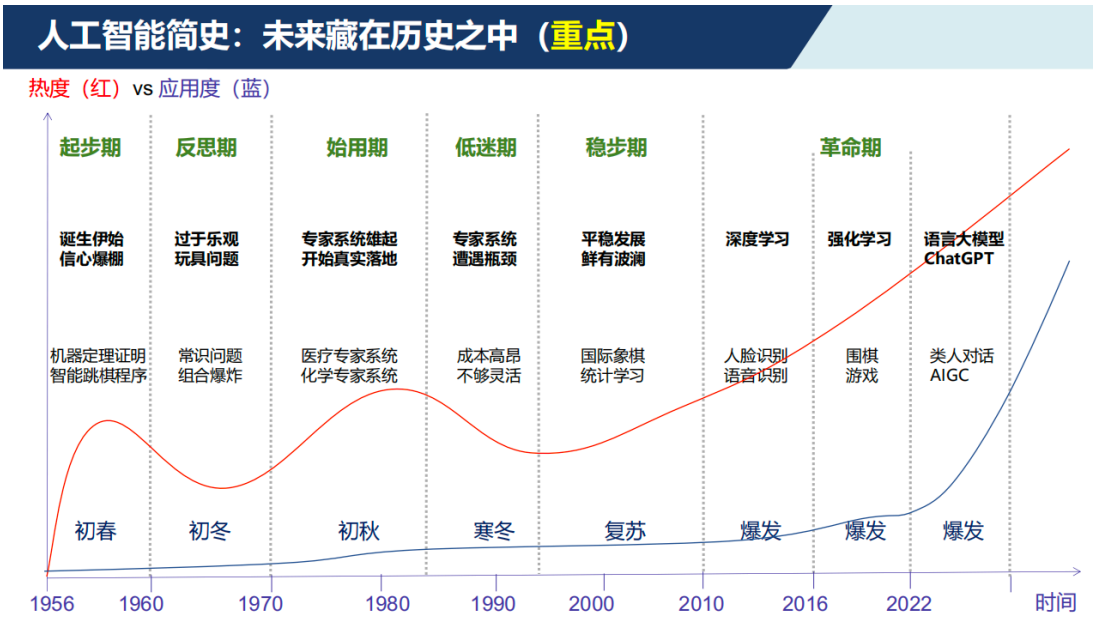


图 1.4: 人工智能历史

第二章　机器学习基础与有监督学习

2.1　机器学习介绍

2.1.1　机器学习的定义

这里留一个对概念的初步印象就好，不必细究。

- 假设用 P（例如，识别正确率）来评估计算机程序在某类任务 T（例如，识别好瓜与坏瓜）上的性能，若一个程序通过利用经验 E（例如，标注好的好瓜与坏瓜图片）在任务 T 上获得了性能改善，则我们就说关于 T 和 P，该程序对 E 进行了学习。也就是**利用经验改善任务性能**。用公式体现就是如下图所示。可以简单理解成利用数据学习（更新）模型的参数，使其更好从输入得到输出。

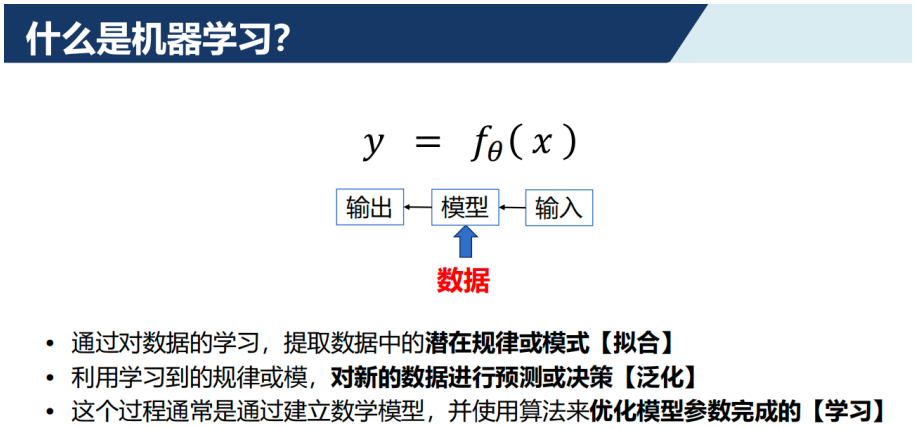


图 2.1: 机器学习

- 传统机器学习流程：明确问题 → 数据选择 → 特征工程（对原始数据分析处理转化为模型可用的特征以描述规律）→ 模型训练（用既定方法学习数据经验，并调整模型）→ 模型评估 → 模型决策。

- 课件中应该介绍了一个**贷款诈骗**的例子。这是一个典型的分类问题，那里还会提到，我们放在第2.3.5节讲解。实际上这部分与机器学习关系不大，也不需要逻辑回归的知识，更像一个纯统计学的指标，有兴趣可以提前看。25 春的模拟题上出现了与之相关的背景，但感觉不需要这部分知识完全也能解答。

2.1.2 相关名词逻辑关系

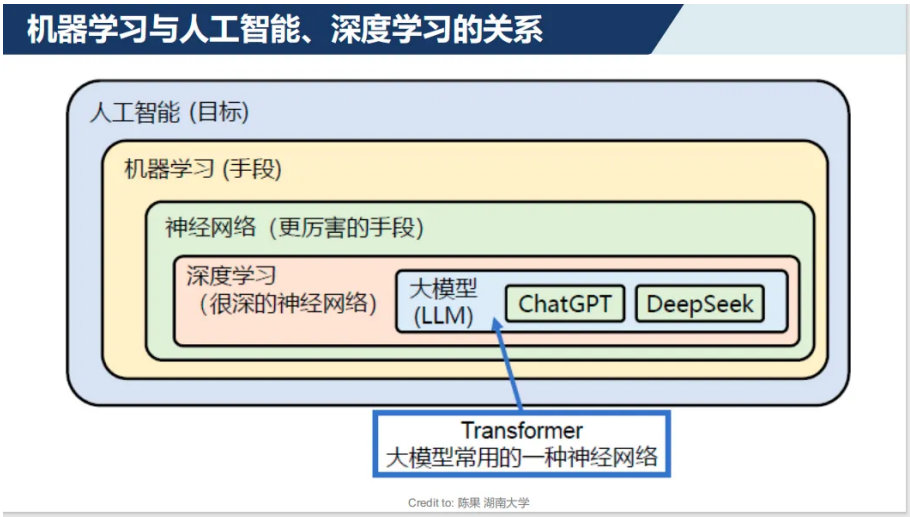


图 2.2: 相关名词逻辑关系

- 同样的数据的不同表达会直接决定模型是否能完成学习。找到好的数据表示是机器学习的核心方法。**表示学习**指的是模型自动从数据中提取特征或表示的方法。**神经网络**是表示学习的最重要实现方式。

- 设计算法时目标往往是分离出能解释观察数据的变差因素。这些因素往往不能直接被观察，例如语音中的说话者年龄信息，需要复杂的接近人类水平的理解来辨识。**深度学习**通过简单表示的堆叠来表达复杂表示。

2.1.3 机器学习基础应用

- ppt 上介绍了诸多应用。其应用不仅在于搜索引擎，自动驾驶这些科技领域，还在于一些画作鉴别，古文献修复这些传统领域。

- 老师上课推荐的网站：机器学习二级任务演示中演示了一些文档提取、文本生成等二级任务的演示，机器学习一级任务演示中演示了分类、回归等这些一级任务的演示。

2.1.4 机器学习相关基础概念

以下先按**有监督学习**的模式介绍相关概念（其余的如无监督学习等可以看成是有监督基础上做一些变化，核心概念类似）。

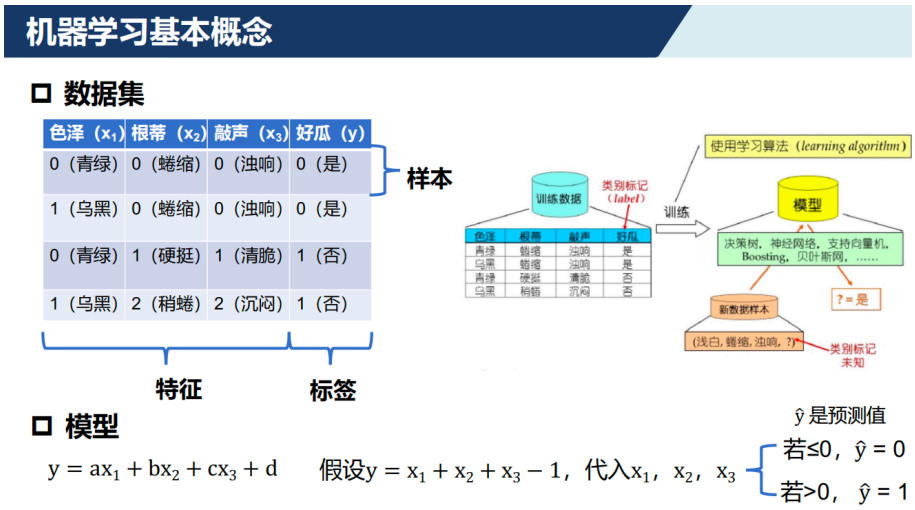


图 2.3: 基本概念

- 上图中的 $y = ax_1 + bx_2 + cx_3 + dx_4$ 表示一个**模型**，通过输入 (x_1, x_2, x_3, x_4) 得到输出（预测）y。其中 (a, b, c, d) 称为模型的**参数**，也就是图2.1中的 θ。
- 不同的参数会导致不同的预测内容。我们需要调整参数[1]。如何评价参数的好坏？利用**损失函数**。损失函数种类的选取依赖于对误差分布的预测，在第2.2.2节中我们介绍了为什么回归问题使用平方损失函数，这里先不必细究。

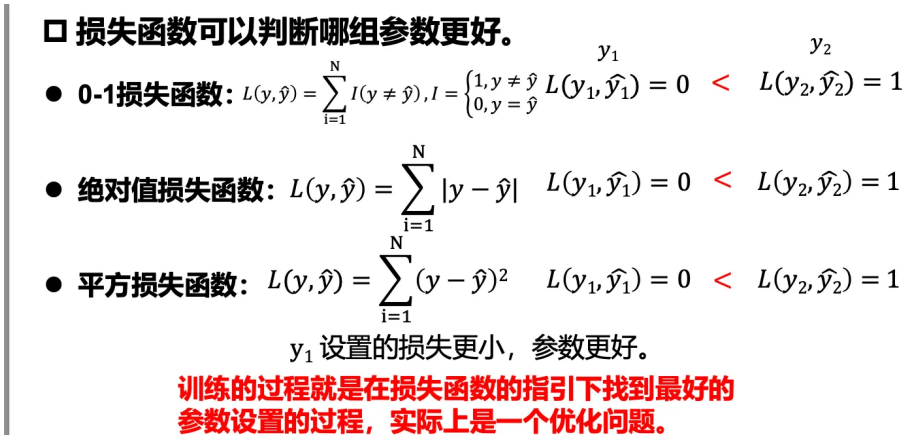


图 2.4: 损失函数

- 有监督学习下的**模型训练**过程就是，利用部分已经知道正确输出（称为**标签**）的输入数据，利用现有模型得到其预测的输出，然后比较正确输出和预测输出的差距，再**更新参数**（也就是**学习**）。更新参数的方法参见第3.2.1节，暂时不必细究。
- 参数的更新并不是使得损失函数越小越好，可能出现**过拟合**。见下。
- **数据集**分类：一种是上述过程中用于训练模型的数据，称为**训练集**（理解成做练习题）；一种是在模型训练完成后，在评估阶段评估模型性能，称为**测试集**（整个训练和调参过程中被**严格隔离**，为了测试已经训练好的模型的性能，理解成高考）。

[1]后续课程中会经常听到超参数这个词。其与参数是完全不同的概念。超参数是模型训练**开始前**就设定好的参数，它们控制着学习过程本身。例如：学习率（可以理解成更新参数的速度）、神经网络的层数和宽度等。它们**不能**从训练数据中自动学习。而参数是模型在训练过程中不断自动更新的，用于从输入得到输出。

- **过拟合**指的是模型对训练集的数据过度依赖（理解成死记硬背练习题），失去了泛化的能力，也就是在训练集上表现良好，而在测试集上表现差劲（也就是对新题目懵了）。这里留一个基本印象就好，第2.3.4节会进一步解释其表现、机理、缓解方法等等。

拓展内容

在数据集的设置上有一种缓解过拟合的方法，也就是设置**验证集**。从训练集中抽取一部分数据，不用于训练，在训练好后用于模拟评估。（理解成模拟考）部分方式及作用如下：

作用	训练方法
监控模型是否发生过拟合	在训练过程中（尤其是深度学习的时代），我们定期（如每个训练周期后）计算模型在**训练集**和**验证集**上的误差/准确率，并绘制学习曲线。如果训练误差持续下降，但**验证误差在达到一个最低点后开始上升**，这说明模型开始”死记硬背”训练数据，对新数据的泛化能力变差。此时，我们通常应该停止训练（**早停法**），选择验证误差最低时的模型状态。
调整模型超参数	我们将不同的超参数组合（如不同的学习率、不同的网络结构）视为不同的”学习方法”。对每一种组合：用**训练集**训练模型，再用**验证集**评估这个训练好的模型的表现（如计算准确率、损失值），选择在**验证集**上表现最好的那组超参数。
选择最佳模型	我们用相同的训练集训练所有候选模型，然后用同一个验证集评估它们的性能。选择在验证集上表现最优的模型作为最终候选，送去参加最终测试。

2.1.5 机器学习的分类

图 2.5: 机器学习的分类

- 监督：数据带有”标签”或”答案”。

- **监督学习**：有标签（数据 + 答案），学习”输入”到”输出”的映射关系。可分为**分类**（预测离散值，如垃圾邮件处理，图片识别）和**回归**（预测连续值，如房价，股票走势等）。

- **无监督学习**：无标签，这种方法不依赖外部信息，只关注模型输出的结果本身是否具有某些良好的数学统计特性。它主要用于**聚类**（划分数据集，目标是”物以类聚”，即同一组内的数据点非常相似，不同组之间的数据点非常不相似）和**降维**（任务目标是在尽可能保留原始数据重要信息的前提下，减少数据的特征数量（维度））。

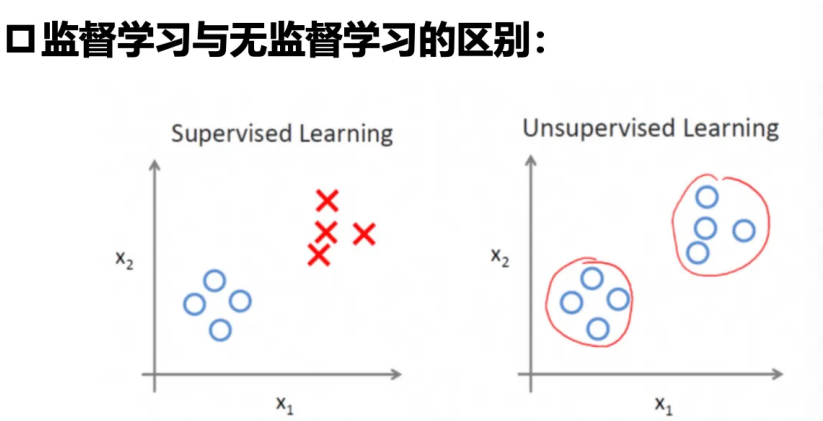


图 2.6: 有监督与无监督

- **半监督学习**：少量标注数据，大量未标注数据。

- **自监督学习**：模型自己学习如何预测自己的输出，例如图像识别中的自学习。详见第六章。

- **强化学习**：经验 E 是由**计算机与环境互动获得**的，计算机产生行为，同时获得这些行为的结果，我们的程序只需要**定义这些行为的收益函数**，对行为进行奖励或惩罚。通过设计算法让计算机改变自己的行为模式去最大化收益函数，完成机器学习的过程。（典例是 AlphaGo）

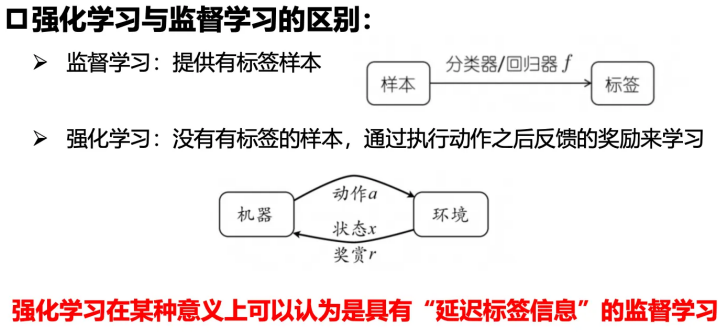


图 2.7: 强化学习

总结

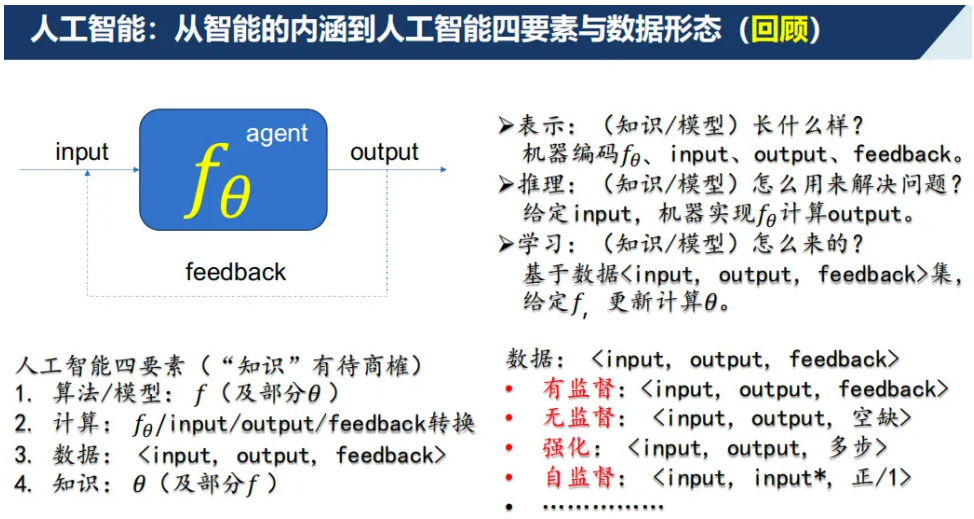


图 2.8: 机器学习介绍总结

2.2 线性回归

前言

- 这里由于科大学生有扎实的数理基础，跳过了 25 春、秋中的数学基础部分。（数学基础包括线性代数和最基础的概统，这是由于第一次授课是 25 春面向大一下的少院同学，后续授课中预计该部分可能删除。）
- 二三两节分别对应有监督学习中最经典的两类任务：回归（预测连续值）——线性回归，分类（预测离散值）——逻辑回归（之所以也称作回归是因为其建模形式和思路来源于数学中的回归）。
- **回归 (regression)** 实际上是一个数学概念，可以理解成通过一个参数化的函数（通常是线性函数）来近似描述因变量与自变量之间的关系，并通过优化某种准则（如最小二乘、极大似然）来估计参数。
- 在学习这两节时，**不必过于纠结其中的数学原理，关注其建模方法即可**。这两节均按照如下逻辑展开：

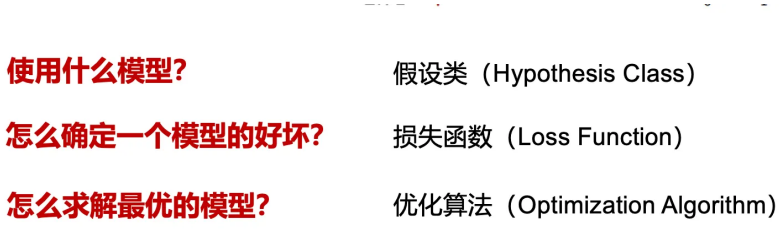


图 2.9: 线性回归与逻辑回归的逻辑框架

2.2.1　假设类

- **概念**（这里 25 秋版 ppt 似乎写的有些前后矛盾，此处已经修正）：**假设**是我们对于世界如何运行的猜想和前提。**模型**则是假设的具体数学体现，核心在于预测器 f_θ（如图2.1所示），包含了特征提取器（可以理解成函数的形式）和对应参数（如系数等）。例如我们假设 x 的某个分量与 y 呈现周期的关系，模型中的特征提取器可能要使用 $\cos x$ 做分量。

- 回归模型：

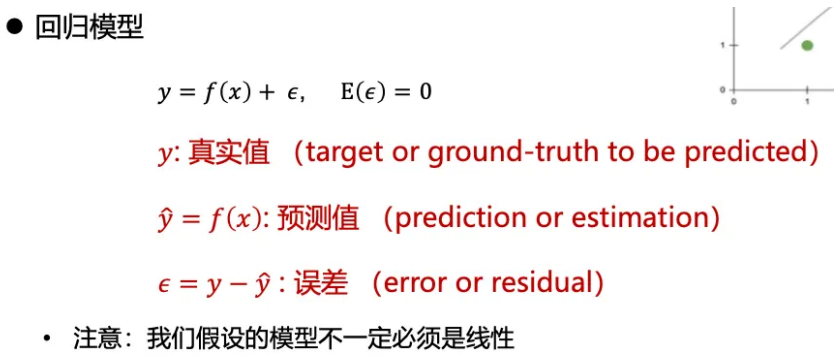


图 2.10: 回归模型

- **线性回归模型**：最简单的形如 $y = \mathbf{w}^\top \mathbf{x} + \epsilon$。但一般我们研究的线性指的是关于参数和特征提取器线性（不一定关于自变量 x 线性），一般形如

$$y = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + \epsilon$$

  其中 $\mathbf{w}$ 指的是**参数向量**（权重），模型关于这些参数是**线性的**，而 $\boldsymbol{\phi}(\mathbf{x})$ 表示**特征提取器**，将原始输入 $\mathbf{x}$ 映射到特征空间。核心特性在于偏导数 $\frac{\partial f}{\partial w_i}$ 不依赖于 w_i。这种简单形式导致其有解析解（当然一般采用后面介绍的梯度下降）。

- 一些例子：$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_d x^d + \epsilon$（多项式回归），$y = w_0 + w_1 \cos(x) + w_2 \cos(2x) + w_3 \cos(3x) + \cdots + w_d \cos(dx) + \epsilon$（含有周期特征的回归）。关于如何寻找更佳特征的方式有一门学科叫做**特征工程**。但现在**深度学习**比特征工程更佳。

拓展内容

按照严格的统计学知识，更加广义的**线性模型**指的是决策边界线性。按这个定义，**线性回归模型**和第2.3.1节中介绍的**逻辑回归模型**都属于线性模型。真正的非线性回归是含有非线性激活函数的神经网络那样，没有线性决策边界（可以处理非线性可分问题）。第三章会讲解。

2.2.2　损失函数

在第2.1.4节中我们介绍了损失函数的意义。这里介绍一下平方损失函数。

- **形式**（注意要熟悉其矩阵形式，能够熟悉最基本的矩阵求导，这在后面的反向传播算法会用到）：下图中，　的每一行表示将每一个数据点代入 $\phi(x)$，乘以 $\mathbf{w}$ 就得到预测值。

$$\begin{aligned}\text{TrainLoss}(\mathbf{w}) &= \frac{1}{|\mathcal{D}_{\text{train}}|}\sum_{(x,y)\in\mathcal{D}_{\text{train}}}(f(x)-y)^2 \\ &= \frac{1}{|\mathcal{D}_{\text{train}}|}\sum_{(x,y)\in\mathcal{D}_{\text{train}}}(\mathbf{w}\cdot\phi(x)-y)^2 \\ &= \frac{1}{|\mathcal{D}_{\text{train}}|}(\mathbf{y}-\Phi\mathbf{w})^\top(\mathbf{y}-\Phi\mathbf{w})\end{aligned}$$

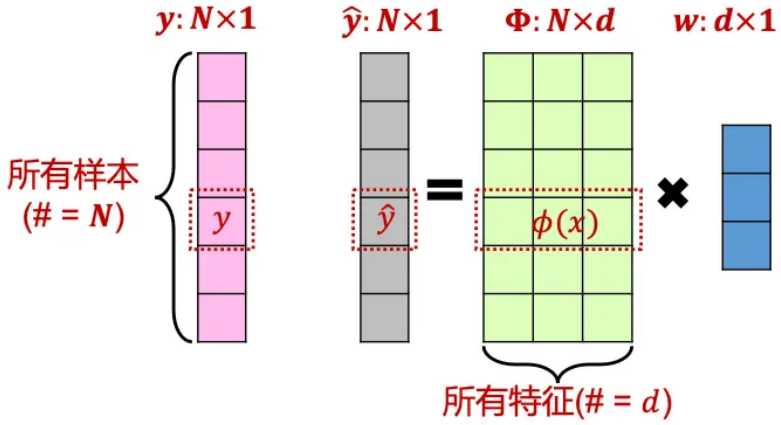


图 2.11: 平方损失函数

拓展内容

关于为什么使用平方损失函数。一方面是其可导，有下界；核心原因如图片所示。我们之前提到，**损失函数的选取依赖于所认为的数据分布**。现实中我们经常假设数据服从高斯分布。

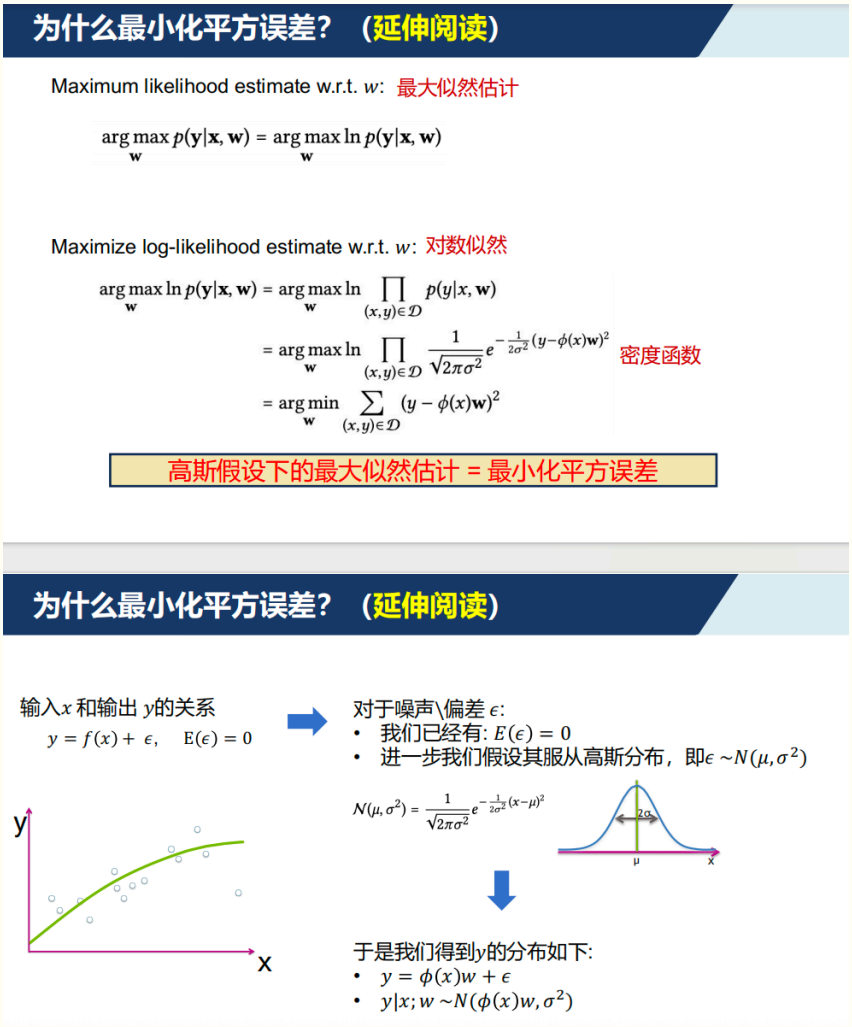


图 2.12: 使用平方损失函数的原因

当然也可以根据已知的采用一些其他的损失函数，比如绝对值损失函数。其特点是：个别数据大偏差不特别反映。但存在可能没有最优解的问题。而三次方和四次方的极值可能不是最小值点，且统计意义模糊，往往不作为损失函数。

2.2.3　优化算法

机器学习的误差来源

1. **模型误差**：模型定义与现实分布的差距。是使用线性模型、决策树、还是神经网络等，进一步的包括线性模型中特征提取器的选择以及神经网络的层数，以及对应的参数等，与现实存在误差。

2. **估计误差**：数据有限。训练样本最佳参数与总体样本最佳参数必然存在误差。可以通过高质量数据减少或者正则化技术防止过拟合等等。

3. **优化误差**：给定损失函数和有限的训练集时，寻找（在训练集上）损失最小的参数的过程（可能会陷入局部最优，以及学习率问题等等）。优化方法包括调整优化器超参数（学习率、动量），使用更先进的优化器，增加训练轮数，检查梯度。

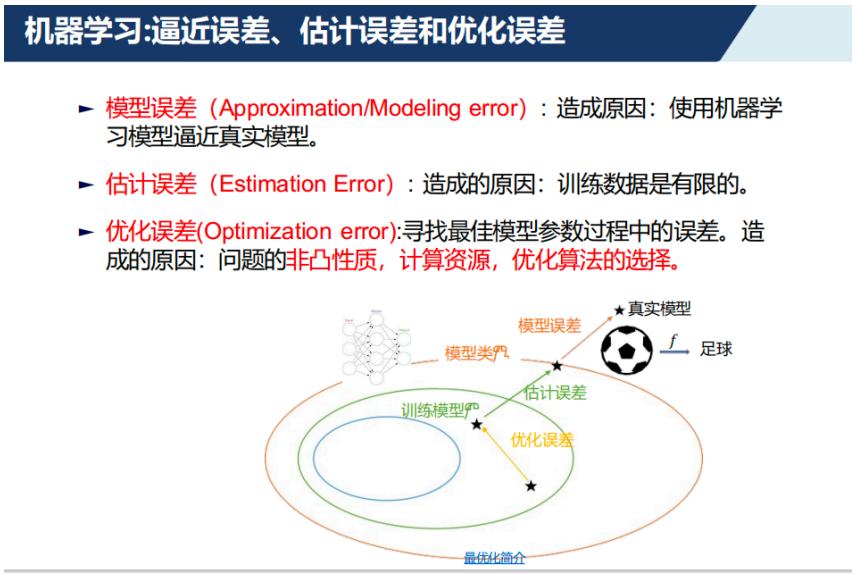


图 2.13: 机器学习的误差来源

注

损失函数的选择（例如，对于分类任务用交叉熵而非均方误差）实际上**同时影响模型误差和优化误差**（本身不直接导致）。

- 对于模型误差，给定一个模型的架构（如使用线性模型并给定特征提取器形式），损失函数不同会导致理论最优参数不同，进而影响其描述现实的能力。
- 对于优化误差，给定优化方式（包括优化器，超参数等），损失函数类型会影响了优化路径的曲率和梯度质量，进而影响了算法逼近其最优点的能力。

一种常用的优化方式：梯度下降算法

优化算法指的是在训练过程中，利用损失函数更新参数的方法。**梯度下降**是最常用的优化算法之一，它通过迭代的方式更新参数，逐步降低损失函数的值。基本步骤如下：

1. **初始化参数**：通常将权重 $\mathbf{w}$ 初始化为零或随机小值。
2. **迭代更新**：对于每个迭代（称为 epoch），计算当前参数下的梯度，然后更新参数：

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$$

   - η 是学习率（learning rate），控制更新步长的大小。
   - $\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$ 是当前点损失函数的梯度。
   - 可以理解成数列递推趋向于不动点（$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) = 0$ 的点）的过程。
3. **停止与收敛**：
   - 理论上对于凸函数，梯度为 0 时算法收敛。
   - 实际运用中一般会设定阈值/迭代次数。达到一定迭代次数或者梯度模长小于阈值时停止。

在线性回归模型中的梯度下降优化

- 首先由于线性回归的模型简单，实际上有解析解。如下图。

令　　$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) = \frac{2}{|\mathcal{D}_{\text{train}}|} \Phi^\top (\mathbf{y} - \Phi\mathbf{w}) = 0$

我们得到参数向量 w 的闭式解

$$\mathbf{w}^* = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$$

图 2.14: 线性回归的解析解

但实际上存在矩阵可能不可逆的问题。即使我们可以通过一些数学手段处理，但求高阶矩阵的逆计算量并不小，而且此方法对于非线性模型难以推广。

- 更广泛的解法是采用梯度下降。见如下 ppt。

图 2.15: 梯度下降过程

改进：随机梯度下降　该部分原先在第三章神经网络部分介绍。笔者认为放在这里更加连贯。

- 数据集很大时，考虑到硬件的存储空间有限，无法全部放入内存。使用全部训练数据会导致速度很慢。
- 所以采用每次算一小部分样本。GPU 运算时通信设备载入下一部分数据，提高了计算效率。公式如下：

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \cdot \frac{1}{B} \sum_{i=1}^{B} \nabla_{\mathbf{w}} L(\mathbf{w}; x^{(i)}, y^{(i)})$$

- 当然其学习率的调整和一般的梯度下降一样，存在一些困难。可以参见下面 Adam 部分。

关于学习率的注记

1. 学习率（`learning_rate`）在代码中经常定义成 `lr`、η、α 等。在本节优化算法中，与**步长（step_size）**是近义词。但是注意与其他语境（如第三章中的 CNN）区分，那里的**步长（stride）**是完全不同的含义。
2. 学习率大小很重要。如果学习率太大，更新步长过大，可能会跳过最小值，导致损失函数震荡甚至发散；如果学习率太小，更新步长过小，收敛速度很慢，可能需要很多次迭代才能达到最小值。

3. 定义步长的策略：

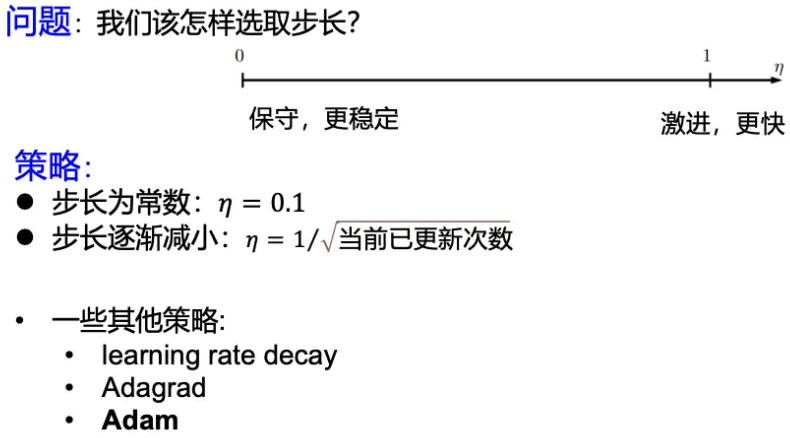


图 2.16: 学习率/步长策略

这里我们结合 AI 对 Adam 做一些简介，在后续大作业中会用到 Adam 优化器。

拓展内容：Adam 优化器（Adaptive Moment Estimation）

（拓展内容指的是优化器的具体原理（动量那些东西）。基本的（随机）梯度下降的问题需要了解，第三章 3.2 节 ppt 中出现了）
在前面（随机）梯度下降里，我们看到参数更新的核心形式都是

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \cdot g_t$$

其中 g_t 是当前（小批量）梯度。难点主要集中在两类：

1. **学习率很难选**：过大会震荡/发散，过小会收敛慢。
2. **不同参数" 合适的步长" 不一样**：有的方向梯度一直很大，有的方向梯度很稀疏、偶尔才出现。

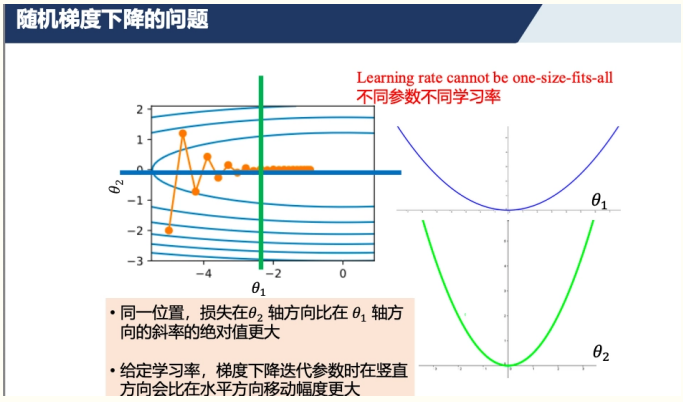


图 2.17: SGD 的问题

Adam 可以理解为两种改进思路的结合：

- **动量（Momentum）**：对梯度做” 平滑的历史平均”，减少来回震荡，让更新方向更稳定。
- **自适应学习率（RMSProp 风格）**：对梯度平方做” 历史平均”，让每个参数都有自己的步长尺度；梯度大的方向自动” 刹车”，梯度小/稀疏的方向相对走得更快。

1）核心变量：一阶矩与二阶矩
记第 t 步的梯度为 $g_t = \nabla_{\mathbf{w}} L_t(\mathbf{w})$（这里 L_t 可以是一个 mini-batch 的平均损失）。Adam 维护两组” 滑动平均”：

- **一阶矩（动量项）**：$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
- **二阶矩（尺度项，近似梯度平方的平均）**：$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$

其中 $\beta_1, \beta_2 \in (0,1)$ 是衰减系数，g_t^2 表示按元素平方。
直观理解：m_t 负责” 朝哪个方向走”（方向盘）；v_t 负责” 每个参数该走多大步”（智能刹车）。
2）偏差校正与参数更新
由于初始时 $m_0 = v_0 = 0$，前几步的滑动平均会偏小，所以 Adam 进行偏差校正：

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

最终更新为：

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

其中 ϵ 是一个很小的常数（防止除零，也让数值更稳定）。
3）优点、适用场景与一个提醒

- **省心**：默认超参数常常就能用（例如学习率 `lr=0.001`，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\epsilon = 10^{-8}$）。
- **收敛快且稳定**：相比纯 SGD，更不容易被” 锯齿状” 梯度带着震荡。
- **对稀疏梯度友好**：在 NLP/推荐等稀疏特征场景里通常表现很好。

提醒（了解即可）：在某些对泛化要求极高的任务上（比如图像分类），精心调参的 **SGD+Momentum** 有时最终测试效果会更好；但在绝大多数深度学习工程实践里，Adam/AdamW 是默认首选。
4）总结：Adam 如何” 缓解学习率问题”
Adam 把” 全局学习率 η” 拆成了：

- 一个全局的基准尺度（仍然需要选，但更鲁棒）；
- 再乘上一个按参数自适应的缩放 $1/(\sqrt{\hat{v}_t} + \epsilon)$。

因此它在很多时候**不需要像 SGD 那样反复手动试探学习率**，也能得到不错的训练曲线。

2.3 逻辑回归

2.3.1 假设类

本节的组织架构与 2.2 相同，都是假设类 + 损失函数 + 优化算法。本节会额外讨论一些逻辑回归特有的问题。

情景与原始想法

- 我们考虑这样一类问题：根据年收入，贷款金额，贷款利率等因素推断贷款违约的可能性。
- 一种简单的办法是，线性分类器。定义 score(x) = 各特征加权和，大于 0 表示会违约。(这里的 h 就是上一节提到的特征提取器)

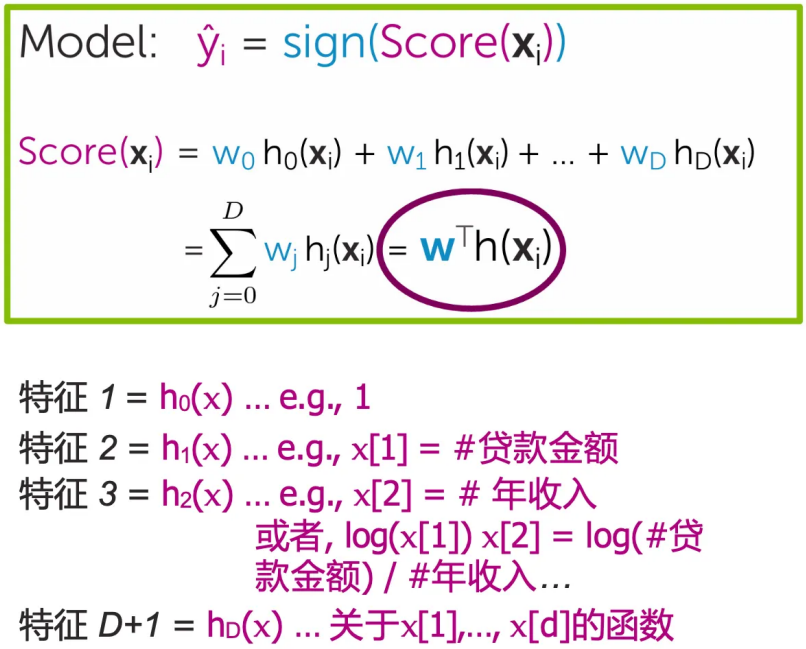


图 2.18: 线性分类器

- 进而得到一个**决策边界**（将 score 正负分开的界面）

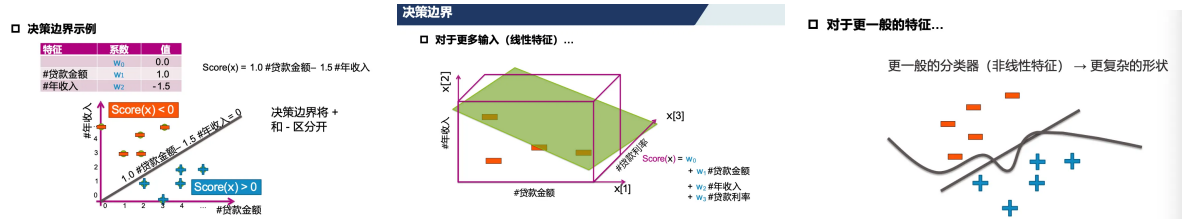


图 2.19: 决策边界

- 但是以上的预测有多确定？以及如何得到一个有广泛评判性的模型？这需要使用逻辑回归学习概率。

逻辑回归的模型与概念

- 逻辑回归的目标是从数据中学习条件概率。在训练数据上通过寻找最优参数找到最佳模型。
- 如何预测概率？一个简单的想法是利用 score(x)（取值从 $-\infty$ 到 $+\infty$），越大表示 $y=1$ 可能越大。

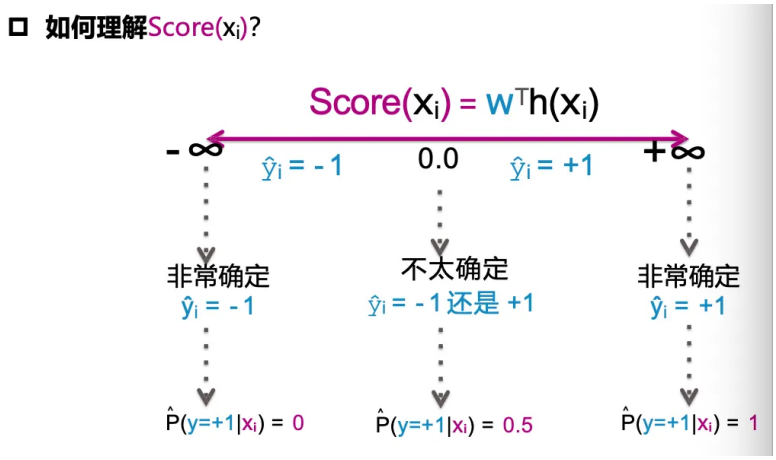


图 2.20: score 的理解

- 为了方便量化大小，我们要寻找一个将 $(-\infty,+\infty)$ 映射到 $(0,1)$ 的函数。这个函数称为**逻辑函数（Logistic / sigmoid / logit）**。其优点在于：光滑性好；0 附近敏感。

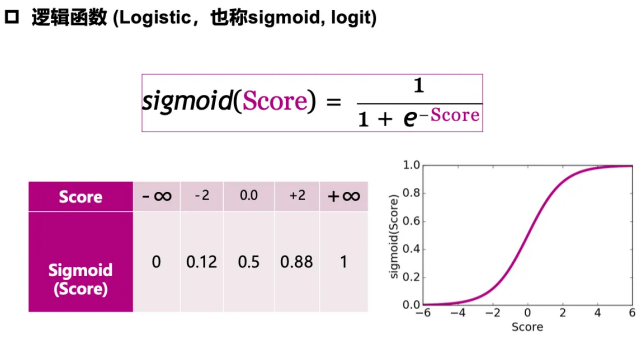


图 2.21: Logistic 函数

- 由此得到逻辑回归的模型：

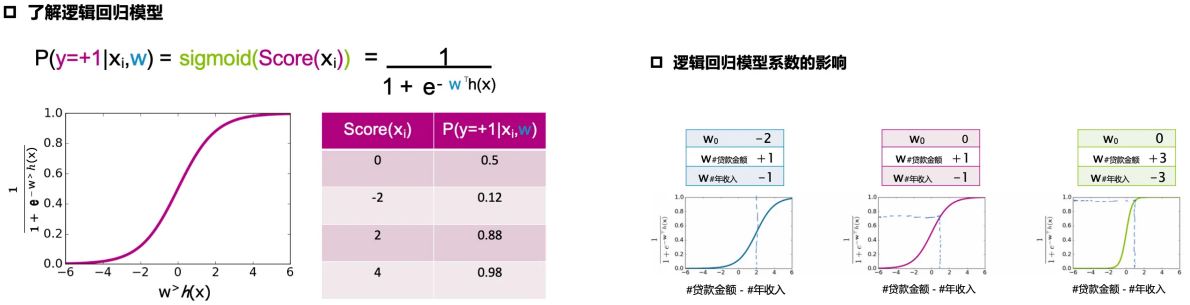


图 2.22: 逻辑回归模型

- 与回归模型的对比：

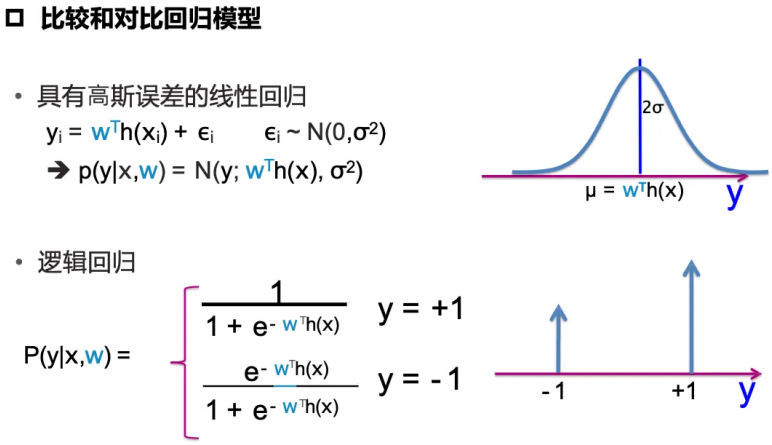


图 2.23: 逻辑回归与线性回归的对比

2.3.2 逻辑回归的损失函数与梯度优化

基本的逻辑和线性回归是相同的，只是这里我们换用了一种损失函数，然后再采用梯度下降。

- 损失函数定义时基本的想法是，我们希望实际值 $y = +1$ 时预测值 $P(y = +1)$ 大，实际值 $y = -1$ 时，预测值 $P(y = -1)$ 大。将所有样本乘起来，自然得到以下损失函数：

