人工智能引论课堂笔记

信息科学技术学院 Seto1

2025 春

前言

主题	知识点与考点
Part2: 编程与数学基础	样本空间，随机事件，古典概型，条件概率，全概率公式，Bayes公式，事件的独立性，随机变量，数学期望、方差和标准差
Part3：逻辑推理与搜索	深搜、广搜，搜索树/图，UCS，启发搜索（A*），蒙特卡洛树搜索，minimax and its alpha beta variant（以及和后面多智能体博弈的关联），CSP问题和SAT问题（DPLL和CDCL，不考）
Part4：机器学习	k近邻、线性回归（平方损失、梯度下降）、逻辑回归（最大对数似然框架）、Softmax回归、L2正则化（L1不考）、k折交叉检验（不考）、决策树与随机森林（信息增益、增益率、基尼系数；标签/特征离散化不考）、多层感知机（能够模拟前向传播和反向传播，包括线性层、ReLU/Sigmoid激活函数、Softmax层）
Part5：计算机视觉	CNN，图像分类数据集（不考），图像特征：HOG特征，图像分类器：线性分类器、最近邻法、SVM（不考）、神经网络（不考），三维重建：针孔模型、摄像机几何、三维重建
Part6：自然语言处理	上下文无关的文法、句法分析（CYK算法）、n元模型（不考）、分词（不考）、基于马尔科夫模型的文本生成（不考）、词袋模型、朴素贝叶斯模型、信息检索（tf-idf）、词表示（one-hot，word2vec）、基于神经网络的自然语言处理方法（了解原理）、Transformer
Part7：知识图谱	知识图谱知识抽取：实体抽取（不考）、关系抽取（不考），知识图谱知识表示：逻辑表示法、一阶谓词逻辑，知识图谱知识推理：逻辑等价式、自然演绎推理、归结演绎推理，GNN（不考）
Part8：智能机器人	机器人概览（不考）、机器人硬件（不考）、机器人智能算法与系统：定位与地图创建（MCL算法）、运动规划（RRT算法）、运动控制（PID算法）
Part9：多智能体	强化学习：问题定义，Bellman方程，策略估值、策略提升、策略迭代、价值迭代、Q-Learning（迭代计算过程） 博弈论：非合作博弈，囚徒困境问题，纳什均衡（根据定义判断，以及对零和博弈计算），多智能体强化学习（不考）
Part10：仿真	几何表达：隐式表示与显式表示，SDF，三角网格；绘制：lambertian反射模型，相机投影模型（透视投影、正交投影），着色计算；仿真：空间和时间的离散化，显式/隐式/半隐式欧拉积分，稳定性问题（不考）；动画（不考）：角色表示，动作捕捉，动作生成方法

Seto1

2025 年 6 月 6 日

目录

第一章　基础知识

1.1　数学基础

概率论基础 样本空间和随机事件；古典概率和条件概率；全概率公式（由因求果）和贝叶斯公式（执果求因，已知结果发生对导致结果发生的因素的可能性大小重新修正）

随机变量： 一个将随机事件映射到实数域上的单值实函数，$X = X(\omega), \omega \in \Omega$.

分布函数： $F(x) = P(X \leq x), x \in R$, 描述随机变量取值的规律.

离散型随机变量（取值有限/无穷可列），将 $P(X = x_k) = p_k$ 称为 X 的**分布律**, 于是 $F(x) = \sum_{x_i \leq x} p_i$.

$\rightarrow$ **一维连续型随机变量**: 连续函数 $F(x) = \int_{-\infty}^{x} f(t)dt$,$f(t)$ 称为 X 的**概率密度**（利用 $\int_{-\infty}^{+\infty} f(t)dt = 1$ 计算）.

数学期望

1) 离散型:$E(X) = \sum_{k=1}^{+\infty} x_k p_k$（要求该级数收敛）

2) 连续型:$E(X) = \int_{-\infty}^{+\infty} xf(x)dx$.

方差 $D(X) = Var(X) = E\{[X - E(X)]^2\} = E(X^2) - E^2(X)$，刻画随机变量对于数学期望的平均偏离程度, 标准差 $\sigma(X) = \sqrt{D(X)}$.

1) 离散型 $D(X) = \sum_k [x_k - E(X)]^2 p_k$.

2) 连续型 $D(X) = \int_{-\infty}^{+\infty} [x - E(x)]^2 \cdot f(x)dx$.

协方差 $Cov(X, Y) = E\{[X - E(X)][Y = E(Y)]\} = E(XY) - E(X)E(Y)$.

相关系数 $\rho_{XY} = \frac{Cov(X,Y)}{\sqrt{D(X)}\sqrt{D(Y)}}$，反映随机变量之间线性相关程度，0 表示独立，1 表示线性相关.

1.2　Python 基础

见程设笔记。

第二章　搜索

2.1　搜索概述

搜索任务主要包括目标、状态（开始状态、目标状态、当前状态）、动作、（状态）转移模型（当前状态随动作如何变化）、动作代价函数，而描述一个搜索问题的解包括其动作序列（路径）和总代价。

最基本的暴力搜索包括深搜和广搜，下面分别描述这两种搜索模式过程：

广搜：根节点入队列（先进先出），循环 [取队头，检验是否为目标，若是则成功跳出，否则将其所有未检验过的子节点入队] 直到队列为空，回传失败。

深搜：根节点入栈（后进先出），循环 [取栈顶，检验是否为目标，若是则成功跳出，否则检验是否有未检验过的子节点，若是则将某个子节点入栈，否则删除当前节点] 直到栈为空，回传失败。

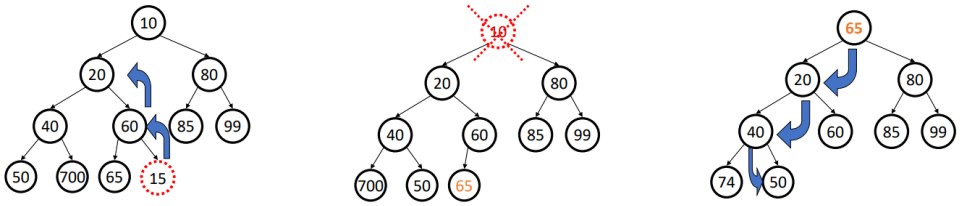
除此以外，在智能体的行动中，还需要一些复杂情况/优化手段，这就是接下来几节所关注的：

- 为什么我们平等对待每一个动作？→ 启发搜索
- 当有一个对手和我们博弈的时候？→ minimax 和剪枝
- 如何系统性的用符号表达状态？→ 逻辑
- 如果环境的转变是不确定性的？→ 不确定性和概率推理
- 搜索空间太大了？→ 蒙特卡洛搜索

2.2　UCS 和 A*

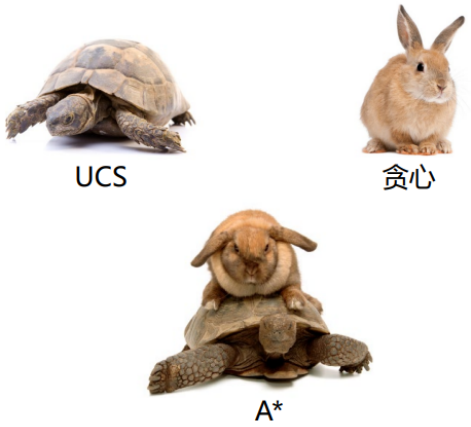
我们指出，在处理最优解问题时，深搜是一种很劣的算法，因为它只能保证找到一组解，却既不能保证其步数最少，也无法保证其成本最小。而广搜可以通过遍历层数保证步数最少，却在有边权问题上无法处理。于是我们提出 UCS（**一致代价搜索**），通过维护按总成本为优先值的优先队列均匀地延展代价轮廓，我们指出（证明将在后续给出）其一定是完备（有解一定能找到）且最优的。

* 了解：**优先队列**是通过维护一个最小堆实现的。最小堆是一棵根节点值总小于叶子节点的完全树。维护操作包括插入（先将其插入最底层，**再依次向上比较交换**，见下左图）和删除（删除根节点，将最底层节点更新到根，**再依次向下比较交换**，见下中右图）



UCS 过程：起点入队，循环 [取队头，若是目标则跳出，将队头标记为**到达过**，对每个子节点，若**未考虑过或考虑过但未到达且代价更小**，则入队/更新节点（update）] 直到队列为空。

但 UCS 仍有其局限性，由于其没有考虑关于目标的信息，其仍然属于盲目搜索，下面介绍利用目标信息的**知情搜索**。一般来说，我们采取的目标信息是**启发**，也即估计当前状态离目标状态离目标还有多远的方程。自然地，我们有极端的贪心搜索：永远扩展启发最小的节点，但其既未考虑走到该点的成本，且启发函数不一定准确，可能退化成一个很劣的算法。那该怎么办呢？如下图：



A* 算法根据积累的代价 $g(n)$ 和启发函数 $h(x)$ 之和决定顺序，从而将两种因素很好地结合，但启发估计不准确的问题依旧存在，特别是当我们对某处作出了**悲观的估计**，可能会使该节点错误地不被扩展，从而破坏最优性。因此我们指出，启发 h **必须是可接受的（乐观的）**，也即满足 $0 \leq h(n) \leq h^*(n)$(真实代价).

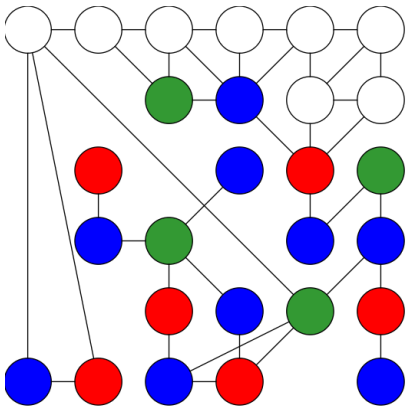
证明乐观的启发在搜索树上最优是容易的：我们从不会高估最优路径上任何一点的代价 $f(n)$，且任何非最优目标的代价就是真实代价，于是这之间的相对大小关系没有改变，**最优路径上的点一定比非最优目标先出队**。

但是在搜索图上，情况发生了改变，我们需要额外考虑 “**边一致性**”，即任意单边的启发值都应是乐观的 $h(A) - h(C) \leq cost(A \to C)$, 这保证了在一条路径上 $f(n)$ 是单调不降的。在搜索树上不需要考虑这一点的原因是树上到任意状态的路径是唯一的，而图中除了有非最优状态，还有**非最优路径**的影响。

2.3　逻辑和 CSP

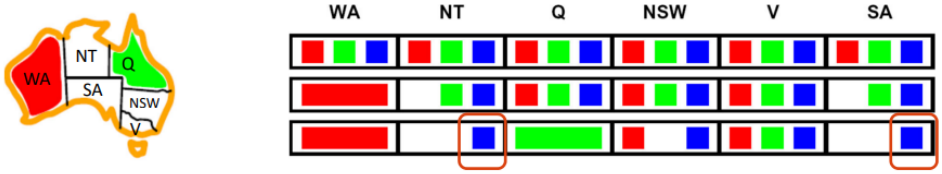
在这一节中，我们研究一种特殊的搜索：**约束满足问题（CSP）**，经典的例子是地图着色/排课表，主要内容是一系列**变量**在**域**中取值（可以是（离散变量）有限/无限域/连续变量（高铁到站时间），但这节中只研究有限域），需要满足一系列**约束**（一元，二元，……），最终实现**目标**：所有变量的满足约束的赋值方式。

我们先研究二元约束问题，一个朴素的想法是一个一个变量赋当前不冲突的值，遇到冲突一层层回溯，但下面这种情况会使其陷入无穷的无效搜索（左上角格子其实已经无路可走，但程序仍机械地在下一行反复回溯）：



于是考虑优化。通过人类玩数独的经验，我们自然地认识到，应该挑选那些有最多约束（还有最少可能）的变量进行赋值，这就是**最少剩余值启发（MRV）**。第二重优化是在赋值时，选给剩下的变量留下更多可能的值（这是因为更多可能值更可能引向一个成功的状态，这可能与人类优先主动压缩搜索空间的策略不同，是因为人类不善于回溯），这称为**最少限制值启发（LCV）**，这两重优化可以使 n 皇后问题规模从 25 上涨至 1000.

需要注意的是，我们在 MRV 中，事实上已经使用了剪枝：从已赋值的值出发，我们将未赋值节点中会导致冲突的节点去掉，从而实现了**提前检查**。但我们的剪枝仍然没有完成，例如在下面的例子中，程序无法有效识别出 NT 和 SA 不能同时为蓝色：



我们的解决方案是**约束传递**，或者更清楚地，在每次赋值后检查每条边的**一致性**：对于某条边 $X \to Y$（这里指的是约束图中的边，也即 X, Y 之间有约束），它是一致的当对每个 X,Y 都有某个不违反约束的赋值方式。

实现检查边一致性的典型算法是 AC-3 算法：所有边（双向）入队，循环 [取队头边 $X \to Y$，检查 X 每个取值，删去 Y 无对应取值的取值；若执行了删除，则将那些终点为 X 的边 $Z \to X$

入队（传播修改）] 直到队列为空。这种剪枝比前向检查更强（我们指出，剪枝显然仍不彻底，但彻底剪枝本来就是 NP-hard 的），但不代表前向检查就没用（在约束稀疏的情况下，前向检查效率更高；有时也会将两者结合使用 Light first，Heavy later 的策略.

上述回溯搜索可以推广到多元约束，是解决 CSP 问题的通用解法。这里给出另一种思考维度：很多 CSP 可以用**逻辑**来表示，从而转化为 **SAT 问题**（可满足性问题，是否存在布尔赋值组合使逻辑约束均满足），此后可以使用 SAT 问题上更高效的算法：**DPLL 算法和 CDCL 算法**（矛盾指引的子句学习）

（命题）逻辑就是给每个**变量**赋布尔值，而用**连接符**与或非（$\wedge, \vee, \neg$）连接组合成逻辑关系（或者说，用**字符**（常量/变量/它们的非）组合成**语句**），所有变量的赋值组合称为**世界**。