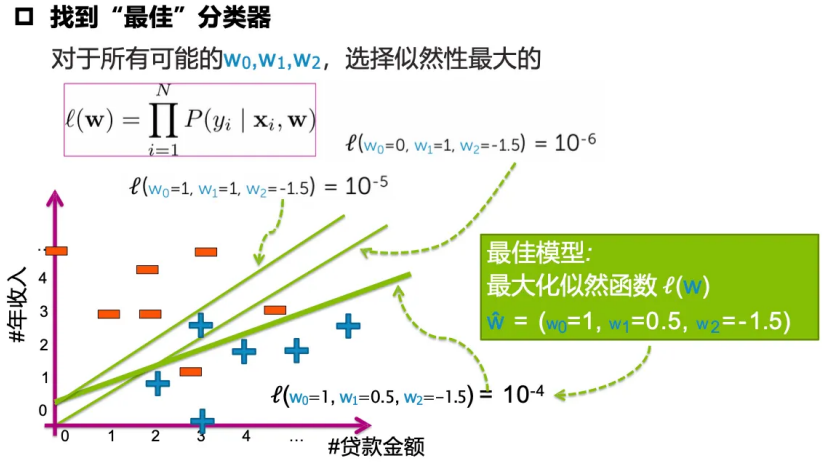


图 2.24: 逻辑回归的损失函数

- 实际中乘积运算比较复杂，我们对上述 $l(\mathbf{w})$ 取对数，也就是对数似然，记作 $ll(\mathbf{w})$；同时，这里 $l(\mathbf{w})$ 应该是越大越好，为了符合损失函数尽可能小的惯例，我们往往考虑负对数。然后再采用与线性回归相同的梯度下降。如果是直接考虑 $ll(\mathbf{w})$ 就是梯度上升。

- 具体操作如下图。表格中左上角和右下角表示实际数据对应的预测概率大，对梯度贡献小（也就是这个数据点上倾向于不需要更新）。

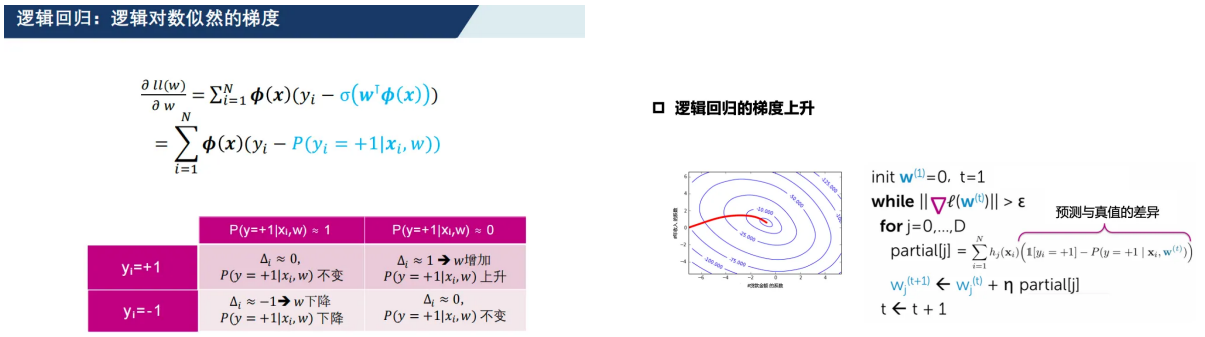


图 2.25: 逻辑回归的梯度优化

2.3.3　多分类问题

之前我们只讨论了二分类问题。实际上存在多分类问题。首先注意多分类与多标签的区别，多分类指的是”一张图片只能属于一个类别”（单选题）（类别之间互斥）所有输出概率之和为 1，多标签指的是”一张图片可以同时有多个标签”（多选题），每个输出的概率独立。

多分类的处理方式 1：One-vs-All 方案

图 2.26: One-vs-All 方案

多分类的处理方式 2：交叉熵损失

- 每一个可能的类别对应一组参数 $\mathbf{w}_i$，合起来就是下图的 $\mathbf{w}$，也就是 $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_D)^{\top}$。对单个样本 $\phi(x)$，第 i 个类别的得分 $\text{Score}_i(x) = \mathbf{w}_i^{\top}\phi(x)$。（以上就是二分类相同操作）。然后定义概率时对 sigmoid 函数做一个推广，改成 **softmax 函数**（很常用，后面还会用到）。如下图。得到每个类别的预测概率，选择其中最高的作为预测。

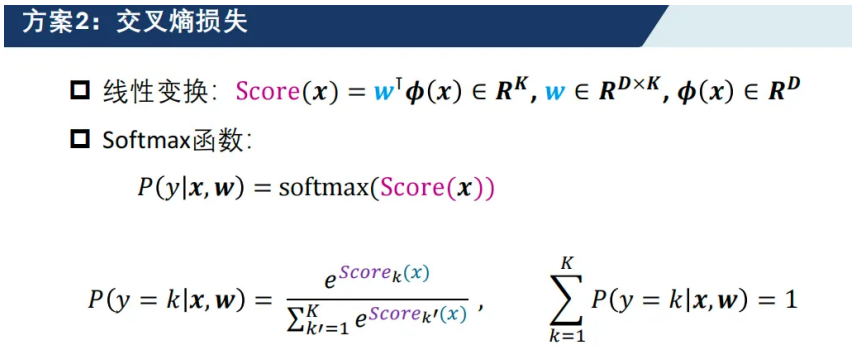


图 2.27: Softmax 函数

- 同时损失函数也要做相应的修改，改用**交叉熵损失**。对单个样本，交叉熵定义成 $L = -\sum_{i=1}^{C} y_i \cdot \log(p_i)$，（$C$：类别数量，$y_i$：真实标签的 one-hot 编码（只有真实类别为 1，其他为 0），p_i：模型预测的各类别概率）也就是之前二分类损失函数（$l(\mathbf{w})$ 取负对数）的自然推广。我们算一个小例子：

```
真实标签：y = [0, 1, 0]  （第二类）
模型预测：p = [0.1, 0.7, 0.2]
L = -[0*log(0.1) + 1*log(0.7) + 0*log(0.2)] = -log(0.7)    0.357
```

考题提示

25 秋的试卷上考过这样的一个判断题，大意是说，多分类问题中影响交叉熵损失的只有对真实标签的预测概率。如果知道交叉熵损失的表达式，很容易知道是正确的。

- 而全体样本的交叉熵就是所有样本交叉熵的求和。

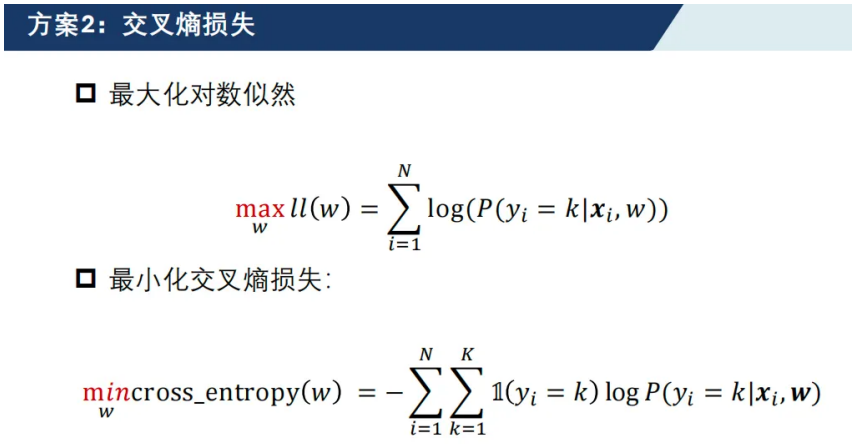


图 2.28: 交叉熵损失

两种方式的对比

增量友好：指的是增加类别时便于更新。

两种方案的对比

	One-vs-All	交叉熵损失
概率解释	独立计算	联合归一化
优化方式	分别优化	整体优化
优点	• 实现简单 • 并行训练优势 • 增量友好	• 概率一致（保证 $\sum_{k=1}^{K} P(y=k\|\boldsymbol{x},w)=1$）
缺点	• 概率不一致（可能 $\sum_{k=1}^{K} P(y=k\|\boldsymbol{x},w)\neq 1$） • 决策边界模糊	• 类别不平衡敏感

图 2.29: 两种多分类方式的对比

2.3.4　过拟合

定义：随着训练轮次增加，出现了在训练集上表现很好，但在测试集上表现不好。

分类中的过拟合

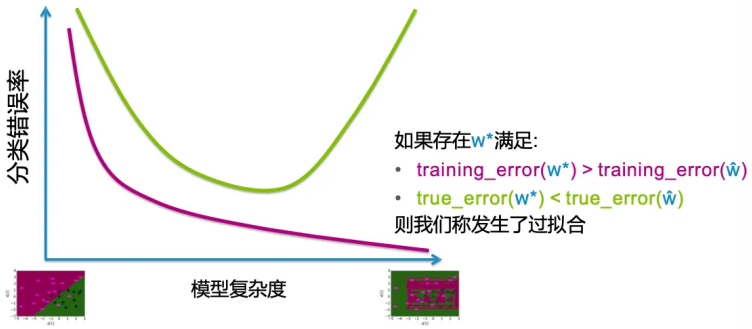


图 2.30: 过拟合现象

成因

通常，过度拟合与非常大的估计系数 $\hat{\mathbf{w}}$ 有关。

逻辑回归中过度拟合的微妙后果

图 2.31: 过拟合原因

- 在第2.2.1节中我们提到，逻辑回归的损失函数虽然关于其参数求导并不是常数，但属于广义的线性模型（GLM），也就是具有线性可分的决策边界。如下公式所示：

$$P(y=1|\mathbf{x})=0.5 \quad \Longleftrightarrow \quad \mathbf{w}^{\top}\mathbf{x}+b=0$$

- 此时存在**无限多组解**都能达到训练误差为 0。由于数学最大似然估计的机制，系数会不断增大。而在现实中几乎不会线性可分，也就是有少量异常点。模型仍会试图用极大系数来拟合。此时，边界会变得十分陡峭，对输入数据的微小变化极其敏感；同时测试集上对靠近边界的样本预测过于自信。预测误差变大。

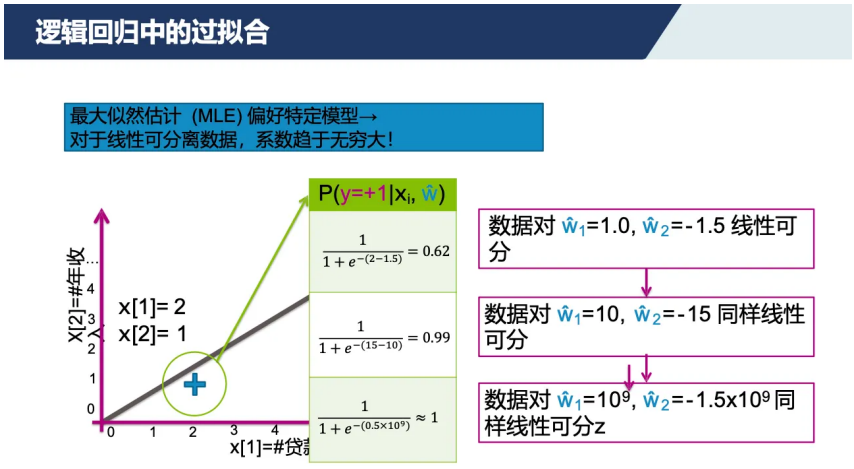


图 2.32: 系数增大导致过拟合

缓解办法

整体的缓解思路：修改损失函数，使其随着系数的增大而增大（加入惩罚项）。

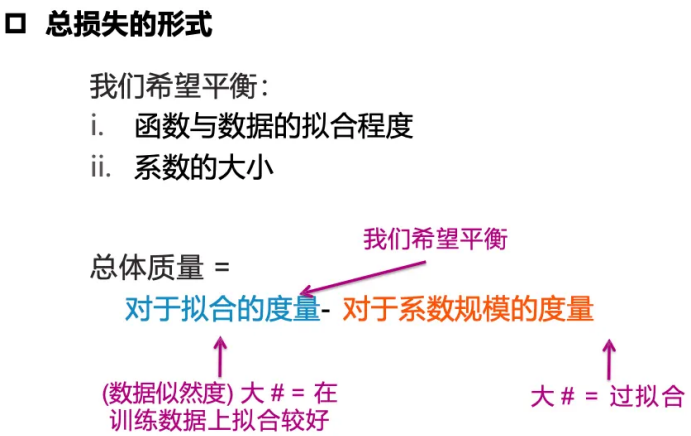


图 2.33: 正则化缓解过拟合

- L2 正则化：

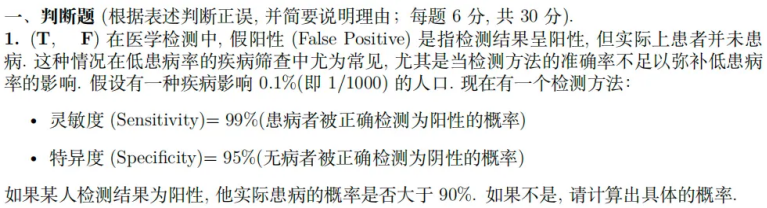
图 2.34: L2 正则化

- L1 正则化：

图 2.35: L1 正则化

2.3.5　如何评估分类模型的好坏

如之前所述，本部分虽然名义上是评测逻辑回归的指标，但从知识上看感觉是一个纯统计学的指标，没涉及什么机器学习的知识。因此个人感觉**不太可能考（仅供参考）**。25 春的模拟题上出现类似背景的题目，但感觉没学过完全可以做。如图。解题方法个人认为与其拿贝叶斯变来变去，不如把那个表格画出来按照已知信息求出所有的占比再带入问题。



一、**判断题** (根据表述判断正误, 并简要说明理由；每题 6 分, 共 30 分).
1. (**T**, **F**) 在医学检测中, 假阳性 (False Positive) 是指检测结果呈阳性, 但实际上患者并未患病. 这种情况在低患病率的疾病筛查中尤为常见, 尤其是当检测方法的准确率不足以弥补低患病率的影响. 假设有一种疾病影响 0.1%(即 1/1000) 的人口. 现在有一个检测方法：

- 灵敏度 (Sensitivity)= 99%(患病者被正确检测为阳性的概率)
- 特异度 (Specificity)= 95%(无病者被正确检测为阴性的概率)

如果某人检测结果为阳性, 他实际患病的概率是否大于 90%. 如果不是, 请计算出具体的概率.

图 2.36: 相关例题

以下部分以一个贷款诈骗的分类预测问题引入相关问题。由于急着和朋友开黑，直接将原先的笔记和对应课件喂给了 GPT，在此基础上做了修改。

贷款诈骗例子与混淆矩阵

为了统一口径，这里把”贷款诈骗”视为**正类（Positive）**，”正常贷款”视为**负类（Negative）**。分类器给出预测后，我们用混淆矩阵统计四种情况：

	预测为正（诈骗）	**预测为负（正常）**
真实为正（诈骗）	TP（真阳性）	FN（假阴性/漏报）
真实为负（正常）	FP（假阳性/误报）	TN（真阴性）

表 2.2: 混淆矩阵

记忆法

- 前面的 T/F 表示”预测是否正确”（True/False）
- 后面的 P/N 表示”预测为正/负”（Positive/Negative）

常用指标（Accuracy / Precision / Recall / F1）

- **Accuracy（准确率）**：整体预测正确的比例

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

类别严重不平衡时（比如诈骗样本极少），Accuracy 可能会”看起来很高但没意义”。

- **Precision（精确率/查准率）**：预测为诈骗的样本里，有多少是真的诈骗

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall（召回率/查全率，也叫 TPR）**：真实诈骗样本里，有多少被成功识别

$$\text{Recall} = \text{TPR} = \frac{TP}{TP + FN}$$

- **F1 分数**：Precision 与 Recall 的折中（调和平均）

$$F_1 = \frac{2\,\text{Precision}\,\text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

贷款诈骗识别的直观含义：

- 更看重**少漏报**（FN 代价高）→ 重点关注 Recall/TPR
- 更看重**少误报**（FP 代价高）→ 重点关注 Precision
- 两者都要兼顾 → 常看 F1

ROC 曲线与 AUC

逻辑回归等模型通常输出”属于正类的分数/概率”。当我们改变分类阈值 τ（概率达到 τ 就被认定为诈骗）时，TP/FP/FN/TN 会随之变化。

- **TPR（真正例率）**：

$$\text{TPR} = \frac{TP}{TP + FN}$$

- **FPR（假正例率）**：真实为负的样本中，被错判为正的比例

$$\mathrm{FPR} = \frac{FP}{FP + TN}$$

以 FPR 为横轴、TPR 为纵轴，把不同阈值下得到的点连起来，就得到 **ROC 曲线**。其中，$(0,0)$ 表示阈值极高，几乎都预测为负类；$(1,1)$ 表示阈值极低，几乎都预测为正类。这两个点在所有曲线中都会出现。

AUC（Area Under the Curve） 是 ROC 曲线下的面积。一个模型对应一条 ROC 曲线（阈值从 0 到 1），AUC 用一个数概括模型在”所有阈值下”的区分能力：

- AUC 越接近 1 → 区分能力越强
- AUC = 0.5 → 接近随机

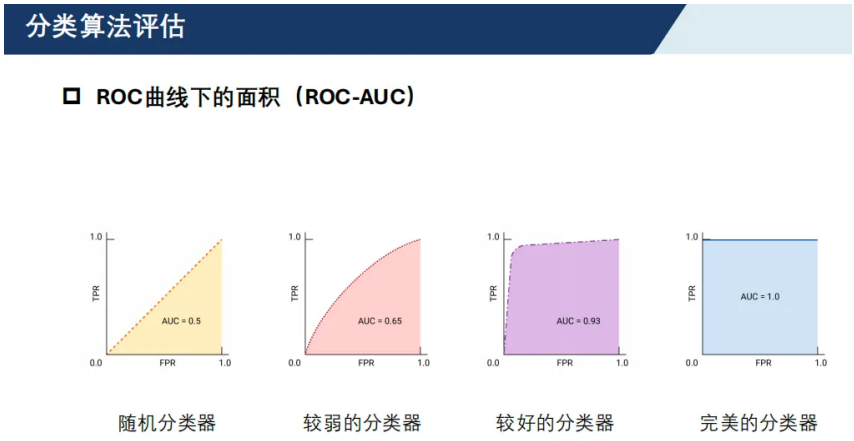


图 2.37: ROC 曲线

使用 AUC 的优势在于，对类别不平衡不敏感。

- 在数据集中正负样本比例不平衡时，准确率（Accuracy）可能误导（如 99% 的准确率可能只因多数类占优）。
- AUC 基于 TPR 和 FPR，这些指标只关注各自类别的样本，因此对类别分布变化不敏感，能更公平地评估模型。

第三章　神经网络基础

3.1　前馈神经网络

3.1.1　神经网络基本介绍

动机

（个人觉得课件上那个图容易引起误解。如下介绍。）

我们的核心任务仍然是学习一个从输入到输出的函数（如第2.1.1节中的图片所示）。第二章中我们介绍了线性模型。但 $y = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + \epsilon$ 本质是**在特征空间 $\phi(\mathbf{x})$ 上做线性回归**。假设我们选择了一组固定的基函数 $\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_m(\mathbf{x})$，那么模型能够表示的所有函数集合是：

$$\mathcal{F} = \left\{ f(\mathbf{x}) = \sum_{j=1}^{m} w_j \phi_j(\mathbf{x}) + \epsilon \;\middle|\; w_j, \epsilon \in \mathbb{R} \right\}$$

它的表达能力完全由 ϕ 决定。这是一个**有限维线性空间**。如果目标函数 f 不在这个空间中，那么无论我们如何优化权重 $\mathbf{w}$，都无法精确表示 f（只能逼近）。

之前我们的方法是采用**特征工程**（设计基函数 $\boldsymbol{\phi}(\mathbf{x})$）或核技巧（隐式定义 $\boldsymbol{\phi}(\mathbf{x})$）。这很依赖于经验和人工。而神经网络的强大之处在于，它通过非线性激活函数的复合，能够自动学习特征表示，从而用更少的参数表示更广泛的函数类。

一个实际的例子是课件上的图像识别任务：

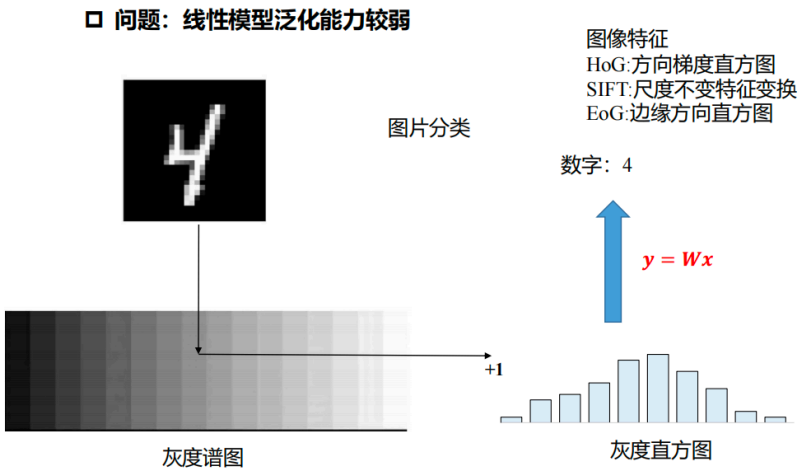


图 3.1: 图像识别任务动机

采用线性模型自然的设计就是 $y = Wx$。但不同图片中数字”4” 的写法、位置都有很多的不

同。模型被迫将同一个数字的所有不同写法都映射到同一个模板上。最终，它学到的模板是这个数字所有训练样本的” 平均形象”。下图中数字”4” 的可视化结果就是所有训练集里”4” 的叠加平均，导致其轮廓模糊、特征不鲜明。也就是上图所说的**线性模型泛化能力弱**。

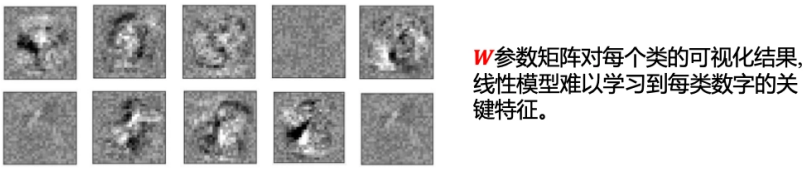


图 3.2: 线性模型的结果

这就导致，在线性模型中，一个特征（比如”4” 的直角）只有在它出现在训练模板的精确位置时，才会被强烈激活。如果测试图片中的”4” 这个直角稍微偏左或偏右，它与模板的匹配度就会急剧下降，从而导致分类错误。而实际上人识别数字的原理，是按照其**轮廓特征**确定的。用之前” 学习从输入到输出的函数” 这种观点看，我们要学习到的函数对位置不敏感、关注轮廓特征。这就是后面**卷积神经网络**设计的动机。

神经元模型

这里课件上的图片很清楚，小问题是下图应该是 $-b$ 而不是 $+b$。

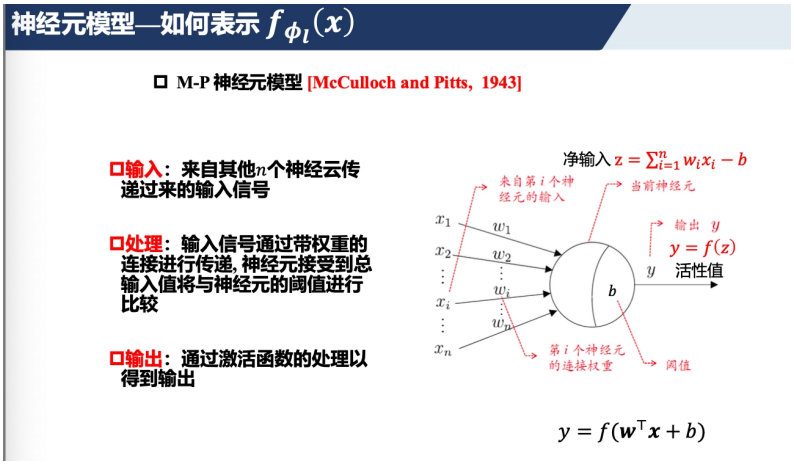
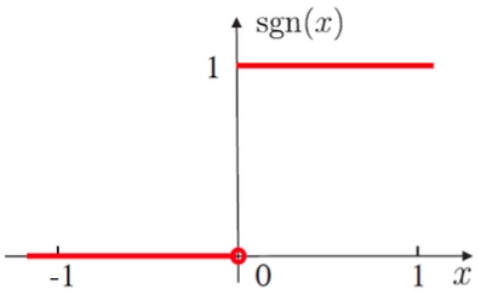


图 3.3: 神经元模型

这是一个与阈值 b 比较的过程。例子中 f 为阶跃激活函数。这种设计思路源于生物体内原始的神经元的模型。当然后文将其推广为偏置就通用写成 $+$。

调整的核心是调整上述的 f，称为**激活函数**。常用的激活函数：

1. **阶跃函数**。模仿人类神经元，但不连续，不光滑，不易优化。



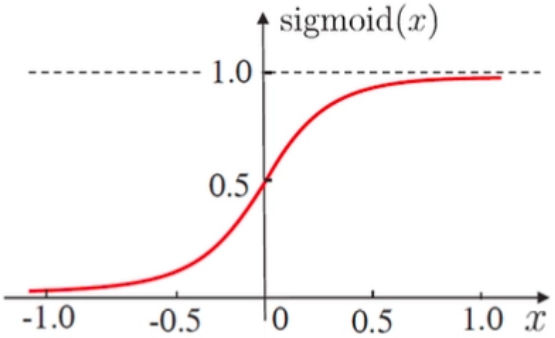
$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

(a) 阶跃函数

理想激活函数是阶跃函数
0表示抑制神经元；1表示激活神经元

图 3.4: 阶跃函数

2. **Sigmoid 函数**。但也存在一些缺点，如非零中心（恒正），$|x|$ 很大时饱和等。



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

(b) Sigmoid 函数

图 3.5: Sigmoid 函数

3. **ReLU 函数**（更贴近人类的思考方式，且梯度很好算。同时 x 前面可以加系数，可以设计和改进。）

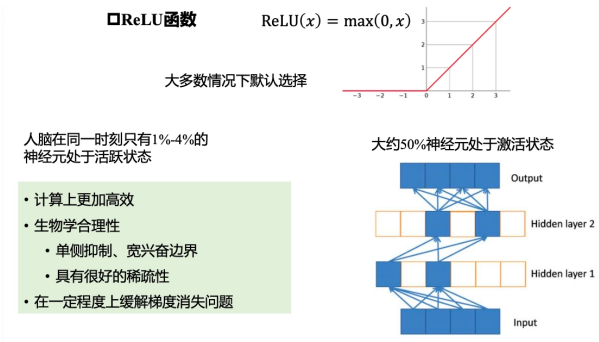


图 3.6: ReLU 函数

补充：在第3.3节深度神经网络中会遇见，Sigmoid 函数导数为 $\sigma(x)(1-\sigma(x))$（σ 函数表示 sigmoid 函数），最大梯度值仅为 0.25，反向传播时梯度会快速衰减。类似的双曲正切函数 $\tanh(x)$ 虽有所缓解，但在 x 较大时也是同样问题。这会导致深度神经网络中的梯度消失问题。此时引入 ReLU（$x > 0$ 时梯度恒 1）以及其改善可以缓解这一问题。因此 ReLU 及其变体在深度学习中十分常用。

其一个变体是 **Swish 函数**。它具有良好的可调节性，在多任务性上表现良好。

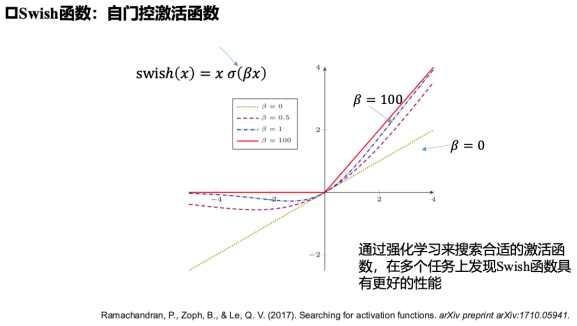


图 3.7: Swish 函数

3.1.2　感知机与全连接神经网络

感知机

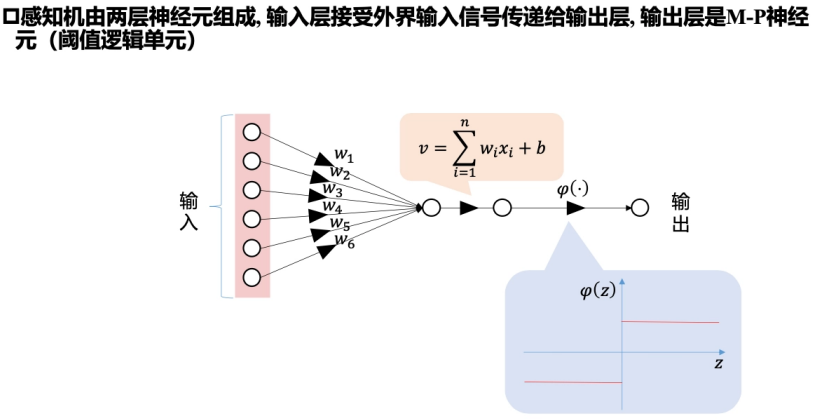


图 3.8: 感知机模型

1. **输入**：接收多个外部信号（$x_1, x_2, \ldots, x_n$）。比如，在判断” 今天是否去打网球” 的问题中，输入可以是” 天气 = 晴、温度 = 热、湿度 = 高”。

2. **加权求和**：每个输入信号都有一个权重（$w_1, w_2, \ldots, w_n$），代表这个信号的重要性。感知机将所有输入乘以其权重后加起来，再加上一个偏置项（b），形成一个总输入 $v = \sum w_i x_i + b$。偏置项可以理解为启动这个神经元的难易程度。

3. **激活函数**：对这个总输入 v 进行一个简单的变换。最经典的感知机使用阶跃函数作为激活函数（$\varphi(z)$）。

注

注意：感知机（不加定语指的是图片中这种单层的结构）本质上是一个**二分类线性分类器**。其本质还是一个线性模型。多层感知机堆叠在一起就会成为处理线性不可分问题的神经网络。

神经网络基本介绍

- **定义**：神经网络是由具有适应性的简单单元组成的广泛并行互联网络，它的组织能够模拟生物神经系统对真实世界物体所作出的反应。从结构上，**神经网络是由大量感知机（或类似感知机的单元，如后面会介绍的 CNN 等）相互连接而成的网络**。其中，由上图那样的多个感知机堆叠在一起得到的神经网络称为**全连接神经网络（又称多层感知机，MLP）**。

- **知识学习的地方**：参数（机器从数据中学习的经验，设计者不直接参与），连接方式（设计者从中得到的经验，如减少一些冗余的神经元）。

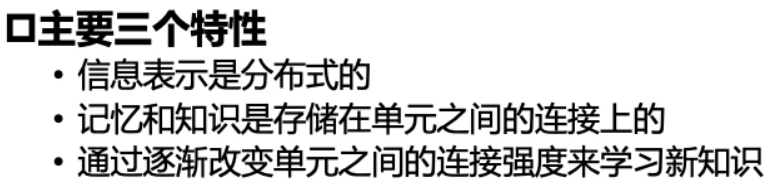


图 3.9: 知识学习

- **网格结构**：

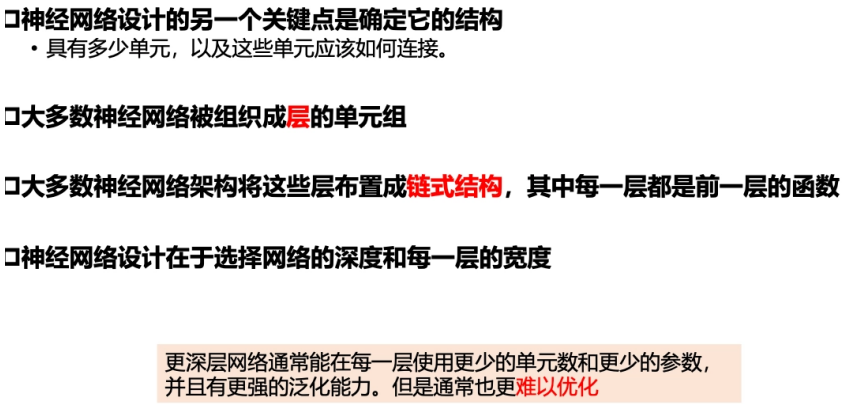


图 3.10: 网格结构

全连接神经网络（MLP，多层感知机）

单层感知机本质是线性模型，无法处理线性不可分问题。多层解决了这一问题。

隐藏层（hidden layer）是对上一层的输出做组合（加权求和）和变换（过非线性激活函数）。所谓多层，就是在输入层与输出层之间多加了一些隐藏层（单层感知机只有一个隐藏层）。

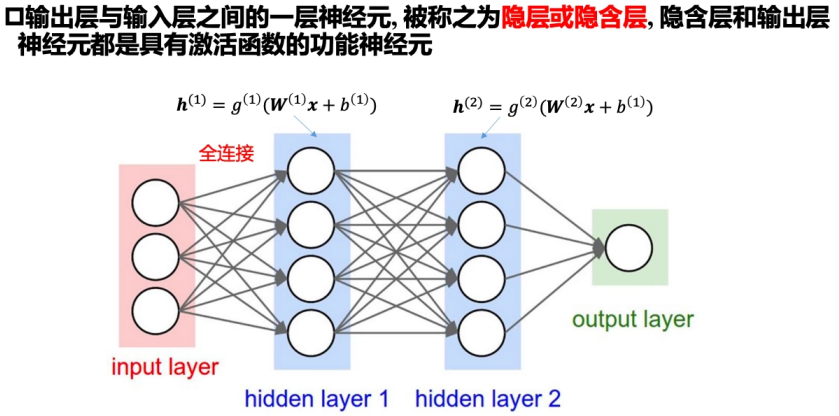


图 3.11: 隐藏层

- **应用举例：异或问题**。与、或、非问题线性可分，一般的感知机学习就可以得到权值向量，但异或问题非线性可分 → 加一个隐藏层得到分类。如下图所示，$[0,0],[0,1],[1,0],[1,1]$ 分别被映到 $[0,0],[1,0],[1,0],[2,1]$。实现了将 $[0,1]$ 和 $[1,0]$ 映射到相同点，三个点自然可以线性可分。

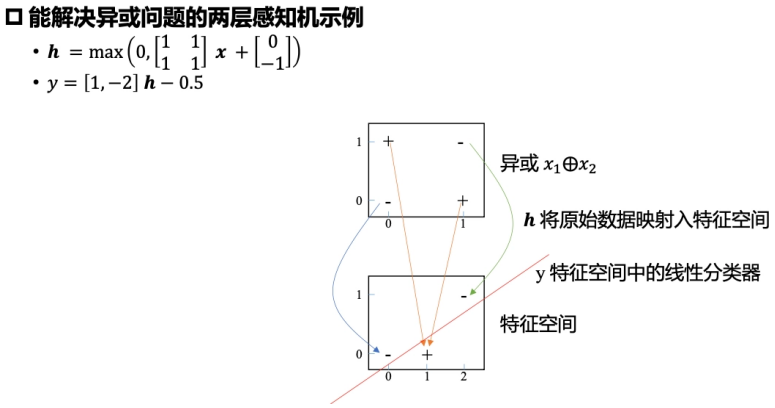


图 3.12: 异或问题

- 但 MLP 也存在一些问题：**参数量巨大**。

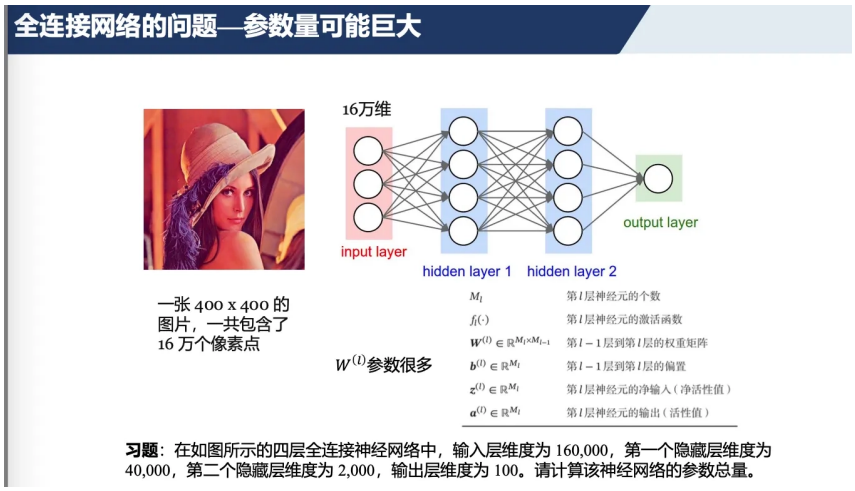


图 3.13: MLP 参数量问题

注意参数量的计算要学会。考试中会考察。本文档的附录部分会和 CNN 部分一起作讲解。

3.1.3 万能近似定理

本节的定理讲的是，从理论上可以证明，神经网络可以近似所需要学习的函数。也就是理论上神经网络可以完全替代前面的特征工程和核技巧。书的 3.1.3 给出了数学化的表述。

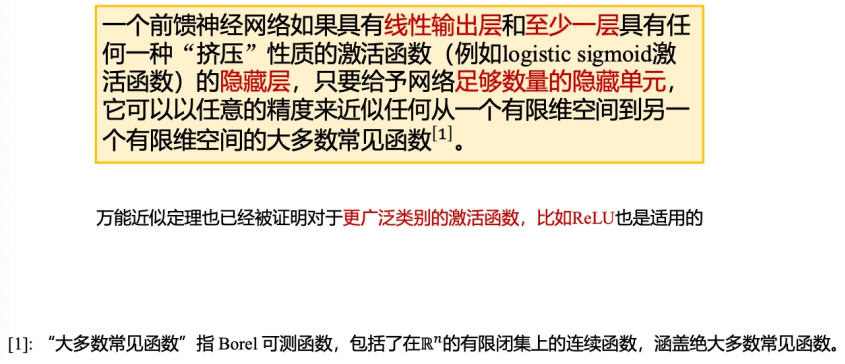


图 3.14: 万能近似定理

- ”足够数量”是多少？指数级。但也可以通过增加深度减少（如下图右图，fully connected 的，深度（层数）是 3 不变，参数量增加（也就是宽度（每层的神经元）增加）并不影响效率，但 convolutional（卷积）11 层效果随尺寸明显增加）。

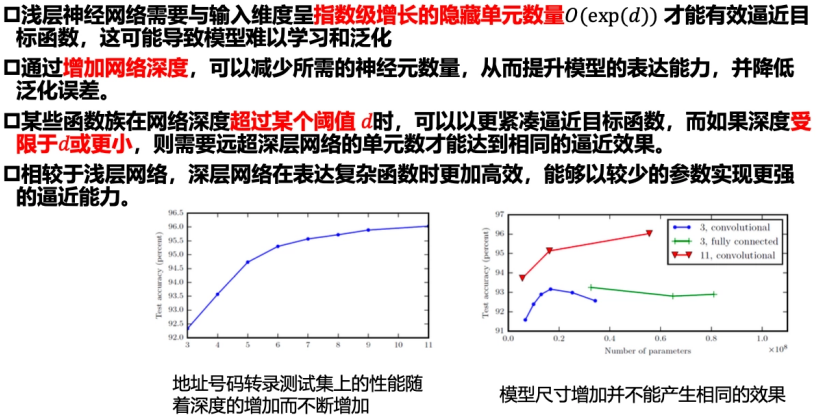


图 3.15: 宽度与深度

- 但本定理也有局限性。只证明存在，没有给出算法。

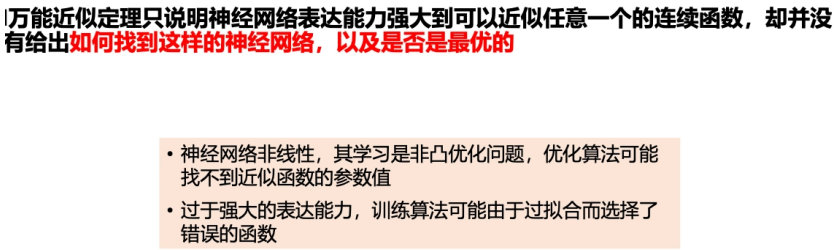


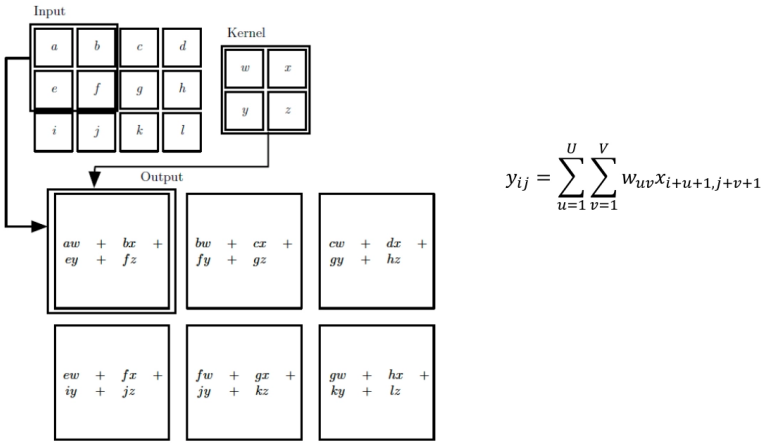
图 3.16: 万能近似定理的局限性

3.1.4　卷积神经网络

卷积的基本概念

在全连接神经网络中，存在参数量巨大的问题。卷积让我们能用很少的参数（一个小的核）来处理很大的输入，因为同样的模式可能在输入的不同位置重复出现。

计算方法：核相当于一个放大镜，在输入的信息矩阵上的移动，将每一块输出成一个元素。例子：



$$y_{ij} = \sum_{u=1}^{U} \sum_{v=1}^{V} w_{uv} x_{i+u+1, j+v+1}$$

图 3.17: 卷积运算

注：

1. 实际中可以改变步幅，也就是不同的移动步长。不说默认是 1。

2. 不同的卷积核可以得到不同的特征。可以理解成，每个卷积核关注点不同，例如一个图片，有些关注色彩，有些关注对比度等等。综合不同的核可以得到一个新图。MLP 中向量的每一个维度，就对应于 CNN 每一个卷积核处理图片得到的一个维度。后面把二者称为**特征维度**。

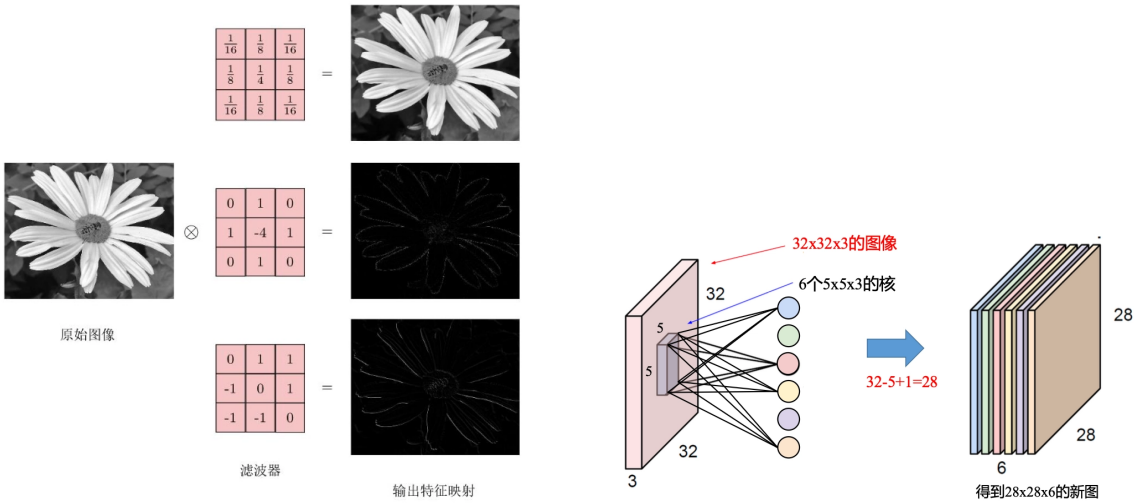


图 3.18: 卷积核与特征图

3. 卷积会使得原输入矩阵的规模减少 → 核不能设计的太大。

填充

填充（Padding）。步长不是 1 时可能无法恰好满足边界，同时会导致输出维度小于输入维度。在周围补几圈 0 来解决这两个问题。填充个数（实际是圈数）：（推导方式用输出尺寸公式就可以）：$(N + 2P - F)/S + 1 = N$，解 P。