一条语句可以被写成**合取范式（CNF）**：$(X_1 \vee \neg X_2 \vee X_3) \wedge (\neg X_3 \vee \neg X_4) \wedge \cdots$（特别地，以 $\wedge$ 连接的内容称为**子句**）或**析取范式（DNF）**：$(X_1 \wedge \neg X_2 \wedge X_3) \vee (\neg X_3 \wedge \neg X_4) \vee \cdots$，二者之间可以通过**德摩根定律** $\neg(P \wedge Q) \iff (\neg P) \vee (\neg Q)$ 和分配律 $(P \wedge (Q \vee R)) \iff ((P \wedge Q) \vee (P \wedge R))$ 互相转换（转换的过程相对复杂，但可以从直观视角观察：DNF 是若干份人类能看懂的通关攻略（走对一个就行了）而 CNF 是更易求解的安全检查机制（只要有一个不满足就失败了））：

$$
\begin{aligned}
(DNF)(X_1 \wedge \neg X_2) \vee (\neg X_3 \wedge X_4) &= \neg(\neg(X_1 \vee \neg X_2) \wedge \neg(\neg X_3 \wedge X_4)) \\
&= \neg((\neg X_1 \vee X_2) \wedge (X_3 \vee \neg X_4)) = (\text{展开})\neg((\neg X_1 \wedge X_3) \vee (\neg X_1 \wedge \neg X_4) \vee (X_2 \wedge X_3) \vee (X_2 \wedge \neg X_4)) \\
&= \neg(\neg X_1 \wedge X_3) \wedge \neg(\neg X_1 \wedge \neg X_4) \wedge \neg(X_2 \wedge X_3) \wedge \neg(X_2 \wedge \neg X_4) \\
&= (X_1 \vee \neg X_3) \wedge (X_1 \vee X_4) \wedge (\neg X_2 \vee \neg X_3) \wedge (\neg X_2 \vee X_4)(CNF)
\end{aligned}
$$

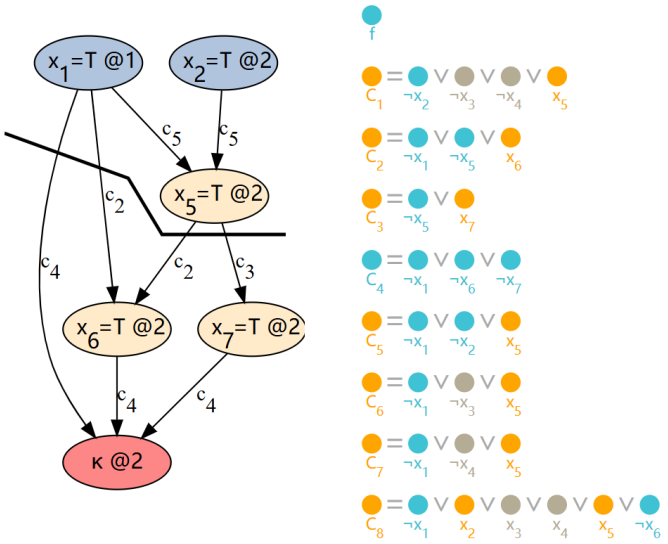
下面我们介绍针对 SAT 问题的两种算法：DPLL 和 CDCL。

DPLL 算法将子句分为真子句、假子句、单字符和其他四种，并不断进行**单字符传递**（当一个子句中其他字符都为假，则将剩下那个字符赋为真），这个过程称为布尔约束传递（BCP），其具体过程是：不断从未赋值变量中（可能启发式地）选择一个，分别赋为真与假进行递归，在每次赋值后，**循环进行单字符传递**，再循环进行**纯字符匹配**（这是指某未赋值的变量在还未确定的子句中极性（是否有非）全部相同，于是将其经过极性操作后全部赋为真，则其互相之间不会冲突且一定最优），操作结束后若有空子句则立判为假，若所有子句都已为真则立判为真，否则继续递归。

但是 DPLL 算法有其局限性。它仅是平凡地做决定，并且无法从冲突中学到除了冲突本身的新内容；同时，它只能回溯一步，这使得它很可能在注定失败的状态里挣扎。

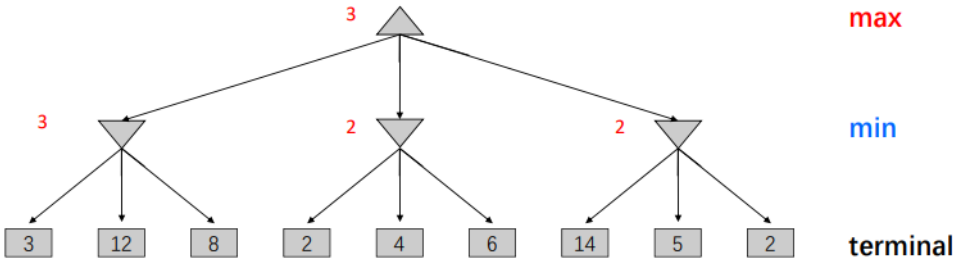
我们提出矛盾指引的子句学习（CDCL）算法加以优化。在这一算法中，我们构建了一张隐含图，其中节点分为猜测的节点、BCP 产生的赋值节点和矛盾节点，节点除了元素和值外，还有层级这样一个属性，在每次猜测赋值时 +1，当遇到矛盾时，它会触发其核心机制——**回溯并学习新子句**。其流程如下：找到与冲突节点直接相连的所有节点，将所有非当前层节点和 [当前层的节点回溯到的 1UIP 节点（**指的是当前层根节点到矛盾节点必经节点中深度最大的**）] 取值取反学到新子句，并回溯到之前取出的**非当前层节点中层数最大的那一层**触发 BCP（特别地，如果没有非当前层节点，则回溯到**第 0 层**），最后推得一个无矛盾的结果或返回失败，如下图，在这次回溯中学

到了子句 $\neg x_1 \vee \neg x_5$:

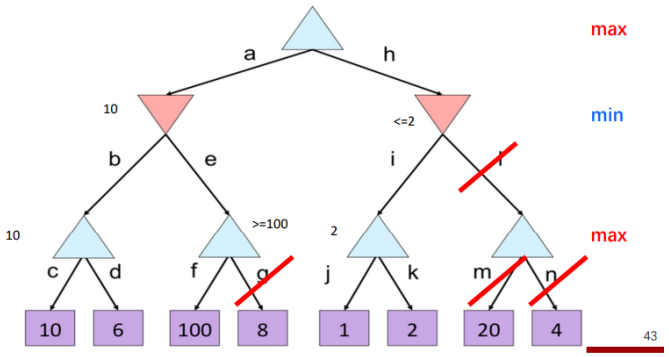


2.4　对抗搜索

对博弈，可以从以下几个维度可以进行分类：确定性/随机性；单一/2 个/多个智能体；是否为零和博弈；是否是完美信息博弈。我们先研究简单的确定性零和博弈——如井字棋、围棋等——的**策略**。先考虑单一智能体的情况，我们可以自然地定义**状态价值**为从该状态出发可能获得的最大效用（分数），于是只要层层向上更新即可。但当有 2 个乃至多个智能体（此时给出效用元组，每个玩家最大化自己的部分）时，由于零和博弈的特性，对方（**如果是理性的**，否则可能需要修改 MIN 节点成为期望节点，但**这样就没法进行后续剪枝**）的决策应当是最小化效用，由此设计出朴素的**极小极大搜索（MiniMax）**，如下图能自然地看出其过程：



但显然，这种算法效率低下，我们需要优化算法，对博弈树进行剪枝，由此提出 **Alpha-Beta 剪枝**：对于 MIN 节点，**设当前更新后值为 n，其祖先 MAX 节点中最大值为 m**，则整个 MIN 节点就不会对那个 MAX 节点产生贡献，进而不会对根节点产生贡献，于是可以直接将后续节点剪掉，MAX 节点同理。于是我们记祖先 MAX 节点最大值为 α，祖先 MIN 节点最小值为 β，则在 MAX-Agent 里对 β 剪枝，并更新 α 即可，可参考下图体会剪枝过程：