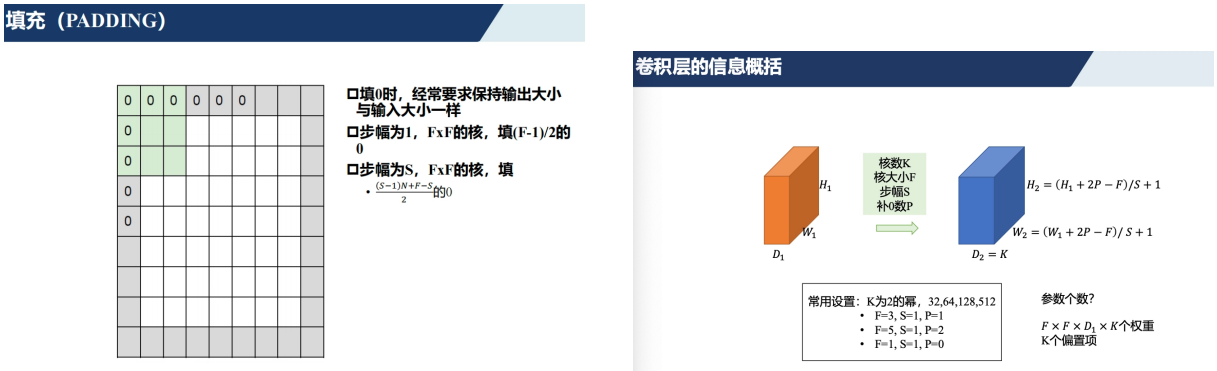


图 3.19: 填充与输出尺寸计算

卷积神经网络（CNN）的特点

本节涉及的网络名称有点多，梳理一下逻辑关系。后面还会提到：

- **前馈神经网络**：指的是一种最基本的网络结构范式。它的核心特征是：信息流从输入层到输出层**单向**、**无反馈**、**无环**地向前传播。

- 它包括多种实现形式，最经典的是**全连接神经网络**（又称为**多层感知机**（MLP）），由多个全连接层堆叠而成的网络结构（最简单最原始的是单层的感知机模型）。还包括**卷积神经网络（CNN）**、Transformer（第五章介绍）等。

在 MLP 的基础上做一些变化：

1. **稀疏连接**：每个输出单元只与输入中的一个局部区域（感受野）相连接（下图上方是卷积神经网络，下方是全连接）。

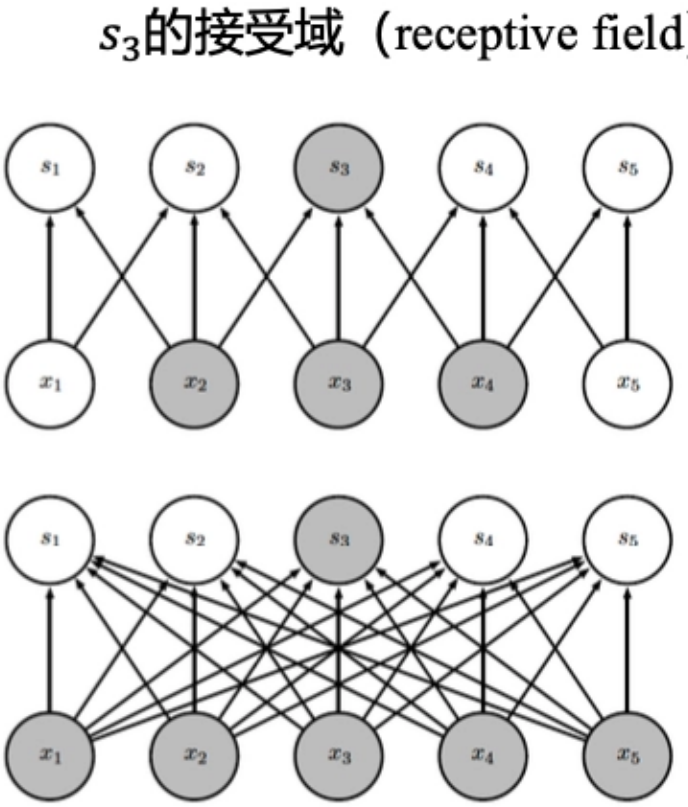
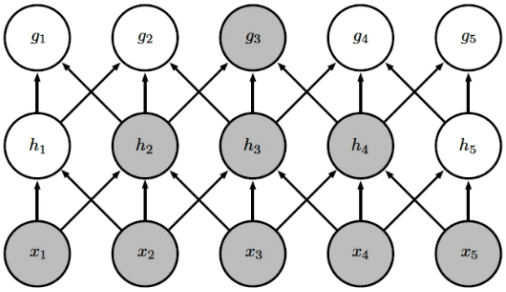


图 3.20: 稀疏连接与感受野

随着网络层次加深更深层次的单元感受野会变大，使得网络既能学习到局部的低级特征也能学习到全局的高级特征。



更深的层中的单元可以间接地连接到全部或者大部分输入图像

图 3.21: 感受野的扩大

2. **参数共享与平移不变性**：卷积核的权重在整个输入上是共享的 → 可以检测同一特征在图像不同位置的出现 → 减少参数数量同时提高模型统计效率。由于参数共享，输入图像发生平移时，输出图像的特征图也会发生相应的平移。对图像识别的任务至关重要。

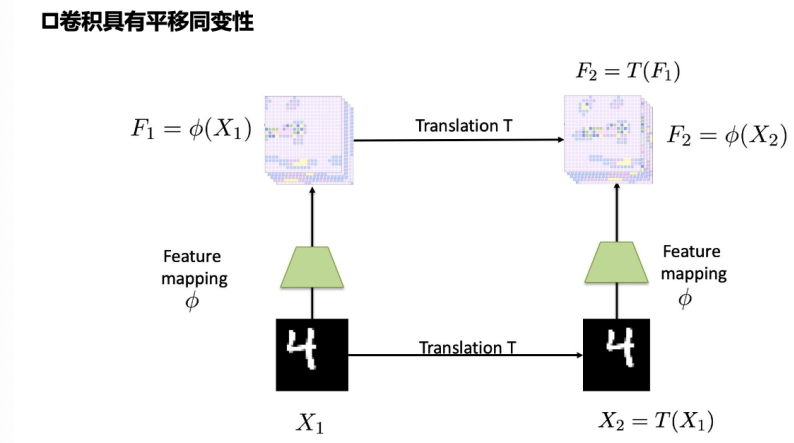


图 3.22: 参数共享与平移不变性

池化

- 卷积层之后会引入一个池化层（下采样层）用于减少特征图的尺寸，从而减少后续层的参数量和计算量。

- 核心思想是用相邻区域的总体统计特征代替该区域的所有细节信息。方式有：

1. 最大池化：取相邻区域的最大值。

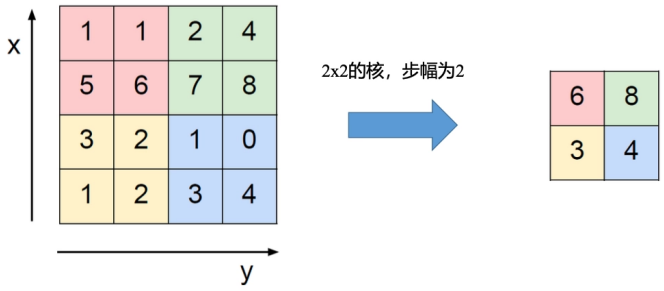


图 3.23: 最大池化

2. 平均池化：取相邻区域的平均值。

- 池化的作用：对局部平移更加稳定。

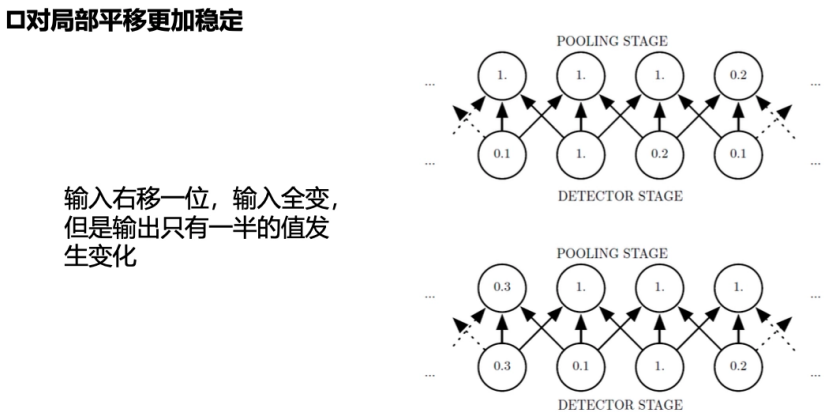


图 3.24: 池化的作用

- 输出尺寸：

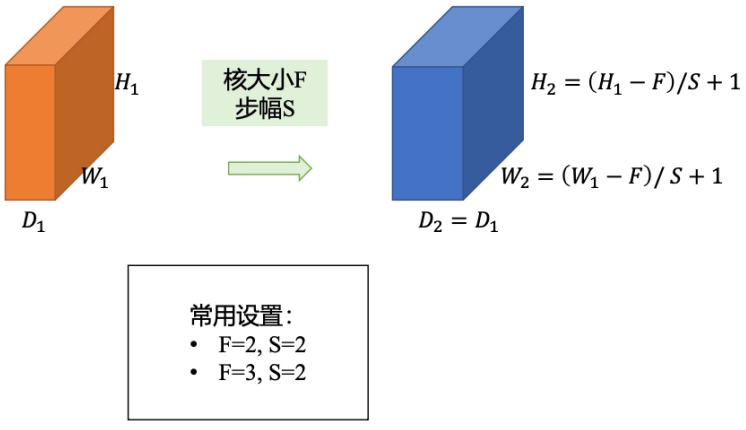


图 3.25: 池化层输出尺寸

- 实例：**LeNet-5**。实际的神经网络就是这样 CNN 和 MLP 的组合。关键点在于关注每层参数量的计算，要掌握，会考察这种参数量的计算，也是对整个流程的一个复习。

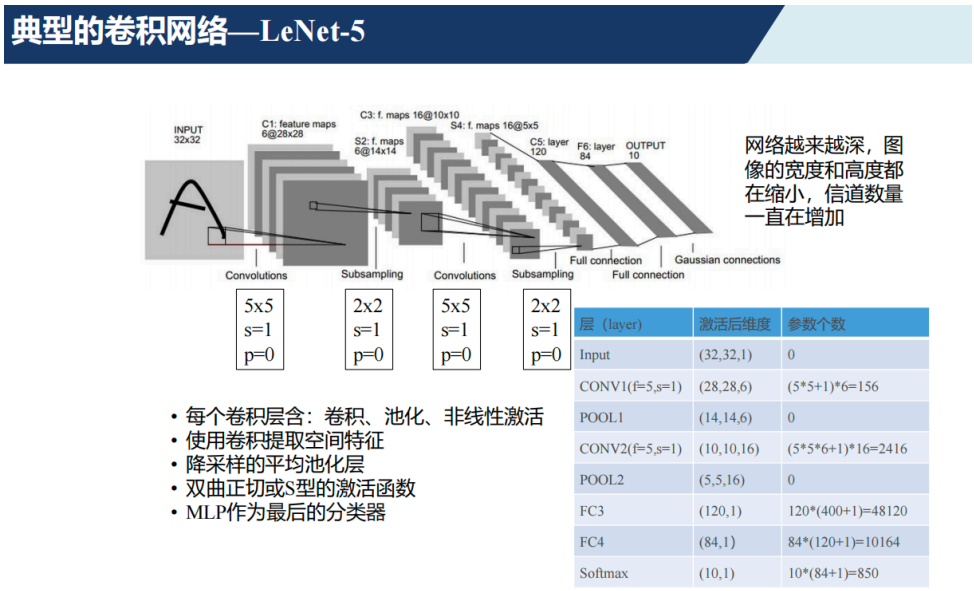


图 3.26: LeNet-5 架构

具体架构上与 MLP 的对比

- **MLP**：输入层（向量）→ 隐藏层 1（全连接 + 激活）→ 隐藏层 2（全连接 + 激活）→ 输出层（如 Softmax）。
- **CNN**：输入层（图像）→ 卷积块（卷积 + 激活 + 池化）→ 卷积块（卷积 + 激活 + 池化）→ 全连接层（或池化 + 输出）。
- 一个卷积核大致对应于 MLP 的一个神经元，卷积块（多个卷积核）的卷积计算过程就对应于 MLP 的组合（加权求和）过程。池化是 CNN 特有的下采样过程。

3.2 神经网络优化

老师在课上提醒：注意 ppt 中有不少符号混用的情形，教材和 ppt 也有所不对应。25 秋 ppt 中改了部分错误，但未全部修改，存在前后概念不一致的问题。（当然在后续授课中在不断改进）

本节很偏计算数学。目的在于实现本课程的” 数学原理” 部分。工程上实现深度学习模型的搭建和训练并不需要本部分知识（直接调用相关的库就可以）。但是原理层面应该知道，以及**反向传播算法可以说是考试的必考点**。

3.2.1 反向传播算法

简介

神经网络的训练优化的逻辑和之前的线性模型基本是一致的，最主要的方法就是梯度下降。但由于神经网络往往设计多个隐藏层，每个隐藏层又包括组合和变换（参见第3.1.2节），因此梯度的计算涉及复合求导，略显复杂。本节主要从两层隐藏层神经网络（MLP 架构）的梯度计算学习原理（实际上其他的多种神经网络架构包括 CNN，后续的 GNN，Transformer 等等都常采用反向传播算法）。笔记中与课件一致，保留了每层间一般维度的计算（也就是隐藏层的输入输

出用 n 维 m 维表示）。考试中一般只会考察 2 维的计算。对于对含大量字母计算头晕的同学可以直接看后面的 2 维计算例子即可。

- 基本计算逻辑：

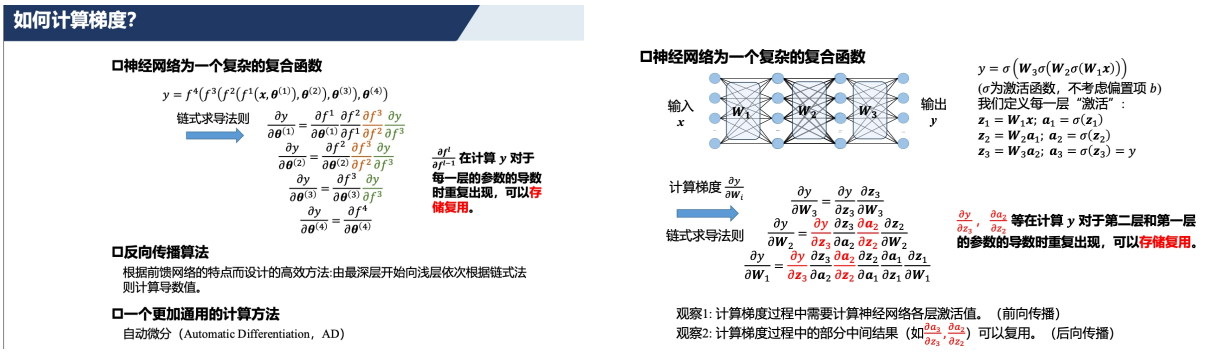


图 3.27: 反向传播的基本计算逻辑

- 由以上的求导过程看出，前项的梯度计算需要后项。这也就是反向传播的算法思想。**反向传播是一种高效计算神经网络中损失函数对所有权重参数梯度的方法**。它利用链式法则，从输出层开始，逐层向后传播误差，计算每一层参数的梯度。

- 前向与反向的对比：

  - **前向传播**：是推理/预测的过程。
    - 方向：从输入层 → 隐藏层 → 输出层
    - 内容：携带的是数据信号。输入数据 x 经过层层加权求和与激活函数，最终得到预测值 $\hat{y}$。
    - 目的：计算当前的损失值 $L(y, \hat{y})$。
  - **反向传播**：是学习/训练的过程。
    - 方向：从输出层 → 隐藏层 → 输入层。
    - 内容：携带的是误差信号（所谓误差就是梯度，误差只是一个叫法。ppt 中取了简单的平方损失函数，得到了误差为最简单的相减的情形）。损失函数计算出的总误差，被分解并分配回每一层、每一个神经元。
    - 目的：计算损失函数对于每一层参数（W, b）的梯度。

- 反向传播并不是独立进行的，它严重依赖于前向传播保存的中间结果。这是一个” 正向计算，反向求导” 的高效过程。同时，在实际计算中，**会保留某些中间计算的结果存储复用**以提高计算效率。这在后面的实际计算中会看到。

实际计算

ppt 中已经写了明细的数学推导。如下图。此处只呈现具体的计算，实际的小例子我们放在附录。再次提醒，如果实际的公式推导有理解障碍，可以先看后面的小例子，考试难度基本上就是小例子的难度（25 春秋时都会允许带计算器）。

注记：

- 建议自己一步一步用复合求导手推一下，理解各个字母的逻辑关系。注意图片中的一些 $e_j^{(2)}$ 之类的数表示后面会反复用到的结果（是复合求导的某些项，往往展开时采用类似于以下的形式

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w}$$

  然后将 $\frac{\partial L}{\partial z}$ 存储起来。这种存储的方式可以避免重复计算。（此处省略了下标，因为两层的推导基本方法都如上述））

- 一个小知识需要记忆：sigmoid 函数的导数是：$\sigma'(x) = \sigma(x)(1-\sigma(x))$
- 图片中 p1 是指前一层的参数更新，p2 是后一层，前一层建立在后一层的基础上。
- 例子中，学习率默认为 1，且没有偏置项。
- 注意用前一层更新再前一层时，要用原始的参数。

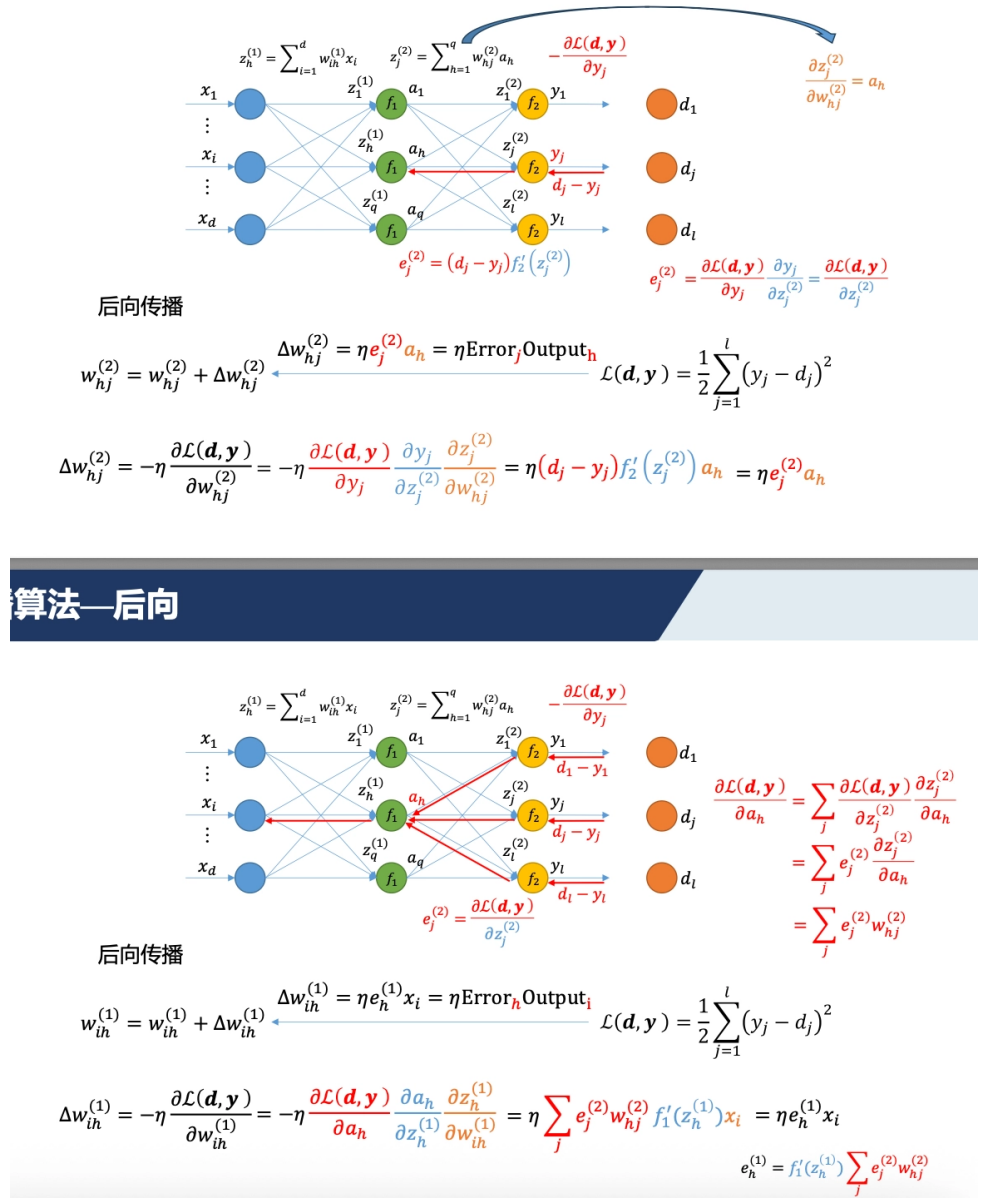


图 3.28: 反向传播实际计算

3.2.2　其他内容简介

本节核心是反向传播，也介绍了一些其他的优化算法注意点等小知识点。做个了解，考试中判断题也许会冷不丁出现。(25 秋考察了 dropout 的一个判断题)

深度学习实现

- 深度学习实现过程（神经网络编程）的三个步骤：定义网络，损失函数，寻找优化方法（如梯度下降）。

- 优先使用 Pytorch，最主流的深度学习模型。语言为 Python。

随机梯度下降相关问题

见第2.2节线性回归部分。

神经网络优化技巧与要点

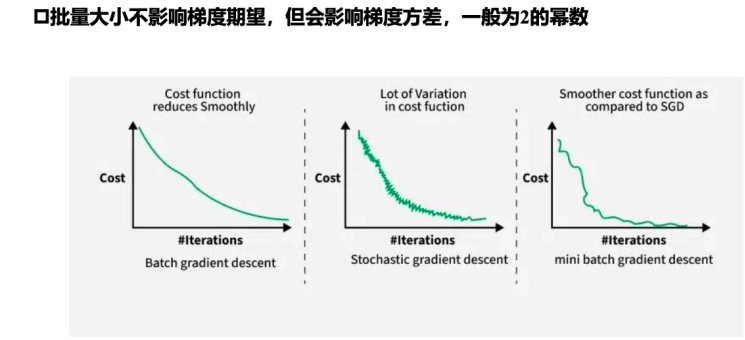
ppt 中还重复介绍了一些学习率、L1L2 正则化技术等等。此处在第2.2节线性回归部分已经介绍了部分略去。

样本随机性

- 在每次迭代时，随机选择 B 个样本，这里的随机性非常重要，但随机性在大规模数据情况下很难满足。

- 实践中通常将样本顺序打乱一次，然后按照这个顺序存储起来。虽然偏离真实随机采样，但不会有严重的有害影响。

批量大小的影响　批量大小指的是上面的提到的，每次选取样本的大小。代码中常用 `batch_size` 表示。



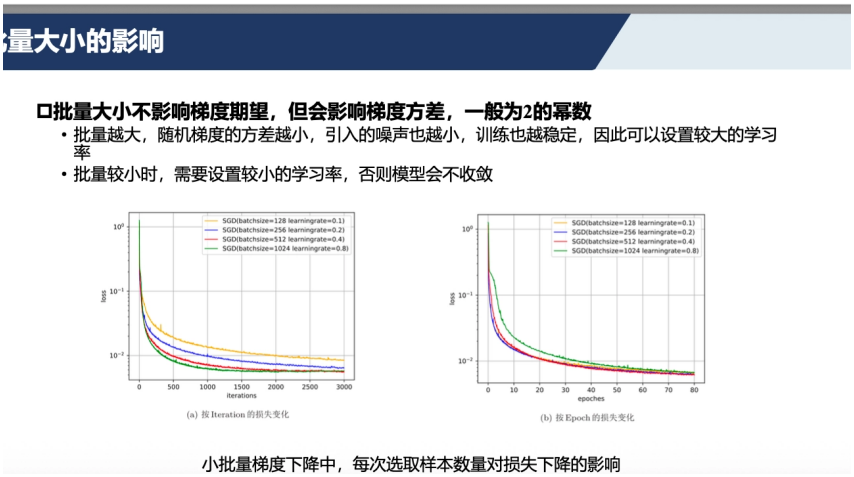


图 3.29: 批量大小的影响

提前终止 简单有效的方法，验证集（参见第2.1.4节）上性能不再提升就停止训练。防止过拟合。

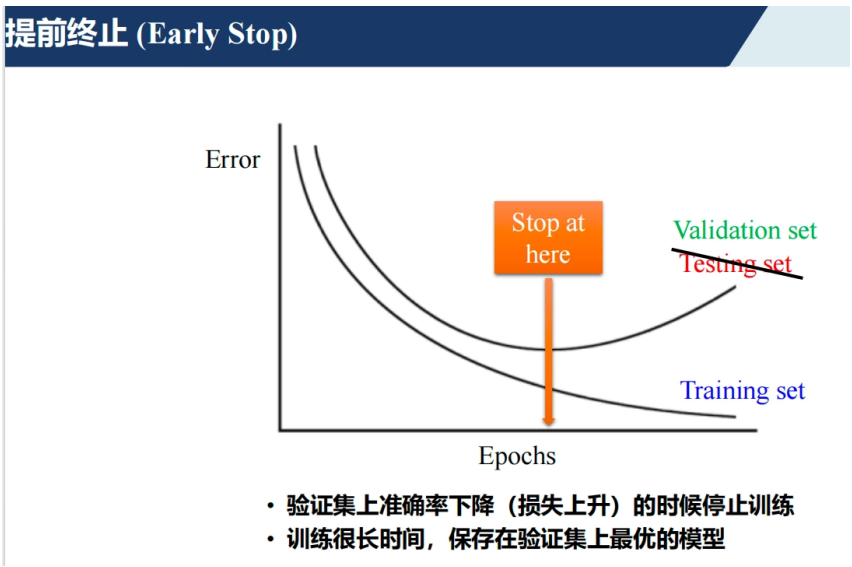


图 3.30: 提前终止

Dropout Dropout 是一种正则化技术，用于减少神经网络中的过拟合。其核心思想是在训练过程中**随机”丢弃”（即设置为 0）一部分神经元的输出**（25 秋判断题：随机丢弃一部分样本

$(\times)$)。具体来说，在每次训练迭代时，以一定的概率 p 将某些神经元的输出置为 0，这使得网络结构在每次迭代中都会发生变化。

实现方式如下：

- **训练阶段**：在取得一个 batch 后，以概率 p 设置一些神经元为 0。例如，输入单元被包括的概率为 0.8，隐藏单元为 0.5。

- **测试阶段**：没有 dropout，即所有的神经元都处于激活状态。为了保持一致性，测试时需要缩放激活函数的输出，使得每个神经元测试时的输出等于训练时的期望输出。如果训练时 dropout 的概率为 p，那么所有的权重要乘以 $(1-p)$。

作用：

- **强制网络有冗余特征表示**：通过随机丢弃神经元，网络被迫学习多种不同的特征组合，从而提高泛化能力。

- **防止特征的 co-adaptation**：避免某些特征检测器只有在其他特定特征检测器存在时才能发挥作用的情况。例如，模型可能学会通过鼻子检测人脸，但如果鼻子信息被擦除，模型必须学习其他特征（如嘴）来检测人脸。

还有一个变体：**DropConnect**。丢一部分权重（连接）。可以记忆为，dropout 丢输出，dropconnect 丢输入。

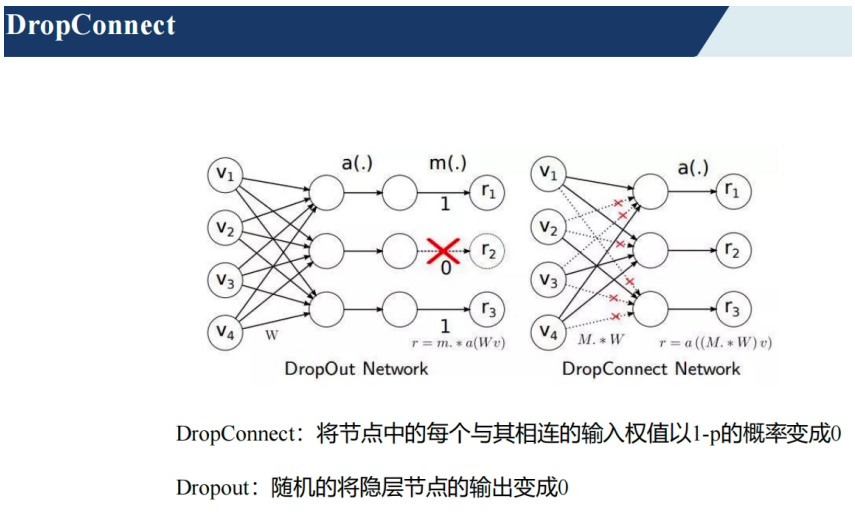


图 3.31: DropConnect

参数初始化

图 3.32: 参数初始化

梯度下降算法是在训练过程中对参数的原始值进行更新。而如果参数初始值已经接近理想值，优化的效率一般会更好。因此往往采用一些初始化的方法技巧。常见的有三类：

- **预训练初始化**：借助大规模数据集上已经预训练好的模型初始化。(核心：迁移学习，复用已经学习到的通用特征，如边缘、语义模式）适用场景主要为：小样本学习（数据量有限时防止过拟合，如医学图像分析）和复杂特征提取（如语义中高层次语义特征可以据此高效提供，自然语言处理中的 BERT)。

- **固定值初始化**：如偏置初始化为 0，权重按照专业领域知识初始化。注意一个问题，参数不可以全部初始化为 0。会导致输出没有区分性。

- **随机初始化**：按照均匀、Gaussian 等分布初始化（基于固定方差)，以及自适应方差的 He 初始化和 Xavier 初始化（可以缓解深度学习中梯度消失问题)。

拓展内容

He 初始化和 Xavier 初始化（会在大作业中用到） **Xavier 初始化：**

对于一个有 n_{in} 个输入、n_{out} 个输出的全连接层：

- **均匀分布**：在区间 $[-r, r]$ 内采样，其中 $r = \sqrt{\frac{6}{n_{in}+n_{out}}}$
- **正态分布**：从 $N(0, \sigma^2)$ 采样，其中 $\sigma = \sqrt{\frac{2}{n_{in}+n_{out}}}$

图 3.33: Xavier 初始化

- 同时考虑了前向传播（输入方差）和反向传播（梯度方差），从而保持前向传播和反向传播中，每层输入和输出的方差大致相等。确保梯度在深层传播时不会过度衰减或爆炸。
- 适用于 **Sigmoid**、**Tanh** 等**对称且饱和区域平缓**的激活函数，对 ReLU 激活函数效果不佳（ReLU 非对称，只输出非负值）。

He 初始化：

- 专门为 ReLU 及其变体设计。只考虑**前向传播**中保持方差稳定（或反向传播）。
- 数学原理与设计：

数学推导（关键步骤）

对于第 l 层：

1. **仿射变换**：$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$
2. **ReLU激活**：$a^{(l)} = \mathrm{ReLU}(z^{(l)}) = \max(0, z^{(l)})$

假设：

- $W^{(l)}$ 元素独立同分布，均值为0
- $a^{(l-1)}$ 元素独立同分布，均值为0
- $W^{(l)}$ 与 $a^{(l-1)}$ 相互独立

推导出：

$$\mathrm{Var}(z^{(l)}) = n_{in} \cdot \mathrm{Var}(W^{(l)}) \cdot \mathrm{Var}(a^{(l-1)})$$

其中 n_{in} 是输入神经元数量。

关键观察：对于ReLU，若输入 x 方差为 σ^2，则输出方差为 $\sigma^2/2$：

$$\mathrm{Var}(a^{(l)}) = \frac{1}{2}\mathrm{Var}(z^{(l)})$$

为了让 $\mathrm{Var}(a^{(l)}) = \mathrm{Var}(a^{(l-1)})$（保持方差稳定），需要：

$$\mathrm{Var}(W^{(l)}) = \frac{2}{n_{in}}$$

具体实现

- **均匀分布**：在区间 $[-r, r]$ 内采样，其中 $r = \sqrt{\frac{6}{n_{in}}}$
- **正态分布**：从 $N(0, \sigma^2)$ 采样，其中 $\sigma = \sqrt{\frac{2}{n_{in}}}$

图 3.34: He 初始化

3.3　深度神经网络

问题：梯度消失

越深的模型越难训练，发生**欠拟合**。

（注意一个误区：欠拟合指的是模型过于简单，以至于无法捕捉数据背后的潜在规律。它在**训练集**上的表现就很差（误差很高），在测试集上自然也很差。最常见原因是模型过于简单，也可能是训练时间不足等等。注意**样本不足**通常导致的是**过拟合**（Overfitting），逻辑：样本太少 → 模型很容易把”噪音”当成”规律”记住 → 训练集得分很高，但测试集得分极低（泛化能力差）。）

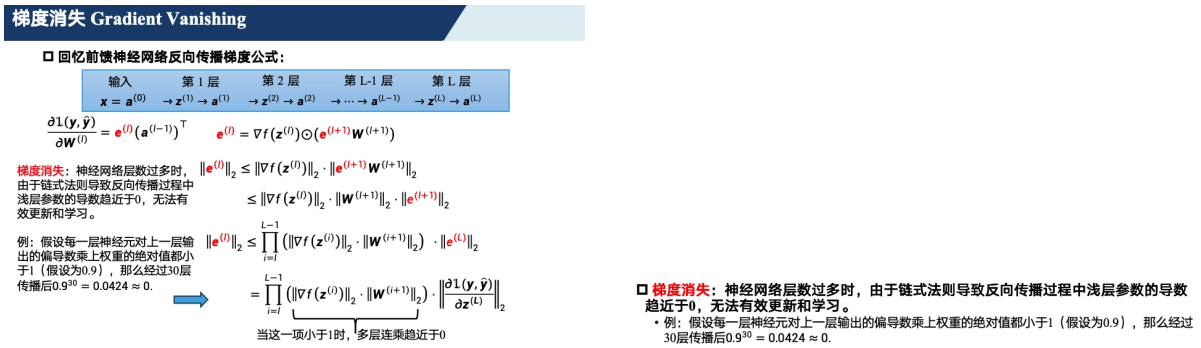


图 3.35: 梯度消失问题

缓解办法：

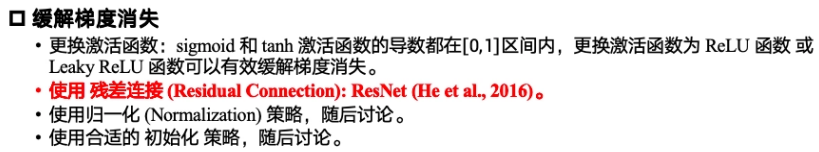


图 3.36: 梯度消失的缓解办法

后面三条随后展开。注意第一条更换激活函数。这实际上是一种很常见的策略，之前在激活函数那里提过。忘记激活函数形式的可以看第3.1.1节中的激活函数。

3.3.1 训练策略一：残差连接

本节不要太纠结细节（实际上我也不会）。大致理解原理即可。个人觉得这节考试上能考个残差连接的名字就不错了。

基本想法

- 神经网络整个部分是在近似一个函数 $y = H(x)$。
- 直觉上深层网络**至少不应比浅层网络差**，因为可以构造一个深层网络：将浅层网络参数复制到前几层，后面层设置为**恒等映射**。
- 但实践中，让深度网络学习恒等映射极其困难。但学习 0 映射是相对容易的，为此我们让其拟合 $F(x) = H(x) - x$。

  理解：之前是直接在 H 的空间中搜索最优函数，而残差坐标相当于做了一个坐标变换，在 F 的空间中搜索，其中 $H = F + \text{identity}$。如果最优解是恒等映射（在深层神经网络中往往特征变化已经很小了，就接近恒等了）：

  - 在 H 空间：需要精确的权重配置才能实现恒等。
  - 在 F 空间：只需让 $F(x) \to 0$，有很多方式实现（权重趋近 0 即可）。

具体架构

加入一个跨层连接。

数学表达变化：

- **传统**：$a^{(l)} = f(W^{(l)}a^{(l-1)} + b^{(l)})$
- **残差**：$a^{(l)} = f(W^{(l)}a^{(l-1)} + b^{(l)}) + a^{(l-k)}$ （k为跳跃层数）

图 3.37: 残差连接

形象的理解就是：

```
输入 → 层1 → 层2 → 层3 → ... → 输出
          (跳跃连接)
```

这一结构强制规定 $H(a^{(l)})$ 必须表示为 $F(a^{(l)}) + a^{(l)}$ 的形式。

作用

梯度消失以及深层带来的信息丢失也得到了缓解：

残差网络的梯度改善

对于残差块：$a^{(l+2)} = f(W^{(l+2)}a^{(l+1)} + b^{(l+2)}) + a^{(l)}$

梯度计算变为：

$$\frac{\partial a^{(l+2)}}{\partial a^{(l)}} = \frac{\partial f(W^{(l+2)}a^{(l+1)} + b^{(l+2)})}{\partial a^{(l)}} + I$$

其中 I 是单位矩阵。

关键改变：

- 梯度表达式从纯连乘 $\prod f'W$ 变成了 $\prod(f'W + I)$
- 即使 $f'W$ 很小，$+I$ 项也能保证梯度至少有一条值为1的通路
- **梯度不会完全消失**，总能通过恒等连接回传

传统网络的信息流

- 每经过一层，原始输入信息都会**被非线性变换修改**
- 深层可能丢失浅层的细节信息
- 如：图像的边缘、纹理等低级特征在深层可能被完全转换

残差网络的信息流

- 浅层特征可以通过跳跃连接**直接传递**到深层
- 网络可以同时利用：
  - **低级特征**（通过跳跃连接保留）
  - **高级特征**（通过非线性层提取）
- 实现了**多尺度特征融合**

图 3.38: 残差连接的作用

课件上介绍了一个 ResNet 的残差连接结构，这是一个来源于论文的成熟网格架构。个人觉得这里没啥考点，了解即可。

3.3.2　训练策略二：归一化

归一化（normalization）是训练过程中对模型的一种干预。考试时对每种归一化的概念清楚即可，不需要记忆具体归一化公式。

方式一：数据归一化

- 原始数据的不同特征（如年龄、工资）可能具有完全不同的尺度（0–100 vs. 10000–100000）。
- 方式包括：标准化（均值为 0，方差为 1）；最大最小值归一化。

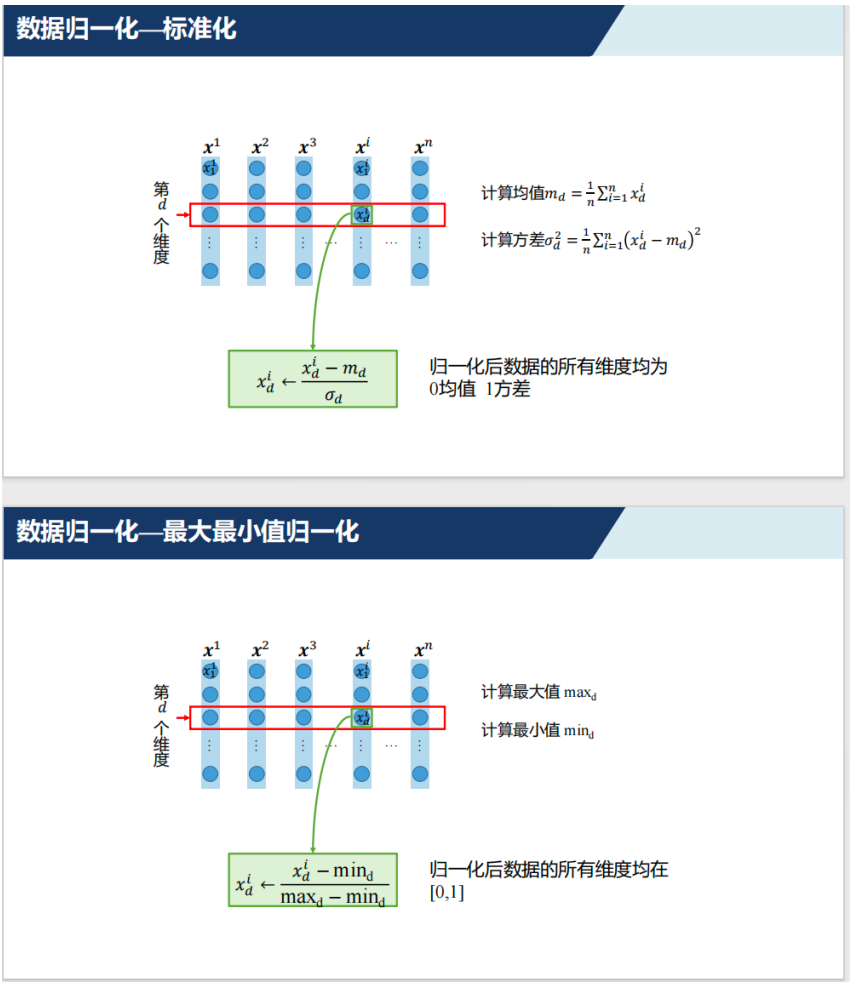


图 3.39: 数据归一化

方式二：逐层归一化

原因：网络会扭曲数据分布。即使输入数据已经完美归一化，经过网络层后分布也会变化。原因在于非线性激活函数会制造新的分布偏移，即**内部协变量偏移**。以下方式是应对出现的具体问题的不同 normalization 方法。

作用在于：稳定梯度，防止过拟合，允许较大学习率，减少初始化影响。

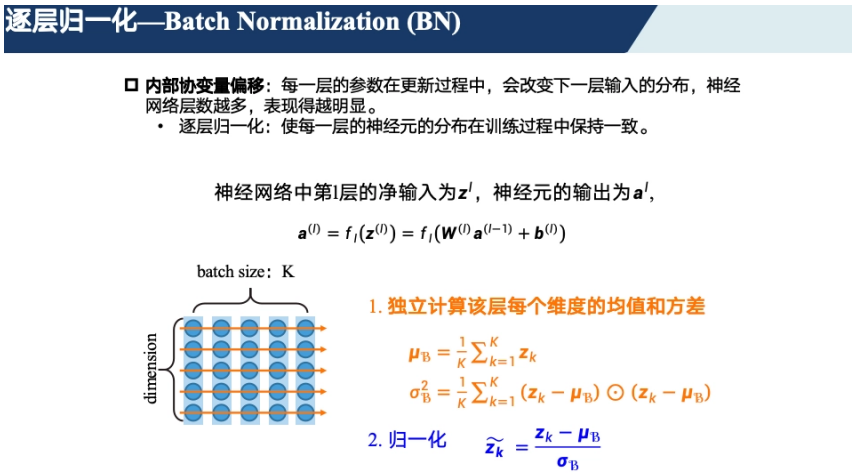


图 3.40: Batch Normalization（一）

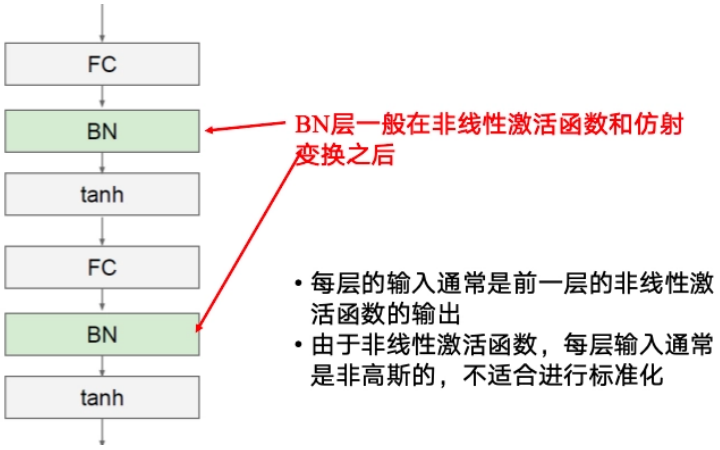


图 3.41: Batch Normalization（二）

BN（Batch Normalization） 核心需要知道：BN 是对一个 `batch_size` 上所有的样本单一特征维度（MLP 中指的是向量的某一个维度，CNN 中则是指一个卷积核对应的维度）做归一化，也就是”**跨样本、不跨维度**”。

注意测试和训练过程的 BN 有不同。

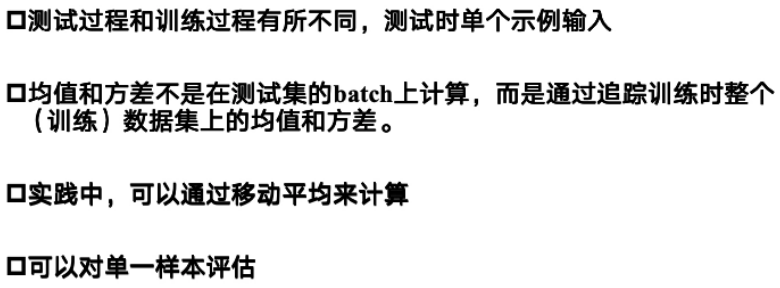


图 3.42: BN 训练与测试的区别

MLP 与 CNN 的 BN 区别——都是不跨特征维度。主要在于 CNN 输入往往是图片，图片的高度宽度是单张图片的内容，需要归一化：

网络结构	输入维度	统计量计算范围（归一化对象）
MLP	(N, D)	对每个特征维度 d，在批量 N 上计算均值和方差
CNN	(N, C, H, W)	对每个通道 c，在批量 N、高度 H、宽度 W 上联合计算均值和方差

表 3.1: MLP 与 CNN 的 BN 区别

LN（Layer Normalization） 动机如下（pad 是自然语言处理中用于填充的符号。为了使得不同句子的长度能够组成一个 batch 所以要填充。如果用 BN，沿着每个维度都归一，大量的 pad 会拉低均值使得统计量失真）：

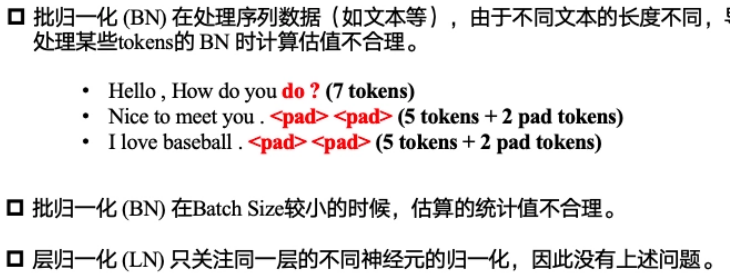


图 3.43: Layer Normalization 动机

层归一化（LN）只关注同一层的不同特征维度（MLP 中是不同神经元，CNN 中就是不同卷积核）的归一化，不受样本大小计算。具体方式：单个样本内部，沿着特征维度计算均值和方差。每个样本的每个位置独立计算。pad 位置的向量会基于它自己的值归一化，不影响真实词位置。