这的确加快了搜索效率，但仍然受资源限制无法走到叶子节点，于是我们可以转而部分牺牲最优性，进行**深度受限搜索**，对于非叶子节点用估值函数替代终态分数（显然，估值函数——尽管进行了机器学习——仍不能做到完美，于是深度的选取就成了最优性和事件复杂度之间的权衡）。

2.5　蒙特卡洛搜索

本节将通过蒙特卡洛搜索算法解决特大状态空间和不确定性。我们面前有一组老虎机，需要探索出选哪个收益最大（或者说亏的最少?），显然，**次数越多越准确**（这就是所谓的**蒙特卡洛方法**），所以我们要进行反复试验，但在每个时间点选哪个是一个令人头疼的问题——我们似乎应当抓住目前平均最优的一直试验下去，于是得到 $t+1$ 时刻对 a 的估值:

$$Q_{t+1}(a) = \begin{cases} \frac{1}{N_t(a)} \sum_{\{\tau|\tau \le t, A_\tau = a\}} R_\tau, & N_t(a) > 0 \\ 0, & N_t(a) = 0 \end{cases}$$

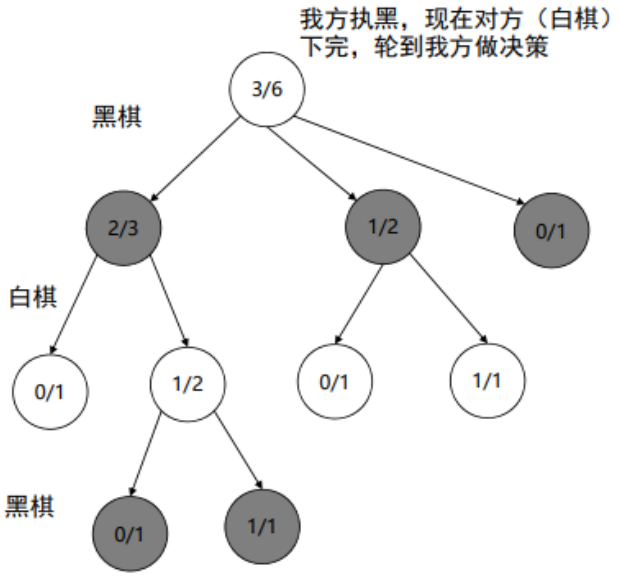
但若仅根据此估值判断，可能目前看上去不优的选择**是因为试验次数不足产生了偏差**，一个极端的例子是 A 一次输了 10 元，B10 次输了 5 元，那就算 A 实际上是一个能让你赚钱的老虎机，我们也永远不会知道。于是，我们需要在选择中加入对少试验动作的奖励，一种很自然的想法是 **ϵ-greedy 算法**：$1-\epsilon$ 概率选择估值最高的动作（利用），ϵ 概率**从其他中任选一个**（探索）。

但这样无法区别最优外的其他动作（例如，这会使一个还不错的动作和一个极其差的动作有相同的计算概率），为了更多地去尝试那些**更有希望成为更优的动作**，提出**最大置信（UCB）**以平衡不确定性:

$$A_{t+1} = argmax_a[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}]$$

其中 $Q_t(a), c\sqrt{\frac{\ln t}{N_t(a)}}$ 两项分别平衡了估值的大小和次数的多少，尤其在动作选择数较多的情况下明显优于 ϵ-greedy 算法（当然，后者也非一无是处，其写法简单且与神经网络配合较好，应用同样广泛）。

于是，我们通过 UCB 进一步扩展到如下图的博弈树（注意每个节点代表这个玩家自身的获胜次数/通过该节点次数），下面我们将在这棵树上进行蒙特卡洛搜索:



首先，从根节点开始，依次把经过的节点当作老虎机处理，**选择**UCB 值最高的节点向下递归，直到叶子节点。此时对此叶子节点进行**扩展**（可选常见的单子节点扩展或全子节点扩展），将该节点置为 0/0. 接下来，使用随机/弱策略（如 ϵ-greedy）**模拟**一次盘面（也不一定会模拟结束，可能到最大步数后返回启发值），此后**回溯**，对路径上每个节点更新参与值/获胜数，如此往复。

著名的 AlphaGo 事实上就是用的优化后的蒙特卡洛搜索，仅有几处不同：在选择阶段的 UCB 函数中，利用人类对局数据训练出的 Policy 网络，使用 $\frac{cP(t)}{1+N(t)}$（$P(t)$ 是训练出的权重）替代了原本的探索项；在模拟阶段，用快速落子（本来是更弱的选择策略，直到 AlphaZero 将其与 Policy 网络合并）策略代替随机选择，并同时使用 Value 网络给出估值（胜率）（在 AlphaZero 中，**Value 网络与 Policy 网络合并成了一个大型网络**），回溯阶段综合模拟结果和估值结果回传价值。至于估值是如何给出的，我们将在下一章中介绍。

第三章　机器学习

3.1　机器学习基础和线性回归

我们将机器学习按任务分为分类/回归，依据为标签是连续（回归）/离散（分类）值

模型评估：给定全部数据，按（9：1 左右）比例随机划为/按时间顺序划分为训练集和测试集，于是可分为**训练误差和测试误差**（衡量泛化能力）。若训练完成后，训练误差仍然很大，则称为**欠拟合**，若测试误差远远大于训练误差，则称为**过拟合**。

我们先提出一个最简单的机器学习模型：**k 近邻算法**，即用训练样本中离其最近的 k 个样本中占多数的标签来预测，其不需要训练且较简单（只需欧氏距离），但需要存储所有样本、计算所有样本到其距离，且所谓距离函数难以评估，并且（最严重的是）在高维度计算中，能包住 k 近邻的超立方体边长过大，导致**维度灾难**。

上面这种模型属于非参数化模型（需要保留训练样本），而我们主要讨论参数化模型（训练好参数后，可以丢弃训练数据，仅依靠模型参数，写成 $y=f(x)$ 来预测新样本）

最简单的参数化模型是**线性模型** $f(x)=w^Tx+b$, 其中参数 w,b 分别称为**权重**和**偏置**（提供一个基础预测值，使其不需要过原点）。其训练方式是通过**最小化损失函数**，我们通过在训练集上一般取平方损失函数（也称为最小二乘法）：

$$\min_{w,b}\frac{1}{n}\sum_{i\in[n]}L(f(x_i),y_i)=\min_{w,b}\frac{1}{n}\sum_{i\in[n]}(f(x_i)-y_i)^2$$

为了优化这个函数，我们进行**梯度下降**：令 $J(w,b)=\frac{1}{n}\sum_{i\in[n]}(f(x_i)-y_i)^2$, 则

$$w\leftarrow w-\alpha\cdot\frac{\partial J(w,b)}{\partial w}\in\mathbb{R}^d, b\leftarrow b-\alpha\cdot\frac{\partial J(w,b)}{\partial b}\in\mathbb{R}$$

即沿梯度的方向走步长为 $\alpha\cdot\nabla J(w,b)$ 的一步，计算得

$$w\leftarrow w-\alpha\cdot\frac{2}{n}\sum_{i\in[n]}e_ix_i, b\leftarrow b-\alpha\cdot\frac{2}{n}\sum_{i\in[n]}e_i, e_i=w^Tx_i+b-y_i(\text{残差})$$

迭代直到 $J(w,b)$ 不再下降（两次差小于 eps）/达到最大次数, 对于一般的函数，梯度下降仅可以得到局部最小值（我们指出，局部最小值与全局最小值的差距通常不大）或（更糟糕地）得到**鞍点**，而梯度下降能得到全局最小值的条件是 $f(x)$ 是**凸函数**，即 $\forall t\in[0,1], f(tx_1+(1-t)x_2)\leq tf(x_1)+(1-t)f(x_2)$, 最小化凸函数的方法就称为**凸优化**.