改进：**RMS Normalization**（了解即可）

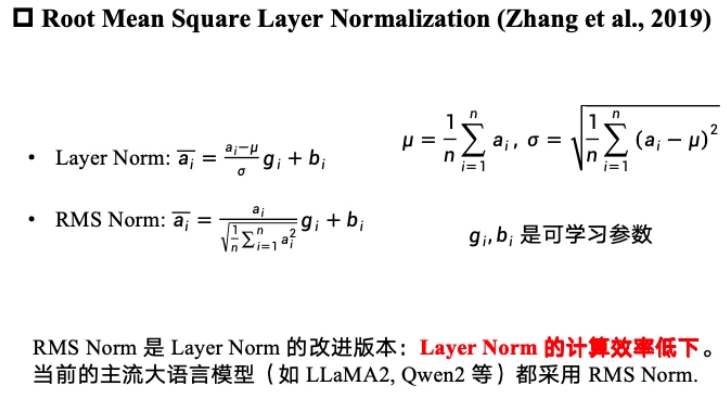


图 3.44: RMS Normalization

方式三：实例归一化

其问题来源于图片中的风格迁移：如何分离内容和风格？

- **内容**：主要由特征图的**空间信息**（哪些神经元在哪些位置被激活）来表征。比如”房子”的轮廓是由特定神经元在图像特定坐标的激活模式决定的。

- **风格**：主要由特征图**每个通道的统计信息**（均值和方差）来表征。比如” 梵高画风” 体现为某些特征通道的均值较高（代表某种黄色调很浓），方差较大（代表笔触对比强烈）。

此时 BN 对样本计算方差均值，会混淆风格图像；LN 计算一个样本所有通道的统计量，混合了不同通道风格。实例归一化（IN）可以看作是最” 局部”、最” 细致” 的归一化方式。自然也就只适用于 CNN。MLP 只有两个输入维度，这种方式相当于没有做操作。

归一化方法	输入维度	统计量计算范围（归一化对象）
Batch Normalization (BN)	(N, C, H, W)	对每个通道 c，在批量 N、高度 H、宽度 W 上联合计算
Layer Normalization (LN)	(N, C, H, W)	对每个样本 n，在所有通道 C、高度 H、宽度 W 上联合计算
Instance Normalization (IN)	(N, C, H, W)	对每个样本 n、单个通道 C，在高度 H、宽度 W 上联合计算

表 3.2: 三种归一化方法对比

3.3.3　训练策略三：初始化

这一部分已经在上一节（第3.2.2节）里面介绍。此处略去。以及后面讲前沿的内容也略去。有兴趣可以自查或看 ppt。

ppt 上的知识点总结如下。注意第三章可以说是考点最多、考频最高的，也是后续章节学习以及大作业实践的重要基础。需要熟练掌握。

图 3.45: 第三章知识点总结

第四章　图神经网络

注：

1. 本章节在 B 系列仅花一节课（3 学时），但是实际上介绍了 A 的 80% 以上的内容，因此有不少知识讲得较为简略。本笔记借助 AI 以及 A 系列课件做了较多补充。A 系列学习时可能少了少量内容，待后来者补充。

2. 这一部分由于难度确实相对较大，以及课时安排等原因，在 25 秋的考试中完全没有涉及。但后续（尤其是 A 系列）也许会在大题中涉及算法的考点。因此追求高分的同学最好也不要忽视此部分内容。但如果复习时间有限，可以优先把时间放在其他部分上。

3. 25 秋冯福利老师：实际上本部分内容极大丰富，需要一学分课时才能讲清楚。本身图神经网络内容极大丰富，慢慢消化。

4. 此笔记内容省略了关于图论的基础知识相关内容。聚焦于如何将机器学习应用于图结构数据。如果觉得基础不足，本章实际上用到的图论前置知识也只有一些基本概念，利用 ppt 和 AI 可以快速补齐。

4.1　图学习基础

4.1.1　图学习简介

- 图是一种描述事物（节点）及其关系（边）的通用语言。ppt 中给出了一些实例。

- **图学习：机器学习的一个前沿分支，其目标是开发能够直接从图结构数据中进行学习和预测的算法**。全称叫做图上的机器学习。也就是在图数据（样本是一个节点集合和邻接矩阵）上进行机器学习。

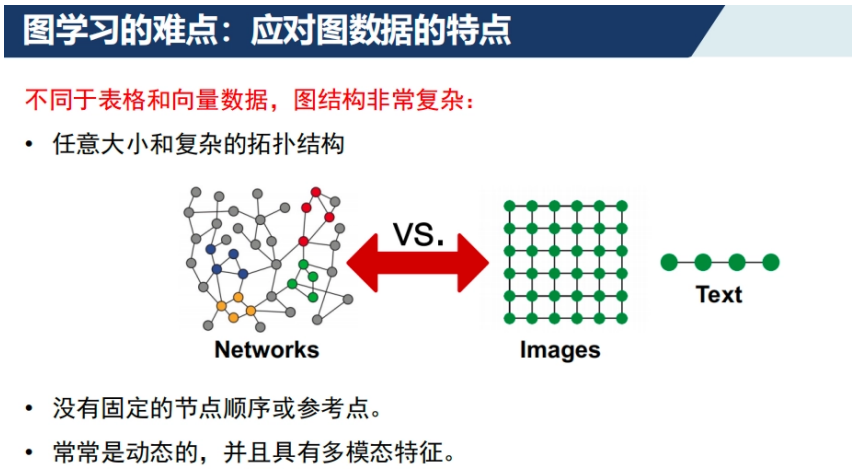


图 4.1: 图学习

- 意义在于，用于解决传统机器学习模型（如处理表格数据的分类器、处理图像的 CNN、处理序列的 RNN）的问题。其在处理图数据时显得力不从心，因为它们无法有效建模图中复杂的**拓扑结构**和**节点间的依赖关系**。

4.1.2　图的表示

- 节点表示实际问题中的具体事物，如用户等。
- 边表示节点间的关系，如社交关系。边的定义决定了问题的性质。例如，在推荐系统中，将”用户–商品”的购买行为定义为边，那么预测新链接就是一个推荐问题。
- 图有不同类型，图表示是一个基于实际问题的建模问题。对问题领域的深刻理解是构建有效图表示的前提。

异构图简介（了解即可）

图论中研究的图往往是 (V, E) 这样的同构图（这里省略了映射函数，直接放在边里面了），节点和边属于同一类型，如全是用户 (V) 和好友关系 (E)。现实世界是复杂的。

一个异构图被定义为：$G = (V, E, R, T)$

- V：节点集合。
- E：边集合。
- R：**关系类型**的集合。边 $(v_i, r, v_j) \in E$，其中 r 是关系类型，$r \in R$。
- T：**节点类型**的集合。每个节点 v_i 都有一个类型 $T(v_i) \in T$。
- 简单来说：异构图就是包含了多种类型节点和多种类型关系的图。

实例：学术图谱

- **节点类型** T：作者、论文、会议、关键词……
- **关系类型** R：撰写、发表于、包含……

- 示例事实：节点：Alice（作者），NeurIPS（会议），一篇关于 GNN 的论文（论文）；边：（Alice，撰写，论文），（论文，发表于，NeurIPS）。

- 价值：可以用于推荐合作者、发现研究趋势、评估学术影响力等。

衡量两个节点间的相关性：**元路径**（同构图中最短路径的推广，但要考虑关系）。元路径是定义在关系类型序列上的一条路径模式。形式：$T_1 \xrightarrow{R_1} T_2 \xrightarrow{R_2} \cdots \xrightarrow{R_k} T_{k+1}$。

经典示例（在学术图谱中）：

- 元路径” 作者–论文–作者”：表示” 合作者” 关系。

- 元路径” 作者–论文–会议–论文–作者”：表示” 在同一会议上发表过论文的研究者”，他们可能研究兴趣相近。

- 元路径” 论文–引用–论文”：表示引用关系。

图学习的三大任务

节点级任务

- 预测图中单个节点的属性或类别。描述节点在图中的结构和位置。

- 核心问题是为节点建立有区分度的特征表示。传统的有：节点度数、节点度重要性（穿过节点的最短路径数量，到达其他节点的平均最短路径长度）、周围子结构、graphlet 计数向量。

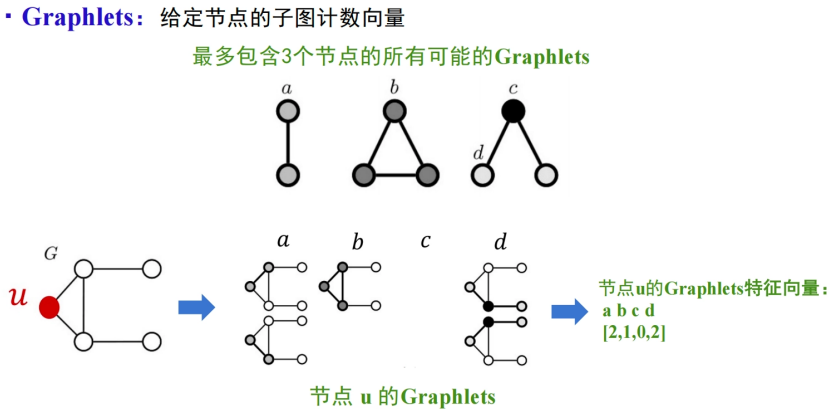


图 4.2：节点级任务

- 实例：**蛋白质折叠**：给定一个蛋白质（图），预测其中每个氨基酸（节点）在三维空间中的坐标。AlphaFold 正是解决了这个节点级回归任务。

图 4.3: 蛋白质折叠

链接级任务

- 预测两个节点之间是否存在或将会产生链接。链接级预测任务旨在基于现有链接预测新的/缺失/未知链接。

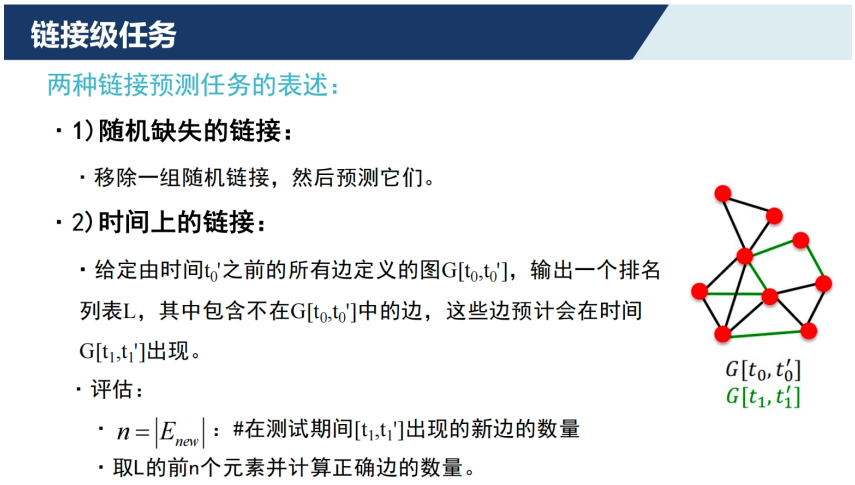


图 4.4: 链接级任务

- 实例：**药物副作用**。给定两种药物（节点），预测它们共同服用时是否会产生副作用（边）。这是一个非常典型的异构图上链接预测问题，具有重大的医疗价值。

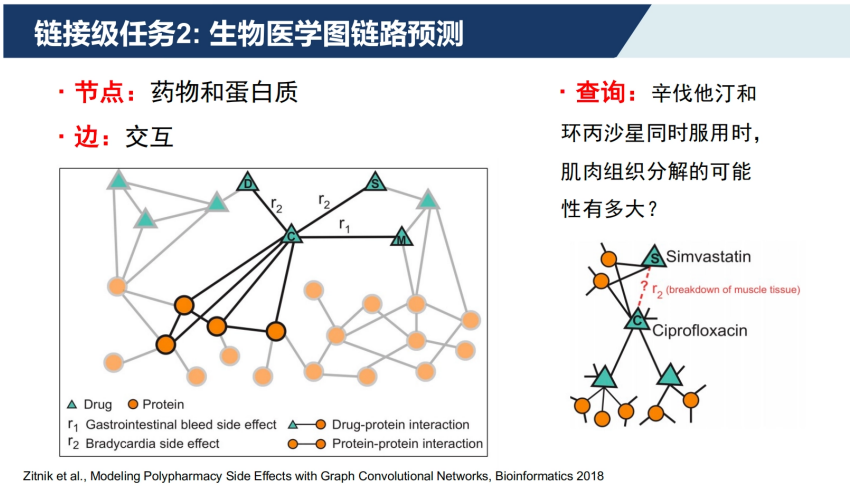


图 4.5: 药物副作用预测

图级任务 对整个图的属性进行预测，或者直接生成一个新的图。

- **核心挑战**：如何将整个图的结构信息**池化**成一个固定大小的表示向量，以供分类器或生成器使用。
- 经典示例：
  - **交通预测**：将城市道路网视为图，预测整个路网的平均速度或拥堵级别。Google Maps 使用图神经网络来预测到达时间。
  - **药物发现**：将一个分子视为图（原子是节点，化学键是边），预测整个分子是否具有某种生物活性（如抗生素性质）。这是一个图分类任务。
  - **物理模拟**：将物理系统（如水、沙、烟雾）建模为图（粒子是节点，相互作用是边），用图网络来模拟整个系统的动态演化。

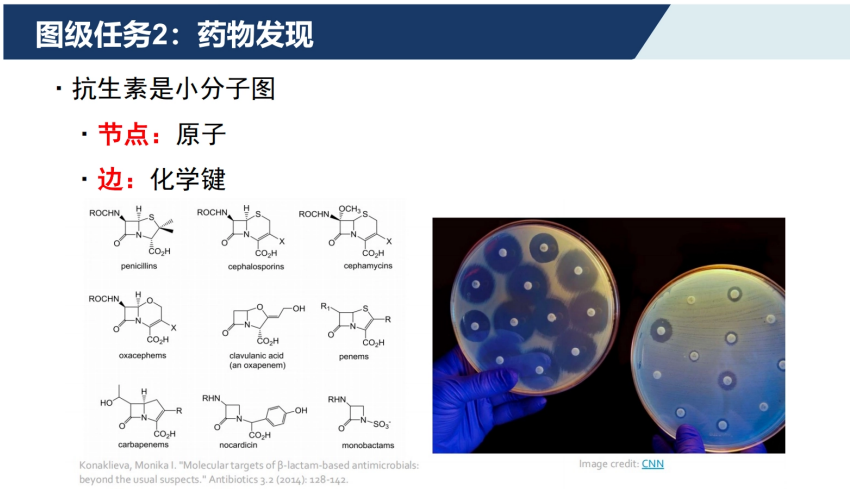


图 4.6: 图级任务示例

总结：

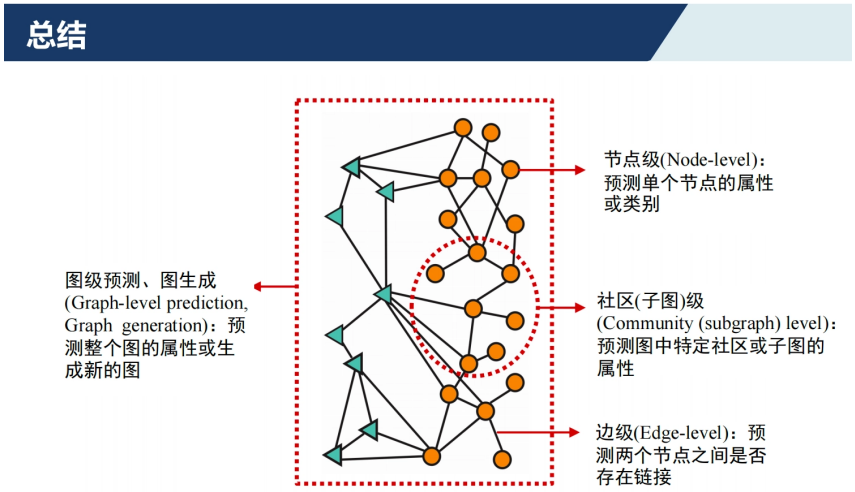


图 4.7: 图学习三大任务总结

4.2　图表征学习

概述

图数据类型简介

- 图数据类型的一个样本包括四类信息：（节点特征向量，边特征向量，全局特征向量和图结构四个部分）。前三者的向量来源于外部的输入，第四者蕴含在节点、边的连接方式中。
- 例如，将一个有机分子表示成图数据：
  - 每个节点的特征向量来源于原子种类、化学信息等。比如说涉及 20 个原子，$C = [1, 0, \ldots, 0]$，$N = [0, 1, 0, \ldots, 0]$，得到一个 20 维的 one-hot 向量。如果需要，还会加上电负性、原子质量等等原子属性维度。（可以理解成异构图中的那个节点类型，但有时这种节点类型会很多）
  - 边特征向量来源于化学键信息（可以理解成异构图中的关系类型）。
  - 全局特征向量来源于分子性质。
  - 图结构的信息来源于原子之间的连接方式。
  - 有时候输入的数据并不完整，这时候我们会采用 pooling 操作补充。比如没有输入边特征向量，就用与边关联的两个顶点向量平均作为边特征向量。

图表征学习任务

- 图学习的目标是**从图数据中得到与任务无关的结构化特征**。具体说就是，比如有一系列不同的节点预测任务：我们要**从图数据本身得到对应节点（边/全局）的特征向量（称为 embedding），这一特征向量要整合图数据的各层级信息**。（下一步我们再利用这个特征向量针对不同问题训练一个预测头，从特征向量得到任务的答案，这一步是容易的）。
- 按照要得到的特征向量种类，将图表征学习划分为**节点嵌入**、**边嵌入**、**图嵌入**三类。

- 从上可以看出，我们的特征向量是否构造的好，判别标准是是否很好地整合了图数据信息，尤其是第四类**图结构信息**。这需要我们构造的损失函数充分结合图的结构信息。这也就意味着说，我们的训练是**自监督学习**：没有使用节点/图/边的特征和来源于任务的标签，而是直接根据图结构信息构造标签来估计嵌入向量。
- 传统中采取特征工程：输入图数据 → (人工) 特征工程 → 特征向量 → 机器学习模型 → 预测结果。费时费力。
- 图表征学习的核心思想是**自动化**这个过程：输入图数据 → (自动) 表征学习 → 嵌入向量 → 机器学习模型 → 预测结果。
- 以下我们重点讨论节点嵌入，并提及与之相似的图嵌入。

4.2.1 节点嵌入

节点嵌入框架

1. **编码器**：最简单的是嵌入查找表。实际上就是直接将每个节点映射成节点特征向量，不采用统一的参数。这显然有很大的局限性，比如优化的时候是直接对每个节点的特征向量优化，计算量巨大。第4.3节我们会结合**图神经网络（GNN）**对编码器进行改进。

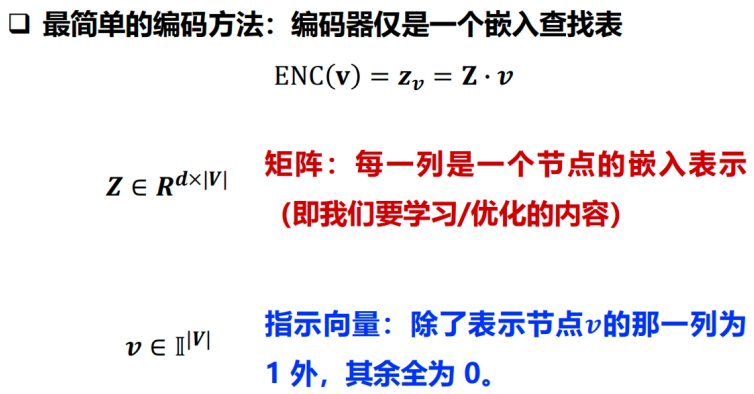


图 4.8: 编码器

2. **定义节点相似性函数**：这一步就是在**充分利用图结构信息，构造图上的相似性函数（可以看成是标签）**。
3. **解码器**：从编码的嵌入向量恢复相似性得分（也就是嵌入空间的相似性），常见的是内积：$\mathrm{DEC}(z_u, z_v) = z_u^\top z_v$。我们**训练的目标就是使得嵌入空间相似性得分和上一步图中的相似性函数得到的得分相近**。
4. **优化目标**：计算得分差，通过优化算法调整编码器的参数。

相似性

在框架中我们提到，训练目标使得嵌入过程中相似性能够保持。嵌入空间中的相似性一般就采用上述的内积相似性，但是图上的相似性衡量有多种方式。本人的另一篇笔记中介绍了一些更基础的相似性解释，以下介绍课件上的随机游走相似性定义。

- 直观感受：如果从节点 u 开始，在图中进行随机游走，经常能走到节点 v，那么我们就认为 u 和 v 是相似的。这种方法能捕捉到**多跳邻居和高阶网络结构信息**，比只看一阶邻居更强大。

（实际上理解到这里，知道要实现相似性保持，对于考试应该就足够了。书上和课件上对后面的公式由于篇幅限制讲得很精简，可能造成阅读障碍。笔者写作时正好学习了相关内容，对书上的多做一些解释，力求解释清楚。有兴趣可以查看）

拓展内容

随机游走的相似性具体实现和解释

- **随机游走的构造**：从 u_0 随机走到一个相邻顶点 u_1，再从 u_1 随机走到其相邻顶点，如此得到一个序列：$u_0, u_1, u_2, \ldots$（这是 DeepWalk 的普通随机游走，node2vec 的随机游走是 BFS/DFS bias，这些不同的策略可以参见上述笔记，但本质都是得到一个序列）。
- 取定一个大小为 w 的窗口（w 是超参数），对每个顶点 u_t 考虑其上下文 $u_{t-w} \to u_{t+w}$。那么在这一次训练中，u_t 的邻居（这里指的是此次随机游走策略 R 下得到的邻居，而不是图上相邻顶点，下同）集合 $N_R(u_t) = (u_i)_{i=t-w}^{t+w}$。在多次训练中，与 u_t 越近的顶点（也就是直觉上相似的顶点）出现的频率会越高，这也就反映了的**图上的相似性**。
- 每一次训练中，我们的目标自然就是，在嵌入空间中，让出现在此次邻居集合中的点与 u 相似性更高。嵌入空间中的相似度定义：$\mathrm{DEC}(z_u, z_v) = z_u^\top z_v$。此处我们要**考虑所有点的相对值，也就是需要把所有候选 $v \in V$ 的分数变成一个合法概率分布** $P(\cdot|u)$，满足 $\sum_{v\in V} P(v|u) = 1$、非负性、单调性（相似度越高，概率越大）。
- 这自然采用 softmax（其同时还有一些概率背景、计算上的优点），也就是

$$P(v|z_u) = \frac{\exp(z_u^\top z_v)}{\sum_{n\in V} \exp(z_u^\top z_n)}$$

- 采用和逻辑回归中类似的方法。使用最大似然，对 u，我们考虑对其所有邻居的概率预测，也就是 $\max \sum_{v\in N_R(u)} \log P(v|z_u)$。然后再利用梯度下降对 z_u 进行更新。
- 考虑全体的损失，也就是 $L = \sum_{u\in V} \sum_{v\in N_R(u)} -\log P(v|z_u)$。当然实际操作中对所有点遍历是极为困难的，会采用一些采样的技巧。此处不必深究。

4.2.2　图嵌入

图嵌入的目标是，学习整个图或子图的低维、稠密向量表示。将复杂的图结构映射为一个固定长度的向量，使得这个向量能够捕捉图的整体结构、属性和功能信息。

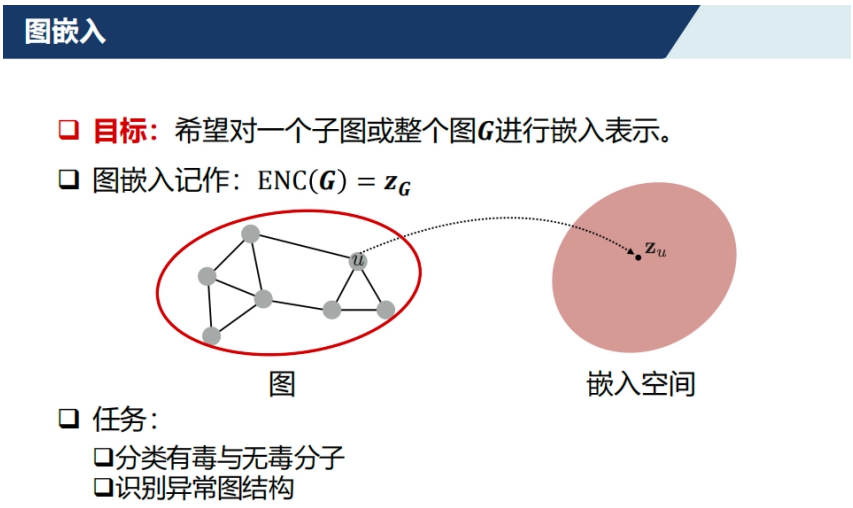


图 4.9: 图嵌入

如下所示，边嵌入、图嵌入都是基于节点嵌入，再加一些简单的操作得到的：

```
节点级任务：
输入图 → GNN → 节点嵌入 → 每个节点接一个分类头 → 输出每个节点的预测

边级任务：
输入图 → GNN → 节点嵌入 → 对每条边(u,v)组合嵌入(拼接/点积) → 分类头 → 输出
    边的预测

图级任务：
输入图 → GNN → 节点嵌入 → 读出函数(Readout/Pooling) → 图嵌入 → 分类头 → 输
    出图的预测
```

图嵌入主流的两种方法：

1. 先节点嵌入，再对所有节点嵌入进行聚合。聚合操作包括求和、平均、最大池化等等。

2. **虚拟节点法**：在原先的（子）图中引入一个特殊虚拟节点，与（子）图中所有节点相连，再对图采用节点嵌入算法。最后得到的虚拟节点嵌入就作为整个（子）图的嵌入。(这种方法往往更加有效，在随机游走中虚拟节点往往会成为所有信息的枢纽，这种引入与所有点都相邻的点是图论中的常见技巧。)

4.2.3 下游任务

学习到嵌入（好的编码器）之后，如何使用？以下给出了一些方式。在学习好嵌入之后使用以下的方式可以实现一些图的预测任务。

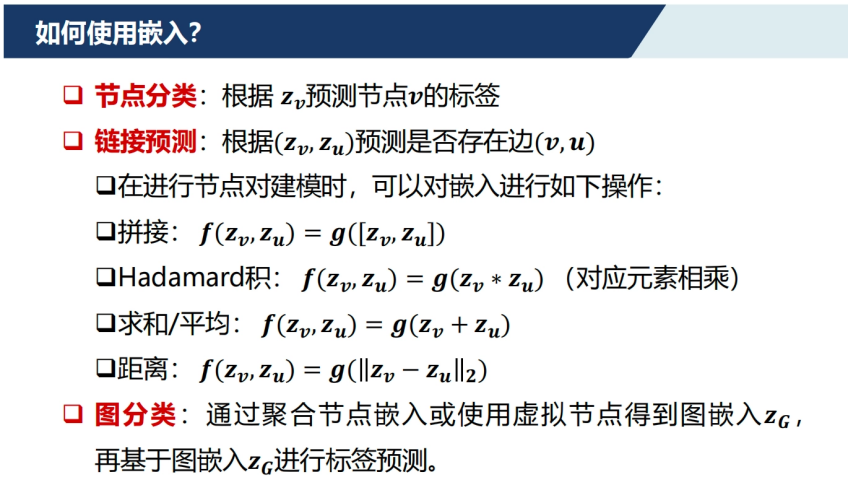


图 4.10: 嵌入的下游应用

4.3　图神经网络

4.3.1　图神经网络的建模

本部分内容除了基于课本，还结合了博客的内容进行讲解。bilibili 上的该视频对这篇博客进行了讲解。有兴趣的同学可以自行查看。

- 之前我们提到，利用图神经网络可以对节点嵌入的**编码器**进行改进。具体而言，GNN 层是对原始的图数据（参见第4.2节）中的（点，边，全局）的信息做一个变换，但不改变整体的连接性，每一层的**输出输入都是图数据**。最后一步再根据最终整合的信息进行节点嵌入。（按照课本和大多数定义，这一步是不算做 GNN 内的，也就是可以说 GNN（指的是层）输入和输出都是图。但也有说 GNN 模型 =GNN 层 + 节点嵌入，按这个说法，输入图数据经过 GNN（指的是整个模型）会得到节点嵌入集合。不必细抠概念，理解即可。）

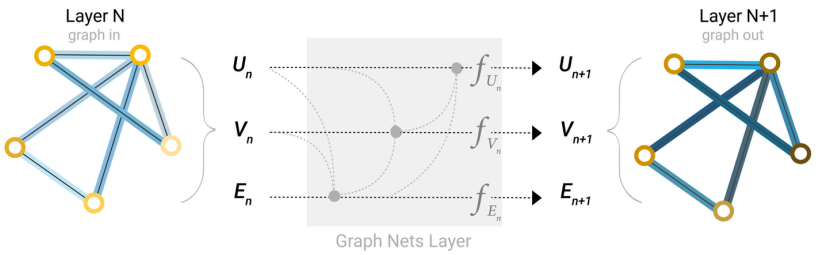


图 4.11: GNN 层

- 做这个变换的动机在于，原始的点、边、全局向量并没有融合图结构的信息，直接学习会造成困难。GNN 相当于对其做一个预处理，使得其融合了图结构信息。因此主流的 GNN 模型采用的核心设计想法是**邻居聚合：一个节点的嵌入，应该由其自身特征和其邻居节点（/边）的嵌入共同决定**。

- 因此 GNN 的每一层输入，实际上结合了邻居节点（边）和自身的信息。结合方式（乘以的权重矩阵是可以学习的）。具体的数学公式可以参见课本（这里写的很清楚了）或下图：

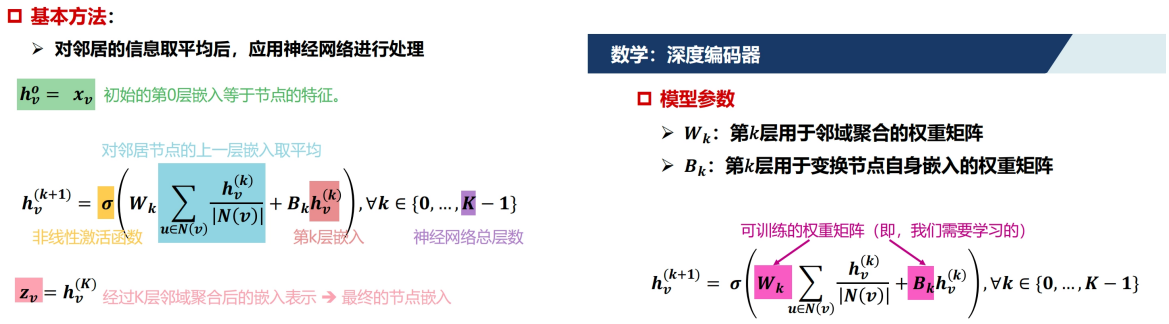


图 4.12: GNN 邻居聚合

4.3.2 图神经网络的优势

- **融合高阶邻居信息**：从上述的流程我们可以看到，对于给定的节点 u，经过第一层 GNN 后，u 的嵌入融合了其自身和相邻点（一阶邻居）的信息；第二层 GNN 得到 u 的嵌入来源于第一层输出的自身和邻居的信息，但第一层输出的邻居已经融合了原始的自身和其邻居的信息，也就是 u 间接融合了其邻居的邻居（二阶邻居）的信息。以此类推，多层 GNN 能够使得 u 融合高阶邻居信息。这一过程实际上和 CNN（参见第3.1.4节）的感受野扩大十分相似。

- **归纳能力**：对于新添节点，只需要对邻接矩阵新添行列。可以很方便的更新。

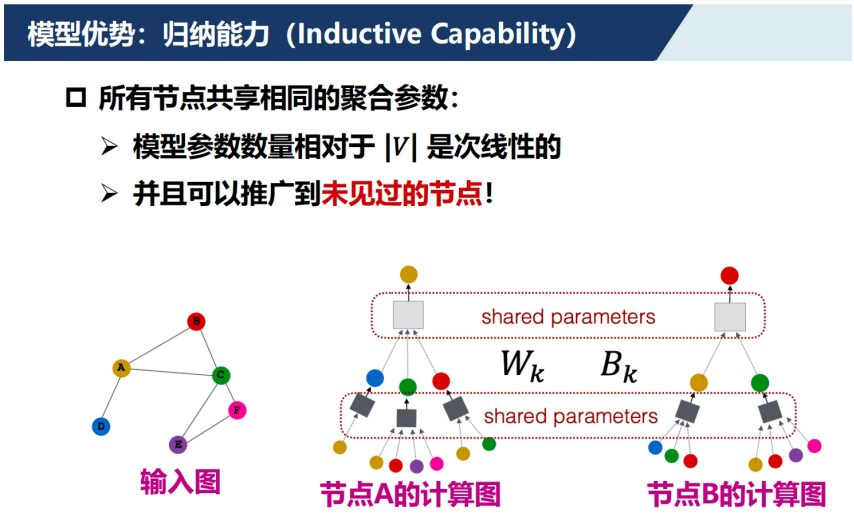


图 4.13: GNN 的归纳能力

4.3.3 训练流程与数据划分

训练流程

GNN 层 + 节点嵌入层本身是一个特征提取器，得到的是融合了图信息和本身信息的嵌入。利用嵌入来解决下游任务（预测、分类等），还需要**预测头**。预测头就是一个简单的神经网络模块，它根据任务的类型，对嵌入进行最后的加工，输出最终的预测结果。

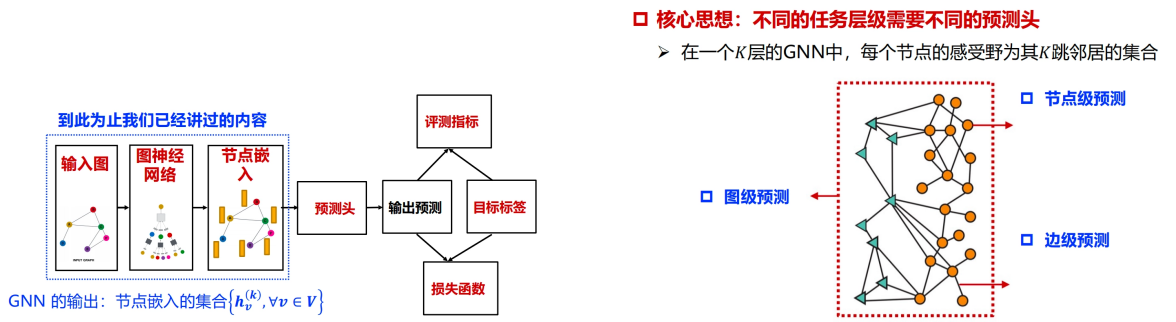


图 4.14: GNN 训练流程

损失函数

- 对于**有监督学习**任务，往往直接使用任务的结果定义损失函数，**同时优化预测头和 GNN**。如分类任务（节点分类、图分类等离散任务）——交叉熵损失，回归任务（分子溶解度等预测连续值）——均方损失。为第三章内容。
- 对于**自监督学习**任务，信号来源于图自身的结构，往往还是采用和第4.2节中相同的方法，如节点对相似性损失或随机游走损失（DeepWalk/node2vec 使用的损失）。

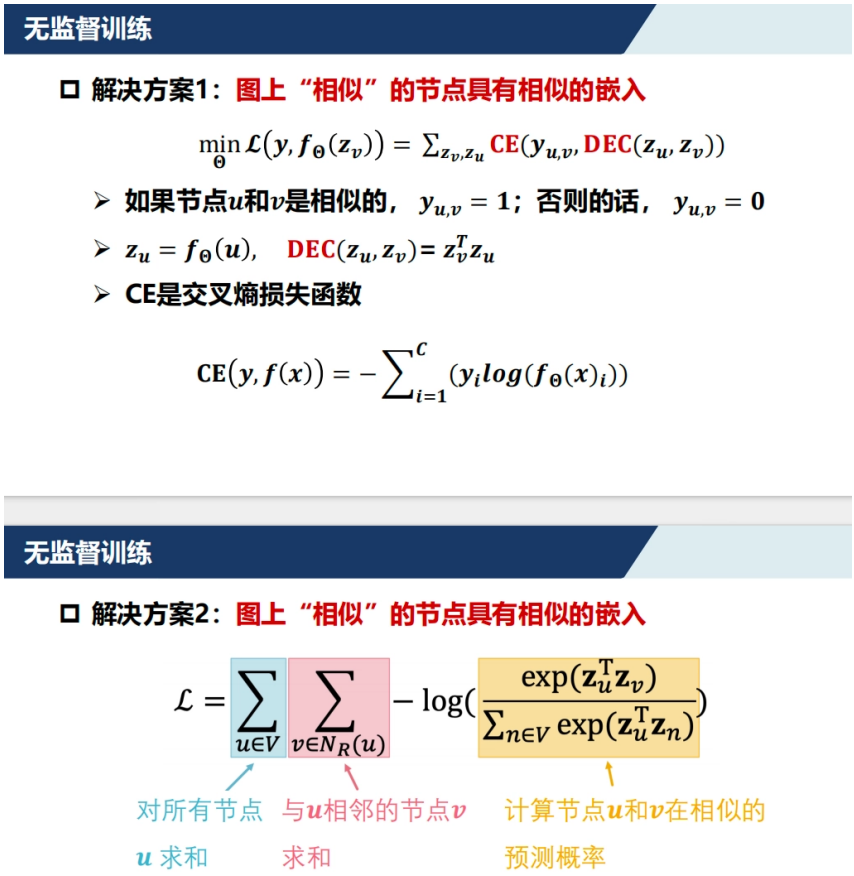


图 4.15: GNN 损失函数

注意这里 ppt 上应该是沿用的 2014 年以前的说法称为无监督（只分组，没有预测的概念），2014 年后提出自监督（自己产生伪标签）后更符合此处。

图数据划分

由于图数据的特殊性（每个数据点（节点）不是独立的（由于邻域聚合，会相互影响）），训练集、验证集、测试集划分不像一般数据那样直接划分样本就可以。主要有以下两种方式：

1. **传导式设定**：模型全过程可以看到整个图，按照节点标签划分训练集、验证集、测试集。（适用于单一图，仅适用于节点和边预测任务）

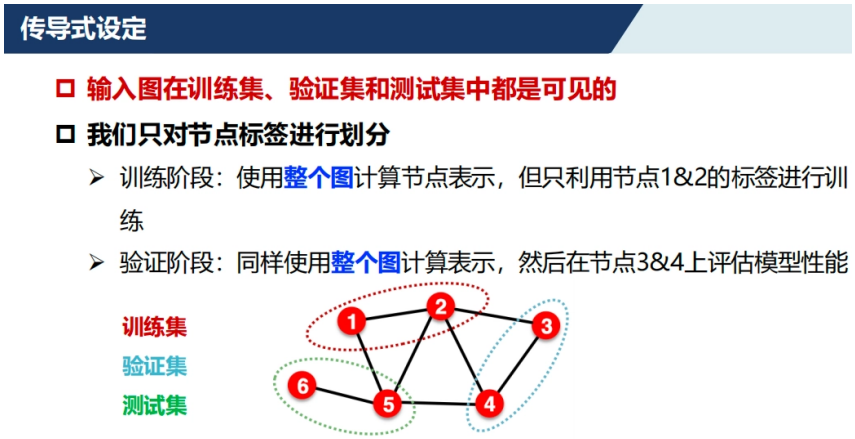


图 4.16: 传导式设定

2. **归纳式设定**：适用于多个独立图。将不同的图分配给训练集、验证集、测试集。（适用于多任务）

图 4.17: 归纳式设定

对比：

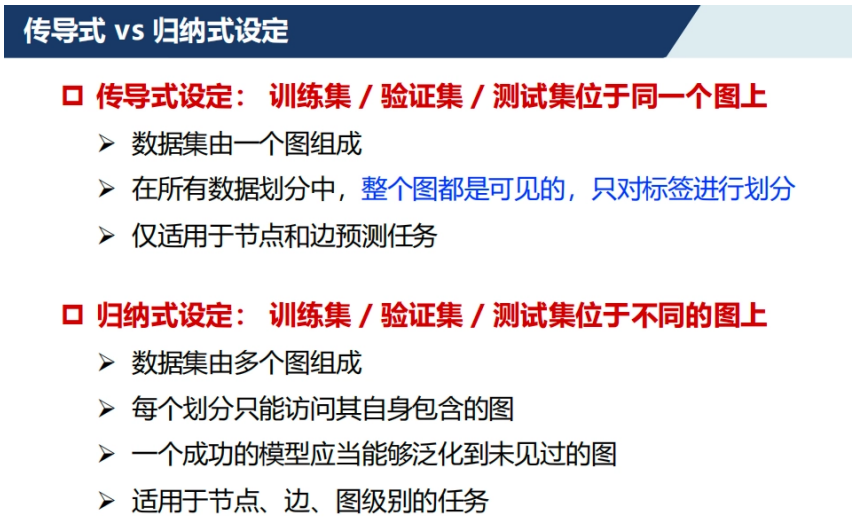


图 4.18: 传导式与归纳式对比

第五章　Transformer

注：

1. 本章相对较为重要，且内容较多，无论是平时学习还是期末突击需要多投入一些时间，重点要掌握几种注意力机制并能计算相关参数。

2. 25 秋的 B 系列课程实验部分的大作业选题之一就是基于 Transformer 的情感分类任务，了解原理对大作业的理解会有一定帮助。

3. 本章内容在理解上依然存在一定难度，但是笔者认为略比第四章简单，学习/突击时建议自己动笔模仿课件绘制示意图，有助于理清知识，防止属于不同小节的知识混在一起，尤其在最后的应用部分要区分开。

4. 部分名词涉及到第六章的内容，不必过分纠结，看完第六章即可。

5.1　注意力与自注意力机制

5.1.1　循环神经网络（RNN）

本小节考察大题的可能性不大，基本熟悉结构即可。

基本结构

- 循环神经网络（Recurrent Neural Networks，RNN）使用一种自我迭代使用的机制，使其无需指定任意固定大小的输入长度，也能处理一维线性序列数据（例如文本）。

- 简单的理解就是，对于一个句子，如果是编码过程，句子拆分成 tokens 后依次给 RNN，每一个 token 进入后可以更新 RNN 中的某一个给定的参数向量（称为隐藏状态）。每个时间步骤运动相同的函数和同一组参数，也即每个时间步使用的权重矩阵是相同的。这里 token 指的是按照给定方式将文本拆分为一个个词元，这些词元就是 token，在第六章中我们还会看到进一步的介绍。

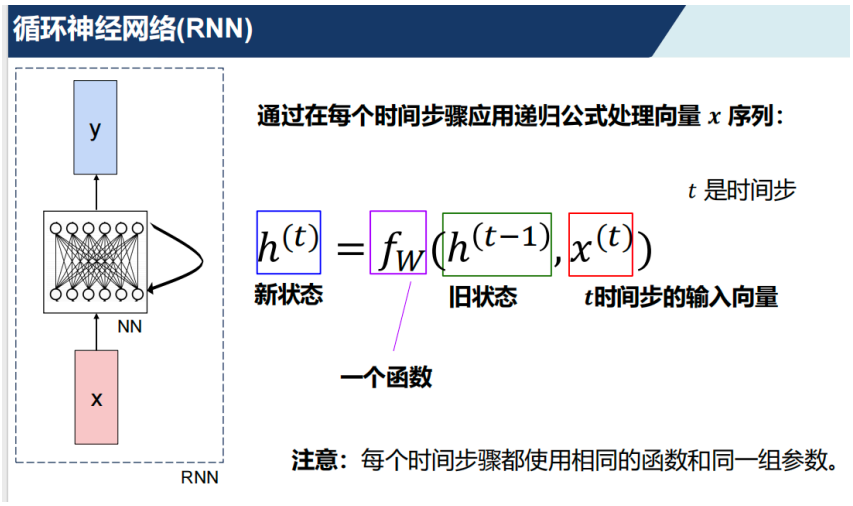


图 5.1: RNN 基本结构

- 每一个时间步都会更新一次隐藏状态（$h^{(t)}$），此更新由上一步的隐藏状态（$h^{(t-1)}$）和当前时间步输入的向量（$x^{(t)}$）决定，而输入向量则由输入的 token 以特定方式转化得到，这里不必细究。如此不断进行下去：（W 是权重矩阵）

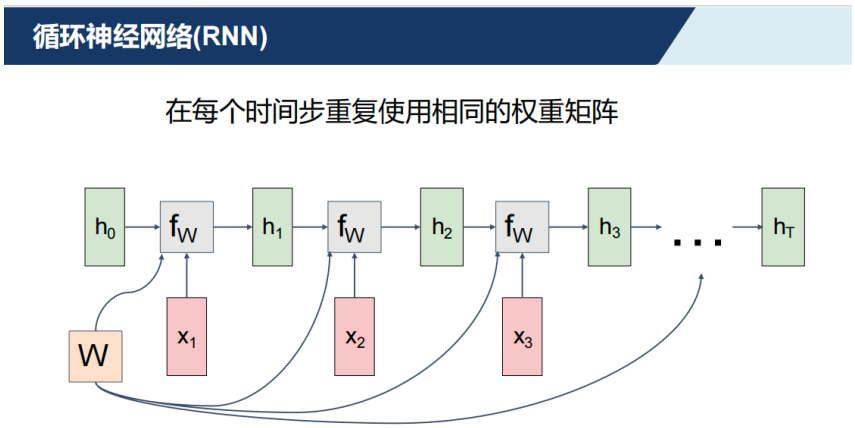


图 5.2: RNN 隐藏状态更新

- 给定时间步长，最终就会得到由 RNN 编码的最终隐藏状态（$h^{(T)}$），可以用于后续操作，它通常聚合了整条输入序列的信息，作为解码器的初始上下文或者语义表示。
- 关于解码器，RNN 自然是备选方案之一，解码器每个时间步都会产生一个输出。$h^{(T)}$ 作为初始隐藏状态，解码器根据前一步输出和当前隐藏状态预测下一输出并更新隐藏状态。

局限性

核心问题是，其迭代的机制导致卷积的层数非常多，更容易会梯度消失/爆炸；且无法并行计算。历史上为了解决，设计了长短期记忆神经网络 LSTM（现在已经不常用）。

长短期记忆神经网络 LSTM

这个内容讲的实在过于简略而需要学习的内容又太多，考试基本不可能涉及，而且现在不常用，这里只简要介绍，期末突击时不必过多关注。

LSTM 就是加入了一些门，遗忘/更新一些信息，除了如 RNN 中记忆 h 外，还引入一个 C（也是一个隐向量）由遗忘、输入、输出三个门决定。见图 3 右，f 是要遗忘的，i_t 是需要存储的新信息，利用这两个迭代更新细胞（类似于之前神经元的每一层）状态，然后再用输出门 o_t 决定输出到下一层的信息。