我们指出，线性回归函数就是一个凸函数（其关于参数事实上是二次函数），于是通过梯度下降即可很好地得到全局最小值。

3.2　逻辑回归、多分类与正则化

事实上，对于大部分监督学习算法（即给定训练数据的学习）都遵循与线性回归类似的**经验风险最小化框架（ERM）**，仅仅需要选择特定的模型 $f(x)$ 和损失函数 $L(f(x),y)$.

我们来研究二分类问题，其标签只包括 $\{-1,1\}$, 分别表示负类和正类, 显然难以无法直接使用 $f(x)\in\mathbb{R}$ 进行拟合，所以采用 $sign()$ 函数将实数输出转换为 $\{-1.1\}$:

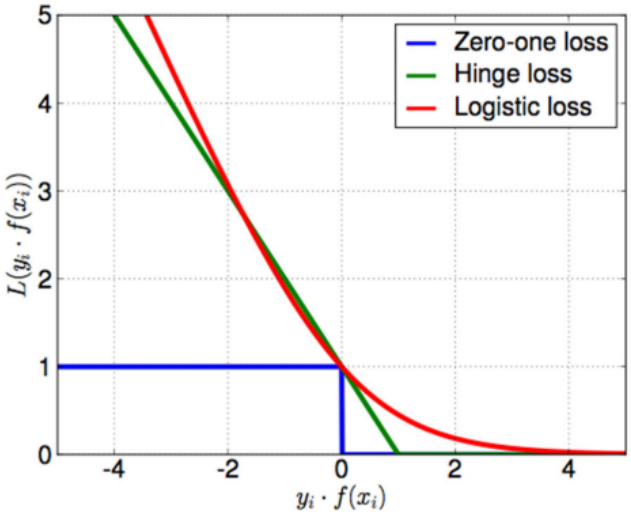
$$sign(f(x))=\begin{cases}1, & if\ f(x)>0\\ -1, & if\ f(x)<0\end{cases}$$

那么应当使用怎样的损失函数呢？最直接也正确的方法是使用零一损失函数:

$$L(f(x),y)=\begin{cases}1, & if\ sign(f(x))=y\\ 0, & if\ sign(f(x))=-y\end{cases}=\begin{cases}1, & if\ y\cdot f(x)\geq 0\\ 0, & if\ y\cdot f(x)<0\end{cases}$$

但是，零一损失函数是阶跃函数，在 0 处不可微不连续，其余处梯度都为 0，无法使用梯度下降优化.

我们先给出，最终的逻辑回归的**交叉熵损失函数** $L(f(x_i),y_i)=\ln(1+e^{-y_if(x_i)})$, 如下图红线所示，其作为零一损失函数的凸上界，最小化这个函数同样可以（通过惩罚错误而鼓励模型做出正确分类从而）最小化零一损失函数（注：图中绿线为另一种常见的替代损失函数——合页损失函数 $L(f(x_i),y_i)=\max(0,1-y_if(x_i))$。



下面使用**最大似然框架**推导损失函数。我们对数据建模 $p(y|x;\theta)$, 进而通过最大化数据的**似然**（预测正确的概率）进行调参。在样本独立前提下，最大似然估计为

$$\max_\theta\prod_{i\in[n]}p(y=y_i|x=x_i;\theta)\Rightarrow\max_\theta\sum_{i\in[n]}\ln(p(y_i|x_i;\theta))(\text{取对数防止概率连乘超精度})$$

举例：抛不均匀硬币 n 次，正面 m 次，估计参数为 p, 则似然为 $p^m(1-p)^{n-m}=e^{m\ln p+(n-m)\ln(1-p)}$, 取导数为 0 即得 $p=\frac{m}{n}$.

于是我们需要寻找合适的 $p(y_i|x_i;\theta)$, 即将线性模型 $f(x)=w^Tx+b\in\mathbb{R}\to[0,1]$. 我们指出，可以采用 **sigmoid 函数**:$\sigma(x)=\frac{1}{1+e^{-x}}$, 此函数满足模型绝对值较大时趋近于 0/1，且**关于 (0,0,5) 中心对称**. 于是

$$p(y=1|x;w,b)=\sigma(f(x))=\sigma(y\cdot f(x)),p(y=-1|x;w,b)=1-\sigma(f(x))=\sigma(-f(x))=\sigma(y\cdot f(x))$$

所以 $p(y|x;w,b)=\sigma(y\cdot f(x))$ 恒成立.

于是即最大化 $\sum_{i\in[n]}\ln(\sigma(y_i\cdot f(x_i)))$, 代入 $f(x_i)=w^Tx_i+b$ 得须最大化 $-\sum_{i\in[n]}\ln(1+e^{-y_i(w^Tx_i+b)})$, 转化为最小化即推出之前给出的交叉熵损失。

于是自然地通过梯度下降，得出逻辑回归训练方式，训练完成后用 sign 函数进行判断即可:

$$\frac{\partial J(w,b)}{\partial w}=-\frac{1}{n}\sum_{i\in[n]}\frac{e^{-y_i(w^Tx_i+b)}}{1+e^{-y_i(w^Tx_i+b)}}y_ix_i=-\frac{1}{n}\sum_{i\in[n]}(1-p(y_i|x_i;w,b))y_ix_i$$

$$\frac{\partial J(w,b)}{\partial b}=-\frac{1}{n}\sum_{i\in[n]}(1-p(y_i|x_i;w,b))y_i,\text{梯度下降迭代更新即可}$$

但对于多分类问题，若仍使用 k 个二分类器代价太高且（由于正负类比例失调）容易摆烂全判负类. 于是提出 **Softmax 回归**，通过共同训练 k 个模型解决多分类问题. 此时需要满足 $\sum_k p(y=k|x)=1$, 于是不能用 sigmoid 函数，转而使用 softmax 函数将概率归一化:

$$p(y=k|x)=\frac{e^{f_k(x)}}{\sum_{j\in[K]}e^{f_j(x)}}$$

其中指数函数的作用是放大正类的概率，即使得若 $f_k(x)\gg f_j(x),\forall j\neq k$, 则 $p(y=k|x)\approx 1$.

再依据此函数使用最大对数似然/最小化交叉熵损失估计参数

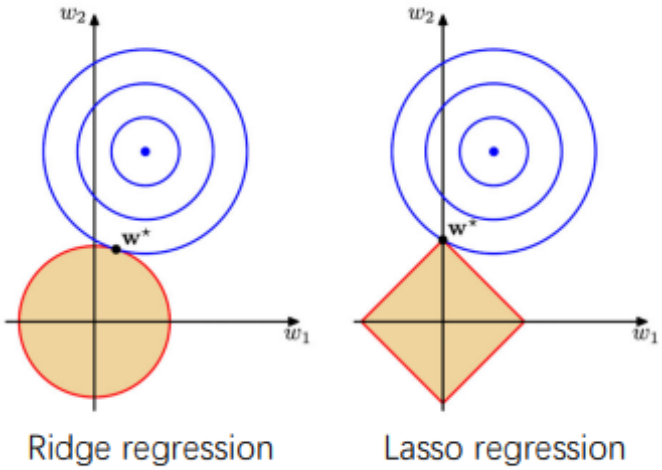
$$\max_\theta\sum_{i\in[n]}\ln(p(y_i|x_i))\Rightarrow\min_\theta\frac{1}{n}\sum_{i\in[n]}(\ln(\sum_{j\in[K]}e^{f_j(x_i)})-f_{y_i}(x_i))$$

在实际应用中，在参数较多时，简单的训练往往会导致过拟合，其原因通常是：某几个特征维度**支配**了预测，即权重过大；或存在大量无用维数。我们可以使用**正则化**解决过拟合问题:

$$\min_f\frac{1}{n}\sum_{i\in[n]}L(f(x_i),y_i)+\lambda\cdot R(f)$$

其中 $\lambda\cdot R(f)$ 称为正则化项，用于惩罚复杂的模型，λ 称为**超参数**，在参数前预先指定（太小容易过拟合，太大容易欠拟合，但我们需要指出：正则化一定会使训练误差变高，但通过合适的选取超参可能使测试误差变低）。

正则化一般分为 **L2 正则化和 L1 正则化**，L2 正则化中 $R(f)=\|w\|_2^2=w^Tw$, 其会放大从而惩罚过大的权重，使权重分配更平均；L1 正则化中 $R(f)=\|w\|_1$, 其鼓励稀疏的 w，即多数维度变为 0。下图可以直观地解释二者的用途（观察正则化项与等值线的最先接触点）:



其表示了带正则化的线性回归，左图表示线性回归 +L2 正则化，也称岭回归；右图表示线性回归 +L1 正则化，也称 Lasso 回归。而对于逻辑回归，其完整形式是交叉熵损失 +L2 正则化，从而梯度实际上是 $\frac{\partial J(w,b)}{\partial w} = -\frac{1}{n}\sum_{i\in[n]}(1-p(y_i|x_i;w,b))y_i x_i + 2\lambda w$, 同时还有**支持向量机**（SVM），即合页损失 +L2 正则化（其关注超平面距离最近的点距离最远，也即**最大间隔准则**，使模型容错率更强（边界附近不会有很多点），其优化主要针对关键点）.

针对超参数，需要在训练集和测试集之外引入**验证集**，大约 8-1-1 占比，对超参数选择在训练集上训练参数，在验证机上验证误差，选择最优的验证机上超参数进一步在测试集上测试泛化能力.

最后，有时总数据量比较少，10% 的验证集不足以准确地验证模型性能，为解决此问题，我们介绍 **K-折交叉验证**，即将除测试集以外的样本分为 K 份，并取 K 次，每次取一份为验证集，其他为训练集，取这 K 次的平均结果作为超参数效果参考（优点是性能好，缺点是在大样本上慢——以至于到更大样本上的大模型中，我们连验证集都免去了）

3.3　决策树与随机森林

在上一讲中提到的线性模型仅能给出简单（单层）的 “分类树”，或简单的线性分隔超平面——那么是否可以设计出具有比 $sign(w^T x + b)$ 逻辑更复杂的分类模型？我们介绍**决策树**，其中根节点是样本全集，每个非叶节点对应于一个属性测试，而叶节点对应于决策结果（取多数类）

为了确定合适的属性进行划分，我们希望每次划分使得子集的**纯度**不断升高，这样，只需要递归地使用最好的属性划分即可。那么如何衡量一个集合的纯度呢？我们引入**熵**的概念：

考虑 4 个符号 abcd，用平均 2bits 编码每个符号，但如果我们知道 $p(a) = 0.5, p(b) = 0.25, p(c) = p(d) = 0.125$, 则我们可以用 0 编码 a,10 编码 b,110 编码 c,111 编码 d, 则 bits 数期望为 1.75，于是我们指出，在信息论中，**确定性越高的事物信息量**（形式化地，需要多少比特进行编码）**越低**（极

端地来说，100% 发生的事情没有任何信息量）由此，我们定义信息论中的**熵**:

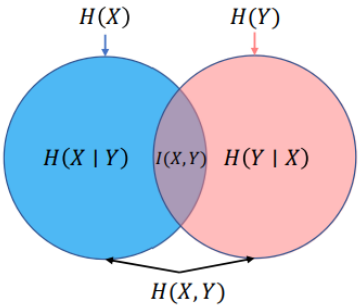
$$H(X) = -\sum_x p(x)\log_2 p(x)$$

当且仅当 $p(x)$ 为一个确定分布时 $H(X) = 0$, 而当所有事件等概率时达到最大熵 $\log_2 n$.

而在多参数情况下，我们自然有**联合熵** $H(X,Y) = -\sum_x \sum_y p(x,y)\log_2 p(x,y)$ 表示联合概率分布的不确定程度. 同时有**条件熵**:$H(Y|X) = \sum_x p(x)H(Y|X=x) = -\sum_x p(x)\sum_y p(y|x)\log_2 p(y|x) = -\sum_x \sum_y p(x,y)\log_2 p(y|x)$, 注意到我们有

$H(X,Y) = H(X)+H(Y|X) = H(Y)+H(X|Y) = H(Y,X)$(可形式化地用“X,Y 分别用几个比特”理解)

同时，有**互信息**: $I(X,Y) = H(X) + H(Y) - H(X,Y)$, 衡量 X,Y 的相关性（X,Y 独立时 $I(X,Y) = 0$），几种熵的关系可从下图中看出:



自然地，我们可以用信息熵度量纯度:

$$H(D) = -\sum_{k\in[K]} \frac{|D_k|}{|D|}\log\frac{|D_k|}{|D|}$$

以及属性 A 对集合 D（标签为 Y）的**信息增益**:

$$g(D,A) = H(D) - \sum_{i\in[m]} \frac{|D^{A=a_i}|}{|D|}H(D^{A=a_i}) = H(Y) - H(Y|X) = I(Y,X)$$

于是，分别计算各属性带来的信息增益，即可判断出选择的属性。但这会带来一个问题，若以“序号”作为一个属性，则其划分后**熵为 0**，会被优先选择（也即前面提出的信息增益准则倾向于选择更多取值可能的属性），于是提出新的划分准则**增益率**，对取值可能过多的属性进行**惩罚**:

$$g_R(D,A) = \frac{g(D,A)}{H_A(D)}, H_A(D) = -\sum_{i\in[m]} \frac{|D^{A=a_i}|}{|D|}\log\frac{|D^{A=a_i}|}{|D|}$$

注意区分 $H(D^{A=a_i})$（即属性 A 分类下对于标签 Y 的纯度）和 $H_A(D)$（属性 A 本身的纯度）!