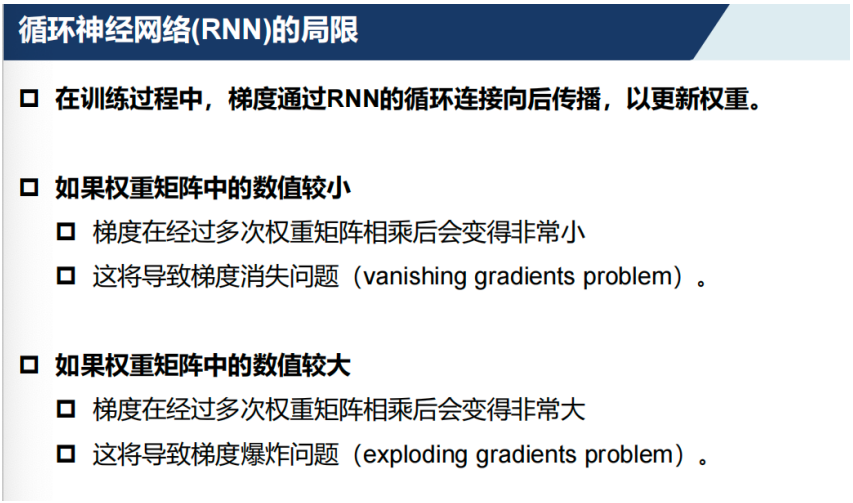


图 5.3: RNN 局限性

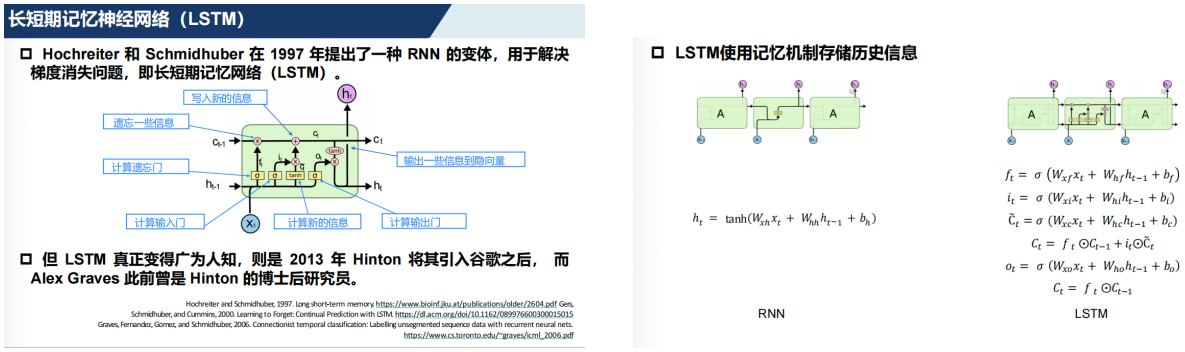


图 5.4: LSTM 结构与机制

5.1.2 注意力机制

本小节内容比较重要，虽然 25 秋考试没有涉及计算，但是笔者认为这部分的原理较为简单，掌握它们还是有必要的。

- 在处理图片或者文本时，往往会遇到部分内容相对更加重要的情况，但是这些位置是未知的。

- 注意力机制是一种思想，核心就是不同的向量要给不同的权重（注意力）。不止用于 RNN。在 RNN 中的 h 就是每一次迭代的隐向量，在其他过程可以是其他的。

- **在为特定任务建模文本时，输入文本的某些部分更为重要，而这些重要部分可能出现在输入的任何位置（可能分布集中或分散）**
- **如果重要部分远离当前状态，RNN 很难捕获重要信息**
- **为了利用重要部分中的信息，我们可以为每个输入单元分配一个注意权重，然后计算它们的加权平均值。其中，注意权重越高意味着输入单元越重要**

图 5.5: 注意力机制背景

基本流程

- **前馈注意力计算过程**
  - 前馈过程：$a(\cdot)$是一个可学习的函数，h_t是输入隐向量

$$e_t = a(h_t)$$

  - 计算注意力权重：a_t

$$a_t = \frac{\exp(e_t)}{\sum_{t=1}^{T} \exp(e_t)}$$

  - 计算输出的存储重要信息的**上下文向量c**

$$c = \sum_{t=1}^{T} a_t \cdot h_t$$

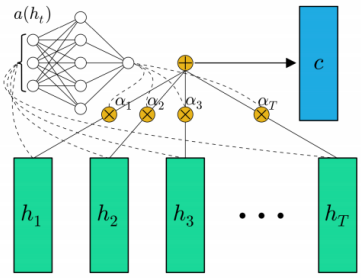


图 5.6: 注意力计算过程（一）

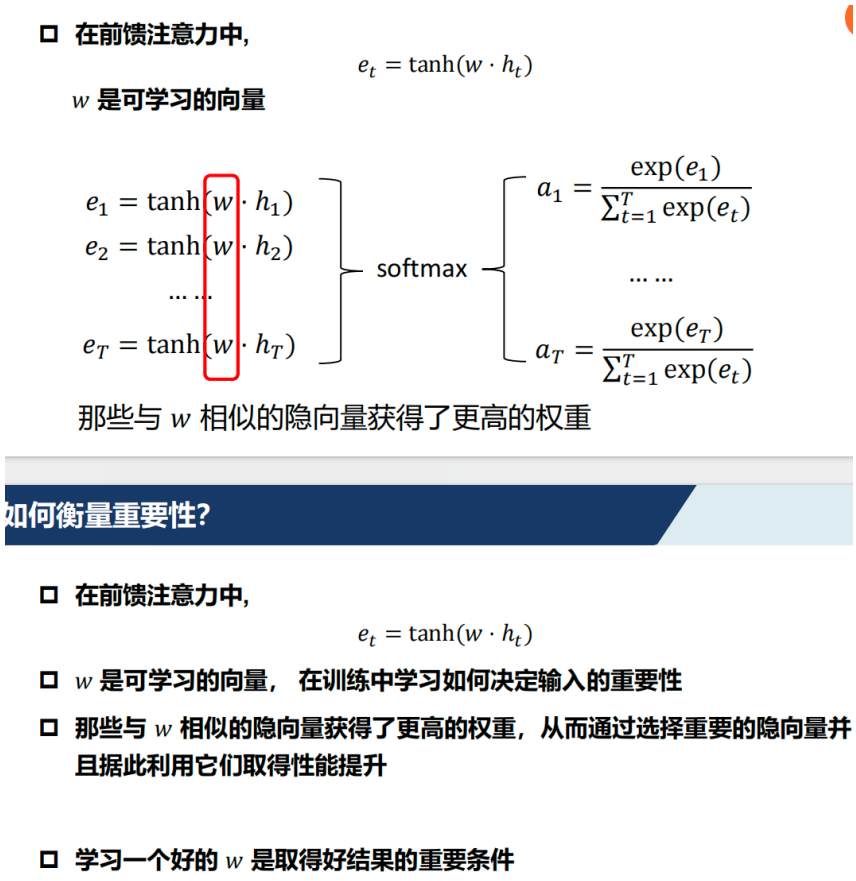


图 5.7: 注意力计算过程（二）

对应代码如下：

```python
import numpy as np

def feedforward_attention(hidden_states, w):
    """ 前馈注意力核心计算逻辑
    :param hidden_states: 隐向量序列，(T, d) 是其形状
    :param w: 可学习权重向量 (d,)
    :return: 上下文向量c (d,) """
    # 1. 计算注意力分数e_t = tanh(w · h_t)
    e = np.tanh(np.dot(hidden_states, w))
    # 2. 计算注意力权重a_t = softmax(e_t)
    a = np.exp(e) / np.sum(np.exp(e))
    # 3. 计算上下文向量c = Σ(a_t · h_t)
    c = np.dot(a, hidden_states)
    return c
```

以 RNN 为例

本部分既是对注意力机制的详细演示，也是对 RNN 的进一步介绍。

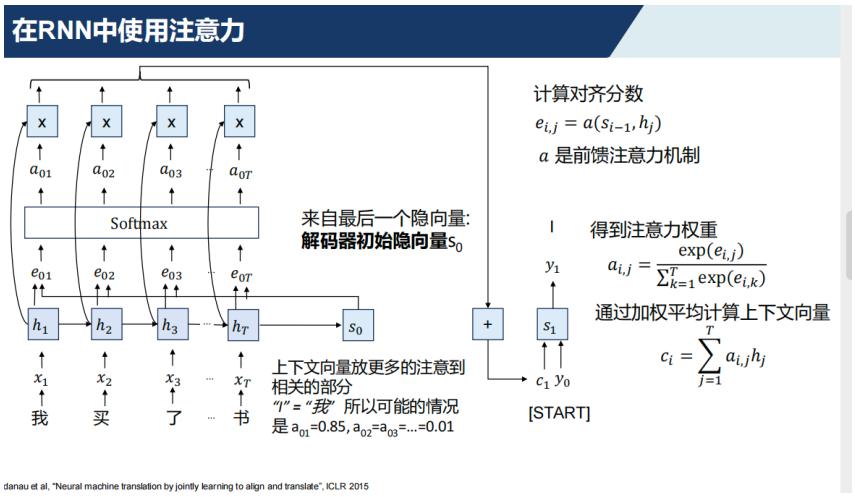


图 5.8: RNN 注意力（一）

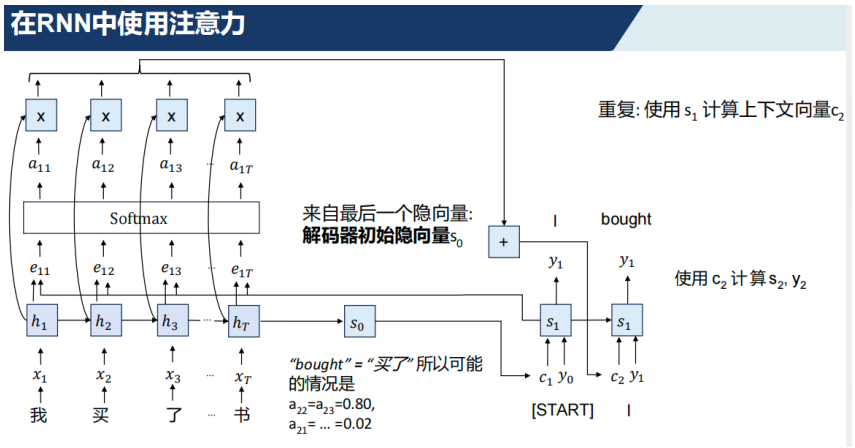


图 5.9: RNN 注意力（二）

图 5.10: RNN 注意力（三）

与上面注意力的计算方式不同的是，RNN 中的注意力权重还与解码器上一步的输出有关，解码器的每一步输出由当前时间步的上下文向量和上一步的输出以及当前时间步的隐藏状态决定。在不同时间步的上下文向量 c_t 不同，那么在解码器生成不同时间步的输出时所关注的重点

便会不同，从而实现对重要部分信息的利用。

总结

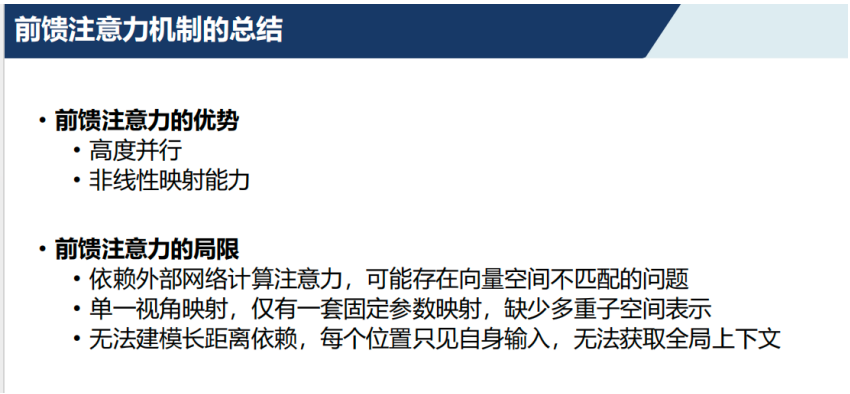


图 5.11: 注意力机制总结

5.1.3 自注意力机制

本小节内容务必掌握，尤其是自注意力机制的计算公式，25 秋期末最后一道大题基本就考察这个。

注：

- 前馈注意力只在**解码时**使用，编码器还是 RNN。
- 自注意力是注意力机制的一种变体，**不依赖 RNN**，完全基于注意力构建序列表示。
- 其与 GNN 有类似之处。GNN 是使用相邻的向量来更新自己，再计算相似度。

基本思想

自注意力机制的核心是让序列中的每个元素都能够直接与序列中的所有其他元素进行交互，从而捕捉全局的依赖关系。

计算步骤

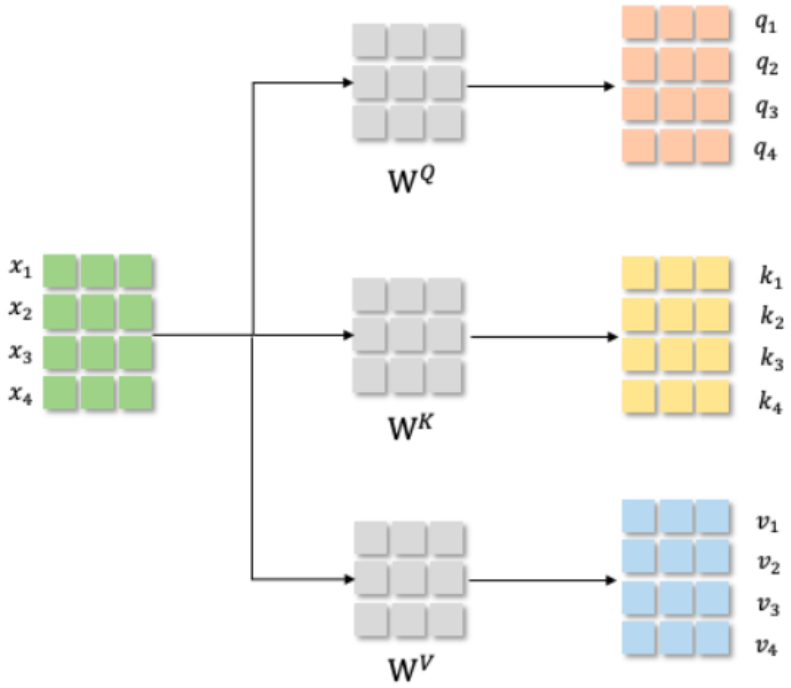


图 5.12: 自注意力机制

1. 生成 Q，K，V 三个向量。 每一个输入的词向量（第六章会有详细介绍）通过三个不同的线性变换生成：

- **查询向量（Query）**：$Q = X \cdot W^Q$
- **键向量（Key）**：$K = X \cdot W^K$
- **值向量（Value）**：$V = X \cdot W^V$

它们的含义如下：

- Q：代表”我要寻找什么信息”
- K：代表”我能够提供什么信息”
- V：代表”我实际包含的信息内容”

Q，K，V 矩阵的维度由对应的权重矩阵决定。

```
import numpy as np
def self_attention_core(X, W_Q, W_K, W_V):
    """ 自注意力机制核心计算（分四步实现）
    :param X: 输入词向量矩阵，形状 (seq_len, d_model)
    :param W_Q: Q的权重矩阵，形状 (d_model, d_k)
    :param W_K: K的权重矩阵，形状 (d_model, d_k)
    :param W_V: V的权重矩阵，形状 (d_model, d_v)
```

```
    :return: 自注意力输出，形状 (seq_len, d_v) """
    # 步骤1：生成Q、K、V向量
    Q = np.dot(X, W_Q)  # (seq_len, d_k)
    K = np.dot(X, W_K)  # (seq_len, d_k)
    V = np.dot(X, W_V)  # (seq_len, d_v)
```

2. 计算注意力分数。 通过 Q 和 K 的矩阵相乘计算相似度：

$$\text{分数矩阵} = Q \cdot K^\top$$

几何意义：分数矩阵 $Q \cdot K^\top$ 中位置 (i, j) 的值表示第 i 个查询向量与第 j 个键向量的点积相似度，即第 i 个词与第 j 个词的相关程度。

```
    # 步骤2：计算注意力分数
    attention_scores = np.dot(Q, K.T)  # (seq_len, seq_len)
```

3. 缩放与归一化。

$$\text{注意力权重} = \text{softmax}\left(\frac{Q \cdot K^\top}{\sqrt{d_k}}\right)$$

为什么要除以 $\sqrt{d_k}$ 缩放：

- 向量投影维度 d_k 越大，点积结果的方差越高（不一定是自身的词向量特征维度，而是投影到 Q 矩阵后的维度，后面的多头自注意力机制中将会看到不同）。
- 大向量会产生更高的 logits，导致 softmax 峰值过尖。
- 除以 $\sqrt{d_k}$ 可以稳定梯度，防止 softmax 饱和。
- 归一化使得每行和为 1，(i, j) 表示第 i 个词对第 j 个位置查询的注意力分布。

```
    # 步骤3：缩放与归一化
    d_k = K.shape[-1]
    scaled_scores = attention_scores / np.sqrt(d_k)
    attention_weights = np.exp(scaled_scores) / \
        np.sum(np.exp(scaled_scores), axis=-1, keepdims=True)
```

4. 加权求和。

$$\text{输出} = \text{注意力权重} \cdot V$$

将归一化后的注意力权重与值向量 V 相乘，得到每个位置的最终表示。（V 的每一行对应一个词的特征向量的投影，这样乘法后每一行是该行对应词的注意力权重对所有词的加权）

```
    # 步骤4：加权求和
    output = np.dot(attention_weights, V)  # (seq_len, d_v)
    return output, attention_weights
```

与前馈注意力的关键区别

特征	前馈注意力	自注意力
输入来源	编码器隐藏状态	同一序列自身
相似度计算	$\tanh(w \cdot h_t)$	$Q \cdot K^\top$
参数使用	单一权重向量 w	三组权重矩阵 W^Q, W^K, W^V
信息交互	单向（解码器关注编码器）	双向（序列内所有位置相互关注）

表 5.1: 前馈注意力与自注意力对比

核心优势

1. **全局信息捕获**：每个位置都能直接访问序列中所有其他位置的信息。
2. **高度并行**：所有位置的注意力计算可以同时进行。
3. **动态权重分配**：根据输入内容自适应调整关注重点。
4. **语义对齐**：在同一语义空间内计算相似度，避免向量空间不匹配。

实际意义

以句子”猫追老鼠”为例：

- 计算”追”的自注意力时，它会同时关注：
  - ”猫”（谁在追）
  - ”老鼠”（追什么）
- 模型能够直接建立”主语–谓语–宾语”的语法关系。
- 无需像 RNN 那样逐步传递信息，避免了长距离依赖问题。

自注意力机制为 Transformer 架构奠定了基础，使得模型能够高效地处理长序列并捕捉复杂的依赖关系。

核心思想：一个词自己的表示都融合其他词的信息。实现方式是基于与自己的相关性，用上下文所有的词汇加权和。实现不同任务时，定义不同的查询向量和权重即可。

应用层级

- 对于新输入的词，使用**训练好的** W^Q、W^K、W^V 生成（与词向量 X 乘）QKV。
- 这些矩阵已经学会了如何提取有用的语义特征。

自注意力机制属于**模型的核心计算层**：

```
输入层 → 词嵌入层 → [自注意力层 → 前馈层] × N → 输出层
                          ↑
                      核心计算层
```

具体层级划分：

1. **输入/输出层**：处理原始文本 ↔ 模型内部表示的转换

2. **嵌入层**：将离散符号转换为连续向量

3. **自注意力层**：核心计算层，建立词与词之间的语义关系

4. **前馈层**：非线性变换，增强模型表达能力

5. **预测层**：输出概率分布，用于具体任务

5.2 Transformer 的编码器与解码器

5.2.1 Transformer 的编码器

注：25 秋课程的大作业选题之一主要就使用了 Transformer 的编码器，相关代码将在大作业中出现。同时期末考试的最后一题涉及到多头注意力机制。

- CNN 与 RNN 处理文本对比：CNN 并行计算，RNN 迭代过程导致其计算需要前面完成才能完成后面的，失去了并行计算的优势。CNN 不同的卷积核可以提取不同的语义。但 CNN 无法处理全局信息。

- Transformer：堆叠了多层的自注意力机制和前馈层。

整体架构

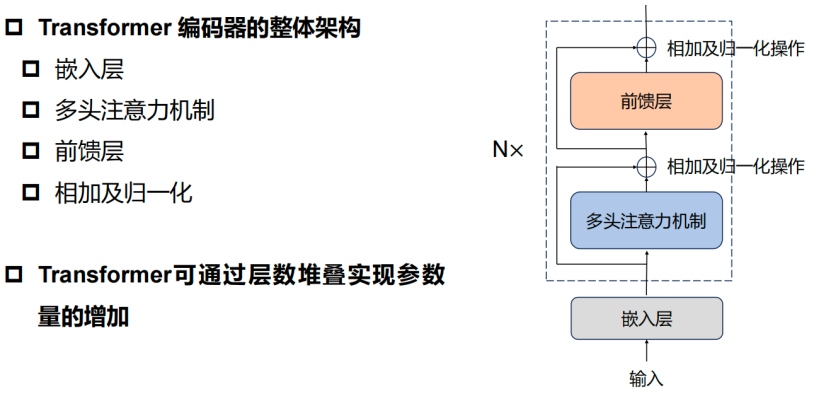


图 5.13: Transformer 编码器整体架构

嵌入层

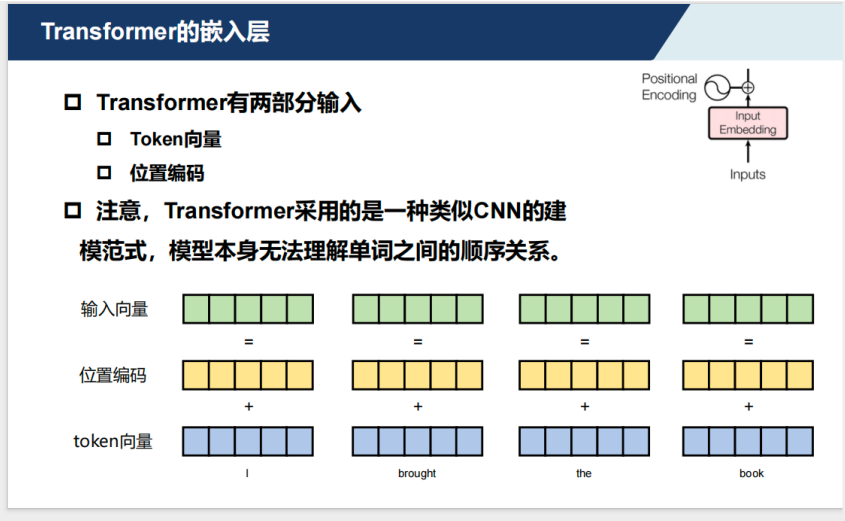


图 5.14: 嵌入层架构

模型没有像 RNN 一样的迭代循环机制，本身无法理解单词的顺序关系，所以需要一个位置向量。

关于 token 向量：

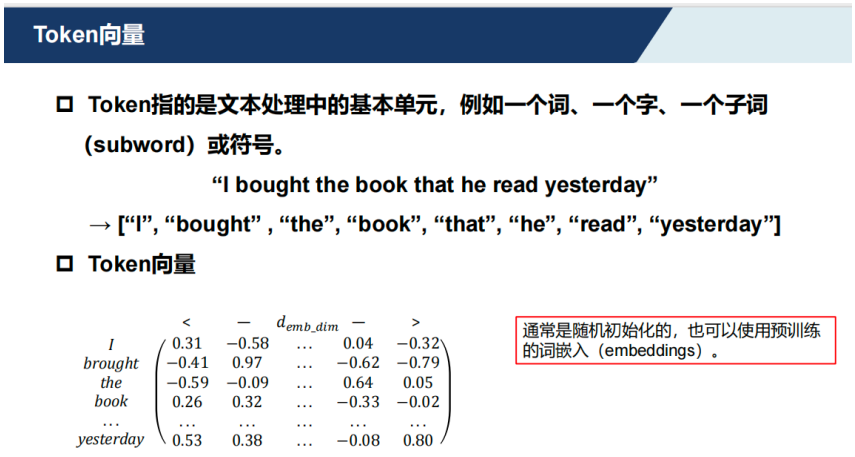


图 5.15: Token 向量

关于位置编码：如前面嵌入层架构所示，每个词对应的位置向量是与 token 加在一起的，也就是位置向量的维度与 token 向量相同。如果使用加一维单独标签的方式，对于同一个词不同位置并没有很好的区分程度。所以有如下要求：

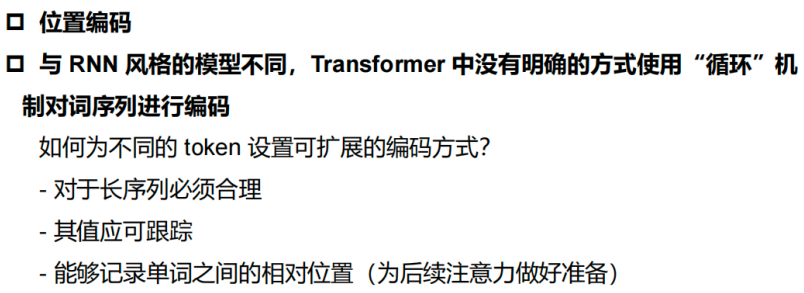


图 5.16: 位置编码要求

早期的 Transformer 使用的是正弦余弦位置编码：

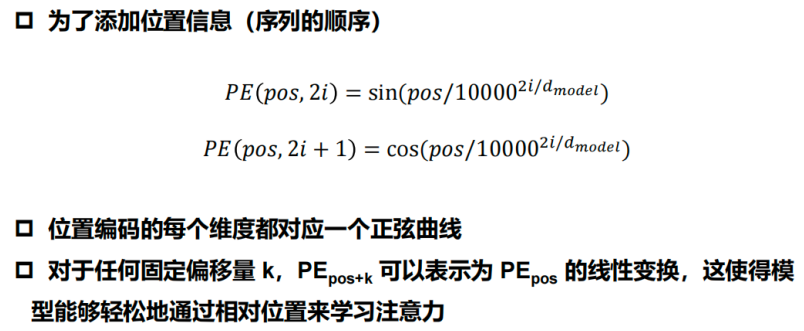


图 5.17: 正弦余弦位置编码

其中：pos 为 token 在序列中的位置（比如第 0 个、第 1 个 token）；i 为编码维度的索引（从 0 到 $d_{\text{model}}/2-1$）；d_{model} 为模型的隐藏层维度，也即 token 向量的维度（例如 512 或 1024）。偶数维度用正弦，奇数维度用余弦。

```python
import numpy as np

def positional_encoding(max_len, d_model):
    """ 位置编码核心实现
    :param max_len: 最大序列长度
    :param d_model: 模型维度（词向量/特征维度）
    :return: 位置编码矩阵 PE, 形状 (max_len, d_model) """
    PE = np.zeros((max_len, d_model))
    pos = np.arange(max_len)[:, np.newaxis]
    i = np.arange(d_model)[np.newaxis, :]
    div_term = np.power(10000, (2 * (i // 2)) / d_model)
    PE[:, 0::2] = np.sin(pos / div_term[:, 0::2])
    PE[:, 1::2] = np.cos(pos / div_term[:, 1::2])
    return PE
```

为什么要这样设计？有很多种设计方式，其有以下优势：

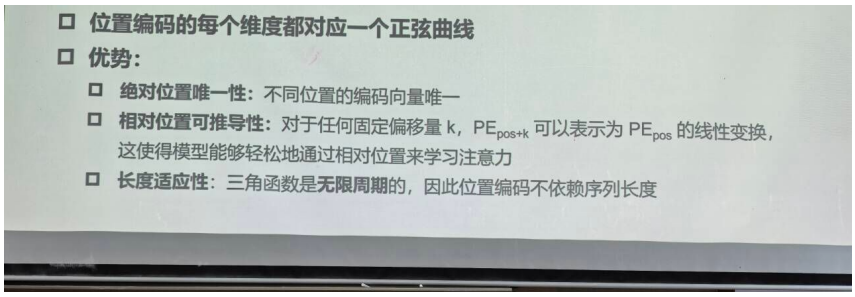


图 5.18: 正弦余弦位置编码的优势

多头注意力机制

注：要重点关注维度变化。

- 动机：信息需要从多个维度抽取意义。以文本为例，有的注意力头关注风格，而有的关注语法。所以需要多个视角。类似于多个卷积核。
- 机制：每个单词用多组不同的 W_i^Q, W_i^V, W_i^K 即可。再用线性变换进行维度拼接，最后输出的向量还是与原始词向量对齐，即与原始词向量维度相同。

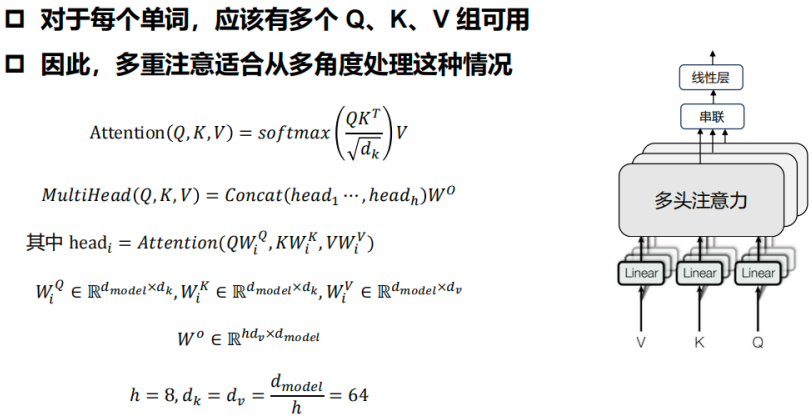


图 5.19: 多头注意力机制

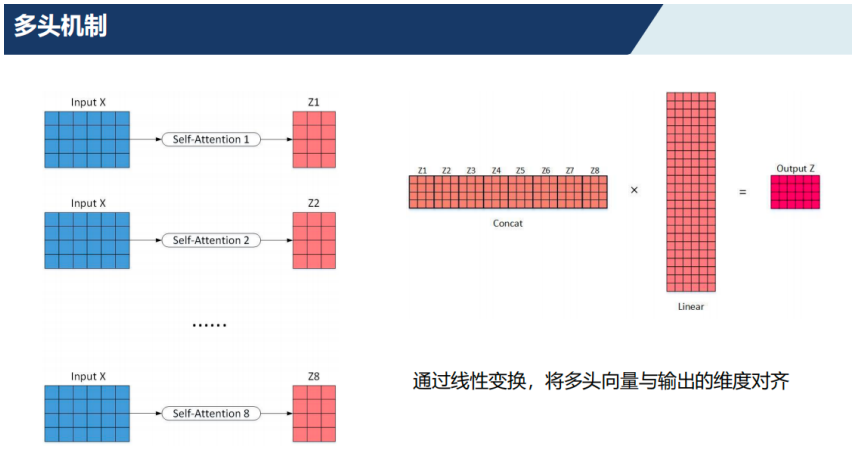


图 5.20: 多头机制详解

实际上，多头注意力机制就是将经过嵌入层后得到的向量（一条文本有多个词向量，自然就组成矩阵）与不同的 W_i^Q, W_i^V, W_i^K 相乘均等地拆分为多个不同的 Q，K，V 向量（对于一条文本序列就是矩阵）并作自注意力计算，这样得到的向量（矩阵）再分别拼接并经过线性变换投影回一开始输入多头注意力的向量的维度。

```python
import numpy as np
def multi_head_attention(X, W_Q_list, W_K_list, W_V_list, W_O):
    """ 多头注意力核心实现
    :param X: 输入矩阵 (seq_len, d_model)
    :param W_Q_list: 每个头的Q权重列表，长度为h
    :param W_K_list: 每个头的K权重列表，长度为h
    :param W_V_list: 每个头的V权重列表，长度为h
    :param W_O: 输出线性层权重 (h*d_v, d_model)
    :return: 多头注意力输出 (seq_len, d_model) """
    h = len(W_Q_list)
    heads = []
    for i in range(h):
        Q_i = np.dot(X, W_Q_list[i])
        K_i = np.dot(X, W_K_list[i])
        V_i = np.dot(X, W_V_list[i])
        head_i, _ = scaled_dot_product_attention(Q_i, K_i, V_i)
        heads.append(head_i)
    concat_heads = np.concatenate(heads, axis=-1)
    output = np.dot(concat_heads, W_O)
    return output
```

与 CNN 的对比：

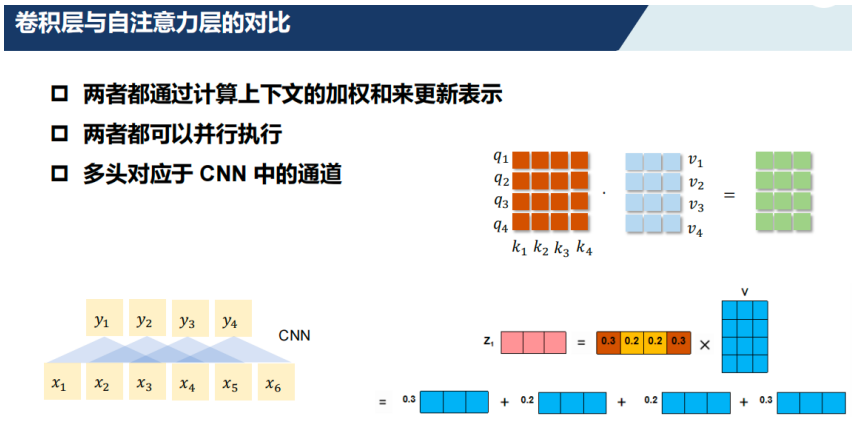


图 5.21: 多头注意力与 CNN 的对比

前馈层

原理：利用一个 ReLU 函数。类似于神经网络中的 dropout。

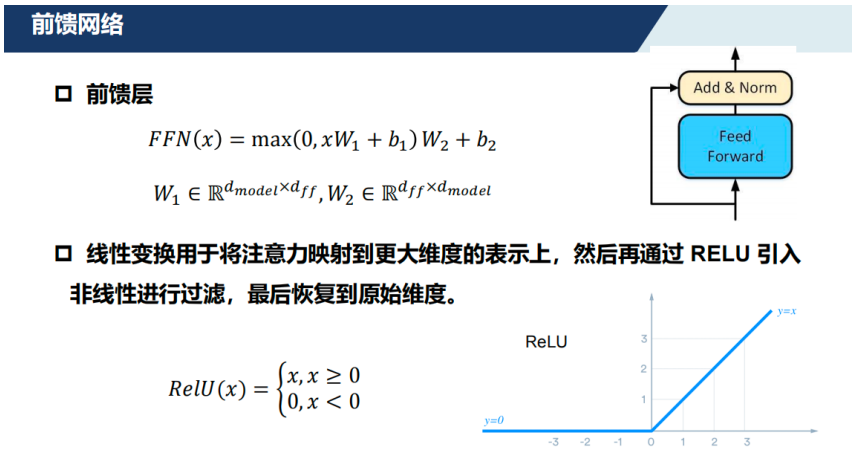


图 5.22: 前馈层

相加与归一化

这一部分内容其实是第三章神经网络优化的具体应用。

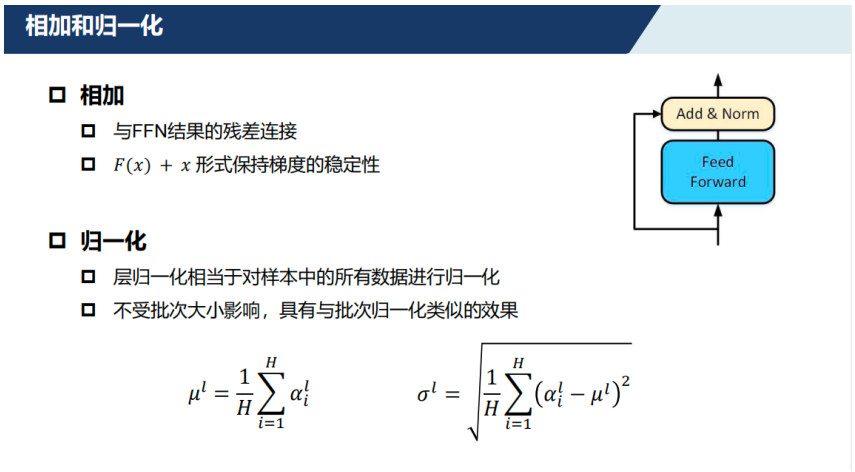


图 5.23: 相加与归一化（一）

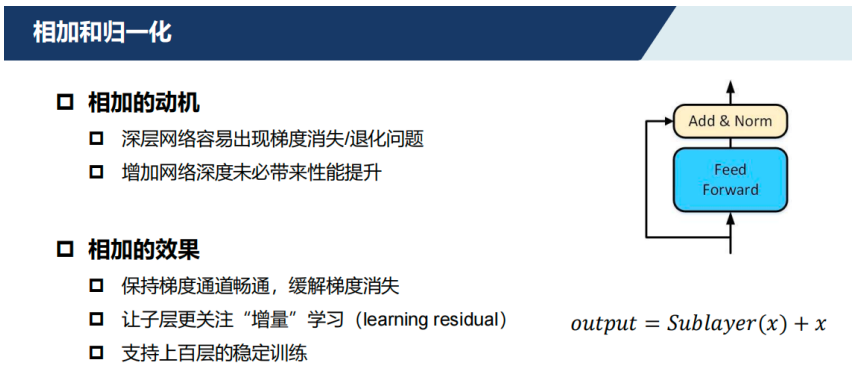


图 5.24: 相加与归一化（二）

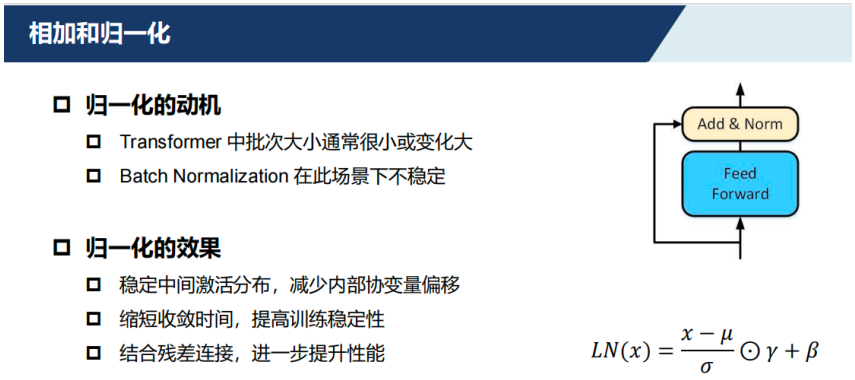


图 5.25: 相加与归一化（三）

总结

最终的输出：每一个 token 变为了上下文表示，携带全局信息，用于下游任务。

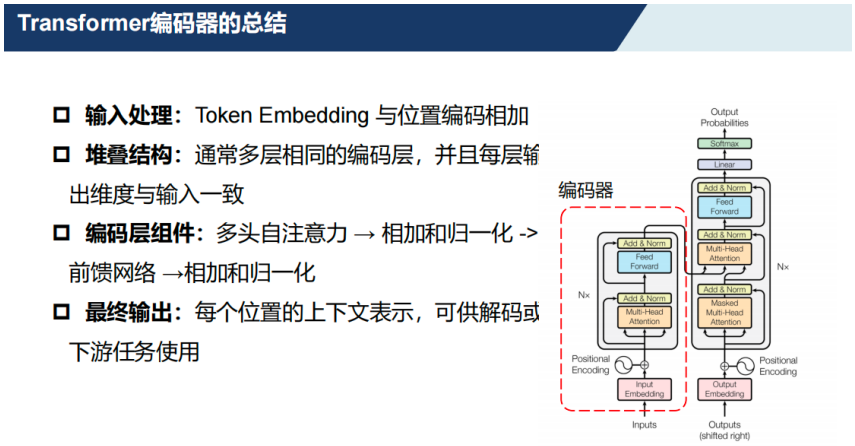


图 5.26: 编码器总结（一）

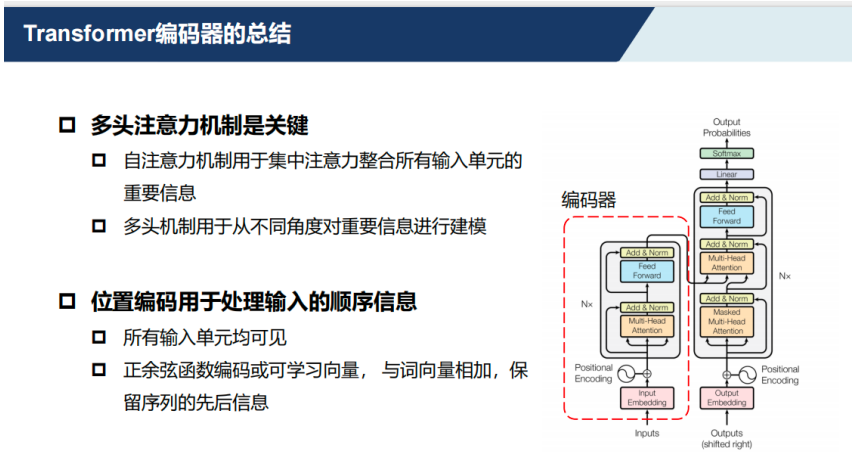


图 5.27: 编码器总结（二）

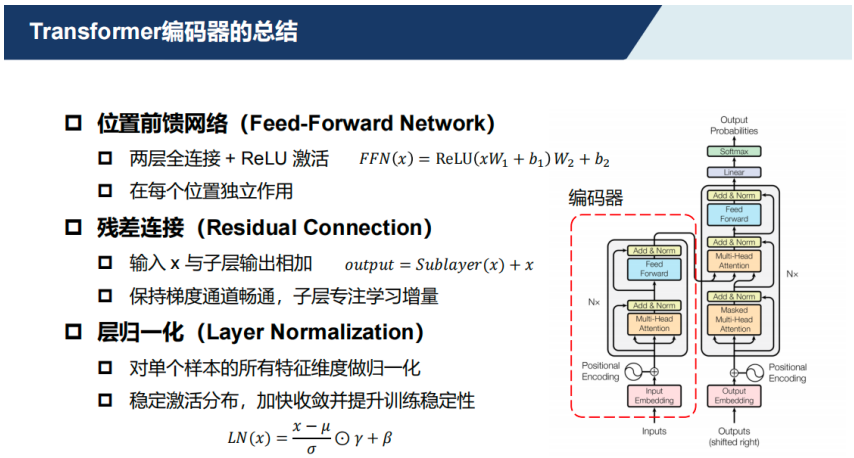


图 5.28: 编码器总结（三）

5.2.2　Transformer 的解码器

注：相对编码器而言需要关注的内容较少，也不必过分纠结具体细节。

与编码器类似，由下图（编码器右侧为解码器结构）可知，不同之处在于，多了一个 masked（掩码多头注意力机制）层，以及 multi-head 层的输入同时用了上一层以及由编码器输出的结果（交叉多头注意力机制）。

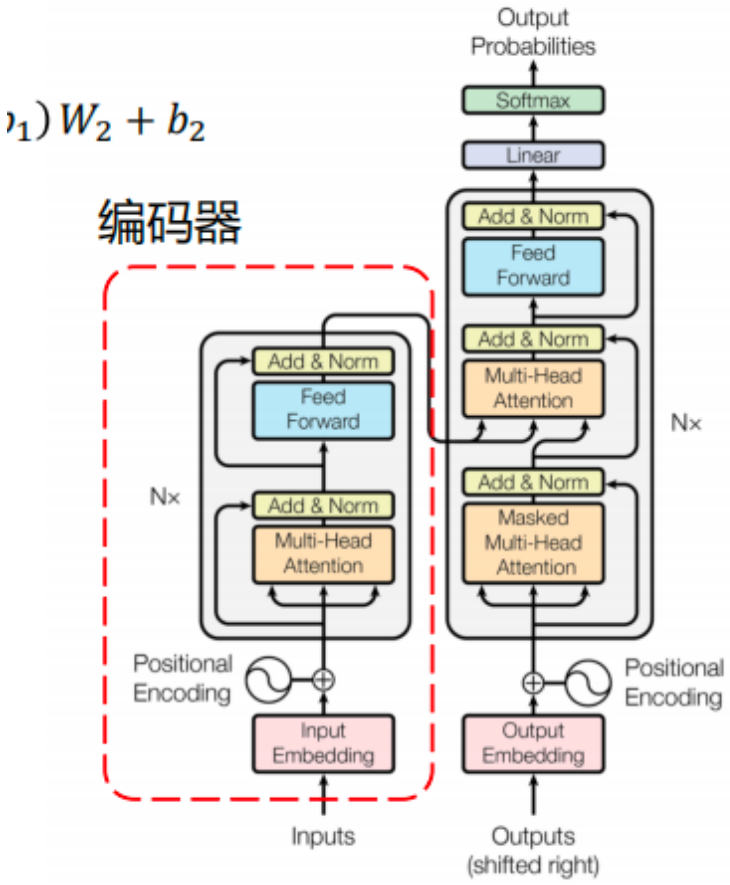


图 5.29: Transformer 解码器结构

Masked（掩码多头注意力机制）

- 解码器要进行迭代的输出，某一步的输入（Output Embedding 叠加位置编码）为之前所有步的输出，注意和来自编码器的输入是独立的。

- 在模型训练过程中，运用一种名为 teacher forcing 的训练方法。每一步预测的结果可能不正确，但是为了让下一步更好的训练，要将正确答案给下一步（而不是像测试过程中一样，将之前步的预测结果给下一步）。也就是，训练过程是并行的。这样可以提高效率。（测试的过程是迭代的）