除了信息熵之外，还可以通过**基尼指数**度量纯度:

$$Gini(D) = \sum_{k\in[K]} \frac{|D_k|}{|D|}(1-\frac{|D_k|}{|D|}) = 1 - \sum_k (\frac{|D_k|}{|D|})^2 \Rightarrow Gini(D,A) = \sum_{i\in[m]} \frac{|D^{A=a_i}|}{|D|}Gini(D^{A=a_i})$$

而现实分类问题中，**属性和标签都可能是连续的**（例：认真工作 → 每天工作时长；是否为好科学家 → 科学家水平打分）。针对这两种情况，分别有以下策略：

针对属性的连续，可以使用**二分法**，在每两个取值间尝试划分（转化成二分类问题），选出增益率最大的划分点。

针对标签的连续，可以使用**回归树**代替决策树，将每个节点的标签集合转化为**标签平均值**以进行分类，此时通过 **L2 Loss** 重新定义纯度：

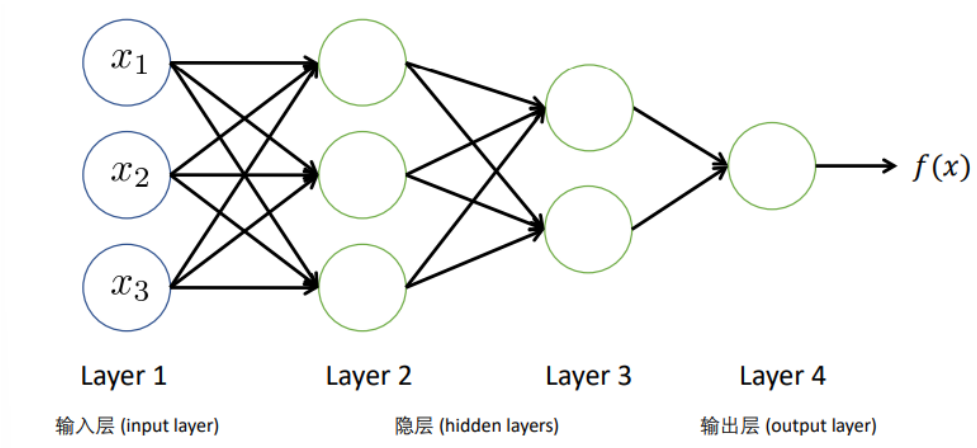
$$L(D)=\sum_{j\in D}(y_j-\overline{y}_D)^2(\text{类似方差})\Rightarrow L(D,A)=\sum_{i\in[m]}L(D^{A=a_i})$$

进一步推广，很多时候单一模型泛化能力及鲁棒性不足，于是提出**集成学习**，将多个（**好而不同的**）个体学习器结合。最经典的例子是**随机森林**，其构建多个搜索树，并通过样本扰动（随机采集不同的训练集）和属性扰动（只选一部分样本特征进行划分）实现独立性，最后选择多数类作为预测结果。

3.4　神经网络与反向传播

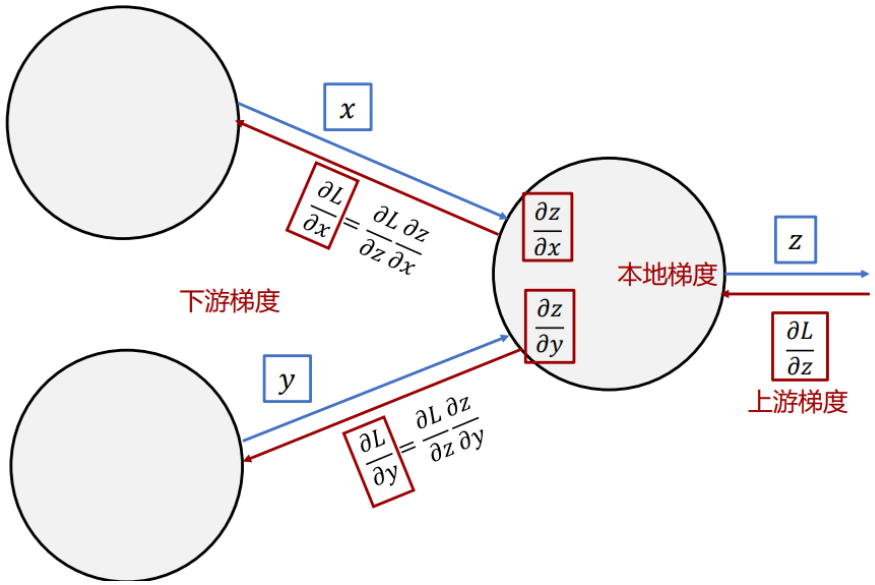
此前介绍了逻辑回归作为分类问题的基础，但其归根结底是线性模型，对**线性不可分问题**（例如 xor 操作）难以解决。这导致了第一次 AI 寒冬，为解决此问题，提出了（至少从名字上）模拟人脑的**神经网络**，感知器以及简单的线性回归/逻辑回归都可视为单层神经网络。

最简单的多层神经网络是如下图所示的**多层感知器**（MLP）（更容易顾名思义的名字是**全连接神经网络**）。事实上，简单的多层神经网络反而是常见的，因为对于神经网络的水平，设计的复杂程度是次要因素，**隐层的维数**（神经元个数）和**深度**（层数）才是主要因素.

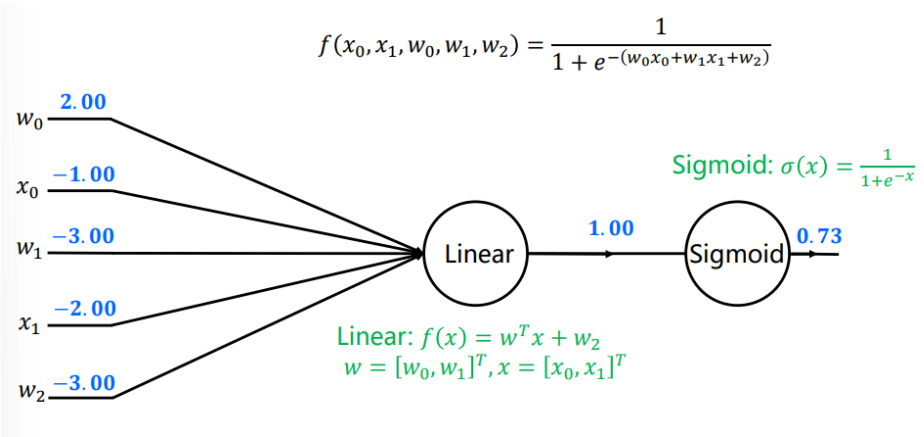


我们用一个公式表示（**除输出层之外）正向传播**的过程:$a_i^{(k+1)}=g(w_{1i}^{(k)}a_1^{(k)}+w_{2i}^{(k)}a_2^{(k)}+\cdots+w_{si}^{(k)}a_s^{(k)}+b_i^{(k)})$, 其中 $a_i^{(k)}$ 分别为第 k 层的参数（输入即为 $a_i^{(1)}$），$w_{ij}^{(k)}$ 代表从第 k 层第 i 个参数传到 $k+1$ 层第 j 个参数的权重，而 $g(\cdot)$ 代表一**个激活函数（使得结果非线性）**. 激活函数有很多种，如 sigmoid、tanh 等，而最常用的激活函数为 $ReLU(x)=\max(0,x)$, 其在大部分问题上表现良好. **输出层的正向传播**是与**任务**挂钩的（二分类 →sigmoid 函数；回归问题 → 直接输出线性组合）