图 5.30: 解码器训练过程

但是，这样在训练时，会导致正确答案也被送入。因此在训练的时候，要屏蔽后面的输入，否则会使得训练无效。也就是掩码注意力机制。

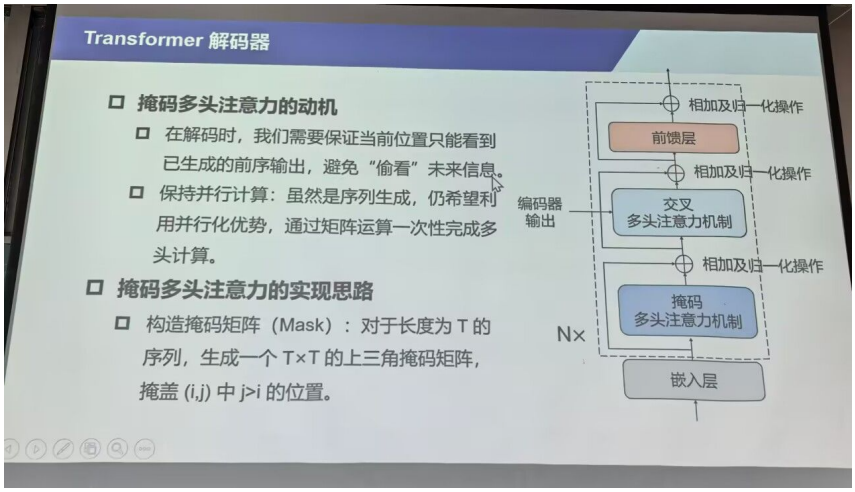


图 5.31: 掩码注意力机制原理

实现方式：置 0（引入掩码矩阵 Mask matrix），经过掩码后的 $QK^\top$ 再参与到后续的自注意力机制中。

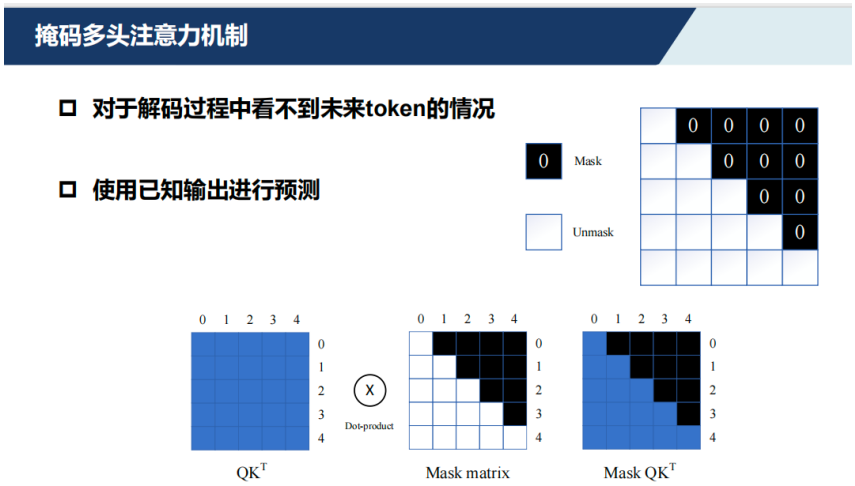


图 5.32: 掩码矩阵实现

```python
import numpy as np

def scaled_dot_product_attention(Q, K, V, mask=None):
    """缩放点积注意力（支持掩码）"""
    d_k = Q.shape[-1]
    scores = np.dot(Q, K.T) / np.sqrt(d_k)
    if mask is not None:
        scores = np.where(mask == 0, -1e9, scores)
    attn_weights = np.exp(scores) / \
        np.sum(np.exp(scores), axis=-1, keepdims=True)
    output = np.dot(attn_weights, V)
    return output, attn_weights

def create_causal_mask(seq_len):
    """生成下三角掩码矩阵（用于解码器自注意力）"""
    mask = np.tril(np.ones((seq_len, seq_len)))
    return mask  # 下三角为1（可见），上三角为0（不可见）
```

交叉多头注意力

注：这不是自注意力机制。

- 在掩码多头中的输入中，没有编码器的信息。(掩码器的输入在训练过程为正确答案，测试时是上一步的预测结果)

- 因此要加入编码器的信息，作为 K 与 V。而掩码多头的输出作为 Q，与其计算相似度，再加权归一化输出。(这里 25 秋使用的教材的图示有误，注意区分 Q，K，V 分别来自编码器还是掩码多头的输出)

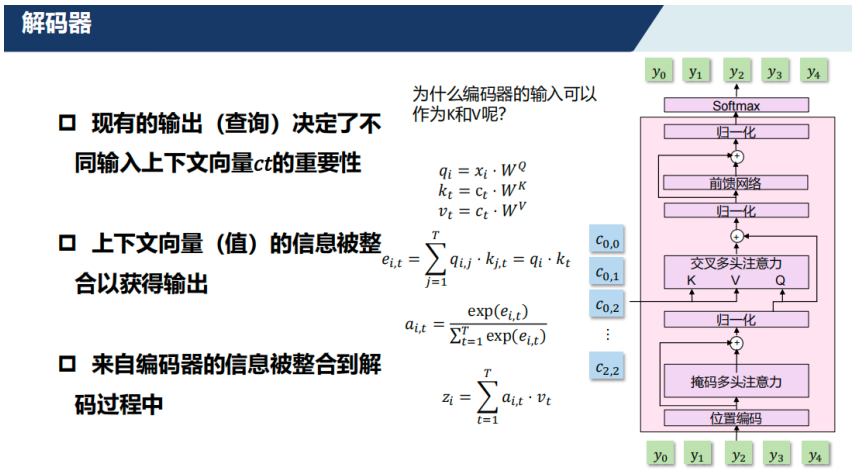


图 5.33: 交叉多头注意力（一）

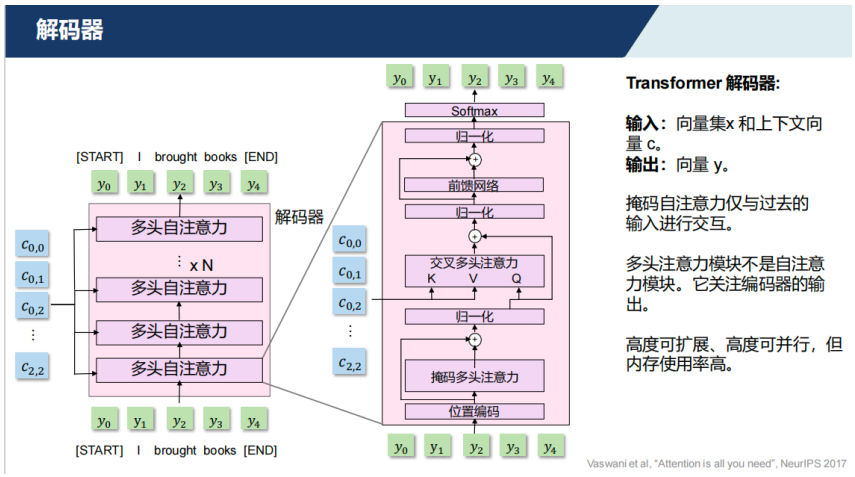


图 5.34: 交叉多头注意力（二）

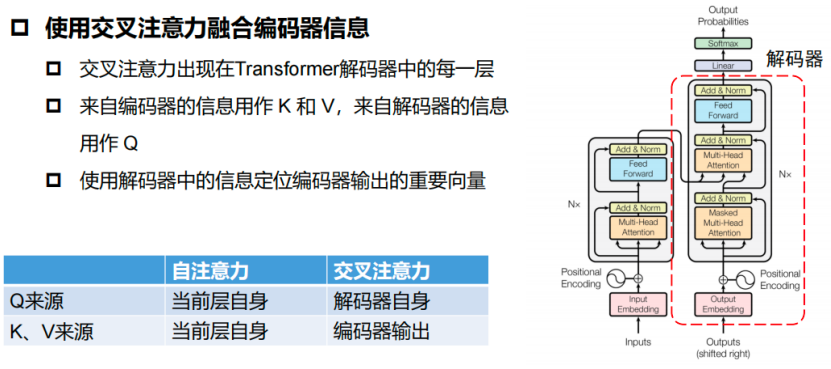


图 5.35: 交叉多头注意力（三）

5.3 Transformer 的典型应用与跨模态能力

重新理解 token 与典型应用

- **文本的 token**。用子词可以平衡文本量和语意。现在已经有成熟算法 BPE（也即前面提到的特定的拆分词元的方式）。

图 5.36: 文本 token

- **图像的 token**。传统方法直接输入的是图像的 RGB 像素值，无法使用 Transformer 编码器或者解码器。于是引入 patch 的概念。将图像划分成小块，每一个小块转化成向量再用序列堆叠。这种方式也可以应用到 DNA 序列，语音，视频等其他场景。

图 5.37: 图像 token 与 patch

- 具体应用时要考虑数据应用的形态。

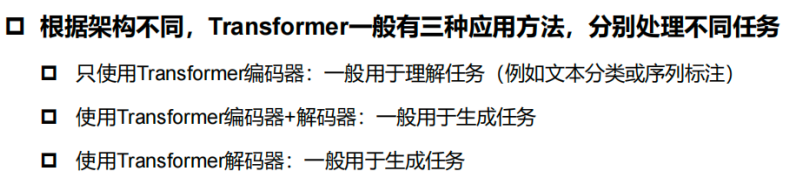


图 5.38: Transformer 应用形态

	优势	局限
编码器	双向上下文信息编码	无法做生成任务
编码器+解码器	可以做理解与生成任务	架构复杂，工程实现存在限制
解码器	可以做理解与生成任务，易于拓展模型大小	单向编码，存在理解局限

图 5.39: 编码器与解码器应用比较

- **Transformer 可以应用于各种不同类型数据的处理任务**
  - 文本、图像、音频...
- **Transformer在文本处理上的应用**
  - Transformer有多种文本处理模型变体（Transformer编码器、编码器+解码器、解模型）
  - 不同变体有各自的优势以及局限性
- **Transformer在图像处理上的应用**
  - 以编码器为主的模型架构，可应用于图像分类等任务

图 5.40: Transformer 典型应用（二）

Transformer 的跨模态推理

注：了解为主。

- 跨模态：例如，给图片拟标题，以文本生成/搜索图片。但文本和图片的编码方式（图是子图的 feature，文本是词向量）不同。
- 所以要尝试将图和文本编码到同一个空间中，并且有一个特定机制来对齐不同特征。（例如，图片中的猫对应文本中的猫）
- 方式：使用 CLIP（损失函数设计见后文）（这里的数据来源于大量已知的配对案例，可以将其打乱之后推断）。

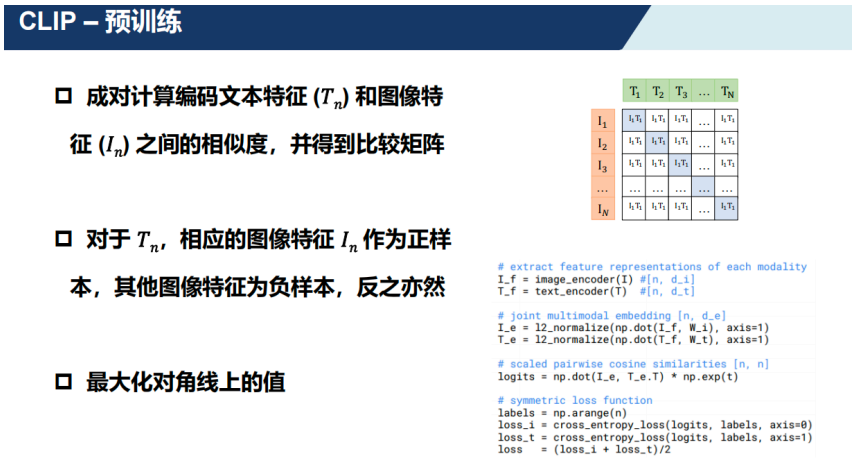


图 5.41: CLIP 模型

- 优点：鲁棒性，泛化性（例如，多个不同风格的香蕉）很好。

图 5.42: CLIP 优点

总结

核心维度	关键内容
核心机制	自注意力机制；Q/K/V 计算；多头并行计算
核心架构	编码器 + 解码器；编码器：理解输入序列；解码器：生成输出序列
关键补充	位置编码；注入序列位置信息（弥补无递归缺陷）
核心优势	并行计算（替代 RNN 串行）；长依赖捕捉（注意力无距离衰减）
典型应用	NLP（翻译/文本生成）；多模态（图文生成）；CV（ViT 图像任务）

表 5.2: 第五章总结

第六章　自监督学习

6.1　自监督学习概述

6.1.1　基本概念与范式

自然语言处理（NLP）中要求我们表示句子中的单词。表示单词最原始的想法：one-hot 向量和词汇表（这个和图表征学习节点嵌入（第4.2.1节）最原始的想法很相似）。很明显的问题在于维度大，且正交无法表示相似性。

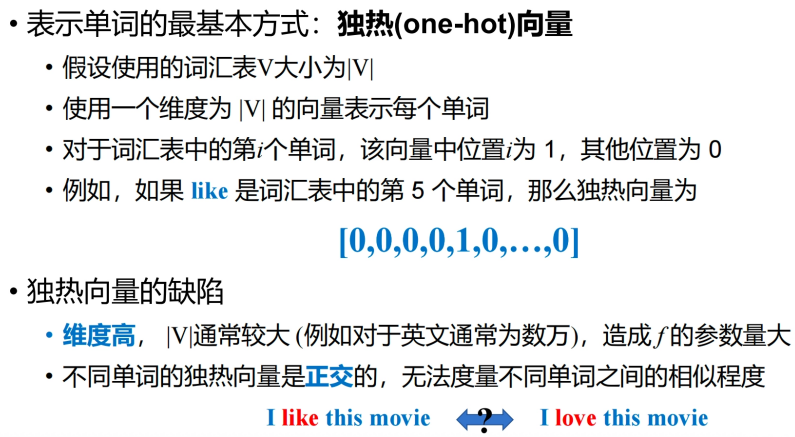


图 6.1: One-hot 表示

如何改进？构造一个模型 f_e。早期采用特征工程方式（这个在第三章中提过，具有很大局限性），现采用**通过模型学习构造 f_e，这一方法被称为表征学习**。

现代 NLP 中表征学习常采用自监督学习实现，也就是**从无标签数据集中构建标签集合，再采用与有监督学习类似方式训练**。（之前我们提过，标签指的是数据中人为标注的目标答案（监督信号））

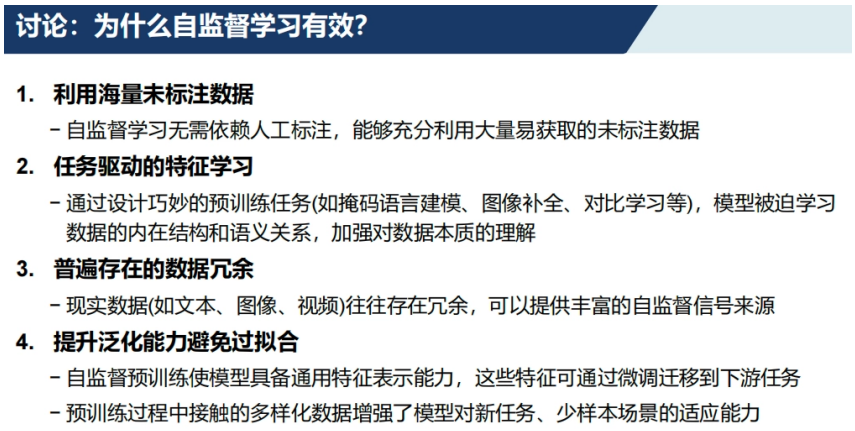


图 6.2: 自监督学习

自监督学习的训练策略

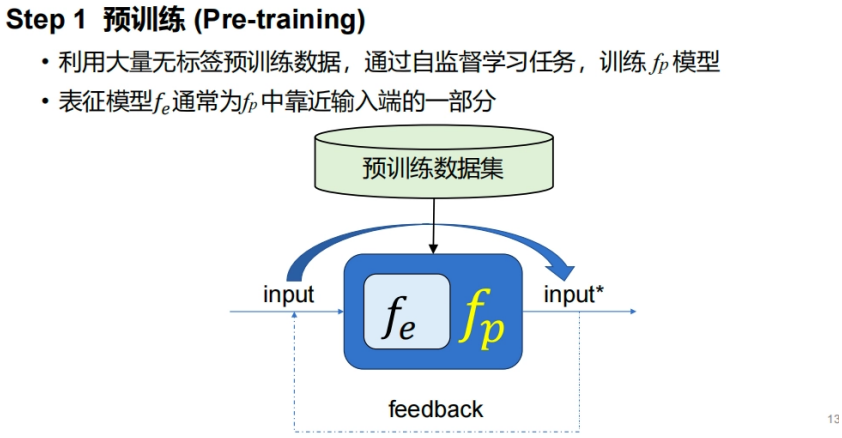


图 6.3: 预训练

1. 预训练

2. 迁移 先介绍一下，表征模型（f_e）和任务模型（f）:

- **表征模型**指的是特征提取器，将原始输入（如文本、图像、图节点）转化为富含语义信息的**低维稠密向量**（即”表征”或”嵌入”）。**是模型的核心**。
- **任务模型**指的是位于表征模型之上的**任务特定处理器**。它接收表征模型输出的特征向量，并做出最终的任务决策（如分类、回归、生成）。
- 例如，GNN 的主体是表征模型，预测头是任务模型。

迁移包括:

- 微调（固定 f_e，利用任务相关的标签数据微调 f 参数）
- 联合微调（f_e, f 都调）
- 上下文学习（不更新参数，只输入几个任务样例让模型熟悉）

- 零样本输入（不提供任何输入，常用于大语言模型）

考题提示

在 25 秋的期末考题判断中考到了上下文学习的概念，大概的表述是说上下文学习会更新参数（×）。

注意预训练和迁移不是同一个阶段的策略。

- 对于一个给定的空白模型（比如 Transformer），我们在这个具体的模型上，**执行”预训练”这个策略**——用海量无标签数据，以自监督任务（如 BERT 的掩码语言建模）为目标进行训练。

- 完成后得到了一个**权重文件**。这个文件里记录了模型所有层的参数值。这个实体，我们称之为**”预训练模型”**（例如 `bert-base-uncased`）。（注意与预训练这个策略指代有所区别）

- **迁移的对象是模型实体**：在迁移阶段，我们加载这个预训练模型的权重文件，来初始化我们下游任务的模型。之后无论是”微调”还是”特征提取”，我们操作的对象都是这个已经具备强大通用能力的、具体的模型实体。

6.1.2　自监督学习的框架

如何从无监督（无标签）数据集中构建有监督学习任务？根据任务分类，预测 → 自预测，分类 → 对比学习。通过一些预训练任务（如对比学习、掩码等）可以让模型提取数据中的特征。

自预测

自预测指的是单条样本内预测，也就是盖住一部分样本让模型自己预测。具体有以下几类：

1. 自回归生成：

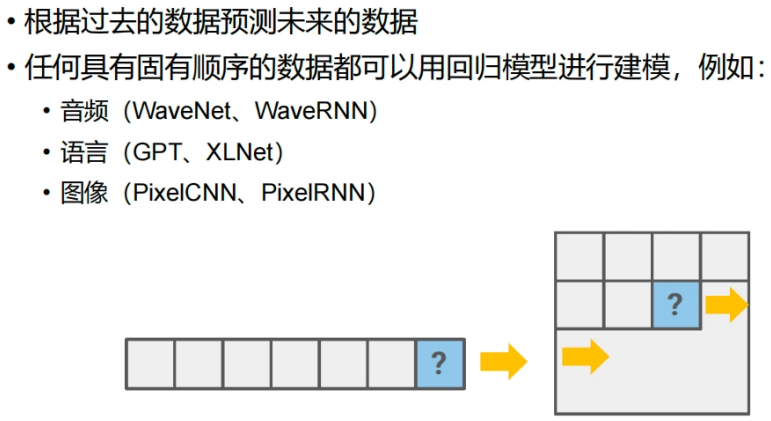


图 6.4: 自回归生成

2. 掩码生成：

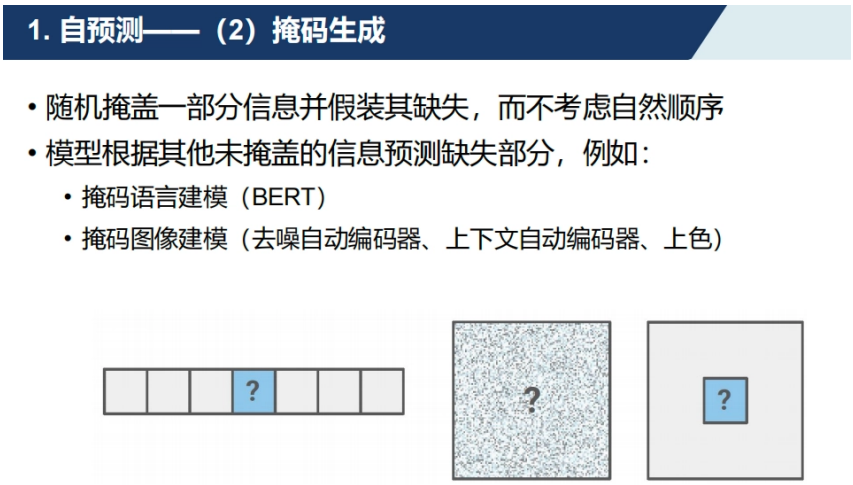


图 6.5: 掩码生成（一）

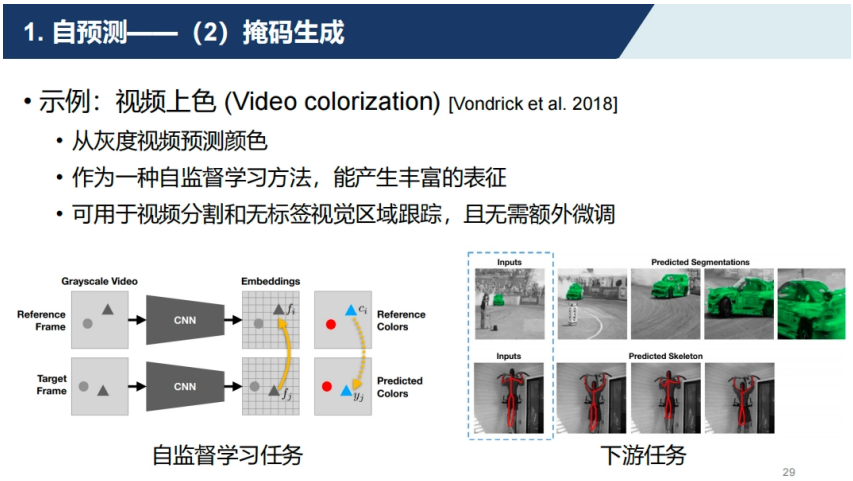


图 6.6: 掩码生成（二）

3. 内在关系预测：

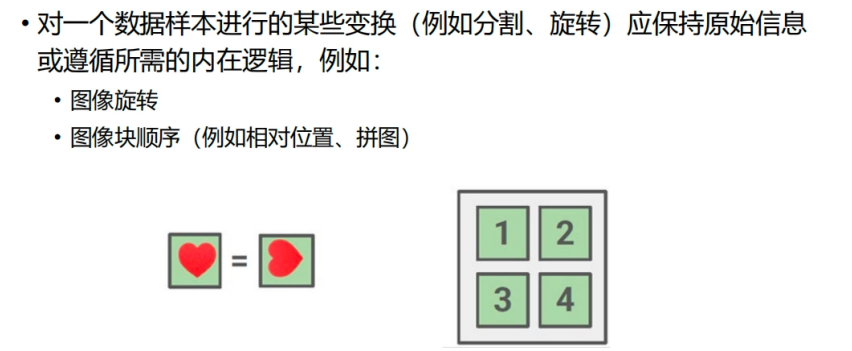


图 6.7: 内在关系预测

对比学习

对比学习关注多条样本间的数据关系，比较样本间的关系。核心是学习一个嵌入空间，使得相似样本靠近、不相似的远离。具体方式如下：

1. 样本间分类：

图 6.8: 样本间分类

2. 特征聚类：

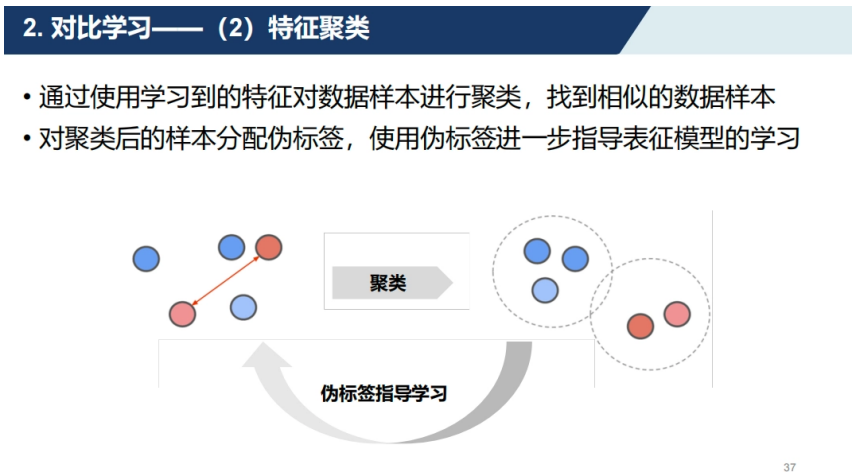


图 6.9: 特征聚类

3. 多视角编码：

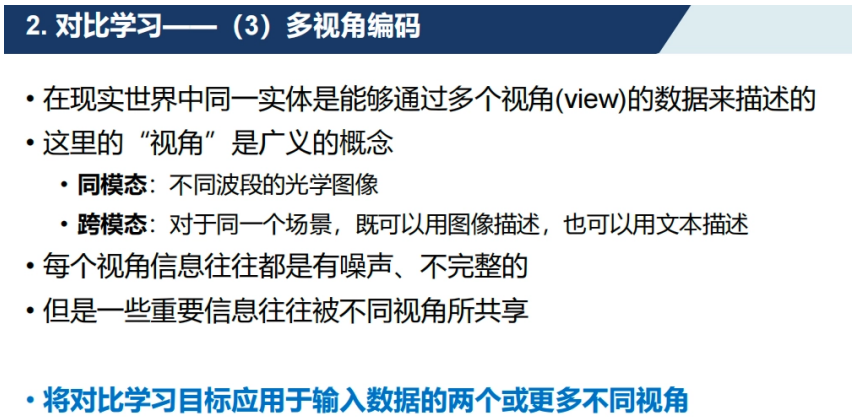


图 6.10: 多视角编码

6.2 word2vec 和 BERT 模型

词向量（单词表征向量，词嵌入，embedding）的概念如下。本节谈论的两个模型是得到单词表征向量的自监督学习模型。（地位上和 GNN 从图数据中学习嵌入是一样的）

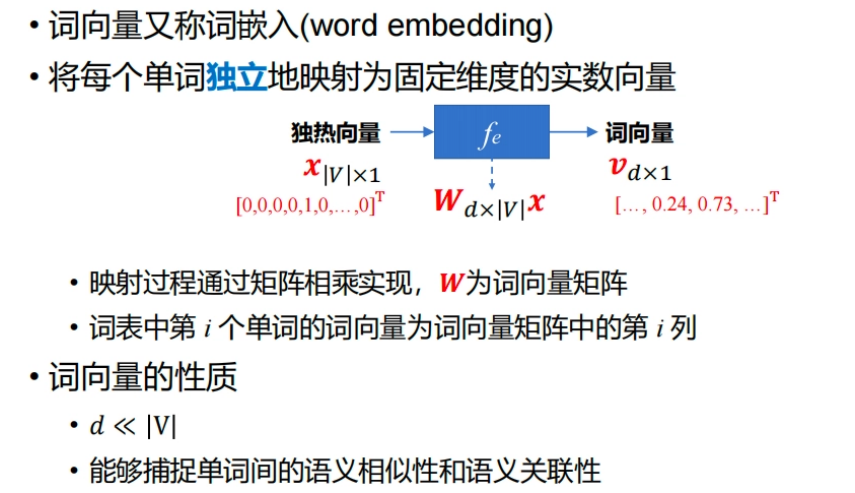


图 6.11: 词向量（embedding）

6.2.1 word2vec

注：

1. ”2” 指的是 to，类似的”4” 指的是 for（AI4Science）。

2. Word2Vec 是一个总称/工具包（模型），它提供了两种具体的模型架构来实现词向量的学习：Skip-gram，Continuous Bag-of-Words。二者目标相同但预测任务互为镜像。原理类似。本节主要介绍 Skip-gram。

3. 本部分以及下一部分 BERT 了解建模、原理即可，正常来说不太可能考计算。

模型假设

- 每一个机器学习模型都应该对处理的问题有一个假设（或者换句话说每一个模型最适合处理具有某一类特征的问题），比如 CNN 假设的是空间变换不变性，RNN 假设的是时序的延续性。GNN 的假设是图的对称性（也就是交换顶点序号，图是不变的）。
- Skip-gram 的核心假设源于分布式语义：一个单词的含义可以由它经常出现的上下文（周围的单词）来定义。（这实际上是之前提到的框架中的对比学习）

结构

- **输入层**：一个独热编码向量，用来表示中心词。
- **投影层（隐藏层）**：这里没有激活函数，仅仅是通过一个权重矩阵 W 将高维的独热向量映射为一个低维、稠密的词向量。
- **输出层**：将投影层得到的词向量与另一个权重矩阵 C 相乘，再经过一个 Softmax 函数，输出一个概率分布，表示词汇表中每个单词作为目标上下文词的可能性。

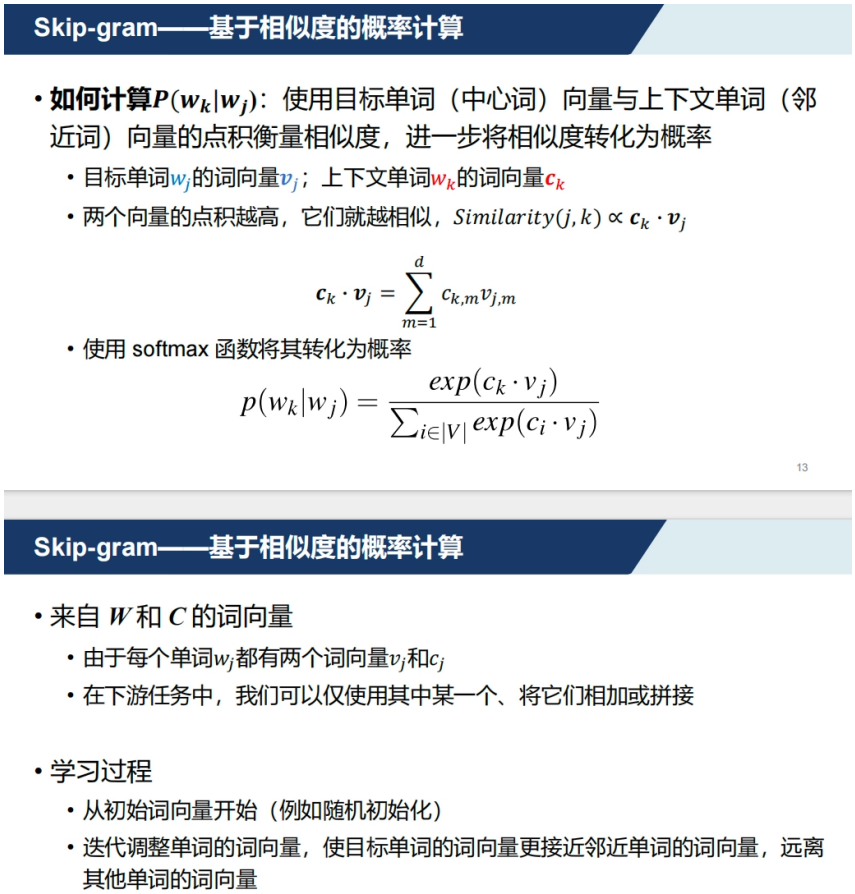


图 6.12: Skip-gram 流程

- 损失函数（负采样：学习表征时常需要正例（真实存在的关系，应该预测得分高）和负例（不相关的，应该预测得分低）。但在实际数据中负例占了绝大多）

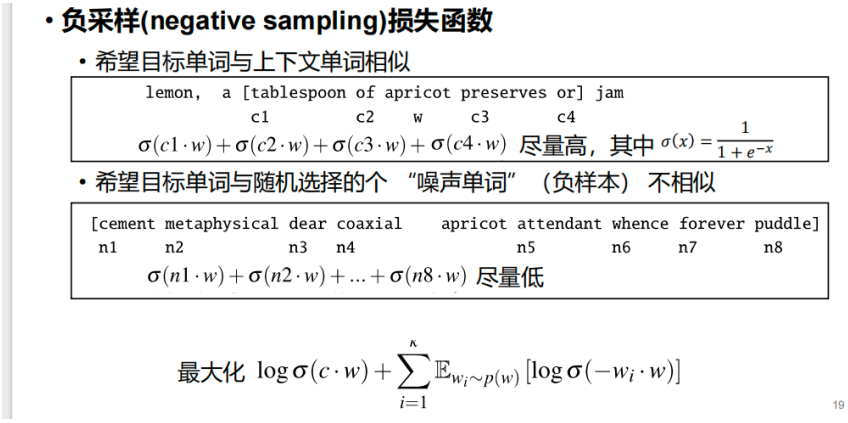


图 6.13: 负采样

关于为什么一个词要采用两个词向量：

1. **不同的语义角色**：一个单词在作为中心词和作为上下文词时，所扮演的语义角色是不同的。前者要蕴含自身完整寓意，后者要用于判断上下文特征。

2. **避免模型”作弊”与训练更稳定**：如果只使用一套词向量，那么在计算 P(银行|银行) 时，模型会简单地将同一个向量的点积最大化，这会导致训练陷入一个平凡的局部最优解。分开两个向量打破了这种对称性，迫使模型学习到更丰富的关系。

3. **更灵活的下游应用**：训练结束后，我们可以根据任务需求灵活选择使用哪种向量，或者将两者结合（相加或拼接）。通常，输入词向量 v 被更广泛地用作该单词的最终表征，因为它更直接地代表了单词本身的含义。

6.2.2 BERT 模型

- word2vec 的问题在于学习到的是每个单词的静态向量表示。这个向量试图捕捉该单词在所有可能上下文中的平均语义，而无法根据当前具体语境进行调整。当出现一词多义时无法很好的解释。

- 设计出的 BERT 模型基于 Transformer 双向编码器（传统的模型只能单向从左往右建模，因为如果简单的双向模型，模型在训练的时候会看到自己，无法学习），可以根据上下文动态调整。

- 输入输出：简单来说，输入的是部分位置被标记（mask）的句子，输出是模型对每一个位置都得出预测结果，但只对标记位置计算 loss。具体如下图：

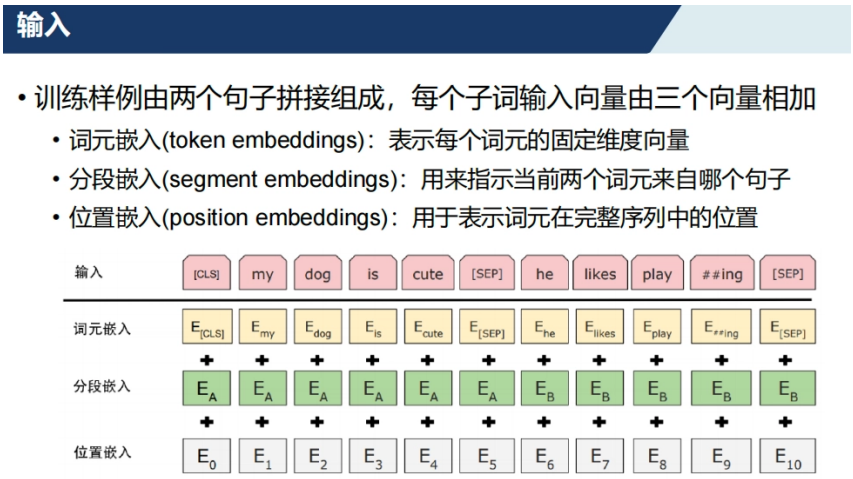


图 6.14: BERT 输入

图 6.15: BERT 输出

- 任务设置（即预训练，动态的表征学习，根据上下文学习迫使模型理解上下文的语义）：主要是掩码语言模型（MLM），原理如下：

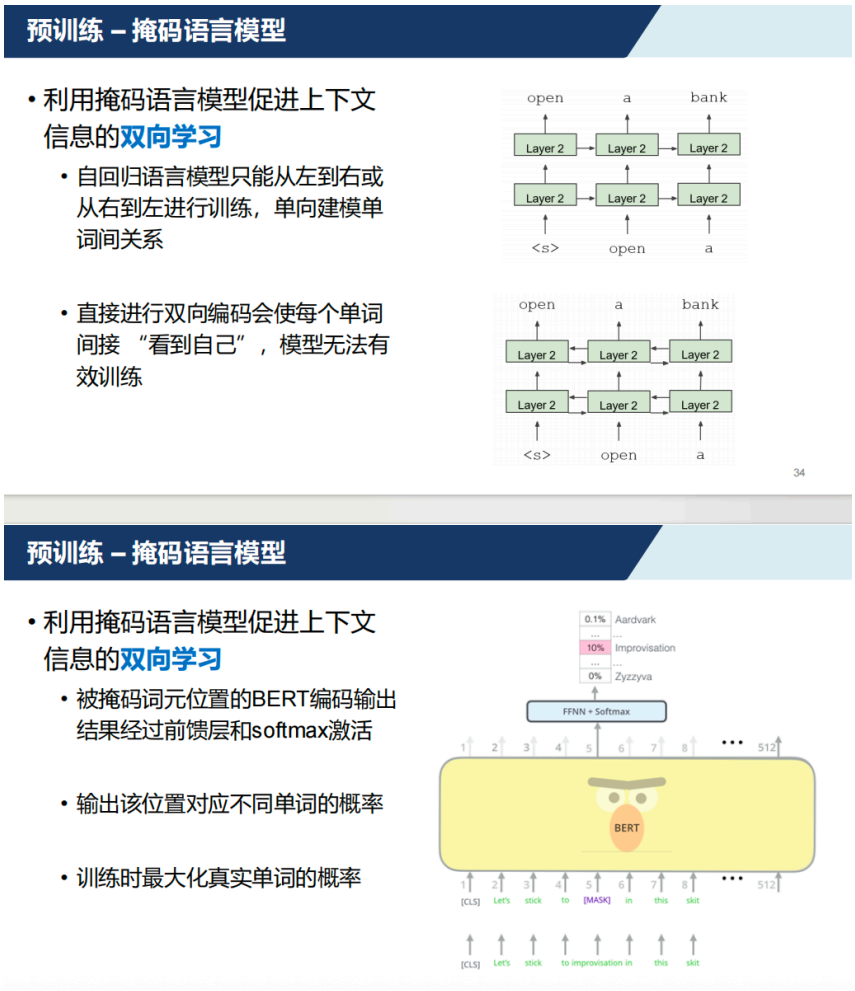


图 6.16: 掩码语言模型（MLM）

同样也存在一些问题。缓解办法是如下图所示的植入一种噪声：

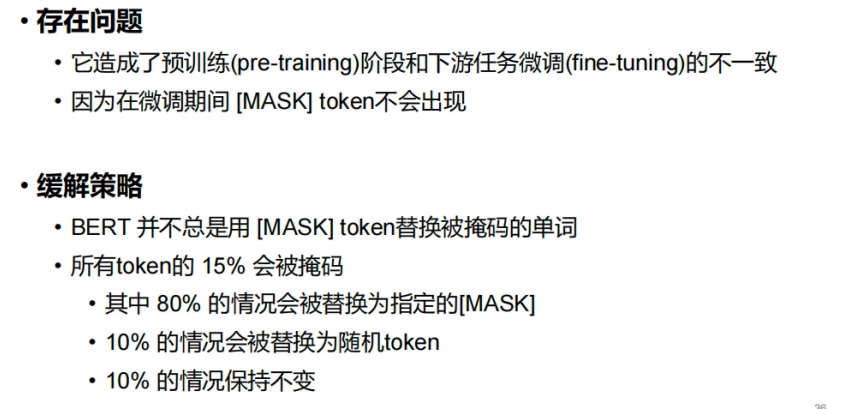


图 6.17: BERT 掩码策略

BERT 的掩码策略实际上是三种噪声类型的混合：

噪声类型	概率	模型看到的	模型任务	目的
完全掩码	12%	[MASK]	根据上下文推理	学习深度理解
随机替换	1.5%	错误单词	纠正错误并推理	提高鲁棒性
保持不变	1.5%	正确单词	仍要预测正确单词	适配下游任务

表 6.1: BERT 掩码策略

注意：这里指的是，共有 15% 的位置被标记为” 测试题”，模型看到的内容不同，但都执行**同一个任务：预测被遮盖掉的原始词**。这个过程强迫 BERT 学会了深度的上下文理解。

还有一种任务是**下一句预测（NSP）**，指的是，给模型一对句子 (A, B)，50% 是 B 真正跟在 A 后面（语料中的下一句），50% 是随机句子。模型学习判断 B 是否是 A 的下一句。这使得 BERT 学到句子级关联性（有利于问答和自然语言推断）。

微调（fine-tuning）：将预训练模型用到下游任务。之前是学习表征模型 f_e，这一步指的是下一阶段训练任务模型 f（参见第6.1.1节）。类似于 GNN 中预测头的地位。

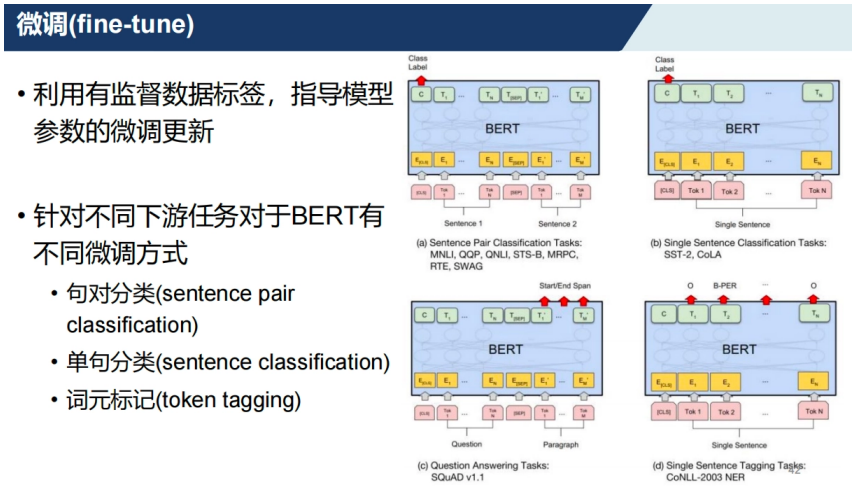


图 6.18: BERT 微调

利用有标签的数据对特定的任务训练几个 epoch（学习率比较小），对参数微调。方式如上。例如，文本分类（情感分析）在 BERT 的输出端取特殊 token [CLS] 的向量，接一个小的全连接层来做分类。

6.3　自回归语言建模

前言

6.2 和 6.3 是当前大语言模型发展的两条主线：

- **6.2（BERT 路线）**：通过**自编码**（掩码语言模型）学习，主要用于**理解**。
- **6.3（GPT 路线）**：通过**自回归**（预测下一个词）学习，主要用于**生成**。

所谓自回归语言建模本质是一种任务定义（给定前文预测下一个词），可以用多种架构实现，现代最常用的是基于 Transformer 的 GPT 系列。

- 语言模型的引入：

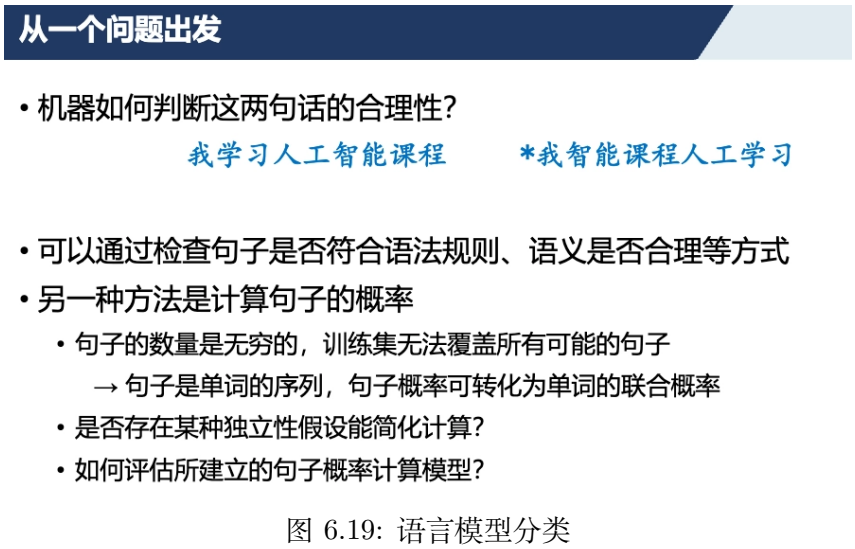


图 6.19: 语言模型分类

什么是语言模型?

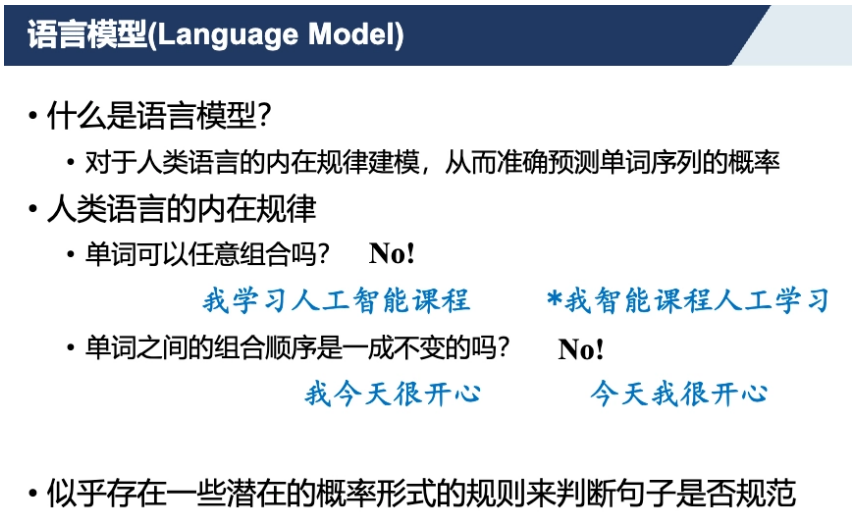


图 6.20: 语言模型

总结:

1. 语言模型（LM）是一种对文本（token 序列）的概率分布进行建模的模型。它的核心目标是学习一种概率分布，使得模型在看到一段文本（甚至是未见过的文本）时，能够准确评估其出现的可能性。

2. 应用上，用其来做文本生成。语言模型能够根据这些学到的概率来”自回归地”生成新文本。

3. 怎么建模?

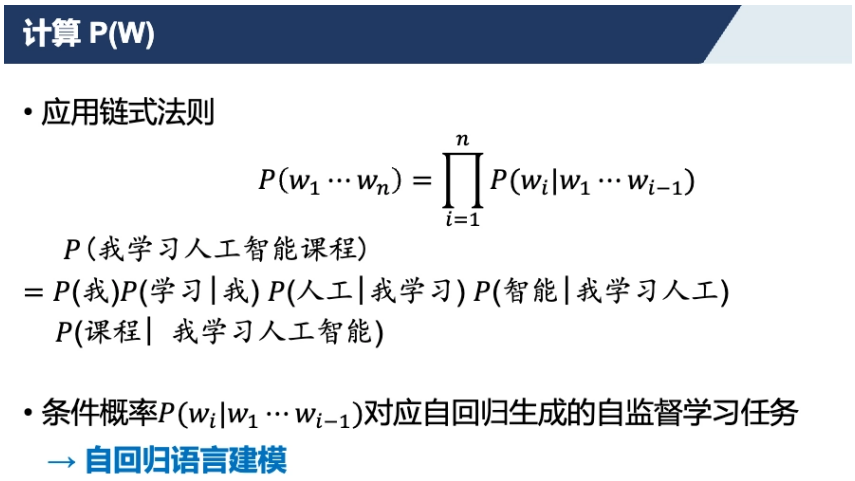


图 6.21: 联合概率

不过联合概率过于复杂，需要建模条件概率：

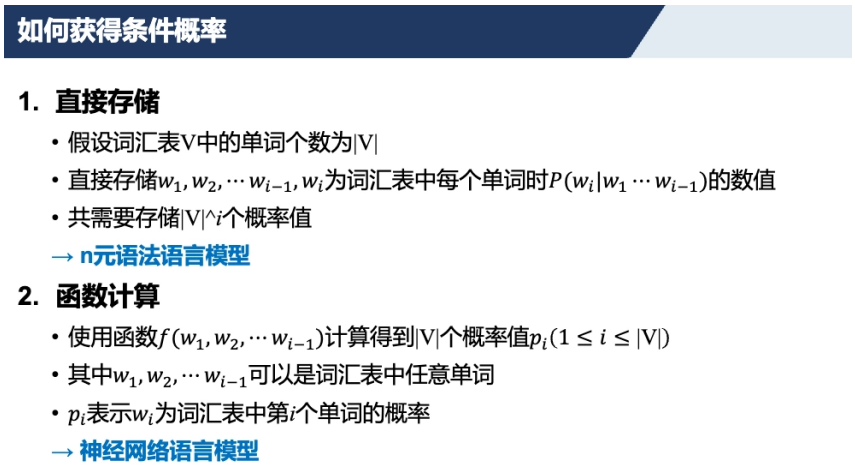


图 6.22: 条件概率

条件概率的获得主要有两种方式：直接存储（对应 n 元语法语言模型）和函数计算（对应神经网络语言模型）。

6.3.1　n 元语法语言模型

这是上图中第一种方法：直接存储。优点是查找很快，但问题在于如果查找距离过长，前面的词关联小，数据过稀疏。

改进方法：引入 Markov 假设，只基于前 n 个词，得到近似的条件概率。实际过程中 N 很小（小于等于 3），长距离处理效果差。

N-gram 模型核心：**马尔可夫假设**。一个词出现的概率，只与它前面有限个词相关。

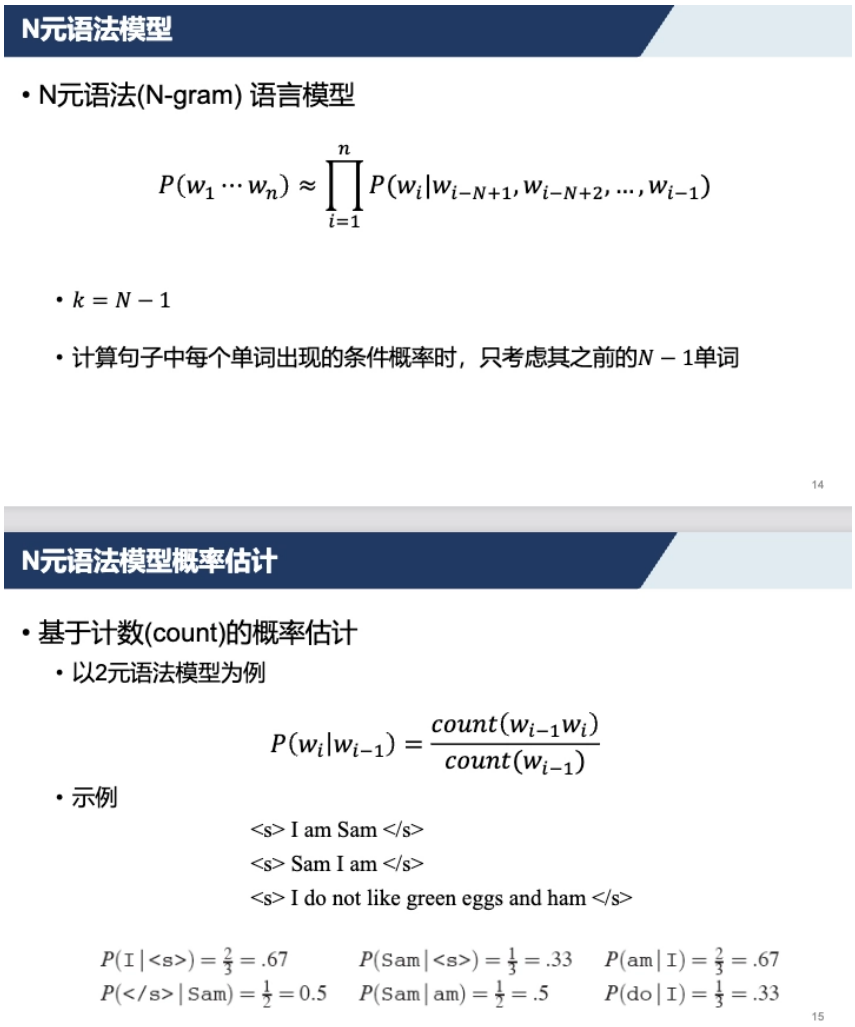


图 6.23: N-gram 模型

这是一个容易考察计算的点。25 秋就考察了这一大题（计算和困惑度的理解）。在附录中会详细讲解。

如何评估模型的性能？采用困惑度（Perplexity）。

困惑度（Perplexity）：评估语言模型好坏的指标。

考题提示

25 秋期末涉及 困惑度的定义和含义在考试中有涉及，需要理解。

N元语法模型的性能评估

- 困惑度（perplexity）
  - 困惑度是基于语言模型计算的测试集概率，再根据单词数量进行归一化得到的结果

$$\begin{aligned} \mathrm{PP}(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}} \end{aligned}$$

  - 链式法则

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

  - 最小化困惑度等同于最大化概率，最好的语言模型是能够最好地预测未见测试集的模型

16

N元语法模型的性能评估

- 困惑度（perplexity）
  - 困惑度越低，语言模型性能越好
  - 例：训练集 3800 万个单词、测试集150 万个单词（《华尔街日报》语料）

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

图 6.24: 困惑度

对困惑度的理解：

- 对某个词 w_i，模型给它的概率是 p_i。把 $1/p_i$ 看成” 等价的选项数”。例如 $p_i = 1/10$，则 $1/p_i = 10$，意思是这一步上模型相当于在 10 个等可能选项中选择（因为每个等可能选项的概率是 $1/10$）。

- 困惑度是这些” 等价选项数” 的几何平均：

$$\mathrm{PP} = \left(\prod_{i=1}^{N} \frac{1}{p_i} \right)^{1/N} = \exp\left(\text{平均的} \log \frac{1}{p_i} \right)$$

模型训练完成后，可以预测下游任务：有了条件概率分布 $P(w \mid \text{context})$，可以按概率采样或取最大概率输出，逐步生成一个句子（从 <s> 开始，按 $P(w_1 \mid \texttt{<s>})$ 采样，接着 $P(w_2 \mid \texttt{<s>}, w_1)$ ……）。

应用举例：

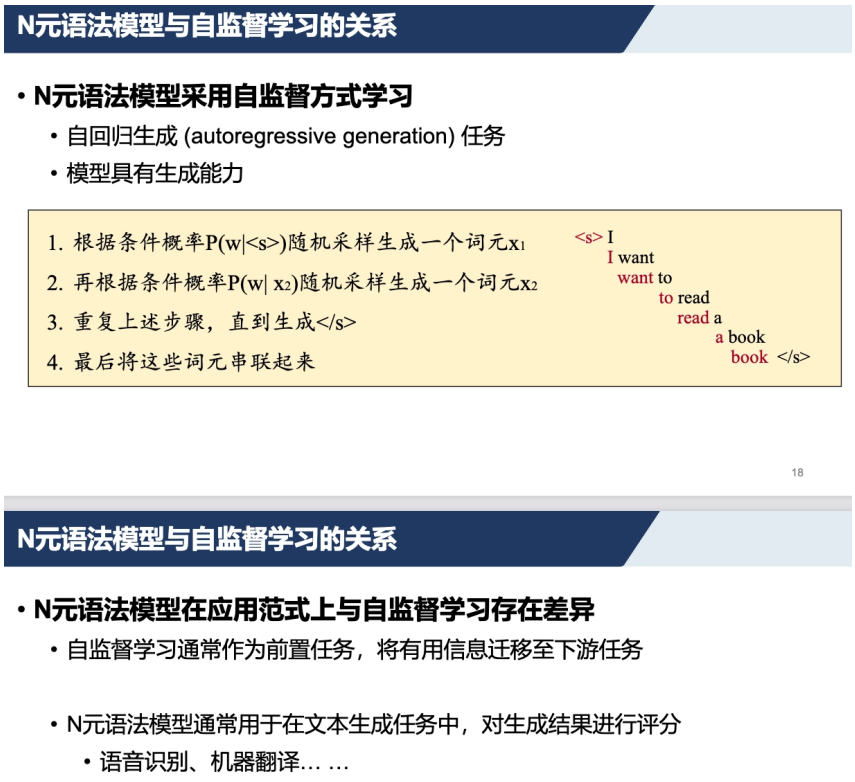


图 6.25: N-gram 模型应用

6.3.2 神经网络语言模型

核心想法：把每个词先映射成一个低维连续向量（embedding），然后用一个神经网络把最近的上下文向量映射为下一个词的概率分布——换句话说，模型学习”给定前文，哪个词更可能出现”。

神经网络语言模型的流程：

1. **输入层**：每个词表示成 one-hot 向量，对应一个索引，不含语义信息。比如词表里第 5 个词就是向量 $[0,0,0,0,1,0,\ldots]$。

2. **嵌入层**（embedding，矩阵 C）：把 one-hot 乘以嵌入矩阵 C（维度 $|V| \times d$），得到一个 d 维向量 z（词向量）。这个 C 是需要估计的模型参数。

3. **隐藏层**（上下文聚合）：把前 $n-1$ 个词的词向量（$z_1, z_2, \ldots$）拼接或平均，然后送入一个神经网络 f（RNN / Transformer 等），得到一个上下文表示 h（一个向量）。

   - 前馈 NN（Bengio 2003）通常把固定窗口的词向量拼接后经过若干全连接层得到 h。
   - RNN/LSTM/Transformer 则能处理更长或可变长度的上下文，得到上下文编码 h。

4. **输出层**：用输出向量矩阵 U（或把 h 与每个词的输出向量内积）得到对每个词的”分数”（logit），然后对所有词做 softmax，把分数转成概率：

$$P(w = v \mid \text{context}) = \text{softmax}_v(U_v^\top h + b_v)$$

   直观理解就是 h 表示”我目前的上下文是什么样”，与词表中每个词的”匹配度”做点乘，得到每个候选词的相对可能性，归一化后就是概率。

5. **生成**：按概率采样或取最大概率词，然后把新词加入上下文重复上面步骤（自回归生成）。

6.3.3　GPT 系列

GPT 系列均采用了**自回归**的架构（6.2 中 BERT 那种自掩码在这里没有直接的使用。ppt 上提到的因果掩码是自回归技术生成手段，与 BERT 的 MLM 有本质区别。）。对于 GPT 要了解一下大致架构，课件的 6.3 节花了不少篇幅解释这里。

<table>
<tr><th></th><th>N元语法语言模型</th><th>前馈神经网络语言模型</th><th>GPT-1</th><th>GPT-2</th><th>GPT-3</th></tr>
<tr><td>自监督学习目标</td><td colspan="5">自回归生成（下一词元预测）$P(w_i|w_1 \cdots w_{i-1})$</td></tr>
<tr><td>模型结构</td><td>马尔科夫假设+直接存储条件概率</td><td>马尔科夫假设+前馈神经网络</td><td colspan="3">Transformer解码器</td></tr>
<tr><td rowspan="2">应用范式</td><td rowspan="2">句子概率评分</td><td rowspan="2">句子概率评分
词向量（副产品）</td><td rowspan="2">预训练+微调</td><td colspan="2">以文本形式生成输出</td></tr>
<tr><td>零样本学习</td><td>上下文学习</td></tr>
</table>

图 6.26: GPT 系列

6.4　大语言模型简介

6.4.1　语言模型发展历程

注意，严格来说，GPT-3 才表现出现行大语言模型的性能。

（该部分可以等看完 6.4 再回头看）GPT-1 严格遵循预训练 + 微调范式。GPT-2 的原始论文中为了对比性能，作者还是做了预训练语言模型那种有监督微调，但是已经尝试用自然语言描述任务，接近 SFT，同时性能也出现了一定的通用性，可以看成是过渡。按照课件上的划分，GPT1、2 均属于预训练语言模型。

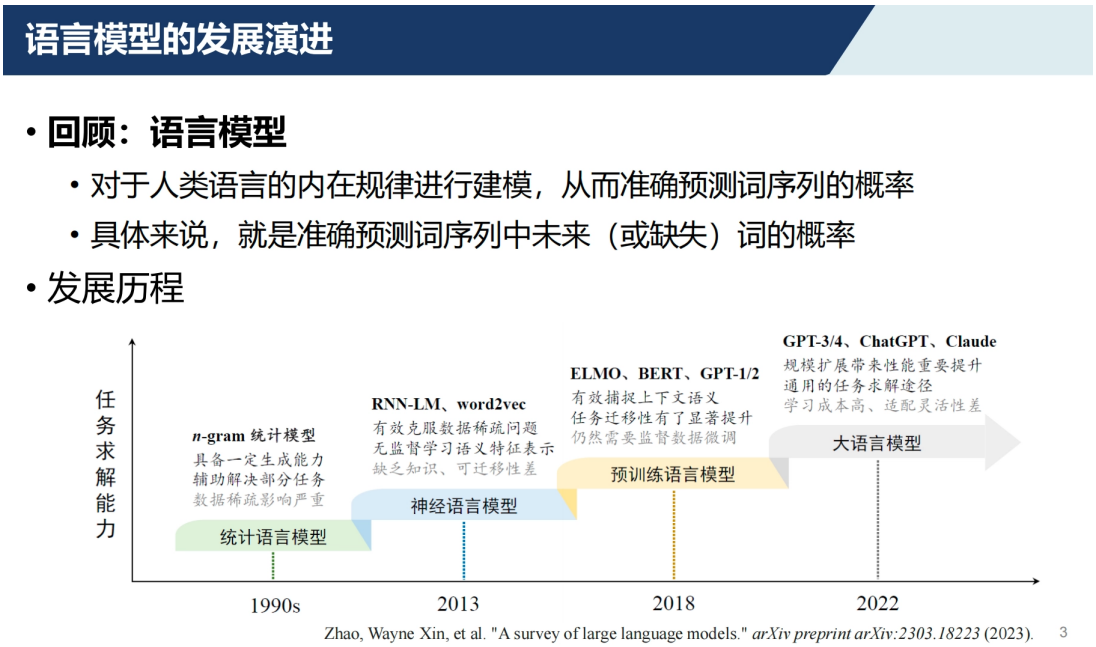


图 6.27: 语言模型发展历程

- GPT-1：证明了” 预训练 + 微调” 范式的有效性。
- GPT-2：扩大规模，发现了 zero-shot 的能力，即不需要微调就能做一些任务。
- GPT-3：进一步扩大到 1750 亿参数，展现了强大的 few-shot 能力（In-context Learning）。

6.4.2 能力特点

相比于上一代预训练语言模型等参数量较小的模型，大语言模型关键在于规模扩展的性能提升远超于模型优化。

扩展法则：函数拟合得到性能与规模、资源的关系，预测投入资源与性能提升的关系。模型性能（loss）与三个因素呈现**幂律关系**：模型参数量 N、训练数据量 D、计算量 C。

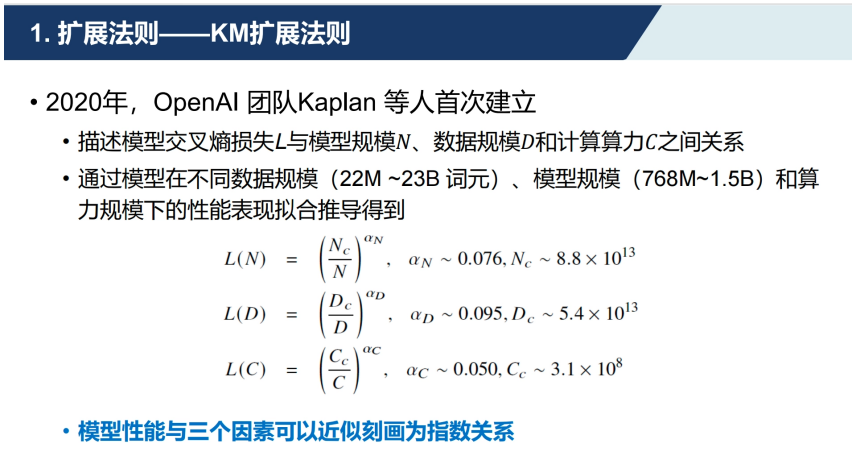


图 6.28: 扩展法则

1. 扩展法则——Chinchilla扩展法则

- 2022年，DeepMind 团队Hoffmann等人提出
  - 指导大语言模型充分利用给定的算力资源进行优化训练
  - 更大范围模型规模（70M~16B）和数据规模（5B~500B 词元）进行实验拟合
  - 模型性能与模型规模、数据规模之间的**另一种指数关系**

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

其中，E、A、B、α、β 为常数

  - 进一步获得获得算力资源固定情况下**模型规模与数据规模的最优分配方案**

$$N_{\text{opt}}(C) = G\left(\frac{C}{6}\right)^{a}, \quad D_{\text{opt}}(C) = G^{-1}\left(\frac{C}{6}\right)^{b}$$

其中，G、a、b 为常数

图 6.29: 扩展法则（续）

扩展法则体现了大模型的涌现能力。模型扩展到一定规模时，性能远远提升。（但是现在存在缺乏理论解释。）典型的涌现能力包括上下文学习，指令遵循，逐步推理。

涌现能力带来了根本的进步：**预训练语言模型**时代，最终交付物是针对每个任务微调后的**模型文件**，**不同任务需要不同的模型**。而**大语言模型**时代，最终交付物是一个通用的**基础模型**，而用户通过不同的**提示词**来”调用”它的不同能力。

6.4.3　构建方法

这里我们与上一时代的预训练语言模型（如 BERT（自掩码）GPT1（自回归），都是预训练 + 微调）加以对比。大语言模型的训练一般分为三个阶段：

预训练（Pre-training）

基本逻辑与预训练语言模型时代相同（都是自回归），但是规模效应引发了质变。在海量无标注文本上进行自回归语言建模，学习通用的语言知识和世界知识。

1. 大规模预训练

- 预训练任务：“预测下一个词元”
  - 绝大部分大语言模型广泛采用
  - 给定一个词元序列$\boldsymbol{u} = \{u_1, \ldots, u_T\}$，基于序列中当前位置之前的词元序列$\boldsymbol{u}_{<t}$，采用自回归的方式对于目标词元$u_t$ 进行预测，即最大化以下概率

$$\mathcal{L}_{\text{LM}}(\boldsymbol{u}) = \sum_{t=1}^{T} \log P(u_t | \boldsymbol{u}_{<t})$$

  - **变体：前缀语言建模**，配合前缀解码器架构
  - 每个文本序列$\boldsymbol{u}$ 会根据随机选择的位置k $(1 \le k \le T)$ 切分为前缀$\boldsymbol{u}_{\text{prefix}} = \{u_1, \ldots, u_k\}$和后缀$\boldsymbol{u}_{\text{suffix}} = \{u_{k+1}, \ldots, u_T\}$，仅后缀中的词元损失会被计入总损失

$$\mathcal{L}_{\text{Prefix}}(\boldsymbol{u}) = \log P(\boldsymbol{u}_{\text{suffix}} | \boldsymbol{u}_{\text{prefix}}) = \sum_{t=k+1}^{T} \log P(u_t | \boldsymbol{u}_{<t})$$

图 6.30: 预训练

有监督微调（Supervised Fine-Tuning, SFT）

注意，这个微调与上一时代的微调（fine-tuning）有本质的区别。上一时代的 fine-tuning 是为了适配具体任务，这里是行为对齐，让模型学会遵循人类指令和对话的通用格式。使用人工标注的高质量”指令–回答”对数据，对预训练模型进行微调，使其学会遵循人类指令。

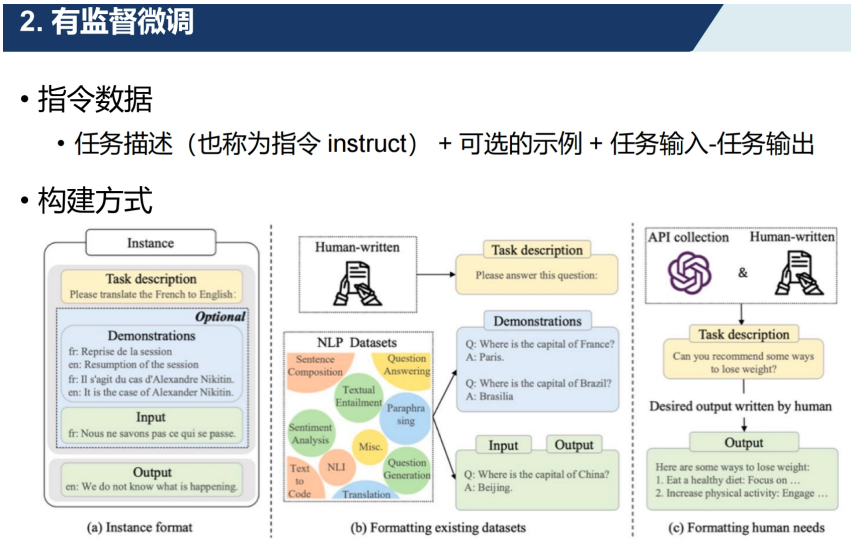


图 6.31: 有监督微调（SFT）

基于人类反馈的强化学习（RLHF）

”学会做人”。通过**基于人类反馈的强化学习（RLHF）**等技术，让模型的回答更符合人类的偏好（有用、诚实、无害）。上一时代无此步骤。通过人类偏好反馈训练奖励模型（Reward Model），再用强化学习（如 PPO 算法）对 LLM 进行优化，使其生成更符合人类价值观和偏好的回答。

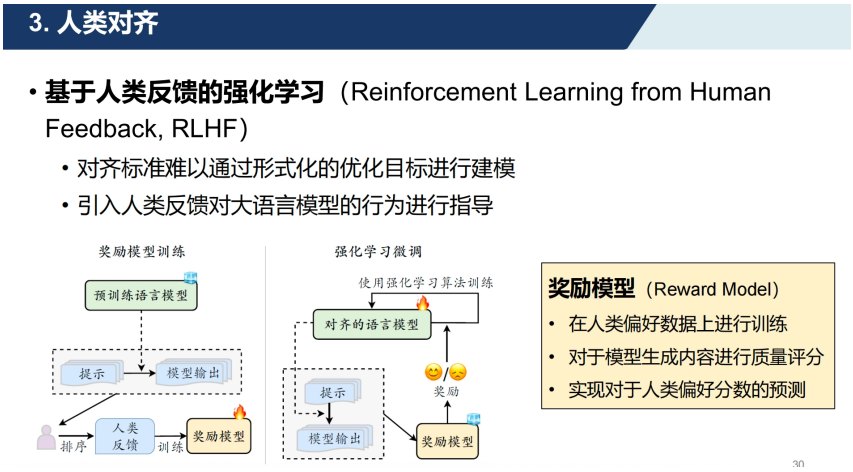


图 6.32: 基于人类反馈的强化学习（RLHF）

第二部分

实验部分

实验部分

实验部分

待完成

本部分将主要分享几个大作业的完成经验。

由**史睿铭**（24 少院数学）负责编写。

另附：**陈明珠**（24 少院 AI）自建的 GitHub 仓库，包含了 25 春的部分大作业、作业等资料，供参考：

https://github.com/Brilliant-Pig-CN/ustc-Mathematical-Principles-and-Algorithms-of-Artificial-Intelligence

附录 A　重要算法

A.1　全连接与卷积神经网络参数量计算

在计算神经网络的参数量之前，我们要搞清楚一个关键的问题：哪些是神经网络的参数？答案是：**所有可以被训练的量（参数）**。这样看来，问神经网络的参数量，其实是问我们：**我们在模型中到底训练的是哪些量？**

1. **全连接神经网络**：全连接神经网络训练的是每一层的矩阵和偏置项。由于每一层的算法都是一样的我们只写出来一层。设一层的输入量为 n_{in}，输出量为 n_{out}，则矩阵大小为 $n_{\text{in}} \times n_{\text{out}}$，偏置项（如果题目明确说不考虑则不用加上这一项）的大小即为输出量的大小，即 n_{out}，因而

$$\text{（每一层的）参数量} = n_{\text{in}} \times n_{\text{out}} + n_{\text{out}}$$

2. **卷积神经网络**：卷积神经网络在卷积层部分训练的是**卷积核和层间偏置项**。需要注意的是：池化层中没有可训练参数。既然这样，那么我们只需要用层的相关参数把卷积核的大小写出来就行了。我们仍然只计算一层，并用卷积核的高度 K_h，宽度 K_w，输入通道数 C_{in}，输出通道数 C_{out} 表示，因此我们有：

$$\text{（每一层的）参数量} = K_h \times K_w \times C_{\text{in}} \times C_{\text{out}} + C_{\text{out}}$$

和上面一样，如果题目明确说不考虑偏置项最后一项就不用考虑。

例题：

1. **在如下左图所示的四层全连接神经网络中，输入层维度为 160,000，第一个隐藏层维度为 40,000，第二个隐藏层维度为 2,000，输出层维度为 100。请计算该神经网络的参数总量。**
2. **在如下右图所示的卷积神经网络中，输入层维度为 160,000，第一个卷积层使用 3 个 5*5 的卷积核，使用max pooling 将特征图尺寸降为 40,000, 第二个卷积层使用 5 个 4*4*3 的卷积核，紧接着使用 max pooling 将特征图尺寸降为 2,000，第三个卷积层使用 5 个 4*4*5 的卷积核，使用 max pooling 将特征图尺寸降为 400，在卷积过程中始终使用padding保持输入和输出维度一致，最后一层使用全连接层将维度展平后降为 100。请计算该神经网络的参数总量。**

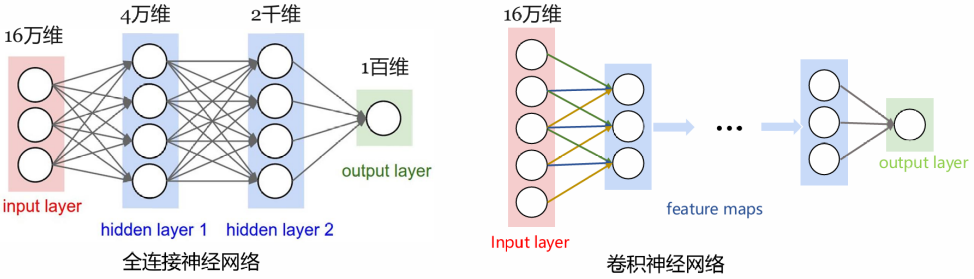


图 A.1: 神经网络参数量计算例题

对第一题而言，代入前面的公式，参数总量为

$$(160000 \times 40000 + 40000) + (40000 \times 2000 + 2000) + (2000 \times 100 + 100) = 6{,}480{,}242{,}100$$

第二题，本题中，参数主要由两部分构成：卷积部分要训练的卷积核以及后面全连接部分要训练的全连接矩阵。后者将 5 层 400 维的图像展平成 100 维的向量，这部分可以视为输入部分大小为 5×400，输出部分大小为 100 的全连接网络，因此整个神经网络总的参数量为：

$$[(5 \times 5 \times 1 \times 3 + 3) + (4 \times 4 \times 3 \times 5 + 5) + (4 \times 4 \times 5 \times 5 + 5)] + (5 \times 400 \times 100 + 100) = 200{,}828$$

A.2　逻辑回归——二分类问题

A.2.1　预测值的计算

二分类问题中，一般的流程是先将样本代入特征提取器，将特征提取器进行线性变换（可能含偏置项）后，得到的值代入函数（通常是 Sigmoid 函数：$\sigma(x) = \dfrac{1}{1+e^{-x}}$，目的是将 $(-\infty, +\infty)$ 上的函数归一化到 $(0, 1)$ 上）得到预测样本标签为 1 的概率，并与分类平面进行比较。具体来说，设样本为 $x = [x_1, \ldots, x_n]$，特征提取器为 $\phi(x) = [\phi_1(x), \ldots, \phi_n(x)]$，线性变换的参数为 $w = [w_1, \ldots, w_n]$，偏置项为 $b = [b_1, \ldots, b_n]$，最后代入的函数为 $\sigma(x)$，则最后的预测概率为 $\sigma(w^T x + b) = \dfrac{1}{1+e^{-(w^T x+b)}}$，得到的就是预测样本标签为 1 的概率。相反的，如果要预测样本标签为 0 的概率，则为 $1 - \sigma$。

A.2.2　优化与正则化

二分类问题优化的目标函数一般是**交叉熵**，其定义为

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

通过观察式子，我们能发现：分类问题中影响交叉熵损失的只有对真实标签的预测概率。如在这个二分类问题中：当 $y_i = 1$ 即真实标签为 1 时，$1 - \hat{y}_i$ 对应一项为 0，只有对真实标签的预测概率 $\hat{y}_i$ 不为 0，影响了交叉熵损失。

为了防止过拟合，常进行 L1 或 L2 正则化以控制参数大小。如 L2 正则化后的目标函数为：

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

A.2.3　样例

1.(15 分) 某银行需构建贷款违约预测模型 (二分类问题, 标签 y=+1 表示违约,y=-1 表示不违约), 基于以下场景回答问题: 某银行需构建贷款违约预测模型 (二分类问题, 标签 y=+1 表示违约,y=-1 表示不违约), 基于以下场景回答问题:

- 特征包括: 年收入 (x_1)、贷款金额 (x_2)、贷款利率 (x_3).
- 采用逻辑回归模型进行验证, 特征提取器为 $\Phi(x) = [1, x_1, x_2, x_3]$.

请回答以下问题:
(1) 写出逻辑回归表达式, 若模型训练后参数为 $w = [w_0, w_1, w_2, w_3] = [1, -0.5, 0.8, 1.2]$, 分别对应 $[1, x_1, x_2, x_3]$ 的权重, 计算样本 $\mathbf{x} = [1, x_1, x_2, x_3] = [1, 5, 3, 0.1]$ 的违约概率, 并判断是否违约. (5 分)

图 A.2: 逻辑回归问题例题

(1) 逻辑回归表达式如下:

$$P(y = +1 \mid x, w) = \frac{1}{1 + e^{-w^T x}}$$

其中 $w^T x = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3$
给定模型参数 $w = [w_0, w_1, w_2, w_3] = [1, -0.5, 0.8, 1.2]$
计算样本 $x = [1, 5, 3, 0.1]$ 的预测值:

$$w^T x = 1 \cdot 1 + (-0.5) \cdot 5 + 0.8 \cdot 3 + 1.2 \cdot 0.1 = 1 - 2.5 + 2.4 + 0.12 = 1.02$$

对应概率为:

$$P(y = +1 \mid x, w) = \frac{1}{1 + e^{-1.02}} \approx \frac{1}{1 + 0.36} \approx 0.735$$

违约概率为 0.735，大于 0.5，因此预测为违约 $y = +1$。

A.3　反向传播算法（含例题）

反向传播算法是训练神经网络的核心算法，其作用是：**高效计算损失函数对所有参数的梯度**。其基本思想是利用**链式法则 (Chain Rule)**，从输出层开始，将误差逐层向前传播，从而得到每一层参数的梯度。所以如果时间紧迫记不住公式或者考场上忘了这堆推导，请各位科大学子相信自己的数理基础，

设神经网络共有 L 层，第 l 层的线性变换为

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

其中 $W^{(l)}$ 为权重矩阵，$b^{(l)}$ 为偏置项，$a^{(l-1)}$ 为上一层的输出。
激活函数记为 $\sigma(\cdot)$，因此第 l 层输出为

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

最终输出为 $a^{(L)}$，损失函数记为 $\mathcal{L}$。

A.3.1 输出层误差

首先计算输出层的误差项

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

若使用 **Sigmoid + 交叉熵损失**，可以得到一个非常简洁的结果：

$$\delta^{(L)} = a^{(L)} - y$$

其中 y 为真实标签。

A.3.2 隐藏层误差传播

对隐藏层而言，根据链式法则有

$$\delta^{(l)} = \left((W^{(l+1)})^T \delta^{(l+1)}\right) \odot \sigma'(z^{(l)})$$

其中 $\odot$ 表示逐元素乘法（Hadamard 积），$\sigma'(z^{(l)})$ 为激活函数的导数。
这一公式表示：**后一层误差通过权重矩阵反向传播，并乘上本层激活函数导数**。

A.3.3 参数梯度计算

得到误差项后，可以直接计算参数梯度：
权重梯度：

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)} (a^{(l-1)})^T$$

偏置梯度：

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

A.3.4 参数更新（梯度下降）

最后利用梯度下降更新参数：

$$W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial W^{(l)}}$$

$$b^{(l)} \leftarrow b^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial b^{(l)}}$$

其中 η 为学习率（learning rate）。

A.3.5 算法流程总结

反向传播的整体流程如下：

1. **前向传播（Forward Pass）**：计算各层 $z^{(l)}, a^{(l)}$
2. **计算输出层误差**：

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

3. **逐层反向传播误差**：

$$\delta^{(l)} = ((W^{(l+1)})^T \delta^{(l+1)}) \odot \sigma'(z^{(l)})$$

4. **计算梯度**：

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)} (a^{(l-1)})^T$$

5. **更新参数**：

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$$

反向传播算法的关键在于：**利用链式法则，使梯度计算复杂度从指数级降低到与网络规模线性相关**。

A.3.6　示例

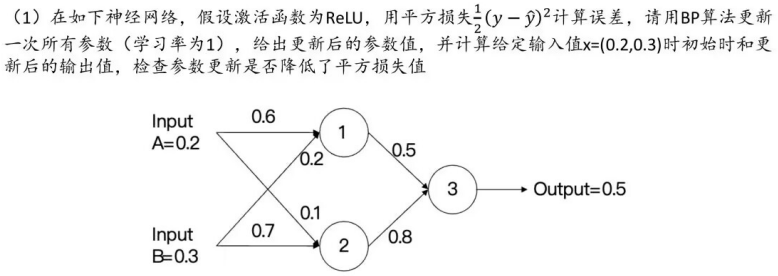


图 A.3: 反向传播算法示例

一、前向传播

隐层加权求和：

$$z_1 = 0.2 \times 0.6 + 0.3 \times 0.2 = 0.18$$

$$z_2 = 0.2 \times 0.1 + 0.3 \times 0.7 = 0.23$$

经过 ReLU 后：

$$a_1 = 0.18, \quad a_2 = 0.23$$

输出层：

$$\hat{y} = 0.18 \times 0.5 + 0.23 \times 0.8 = 0.274$$

初始损失：

$$L = 0.0255$$

二、反向传播

平方损失下：

$$\delta_3 = \hat{y} - y = -0.226$$

误差向前传播：

$$\delta_1 = 0.5\delta_3 = -0.113$$

$$\delta_2 = 0.8\delta_3 = -0.1808$$

由于所有神经元输入均为正，ReLU 导数为 1，因此不改变误差。

三、梯度计算

权重梯度的通式：

$$\frac{\partial L}{\partial w} = \delta \times (\text{该权重的输入})$$

例如

$$\frac{\partial L}{\partial w_{13}} = \delta_3 a_1$$

$$\frac{\partial L}{\partial w_{A1}} = \delta_1 A$$

计算可得各权重梯度。

四、参数更新

梯度下降公式：

$$w' = w - \eta \frac{\partial L}{\partial w}$$

得到更新后的权重：

$$w'_{13} = 0.54068, \quad w'_{23} = 0.85198$$

$$w'_{A1} = 0.6226, \quad w'_{B1} = 0.2339$$

$$w'_{A2} = 0.13616, \quad w'_{B2} = 0.75424$$

五、更新后验证

重新前向传播得到

$$\hat{y}' \approx 0.3206$$

新损失

$$L' \approx 0.0161$$

因为 $L' < L$，说明一次 BP 更新**成功降低了损失函数**。

A.4　n 元语法模型

A.4.1　核心原理（马尔可夫假设）

n 元语法模型基于 **n-1 阶马尔可夫假设**：一个词出现的概率只依赖于它前面的 $n-1$ 个词，即

$$P(w_t \mid w_1, w_2, \ldots, w_{t-1}) \approx P(w_t \mid w_{t-n+1}, \ldots, w_{t-1}).$$

A.4.2　句子概率计算

对于句子 $w_1, w_2, \ldots, w_N$（通常添加起始符 <s> 和终止符 </s> 以处理边界），其概率为所有 n 元条件概率的乘积：

$$P(\text{sentence}) = \prod_{t=1}^{N} P(w_t \mid w_{t-n+1}, \ldots, w_{t-1}),$$

当 $t < n$ 时，需要补足起始符（如 <s>）作为历史词。

对于每一项的 $P(w_t \mid w_{t-n+1}, \ldots, w_{t-1})$，按照条件概率的定义计算即可，即

$$P(w_t \mid w_{t-n+1}, \ldots, w_{t-1}) = \frac{\text{count}(w_t, w_{t-n+1}, \ldots, w_{t-1})}{\text{count}(w_{t-n+1}, \ldots, w_{t-1})}$$

A.4.3　模型评价：困惑度（Perplexity）

困惑度衡量语言模型对测试数据的预测能力，定义为句子概率的几何平均数的倒数：

$$\text{PPL} = P(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}} = \exp\left(-\frac{1}{N}\sum_{t=1}^{N} \ln P(w_t \mid w_{t-n+1}, \ldots, w_{t-1})\right).$$

困惑度越低，说明模型对测试数据的拟合越好，预测不确定性越小。

A.4.4　示例

题目：给定三个句子：

- <s> I like AI </s>
- <s> I like NLP </s>
- <s> AI like math </s>

其中 <s> 和 </s> 分别是句子的起始符和终止符。
(1) 求第一个句子 <s> I like AI </s> 的概率表达式。
(2) 以这三个句子作为语料库，求第一个句子的概率。
(3) 困惑度是什么？请解释之并写出它的数学表达式。

解答：
(1) 第一个句子的概率表达式为：

$$P(\texttt{<s> I like AI </s>}) = P(\text{I} \mid \texttt{<s>}) \times P(\text{like} \mid \text{I}) \times P(\text{AI} \mid \text{like}) \times P(\texttt{</s>} \mid \text{AI})$$

(2) 根据所给三个句子构成的语料库，统计二元语法模型的计数，得到：

- $P(\text{I} \mid \texttt{<s>}) = \dfrac{2}{3}$
- $P(\text{like} \mid \text{I}) = \dfrac{2}{2} = 1$

- $P(\text{AI} \mid \text{like}) = \dfrac{1}{3}$
- $P(\texttt{</s>} \mid \text{AI}) = \dfrac{1}{2}$

因此，第一个句子的概率为：

$$\frac{2}{3} \times 1 \times \frac{1}{3} \times \frac{1}{2} = \frac{1}{9}$$

(3) 困惑度（Perplexity）是衡量语言模型性能的指标，它反映了模型对测试数据的不确定性。困惑度越低，模型对数据的预测能力越好。数学上，对于包含 N 个词的测试序列 $W = w_1 w_2 \dots w_N$，困惑度定义为该序列概率的几何平均的倒数：

$$\text{PP}(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

其中 w_0 为起始符 `<s>`，且概率计算基于训练好的模型。

A.5 Transformer 编码器中的多头注意力

写在前面：**实际上，多头注意力机制就是将经过嵌入层后得到的向量（一条文本有多个词向量，自然就组成矩阵）与不同的 W_i^Q, W_i^V, W_i^K 相乘均等地拆分为多个不同的 Q，K，V 向量（对于一条文本序列就是矩阵）并作自注意力计算，这样得到的向量（矩阵）再分别拼接并经过线性变换投影回一开始输入多头注意力的向量的维度**。

A.5.1 一、输入表示

设输入序列长度为 n，每个 token 的表示维度为 d。

Transformer 中通常将所有 token 组成矩阵

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n \times d}$$

其中每一行 x_i 表示一个 token 的向量表示，第 i 行对应序列中的第 i 个输入。

A.5.2 二、线性映射

每个 attention head 会学习三组参数：W^Q，W^K，W^V，并计算

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

其中 Q：Query（查询），K：Key（键），V：Value（值）。

A.5.3 三、计算注意力分数

注意力分数矩阵为

$$S = \frac{QK^T}{\sqrt{d_k}}$$

其中 d_k 为 key 向量维度，$\sqrt{d_k}$ 用于**缩放点积结果**，防止数值过大。
矩阵 S 的第 (i,j) 个元素表示：第 i 个 token 对第 j 个 token 的相关性。

A.5.4　四、Softmax 计算注意力权重

对矩阵 S 的**每一行**做 softmax：

$$A_{ij} = \frac{\exp(S_{ij})}{\sum_{k=1}^{n} \exp(S_{ik})}$$

得到注意力权重矩阵 $A = \mathrm{softmax}(S)$。
性质：每一行元素之和为 1，表示该 token 对所有 token 的注意力分布。

A.5.5　五、计算注意力输出

注意力输出为

$$Z = AV$$

含义：用注意力权重对所有 value 向量进行加权求和。第 i 个 token 的输出为

$$z_i = \sum_{j=1}^{n} A_{ij} v_j$$

A.5.6　六、多头注意力

若有 h 个 head，每个 head 得到 $Z_1, Z_2, \ldots, Z_h$，多头注意力输出为

$$\mathrm{MultiHead}(X) = \mathrm{Concat}(Z_1, Z_2, \ldots, Z_h) W^O$$

其中 Concat 表示**在特征维度上拼接各个 head 的输出**。若每个 head 的输出为 $Z_i \in \mathbb{R}^{n \times d_v}$，则

$$\mathrm{Concat}(Z_1, Z_2, \ldots, Z_h) \in \mathbb{R}^{n \times (h d_v)}$$

即保持序列长度 n 不变，把不同 head 学到的特征向量在列方向依次拼接起来。W^O 为输出投影矩阵。
作用：融合不同 head 学到的不同注意力模式。

A.5.7　Transformer 编码器多头注意力计算

设输入矩阵

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

即 $x_1 = [1, 0], \quad x_2 = [0, 1]$。
共有两组 attention head。
第一组：$W^Q = W^K = W^V = I_2$
第二组：

$$W^Q = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad W^K = W^V = I_2$$

并且 $d_k = 2,\quad W^O = I_4$。

（1）第一组 head 中 x_1 的注意力权重

由于 $Q = K = V = X$，计算

$$QK^T = XX^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

缩放：$S = \dfrac{QK^T}{\sqrt{2}}$

x_1 的分数为 $\left[\dfrac{1}{\sqrt{2}},\ 0\right]$

softmax 得

$$\left(\frac{e^{1/\sqrt{2}}}{e^{1/\sqrt{2}}+1},\ \frac{1}{e^{1/\sqrt{2}}+1}\right)$$

（2）第二组 head 中 x_1 的注意力输出

$$Q = XW^Q = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

因此 $q_1 = [1, 1]$，又 $K = X$，所以 $q_1 K^T = [1, 1]$。

缩放：$\left[\dfrac{1}{\sqrt{2}},\ \dfrac{1}{\sqrt{2}}\right]$

softmax：$(0.5,\ 0.5)$

注意力输出：

$$z_1 = 0.5\, x_1 + 0.5\, x_2 = [0.5,\ 0.5]$$

（3）多头注意力最终输出

设第一组 head 输出为 $z_1^{(1)}$，第二组输出为 $z_1^{(2)} = [0.5,\ 0.5]$。

拼接：$[z_1^{(1)},\ z_1^{(2)}]$

由于 $W^O = I_4$，因此最终输出为

$$\mathrm{Concat}(z_1^{(1)},\ z_1^{(2)})$$

附录 B　考点汇总

1. 线性回归模型：$y = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + \epsilon$，这里的线性指的是相对于参数 w 线性。

2. 平方损失函数：

$$\begin{aligned}\text{TrainLoss}(\boldsymbol{w}) &= \frac{1}{|D_{\text{train}}|} \sum_{(x,y)\in D_{\text{train}}} (f(x) - y)^2 \\ &= \frac{1}{|D_{\text{train}}|} \sum_{(x,y)\in D_{\text{train}}} (\boldsymbol{w} \cdot \phi(x) - y)^2 \\ &= \frac{1}{|D_{\text{train}}|} (\boldsymbol{y} - \Phi\boldsymbol{w})^\top (\boldsymbol{y} - \Phi\boldsymbol{w})\end{aligned}$$

3. **机器学习的误差来源**（参见第2.1.4节）：模型误差（模型与现实分布的误差），估计误差（样本数量或质量导致），优化误差（优化方法导致）。其中，损失函数的选择（例如，对于分类任务用交叉熵而非均方误差）实际上**同时影响模型误差和优化误差**（本身不直接导致）。

4. **梯度下降**（参见第2.2.1节）是最常用的优化算法之一，它通过迭代的方式更新参数，逐步降低损失函数的值。每一次迭代为：$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \text{TrainLoss}(\boldsymbol{w})$，然而当数据集很大的时候，我们会采用随机梯度下降 $w \leftarrow w - \eta \cdot \frac{1}{B} \sum_{i=1}^{B} \nabla_w L(w; x^{(i)}, y^{(i)})$，其中，$B$ 为抽取样本，η 称为学习率。

5. 学习率大小很重要。如果学习率太大，更新步长过大，可能会跳过最小值，导致损失函数震荡甚至发散；如果学习率太小，更新步长过小，收敛速度很慢，可能需要很多次迭代才能达到最小值。

6. 逻辑函数 $\text{sigmoid}(x) = \dfrac{1}{1 + e^{-x}}$，将 $(-\infty, +\infty)$ 上的函数映射到 $(0, 1)$ 上。

7. 多分类问题的第一个方案为 one vs all，即训练 K 个独立的分类器，分别判断。

8. 多分类问题的另一个方案为交叉熵损失：将线性变换后的 $\text{score}(x)$ 经过 softmax 函数变换

$$P(y = k \mid x, w) = \frac{e^{\text{Score}_k(x)}}{\sum_{k'=1}^{K} e^{\text{Score}_{k'}(x)}}, \quad \sum_{k=1}^{K} P(y = k \mid x, w) = 1$$

   则要优化的函数为交叉熵：

$$\min_w \text{cross_entropy}(w) = -\sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}(y_i = k) \log P(y_i = k \mid x_i, w)$$

特别的，当 $K=2$ 即处理二分类问题时，交叉熵为：

$$\min_{w} \text{cross_entropy}(w) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$

多分类的两种方案的优劣性对比见第2.3.3节。

9. 在逻辑回归中，**过拟合表现为模型参数 w 过大**，使得决策边界过于陡峭。其成因是当数据接近线性可分时，最大似然估计（MLE）会不断推高系数以最大化似然，导致 sigmoid 函数 $\sigma(w^\top x)$ 的输出趋近于 0 或 1，模型对预测变得过度自信。这导致模型对输入特征的微小变化极为敏感，泛化能力下降，测试集误差增大。

10. **过拟合的解决方案**（参见第2.3.4节）：修改损失函数，使其随着系数的增大而增大（加入惩罚项），因此引入 L1，L2 正则化。

11. **评估模型的好坏**（参见第2.3.5节）：一个重要的指标是 AUC，表示 ROC 曲线下的面积，越接近 1，区分能力越强，接近 0.5，代表区分能力接近随机模型。

12. **卷积神经网络**（参见第3.1.4节）的特点：稀疏连接（每个输出单元只与输入中的一个局部区域）、参数共享与平移不变性（检测同一特征在图像不同位置的出现）。

13. **池化**的方式：最大池化与平均池化。作用：对局部平移更加稳定。

14. **反向传播算法**（参见第3.2.1节）的核心思想为复合函数求导。详细推导见附录A.3。再次建议，即使复习时间紧张，也建议**做一下附录 A 中的例题**，并记住两个技巧：sigmoid 函数 $\sigma(x)$ 的导数为 $\sigma'(x)=\sigma(x)(1-\sigma(x))$，以及在计算层数比较多（虽然目前感觉考试中不太会出）的神经网络的时候，可以把前几层的 $\frac{\partial f}{\partial z}$ 存储起来，便于计算。

15. 随机梯度下降中，批量的大小不影响梯度期望，但是会影响梯度方差。批量越大，方差越小，可以设置较大学习率。

16. 防止过拟合——提前终止：在验证集上性能不再提升就停止训练，防止过拟合。

17. **防止过拟合——Dropout**（参见第3.2.2节）：核心思想是在训练过程中**随机”丢弃”（即设置为 0）一部分神经元的输出**。Dropout 还有一个变体：dropconnect，丢失一部分权重（连接）。

18. **参数初始化**（参见第3.2.2节）：参数不能全部初始化为 0。初始化方法有：预训练初始化，随机初始化，固定值初始化。

19. **神经网络问题：梯度消失**（参见第3.3节）——神经网络层数过多时，由于链式法则导致反向传播过程中浅层参数的导数趋于 0，无法有效更新和学习。缓解方法有：更换激活函数，残差连接，归一化，采用合理的初始化策略。

20. **残差连接**（参见第3.3.1节）：残差连接的核心想法是**不直接学习目标映射 $H(x)$，而是学习残差 $F(x)=H(x)-x$**。这样，当最优解接近恒等映射时，网络只需让权重趋近于零（$F(x)\to 0$），比直接精确拟合恒等映射要容易得多。作用：缓解梯度消失及深层带来的信息丢失；浅层特征可以通过跳跃连接直接传递到深层，网络同时利用底层特征与高层特征，实现了多尺度的特征融合。

21. **归一化**（参见第3.3.2节）：主要有数据归一化（标准化，最大最小值归一化），逐层归一化（批归一化 BN，层归一化 LN，实例归一化 IN）。

22. **图学习的三大任务**（参见第4.1节）：节点级任务（单个节点）、链接级任务（两个节点之间是否会有链接）、图级任务（对图的属性预测，或者直接生成新的图）。

23. **图数据类型**：图数据类型的一个样本包含四类信息：节点特征向量，边特征向量，全局特征向量和图结构。**图表征学习**：可以分为节点嵌入，图嵌入和边嵌入。使用嵌入：节点分类，链接预测，图分类。

24. **图神经网络**（参见第4.3节）：核心想法：邻居聚合。

$$h_v^{(k+1)} = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_v^{(k)}}{|N(v)|} + B_k h_v^{(k)}\right), \quad \forall k \in \{0, \ldots, K-1\}$$

其中初始第 0 层嵌入等于节点的特征，即 $h_v^0 = x_v$，公式里的 σ 为非线性激活函数，K 为神经网络总层数，最终的嵌入表示 z_v 为最终的节点表示，即 $z_v = h_v^K$。在图神经网络中，我们要学习的量为 W_k, B_k。

25. **图神经网络的优势**（参见第4.3.2节）：融合高阶邻居信息、归纳能力。

26. **注意力机制**（参见第5.1.2节）：核心思想——给不同的向量以不同的权重（注意力）。**前馈注意力机制**（区别于下一条的自注意力机制）的计算过程：设 $a(x)$ 是一个可学习的函数，h_t 是输入隐向量 $\Rightarrow e_t = a(h_t)$，注意力权重 α_t（注意这里的 α_t 和前文的可学习函数 $a(x)$ 不是一个东西）为

$$\alpha_t = \frac{\exp(e_t)}{\sum_{t=1}^{T} \exp(e_t)}$$

上下文向量 c 为

$$c = \sum_{t=1}^{T} \alpha_t \cdot h_t$$

其中，可学习函数 $a(h_t)$ 的一个典型实现是 $a(h_t) = \tanh(w \cdot h_t)$，其中 w 为要学习的参量。

27. **自注意力机制**（参见第5.1.3节）的算法见相关章节或附录A.5的前五部分，与上一条提到的前馈注意力机制在输入来源，相似度计算，参数使用和信息交互方面有所不同。自注意力机制的核心优势体现在全局信息捕获，高度并行，动态权重分配和语义对齐四个角度。

28. Transformer 没有 RNN 的循环结构，本身无法感知词的顺序，因此需要**位置编码**来注入位置信息。常用**正余弦函数**为每个位置生成一个向量，与词向量相加后输入模型。这样的位置编码称为**正余弦位置编码**，具有绝对位置唯一性，相对位置可推导性和长度适应性的优势。

29. **多头注意力机制**（参见第5.2.1节）：动机：信息需要从多个维度抽取意义。以文本为例，有的注意力头关注风格，而有的关注语法。所以需要多个视角。类似于多个卷积核（所以二者都具有并行执行的特点，都通过计算上下文的加权和来更新表示，多头正对应 CNN 里的多通道）。**多头注意力机制就是将经过嵌入层后得到的向量（一条文本有多个词向量，自然就组成矩阵）与不同的 W_i^Q, W_i^V, W_i^K 相乘均等地拆分为多个不同的 Q，K，V 向量**

（对于一条文本序列就是矩阵）并作自注意力计算，这样得到的向量（矩阵）再分别拼接并经过线性变换投影回一开始输入多头注意力的向量的维度。

30. **相加与归一化**中，相加指的是与前馈层（FFN）残差连接，以 $F(x)+x$ 形式保持梯度稳定性，归一化指的是 LN——层归一化，相当于对样本中所有的数据进行归一化。本质上都是第三章提到过的内容。

31. **解码器**（参见第5.2.2节）中，需要屏蔽当前时间步骤之后的输出：即掩码多头注意力机制。

32. **交叉多头注意力**是为了将编码器的信息引入解码器：以编码器的输出作为 **K** 和 **V**，以掩码多头注意力的输出作为 **Q**，计算相似度并加权求和，从而让解码器在生成每个词时都能关注到输入序列的相关部分。

33. Transformer 也可以处理图像。但是图像的 RGB 值无法使用 Transformer 的编码器或者解码器，于是采用 patch（图像块）的概念：将图像分成小块，每一小块转化成向量再这些嵌入按 patch 在图像中的顺序排列成一个序列，作为 Transformer 编码器的输入。

34. **独热 one-hot 向量**（参见第6.1.1节）的缺陷：维度高，f 参数量大；不同单词的独热向量正交，无法判断相似程度。

35. **自监督学习的训练策略**（参见第6.1.1节）：预训练和迁移。其中，迁移包括微调（固定 f_e，利用任务相关的标签数据微调 f 参数），联合微调（f_e, f 都调），上下文学习（不更新参数，只输入几个任务样例让模型熟悉），零样本输入（不提供任何输入，常用于大语言模型）。

36. **自预测**（参见第6.1.2节）分为两种：自回归生成（根据过去的数据预测未来的数据）、掩码生成（不考虑自然顺序的随机掩盖一部分信息并假装缺失）和内在关系预测（对一个数据样本进行某些变换应保持原始信息或遵循所需的内在逻辑）。

37. **对比学习**（参见第6.1.2节）分为三种：样本间分类，特征聚类，多视角编码。

38. **word2vec**（参见第6.2.1节）的结构分为输入层，投影层，输出层，和损失函数。其中输出层得到的是词汇表中每个单词作为目标上下文词的可能性。**BERT**（参见第6.2.2节）的建模原理基于 **Transformer 编码器**，通过**双向上下文**的理解来学习文本的深层语义表示。其核心在于使用**掩码语言建模（MLM）**和**下一句预测（NSP）**两个自监督任务进行预训练。

39. 条件概率 $P(w_i \mid w_1 \dots w_{i-1})$ 的获取方式主要有两种：直接存储（对应 n 元语法语言模型，参见第6.3.1节）和函数计算（对应神经网络语言模型，参见第6.3.2节）。GPT 系列（参见第6.3.3节）均采取了自回归（即按条件概率预测下一个词）的架构。

40. 大语言模型的**能力特点**（参见第6.4节）：扩展法则：具有极强的涌现能力。**构建方法**：预训练、有监督微调、人类对齐。

附录 C　试卷合集

C.1　25 秋（B 系列）试卷回忆版

由 24 网安**刘昊天**回忆整理。表述可能与原题略有偏差，仅供参考。

C.1.1　一、判断正误并说出理由（10×5 分）

1. 非线性激活函数的主要作用是加快神经网络的收敛速度。
2. 若一个多层神经网络的所有激活函数都被替换成线性函数（如 $f(x) = ax + b$），则无论多深的神经网络的表达能力等价于一个单层线性神经网络。
3. 如果模型训练时训练集上的损失低而测试集的损失高，那么只要增加训练轮数且优化更加充分，验证时的误差最终也会下降至近似训练误差。
4. 处理 one-hot 向量的多分类问题时，交叉熵损失只与真实标签的预测概率有关。
5. 处理多分类问题时，即使不同类别的个数不均衡，单用准确率指标也可以训练出一个好的模型。
6. Dropout 机制是训练时随机丢弃一些样本，以达到增强对不同样本的鲁棒性的目的。
7. BERT 中使用掩码语言模型时，由于会用 mask 替换部分词语，那么训练时不能感知到 mask 词元右侧的上下文。
8. 上下文学习时，模型会根据任务描述快速进行小范围的梯度更新，以识别和执行新的任务。
9. 强化学习的奖励必须是及时的且明确的。
10. Value-based 优化方式直接对策略参数化并优化。

C.1.2　二、解答题（共五道）

第 1 题　卷积神经网络参数量

定义神经网络有两个卷积层和一个全连接层展平维度并二分类输出：

$$\text{input} \to \text{conv1} \to \text{conv2} \to \text{FC} \to \text{output}$$

其中 input 是一个 $64 \times 64 \times 1$ 的灰度图。

- conv1：卷积核 3×3，输入通道 1，输出通道 4，stride=1，padding=1。

- conv2：卷积核 3×3，输入通道 4，输出通道 8，stride=1，padding=2。

均不考虑偏置，请给出这个网络的参数总量。

第 2 题　逻辑回归

预测同学是否挂科（题干与样卷类似）：

1. 请写出预测值 $\hat{y}$ 的表达式并说明其概率意义。
2. 请预测会挂科的概率 p。
3. 在概率阈值为 0.5 的情况下，判断其是否会挂科?

第 3 题　反向传播

与作业题课本 3.3 几乎完全相同，只是损失函数变为 Sigmoid，学习率变成了 0.1，其他参数改了一下。

第 4 题　二元语法模型

给了三个句子：`<s> I like AI </s>`、`<s> I like NLP </s>`、`<s> AI like math </s>`。其中 `<s>` 和 `</s>` 分别是句子的起始符和终止符。

1. 求第一个句子概率表达式。
2. 以这三个句子作为语料库，求第一个句子的概率。
3. 困惑度是什么？请解释之并写出它的数学表达式。

第 5 题　Transformer 编码器

给了 2×2 的输入 $X=\begin{bmatrix}x_1\\x_2\end{bmatrix}$，其中 $x_1=[1,0]$，$x_2=[0,1]$。两组 attention head 都是 2×2 矩阵：

- 第一组：$W^Q=W^K=W^V=I_2$
- 第二组：$W^Q=\begin{bmatrix}1&1\\0&1\end{bmatrix}$，$W^K=W^V=I_2$

且 $d_k=2$，投影矩阵 $W^O=I_4$（I_2、I_4 分别是二阶、四阶单位阵）。

1. 求第一组头作用时 x_1 对全部输入的注意力权重。
2. 求第二组头作用时 x_1 的注意力输出向量。
3. 求多头注意力对 x_1 的最终输出。

C.2　25 春模拟卷（含答案）

25 春模拟卷（试卷及参考答案）以 PDF 文件形式随本手册附带，以下直接嵌入展示。

C.2.1　模拟试卷

2025 年春季人工智能数学原理与算法期末考试 (模拟)

学号 _____ 姓名 _____

一、**判断题** (根据表述判断正误, 并简要说明理由; 每题 6 分, 共 30 分).

1. (**T**, **F**) 在医学检测中, 假阳性 (False Positive) 是指检测结果呈阳性, 但实际上患者并未患病. 这种情况在低患病率的疾病筛查中尤为常见, 尤其是当检测方法的准确率不足以弥补低患病率的影响. 假设有一种疾病影响 0.1%(即 1/1000) 的人口. 现在有一个检测方法:

- 灵敏度 (Sensitivity)= 99%(患病者被正确检测为阳性的概率)
- 特异度 (Specificity)= 95%(无病者被正确检测为阴性的概率)

如果某人检测结果为阳性, 他实际患病的概率是否大于 90%. 如果不是, 请计算出具体的概率.

2. (**T**, **F**) 一个前馈神经网络如果具有线性输出层, 只要给予网络足够数量的隐藏单元, 它可以以任意的精度来近似任何从一个有限维空间到另一个有限维空间的大多数常见函数

3. (**T**, **F**) 基于随机游走的节点嵌入方法可以应用于新的图或者动态变化的图.

4. (**T**, **F**) 在 Transformer 模型中, 多头注意力模块的每个头使用相同的查询 (Q)、键 (K)、值 (V) 矩阵进行计算.

5. (**T**, **F**) 自监督学习是一种特殊的表征学习, 能够从无标签数据集中学习良好的数据表征.

6. (**T**, **F**) Q-Learning 属于基于模型 (Model-Based) 的强化学习算法.

二、**综合题** (根据题目要求写出解答过程, 共 60 分).

1.(15 分) 某银行需构建贷款违约预测模型 (二分类问题, 标签 y=+1 表示违约,y=-1 表示不违约), 基于以下场景回答问题: 某银行需构建贷款违约预测模型 (二分类问题, 标签 y=+1 表示违约,y=-1 表示不违约), 基于以下场景回答问题:

- 特征包括: 年收入 (x_1)、贷款金额 (x_2)、贷款利率 (x_3).
- 采用逻辑回归模型进行验证, 特征提取器为 $\Phi(x) = [1, x_1, x_2, x_3]$.

请回答以下问题:

(1) 写出逻辑回归表达式, 若模型训练后参数为 $w = [w_0, w_1, w_2, w_3] = [1, -0.5, 0.8, 1.2]$, 分别对应 $[1, x_1, x_2, x_3]$ 的权重, 计算样本 $\mathbf{x} = [1, x_1, x_2, x_3] = [1, 5, 3, 0.1]$ 的违约概率, 并判断是否违约. (5 分)

(2) 训练过程中发现模型在训练集上准确率达 95%, 但在测试集上仅为 60%, 请分析可能原因及解决方案. (5 分)

(3) 若引入 L2 正则化, 写出正则化后的目标函数 (二项交叉熵), 并说明正则化参数 λ 的作用. (5 分)

2.(15 分) 在深度神经网络中, 激活函数的选择和初始化方法对训练过程有重要影响. 请结合以下两个问题, 分析激活函数在训练过程中可能遇到的问题及其解决方案.

(1) Sigmoid 激活函数的梯度消失问题

(1.1) 请解释在使用 Sigmoid 激活函数时, 为什么权重初始化过大会导致梯度消失? (3 分)

(1.2) 假设一个全连接层的输出为 $z = \sigma(Wx)$, 其中 σ 是 Sigmoid 函数. 请推导反向传播时 z 的局部梯度 (即 $\frac{\partial L}{\partial z}$), 并说明为什么难以堆叠多层使用 Sigmoid 激活函数的网络. (4 分)

(2) ReLU 激活函数的" 死亡" 问题

(2.1) 说明为什么 ReLU 神经元一旦" 死亡" (激活值为 0) 就很难恢复? (4 分)

(2.2) 假设你正在训练一个深度神经网络, 发现 40% 的 ReLU 神经元在训练过程中始终输出为零. 请提出两种可能的解决方法, 并简要解释其原理.(4 分)

3. (10 分) 已知一张包含四个节点 A, B, C, D 的无向图, 连接关系如图 1 所示. 其中, 每个节点的初始特征向量 $h^{(0)}$ 为:

$$h_A^{(0)} = [1, 0]$$
$$h_B^{(0)} = [0, 1]$$
$$h_C^{(0)} = [1, 1]$$
$$h_D^{(0)} = [0, 0]$$
$$h_E^{(0)} = [1, 0]$$

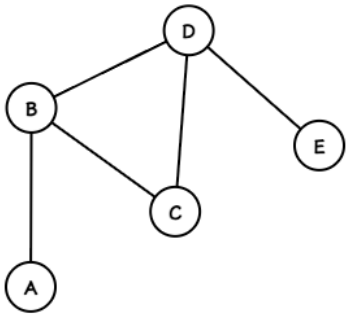


图 1: 无向图

(1) 请简述图神经网络 (GNN) 中消息传递机制及其作用. (5 分)

(2) 使用简单的消息传递机制 (每个节点通过平均其邻居的特征来更新其特征), 形成新的特征向量 $h^{(1)}$, 计算节点 B 经过一次消息传递步骤后的新特征向量. (5 分)

4.(10 分)

(1) 请描述 Transformer 模型中编码器与解码器的自主注意力机制的两个不同点. (5 分)

(2) 已知对于一个 Transformer 编码器, 输入序列长度为 10, 词嵌入维度为 512, 多头注意力头数为 8, 请你计算多头注意力模块中可学习参数矩阵的总参数量.(忽略偏置项) (5 分)

5.(10 分)

Word2Vec,BERT 和 GPT 模型都是自然语言处理领域中典型的自监督学习方法, 请根据课堂所学内容作答下列问题.

(1) 说明 Word2Vec(以 Skip-gram 模型为例) 和 BERT 的 MLM(Masked Language Model) 模型在自监督学习思路的相似性和获取词向量的性质这两方面的差异性. (4 分)

(2)

(2.1) 对比 BERT 模型和 GPT-1 模型在模型结构, 自注意力机制和自监督预训练任务上的差异性. (2 分)

(2.2) 简要阐述 BERT 和 GPT-1 在文本分类下游任务上进行微调的任务设计思路 (输入数据是否需要特殊标记, 特征提取, 分类层). (2 分)

(2.3) 举例简要阐述 GPT-2 模型如何实现预训练后多任务的零样本学习和上下文学习. (2 分)

6. (10 分) 考虑如下简单的网格世界 MDP:

- 状态集 $S = \{s_1, s_2, s_3, s_4\}$;
- 动作集 $A = \{Up, Down, Left, Right\}$;
- 折扣因子 $\gamma = 0.9$;
- 学习率 $\alpha = 0.1$;
- 初始 Q 值表中所有 $Q(s,a) = 0$.

Q-learning 的更新公式为:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

SARSA 的更新公式为:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

(1) 请根据 SARSA 和 Q-learning 的更新公式, 解释为什么 SARSA 是在策略 (on-policy) 学习,Q-learning 是离策略 (off-policy) 学习. (5 分)

(2) 假设初始 Q 值表中 $Q(s_3, Up) = 0.5, Q(s_3, Down) = 1.0, Q(s_5, Left) = 2.0, Q(s_5, Right) = 1.5$, 其余部分全为 0. 考虑如下的交互序列:

1. 从 s_2 执行 $Right$ 到 s_3, 奖励 $r_1 = 5$;
2. 从 s_3 执行 Up 到 s_5, 奖励 $r_2 = 3$;
3. 从 s_5 执行 $Left$ 到 s_4, 奖励 $r_3 = 1$.

若 s_4 为终止状态, 即 $Q(s_4, -) = 0$, 请分别计算 SARSA 和 Q-learning 的 Q 值更新过程. (5 分)

C.2.2　模拟试卷答案

2025 年春季人工智能数学原理与算法期末考试 (模拟)

学号 _____ 姓名 _____

一、**判断题** (根据表述判断正误, 并简要说明理由; 每题 6 分, 共 30 分).

1. (**T**, **F**) 在医学检测中, 假阳性 (False Positive) 是指检测结果呈阳性, 但实际上患者并未患病. 这种情况在低患病率的疾病筛查中尤为常见, 尤其是当检测方法的准确率不足以弥补低患病率的影响. 假设有一种疾病影响 0.1%(即 1/1000) 的人口. 现在有一个检测方法:

- 灵敏度 (Sensitivity)= 99%(患病者被正确检测为阳性的概率)
- 特异度 (Specificity)= 95%(无病者被正确检测为阴性的概率)

如果某人检测结果为阳性, 他实际患病的概率是否大于 90%. 如果不是, 请计算出具体的概率.

解答: F 错误.

定义事件:

- D: 患有该疾病
- T^+: 检测结果为阳性

已知条件:

- $P(D) = 0.001$(患病率为 0.1%)
- $P(T^+ \mid D) = 0.99$(灵敏度为 99%)
- $P(T^- \mid D^c) = 0.95$(特异度为 95%)

由特异度可知: $P(T^+ \mid D^c) = 1 - P(T^- \mid D^c) = 1 - 0.95 = 0.05$ 使用贝叶斯定理:

$$
\begin{aligned}
P(D \mid T^+) &= \frac{P(T^+ \mid D) \cdot P(D)}{P(T^+)} && (1)\\
&= \frac{P(T^+ \mid D) \cdot P(D)}{P(T^+ \mid D) \cdot P(D) + P(T^+ \mid D^c) \cdot P(D^c)} && (2)\\
&= \frac{0.99 \times 0.001}{0.99 \times 0.001 + 0.05 \times 0.999} && (3)\\
&= \frac{0.00099}{0.00099 + 0.04995} && (4)\\
&= \frac{0.00099}{0.05094} && (5)\\
&\approx 0.0194 && (6)
\end{aligned}
$$

因此, 一个检测结果呈阳性的人实际患病的概率约为 1.94%, 即使检测方法有较高的灵敏度和特异度.

2. (**T**, **F**) 一个前馈神经网络如果具有线性输出层, 只要给予网络足够数量的隐藏单元, 它可以以任意的精度来近似任何从一个有限维空间到另一个有限维空间的大多数常见函数

解答:F 错误, 万能近似定理: 线性输出层和至少一层具有任何一种“挤压”性质的激活函数.

3. (**T**, **F**) 基于随机游走的节点嵌入方法可以应用于新的图或者动态变化的图.

解答: F 错误, 随机游走方法（如 DeepWalk、Node2Vec）是传导式（transductive），需在固定图上训练，无法直接为新节点或新图生成嵌入。

4. (**T**, **F**) 在 Transformer 模型中, 多头注意力模块的每个头使用相同的查询 (Q)、键 (K)、值 (V) 矩阵进行计算.

解答: F 错误,Transformer 的多头注意力机制为每个头分配独立的权重矩阵（W_i^Q, W_i^K, W_i^V），以捕捉不同子空间的特征。

5. (**T**, **F**) 自监督学习是一种特殊的表征学习, 能够从无标签数据集中学习良好的数据表征.

解答 T 正确. 自监督学习通过构造伪标签（如预测掩码词、下一词）从无标签数据中学习通用表征。

6. (**T**, **F**) Q-Learning 属于基于模型 (Model-Based) 的强化学习算法.

解答 F 错误. Q-Learning 属于“无模型 (Model-Free)+ 价值式 (Value-Based)+ 离策略 (Off-Policy)”算法, 不依赖对环境状态转移概率和奖励函数的显式建模.

二、**综合题** (根据题目要求写出解答过程, 共 60 分).

1.(15 分) 某银行需构建贷款违约预测模型 (二分类问题, 标签 y=+1 表示违约,y=-1 表示不违约), 基于以下场景回答问题: 某银行需构建贷款违约预测模型 (二分类问题, 标签 y=+1 表示违约,y=-1 表示不违约), 基于以下场景回答问题:

- 特征包括: 年收入 (x_1)、贷款金额 (x_2)、贷款利率 (x_3).
- 采用逻辑回归模型进行验证, 特征提取器为 $\Phi(x) = [1, x_1, x_2, x_3]$.

请回答以下问题:

(1) 写出逻辑回归表达式, 若模型训练后参数为 $w = [w_0, w_1, w_2, w_3] = [1, -0.5, 0.8, 1.2]$, 分别对应 $[1, x_1, x_2, x_3]$ 的权重, 计算样本 $\mathbf{x} = [1, x_1, x_2, x_3] = [1, 5, 3, 0.1]$ 的违约概率, 并判断是否违约. (5 分)

(2) 训练过程中发现模型在训练集上准确率达 95%, 但在测试集上仅为 60%, 请分析可能原因及解决方案. (5 分)

(3) 若引入 L2 正则化, 写出正则化后的目标函数 (二项交叉熵), 并说明正则化参数 λ 的作用. (5 分)

参考答案:

(1) 逻辑回归表达式如下:

$$P(y = +1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

其中 $\mathrm{w}^T\mathbf{x} = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3$.

给定模型参数 $\mathbf{w} = [w_0, w_1, w_2, w_3] = [1, -0.5, 0.8, 1.2]$.

计算样本 $\mathbf{x} = [1, 5, 3, 0.1]$ 的预测值:

$$\mathbf{w}^T\mathbf{x} = 1 \cdot 1 + (-0.5) \cdot 5 + 0.8 \cdot 3 + 1.2 \cdot 0.1 = 1 - 2.5 + 2.4 + 0.12 = 1.02$$

$$P(y = +1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-1.02}} \approx \frac{1}{1 + 0.36} \approx 0.735$$

违约概率为 0.735, 大于 0.5, 因此预测为违约 ($y = +1$).

(2) 模型在训练集上准确率达 95%, 但在测试集上仅为 60%, 可能的原因及解决方案:

- **过拟合**: 模型过度拟合训练数据, 捕捉了训练集中的噪声, 导致泛化能力差.
- **分布偏移**: 训练集和测试集的数据分布不一致.
- **特征选择不当**: 当前特征可能不足以捕捉违约行为的关键模式.
- **样本不平衡**: 违约和非违约样本比例失衡.

解决方案:

- **正则化**: 引入 L1 或 L2 正则化, 减少过拟合.
- **特征工程**: 添加更多相关特征, 如信用历史、债务收入比等.
- **交叉验证**: 使用 k 折交叉验证来更准确地评估模型性能.

(3) 引入 L2 正则化后的目标函数:

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(h_{\mathbf{w}}(\mathbf{x}i)) + (1-y_i)\log(1-h\mathbf{w}(\mathbf{x}i))] + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

其中:

- $h_{\mathbf{w}}(\mathbf{x}i) = P(y=+1|\mathbf{x}_i,\mathbf{w}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}_i}}$
- $\|\mathbf{w}\|_2^2 = w_0^2 + w_1^2 + w_2^2 + w_3^2$(注意: 有时正则化会排除偏置项 w_0)

正则化参数 λ 的作用:

- 控制正则化强度, 平衡模型的拟合能力和复杂度.
- λ 越大, 对权重的惩罚越严重, 模型越简单, 但可能欠拟合.
- λ 越小, 对权重的惩罚越小, 模型可能更复杂, 容易过拟合.
- L2 正则化倾向于将所有权重缩小, 但不会使其变为零, 有助于处理多重共线性问题.

2.(15 分) 在深度神经网络中, 激活函数的选择和初始化方法对训练过程有重要影响. 请结合以下两个问题, 分析激活函数在训练过程中可能遇到的问题及其解决方案.

(1) Sigmoid 激活函数的梯度消失问题

(1.1) 请解释在使用 Sigmoid 激活函数时, 为什么权重初始化过大会导致梯度消失? (3 分)

(1.2) 假设一个全连接层的输出为 $z = \sigma(Wx)$, 其中 σ 是 Sigmoid 函数. 请推导反向传播时 z 的局部梯度 (即 $\frac{\partial L}{\partial z}$), 并说明为什么难以堆叠多层使用 Sigmoid 激活函数的网络. (4 分)

(2) ReLU 激活函数的" 死亡" 问题

(2.1) 说明为什么 ReLU 神经元一旦" 死亡" (激活值为 0) 就很难恢复? (4 分)

(2.2) 假设你正在训练一个深度神经网络, 发现 40% 的 ReLU 神经元在训练过程中始终输出为零. 请提出两种可能的解决方法, 并简要解释其原理.(4 分)

参考答案:

(1.1) 权重初始化过大导致梯度消失的原因 (3 分)

- 当权重 W 初始化过大时,Wx 的绝对值会很大, 导致 Sigmoid 输入 $\sigma(Wx)$ 落在饱和区
- Sigmoid 函数在饱和区 ($|Wx|$ 很大时) 的梯度 $z(1-z)$ 趋近于 0
- 根据链式法则, 反向传播时梯度会乘以这个极小值, 导致梯度指数级衰减
- 最终造成底层网络参数几乎无法更新, 训练停滞

(1.2) 局部梯度推导及多层网络问题 (4 分)

- 局部梯度推导:

$$z = \sigma(Wx) = \frac{1}{1+e^{-Wx}}$$
$$\frac{\partial z}{\partial W} = z(1-z)x$$
$$\frac{\partial z}{\partial x} = z(1-z)W$$

- 难以堆叠多层的原因:
  - 单层最大梯度为 $0.25(z = 0.5$ 时)
  - 对于 n 层网络, 梯度最多衰减为 $(0.25)^n$
  - 深层网络中梯度会指数级消失
  - 导致底层网络参数更新极其缓慢

(2) ReLU 激活函数的" 死亡" 问题

(2.1) " 死亡" 神经元难以恢复的原因 (4 分)

- 定义: 当 ReLU 输入 $Wx+b \leq 0$ 对所有训练样本都成立时, 神经元永久输出 0
- 梯度问题:
  - 前向传播: 输出恒为 0
  - 反向传播: 梯度 $\frac{\partial L}{\partial W} = 0$(因 ReLU 在负区梯度为 0)

- 权重更新停滞:
  - 参数梯度始终为 0
  - 无法通过常规梯度下降恢复
- 特别是使用较大学习率时, 容易因大幅更新导致大量神经元” 死亡”

(2.2) 两种解决方法及原理 (4 分)

- **方法 1: 使用 LeakyReLU/PReLU**
  - LeakyReLU:$f(z) = \begin{cases} z & \text{当} z > 0 \\ \alpha z & \text{当} z \leq 0 \end{cases}$ ($\alpha = 0.01$ 等小正数)
  - PReLU: 将 α 作为可学习参数
  - 优势: 负区保持微小梯度, 避免完全” 死亡”
- **方法 2: 调整初始化策略**
  - 使用 He 初始化:$W \sim \mathcal{N}(0, \sqrt{2/n_{in}})$
  - 偏置项初始化为小正值 (如 0.1)
  - 优势: 增加初始激活概率, 降低” 死亡” 风险
- **其他可选方案**:
  - 使用学习率衰减
  - 尝试 Swish 等新型激活函数

3. (10 分) 已知一张包含四个节点 A,B,C,D 的无向图, 连接关系如图 1 所示. 其中, 每个节点的初始特征向量 $h^{(0)}$ 为:

$$h_A^{(0)} = [1,0]$$
$$h_B^{(0)} = [0,1]$$
$$h_C^{(0)} = [1,1]$$
$$h_D^{(0)} = [0,0]$$
$$h_E^{(0)} = [1,0]$$

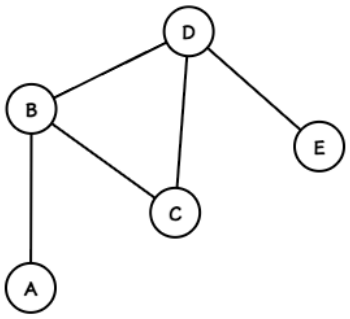


图 1: 无向图

(1) 请简述图神经网络 (GNN) 中消息传递机制及其作用. (5 分)

(2) 使用简单的消息传递机制 (每个节点通过平均其邻居的特征来更新其特征), 形成新的特征向量 $h^{(1)}$, 计算节点 B 经过一次消息传递步骤后的新特征向量. (5 分)

参考答案:

(1) 在图神经网络 (GNN) 中, 消息传递是一个核心机制, 它允许节点从其邻居节点收集信息并更新自身的表示 (特征向量). 这个过程使得 GNN 能够捕捉图的拓扑结构以及节点间的依赖关系.

消息传递通常包含以下两个主要步骤:

- 聚合 (Aggregation): 每个节点收集其邻居节点的信息 (通常是它们的特征向量). 这些信息可以通过不同的聚合函数 (如求和、平均、最大值等) 进行整合.
- 更新 (Update): 节点将聚合到的邻居信息与自身的当前信息结合起来 (通常通过一个神经网络层或简单的线性变换), 以生成该节点在下一层的新表示. 通过多层消息传递, 每个节点可以逐渐聚合到来自更远邻居的信息, 从而学习到更丰富的、与图结构相关的节点表示. 这对于图上的各种任务 (如节点分类、链接预测等) 至关重要.

(2) 根据 GNN 中的消息传递定义,$h_v^{(t+1)} = \text{AGGREGATE}(\{h_u^{(t)}, u \in \mathcal{N}(v)\})$

采用邻居特征平均方法, 所以 $\text{AGGREGATE} = \frac{1}{|\mathcal{N}|}\sum_{u\in\mathcal{N}(v)} h_u^{(t)}$.

则计算得到 $h_B^{(1)} = \frac{1}{3}([1,0]+[1,1]+[0,0]=[2,1]) = [\frac{2}{3},\frac{1}{3}]$.

综上, 节点 B 经过一次消息步骤后得到的新特征向量为 $[\frac{2}{3},\frac{1}{3}]$.

4.(10 分)

(1) 请描述 Transformer 模型中编码器与解码器的自主注意力机制的两个不同点. (5 分)

(2) 已知对于一个 Transformer 编码器, 输入序列长度为 10, 词嵌入维度为 512, 多头注意力头数为 8, 请你计算多头注意力模块中可学习参数矩阵的总参数量.(忽略偏置项) (5 分)

参考答案:

(1) 不同点:

i. 是否包含掩码:

- 编码器的自注意力机制不包含掩码, 因为它处理的是完整输入序列, 所有 token 是并行可见的.
- 解码器的自注意力机制必须使用掩码, 防止在生成当前 token 时看到未来位置的 token.

ii. 输入来源不同:

- 编码器的自注意力机制的 Q, K, V 都来自于编码器的输入序列;
- 解码器的自注意力机制的 Q, K, V 都来自于解码器已生成的 token 序列 (即当前步之前的位置) 并且应用掩码

(2) 已知词嵌入维度为 $d_{model} = 512$, 头数为 8, 首先计算单头的维度 $d_k = d_v = \frac{d_{model}}{h} = 64$.

其次, 每个头包含三个参数矩阵 (W^Q, W^K, W^V), 形状均为 $d_e \times d_k = 512 \times 64$, 参数量大小为 $512 \times 64 = 32768$

最后, 多头注意力将所有头的输出拼接, 纬度为 $d_o = h \times d_v = 8 = 512$ 输出矩阵 W^O 的形状为 $d_o \times d_{model} = 512 \times 512$, 因此输出矩阵的参数量为 $512 \times 512 = 262144$

因此, 多头注意力模块中可学习参数矩阵的总参数量为 $3 \times h \times (d_{model} \times d_k) + d_o \times d_{model} = 786432 + 262144 = 1048576$

5.(10 分)

Word2Vec,BERT 和 GPT 模型都是自然语言处理领域中典型的自监督学习方法, 请根据课堂所学内容作答下列问题.

(1) 说明 Word2Vec(以 Skip-gram 模型为例) 和 BERT 的 MLM(Masked Language Model) 模型在自监督学习思路的相似性和获取词向量的性质这两方面的差异性. (4 分)

(2)

(2.1) 对比 BERT 模型和 GPT-1 模型在模型结构, 自注意力机制和自监督预训练任务上的差异性. (2 分)

(2.2) 简要阐述 BERT 和 GPT-1 在文本分类下游任务上进行微调的任务设计思路 (输入数据是否需要特殊标记, 特征提取, 分类层). (2 分)

(2.3) 举例简要阐述 GPT-2 模型如何实现预训练后多任务的零样本学习和上下文学习. (2 分)

参考答案:

(1) **自监督学习思路的相似性**:Skip-gram: **中心词预测上下文单词** (单向, 窗口内多目标, 而 BERT 的 MLM: **双向上下文预测 mask 词元**, 本质上都是掩码预测自监督任务.

获取词向量性质的差异性:Word2Vec 方法获取的词向量为静态, 无法区分多义词 (词向量与所处上下文无关); BERT 使用 MLM 方法获取词向量可以动态捕捉词义变化 (如 Apple 在水果和 Apple 公司中的差异), 与所处上下文相关.

(2)

(2.1) BERT 使用基于 Transformer 的双向编码器 (仅使用编码器层); GPT-1 则使用基于 Transformer 的单向解码器 (仅使用解码器层, 含掩码自注意力).BERT 的自注意力机制: 每个词可同时关注左右两侧所有词; GPT-1 的自注意力机制: 每个词仅能关注左侧历史词 (未来词被掩码屏蔽).BERT 采用 MLM(掩码语言建模): 随机掩盖 15% 的词, 预测被掩盖词和 NSP(下一句预测): 判断两句子是否连续两个任务进行自监督预训练; GPT-1 则采用语言建模 (language modeling): 从左到右预测下一个词 (自回归生成) 来进行自监督预训练.

(2.2) **BERT**:1. 添加特殊标记 [CLS](句首) 和 [SEP](分隔符), 例如:[CLS] 这部电影很棒! [SEP].2. 使用 [CLS] 标记的最终隐藏状态 (维度 768) 作为句子表征, 因其在预训练中通过 NSP 任务学习聚合全局语义.3. 微调添加一个分类层 (dropout 和全连阶层) 来进行文本分类任务的微调.

GPT-1:1. 直接输入原始文本 (无需特殊标记).2. 通过取最后一个词的隐藏状态 (因单向模型仅最右词累积全部左上下文) 或者对所有词的隐藏状态进行平均池化获得句子表征.3. 添加一个分类层进行文本分类任务的微调.

(2.3) **零样本学习**: 在输入文本中显式定义任务 (Prompt), 例如:" 将以下句子分类为积极或消极:' 这部电影很棒! ' 类别:"GPT-2 根据模式补全为" 积极".

上下文学习: 在输入 Prompt 中提供少量任务示例 (无需梯度更新), 例如情感分类:

文本: 服务很好! → 情感: 积极

文本: 质量太差 → 情感: 消极

文本: 剧情无聊 → 情感: 消极

文本: 特效震撼 → 情感:[待模型补充完整]

6. (10 分) 考虑如下简单的网格世界 MDP:

- 状态集 $S = \{s_1, s_2, s_3, s_4\}$;
- 动作集 $A = \{Up, Down, Left, Right\}$;
- 折扣因子 $\gamma = 0.9$;
- 学习率 $\alpha = 0.1$;
- 初始 Q 值表中所有 $Q(s,a) = 0$.

Q-learning 的更新公式为:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

SARSA 的更新公式为:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

(1) 请根据 SARSA 和 Q-learning 的更新公式, 解释为什么 SARSA 是在策略 (on-policy) 学习,Q-learning 是离策略 (off-policy) 学习. (5 分)

(2) 假设初始 Q 值表中 $Q(s_3, Up) = 0.5, Q(s_3, Down) = 1.0, Q(s_5, Left) = 2.0, Q(s_5, Right) = 1.5$, 其余部分全为 0. 考虑如下的交互序列:

1. 从 s_2 执行 *Right* 到 s_3, 奖励 $r_1 = 5$;
2. 从 s_3 执行 Up 到 s_5, 奖励 $r_2 = 3$;
3. 从 s_5 执行 $Left$ 到 s_4, 奖励 $r_3 = 1$.

若 s_4 为终止状态, 即 $Q(s_4, -) = 0$, 请分别计算 SARSA 和 Q-learning 的 Q 值更新过程. (5 分)

参考答案:

(1) SARSA: 目标值中的 $Q(s_{t+1}, a_{t+1})$ 不是对下一状态所有动作取最大值, 而是用智能体实际在下一个状态 s_{t+1} 真正选择的动作 a_{t+1} 的价值. 这意味着智能体使用同一个策略 (例如 ε-greedy) 去生成下一个动作 a_{t+1}, 同一个策略又用来评估 (更新) 当前动作 a 的价值. 因为评估过程和采样过程都依赖同一个策略,SARSA 就是典型的在策略 (on-policy) 算法.

Q-learning: 目标值中的 $Q(s_{t+1}, a_{t+1})$ 是对下一状态 s_{t+1} 下所有可能动作 a_{t+1} 的价值做最大化, 它完全不考虑智能体当前真正会选哪个动作, 而是「假设」智能体在 s_{t+1} 一定会选最优动作 (即取最大 Q 值的动作). 此时行为策略 (behavior policy) 是智能体实际用来采样数据、选择动作的策略 (可以是 ε-greedy 等), 目标策略 (target policy) 是 Q-learning 在更新时所假设或评估的策略, 即始终选择 Q 值最大的动作. 因为目标值是基于一个与行为策略 (真实采样时用的策略) 无关的最优动作, 所以 Q-learning 属于离策略 (off-policy) 算法.

(2) SARSA:

第一步: 从 s_2 执行 *Right* 到 s_3, 奖励 $r_1 = 5$:

$$\begin{aligned} Q(s_2, Right) &\leftarrow Q(s_2, Right) + \alpha\,[r_1 + \gamma \cdot Q(s_3, Up) - Q(s_2, Right)] \\ &= 0 + 0.1 \cdot (5 + 0.9 \cdot 0.5 + 0) \\ &= \boxed{0.545} \end{aligned}$$

第二步: 从 s_3 执行 Up 到 s_5, 奖励 $r_2 = 3$:

$$\begin{aligned} Q(s_3, \text{Up}) &\leftarrow Q(s_3, Up) + \alpha\left[r_2 + \gamma \cdot Q(s_5, Left) - Q(s_3, Up)\right] \\ &= 0.5 + 0.1 \cdot (3 + 0.9 \cdot 2.0 - 0.5) \\ &= \boxed{0.93} \end{aligned}$$

第三步: 从 s_5 执行 $Left$ 到 s_4, 奖励 $r_3 = 1$:

$$\begin{aligned} Q(s_5, Left) &\leftarrow Q(s_5, Left) + \alpha\left[r_3 + \gamma \cdot Q(s_4, -) - Q(s_5, Left)\right] \\ &= 2.0 + 0.1 \cdot (1 + 0 - 2.0) \\ &= \boxed{1.9} \end{aligned}$$

Q-learning:
第一步: 从 s_2 执行 $Right$ 到 s_3, 奖励 $r_1 = 5$:

$$\begin{aligned} Q(s_2, \text{Right}) &\leftarrow Q(s_2, Right) + \alpha\left[r_1 + \gamma \cdot \max_{a'} Q(s_3, a') - Q(s_2, Right)\right] \\ &= 0 + 0.1 \cdot [5 + 0.9 \cdot \max(0.5, 1.0, 0, 0)] \\ &= 0.1 \cdot (5 + 0.9 \cdot 1.0) = \boxed{0.59} \end{aligned}$$

第二步: 从 s_3 执行 Up 到 s_5, 奖励 $r_2 = 3$:

$$\begin{aligned} Q(s_3, \text{Up}) &\leftarrow Q(s_3, Up) + \alpha\left[r_2 + \gamma \cdot \max_{a'} Q(s_5, a') - Q(s_3, Up)\right] \\ &= 0.5 + 0.1 \cdot [3 + 0.9 \cdot \max(2.0, 1.5, 0, 0) - 0.5] \\ &= 0.5 + 0.1 \cdot (3 + 1.8 - 0.5) = \boxed{0.93} \end{aligned}$$

第三步: 从 s_5 执行 $Left$ 到 s_4, 奖励 $r_3 = 1$:

$$\begin{aligned} Q(s_5, \text{Left}) &\leftarrow Q(s_5, Left) + \alpha\left[r_3 + \gamma \cdot \max_{a'} Q(s_4, a') - Q(s_5, Left)\right] \\ &= 2.0 + 0.1 \cdot [1 + 0.9 \cdot 0 - 2.0] \\ &= 2.0 + 0.1 \cdot (-1.0) = \boxed{1.9} \end{aligned}$$