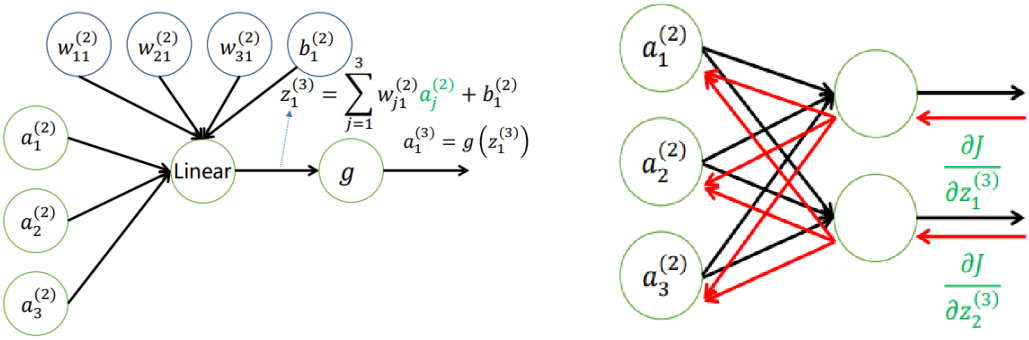
需要思考的下一个问题是：**如何训练参数** w, b? 我们提出**反向传播**，如下图，其将梯度回传化为了**马尔科夫问题**，即第 k 层的梯度只依赖第 $k+1$ 层梯度的结果，例如 $z=xy$, 则利用**上游梯度** $\frac{\partial L}{\partial z}$ 和**本地梯度** $\frac{\partial z}{\partial x}=y$, 于是**下游梯度**即为 $\frac{\partial L}{\partial x}=\frac{\partial L}{\partial z}\frac{\partial z}{\partial x}=y\frac{\partial L}{\partial z}$:



于是，可以将复杂的函数拆分成基元计算（$z=x+y$ 对应直接将梯度复制，$z=xy$ 对应梯度交叉相乘，$z_1=z_2=x$ 对应将梯度汇聚相加，$z=\max(x,y)$ 对应选择性地将梯度复制给最大值），也可拆分为较完整的基元步骤，转化为如下图的计算图：



将其放进神经网络中，如下图，需要使用上一层参数更新后一层 w, b, a 的梯度：



$$\frac{\partial J}{\partial a_j^{(2)}}=\sum_{i=1}^{2}\frac{\partial J}{\partial z_i^{(3)}}\frac{\partial z_i^{(3)}}{\partial a_j^{(2)}}(\text{用于继续反向传播}),\frac{\partial J}{\partial w_{ji}^{(2)}}=\frac{\partial J}{\partial z_i^{(3)}}a_j^{(2)},\frac{\partial J}{\partial b_i^{(2)}}=\frac{\partial J}{\partial z_i^{(3)}}(\text{梯度下降})$$

下面将正向传播、反向传播写成更简洁的**矩阵形式**：

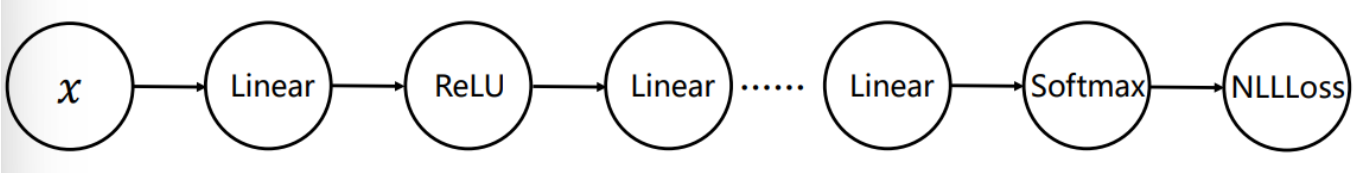
正向传播：$z^{(L+1)}=(W^T)^{(L)}a^{(L)}+b^{(L)},a^{(L+1)}=g(z^{(L+1)})$(这里课件诡异地将 W^T 记作了 W)

反向传播：$$\frac{\partial J}{\partial z^{(L+1)}}=\frac{\partial J}{\partial a^{(L+1)}}\odot(\frac{dg(z)}{dz}\Big|_{z=z^{(L+1)}})$$

$$\frac{\partial J}{\partial (W^T)^{(L)}}=\frac{\partial J}{\partial z^{(L+1)}}(a^T)^{(L+1)},\frac{\partial J}{\partial b^{(L)}}=\frac{\partial J}{\partial z^{(L+1)}}$$

$$\frac{\partial J}{\partial a^{(L)}}=W^{(L)}\frac{\partial J}{\partial z^{(L+1)}}(\text{实际计算的时候注意矩阵尺寸对应相乘即可})$$

有了矩阵形式，就可以将神经网络简化为层与层连接的计算图形式，在此图上前向/反向传播，梯度下降直至收敛即可：



特别地，对于梯度下降环节，原始的**全量梯度下降**可能会因数据规模过大而难以操作，而极端的每次随机抽取一个样本进行**随机梯度下降**又会严重影响模型的准确性，于是提出**小批量梯度下降**，每次迭代抽出一**批样本**，可以通过调整超参数 **batch size** 平衡硬件限制和准确性。

第四章　计算机视觉

4.1　图像分类与卷积神经网络

图像分类，即判断图像所属的类别标签，事实上面临诸多难点——视角变化大、姿态变化多、遮挡严重、类内差异大、尺度变化大……于是尝试设计图像分类算法。

在计算机中，图像由 $I = f(x, y)$ 表示，每个点称为一个**像素**。根据数字图像，大致分为二值图像、灰度图像（一般来说**位深度**为 8，代表储存每个像素需要 8 个 bit，有 256 种灰度）、彩色图像（RGB 三通道分量）

实现图像分类的基本方法是：利用算法对图像数据进行处理，**提取局部特征**，再通过 **BoWs**（Bag-of-Words）进行局部**特征聚类**筛选类内共有特征，形成描述整张图的统计直方图特征，进而训练**分类器**以对新图像进行预测。

下面介绍一些经典的局部特征算子。**HOG** 特征描述子（梯度直方图）希望通过描述边缘的方向梯度以刻画目标的形状（适合行人检测，但无法处理遮挡）。其基本流程如下：将彩色图像通过**色彩变换与灰度校正**（某种意义上的归一化以均衡光照分布）转换成灰度图像；计算每个点的**梯度方向和大小**；将图像分为若干 16×16 的 cell，每个 cell 内**对梯度分布进行投票**（按 θ 每 $20°$ 分类形成 9 维向量）；进而将相邻的 4 个 cell 组成一个 block，组合成 36 维向量并进行 **L2 归一化**（解决局部亮度和对比度变化对特征的影响），最后所有 block 组合成 $36N$ 维向量，也即最终的 HOG 描述子.

Haar 算子（不考）主要用于面部识别，反映的是图像的**灰度变化**（眼睛比脸颊的颜色深）。其实现是通过**计算黑白相间的矩形分别像素和的差值**判断是否分区（由区域的形状可以看出，其对旋转适应不佳），具体计算是通过前缀和实现的。获得多个特征后将每个 Haar 特征作为一个弱分类器，组合成一个级联强分类器即可。具体可以参考这篇博客。

SIFT 算子（不考）通过找出那些在不同方向/亮度/大小下都不容易消失的**关键点**（比如角落、边缘交汇处等），并对每个关键点描述其边缘方向、纹理分布，总结成一个 128 维的 **“指纹”向量**，从而实现了光照旋转尺度不变性。（缺点是若找不到有效的关键点——如边缘光滑时——会很低效）

而分类器是通过计算图像特征与查找库中的 L1/L2 距离，并通过此前讲过的最近邻/k 近邻 + 投票/线性分类（**支持向量机**）/softmax 多分类进行标签预测。

但是上面这些方法都已经是历史了，当下流行的做法是通过**卷积神经网络（CNN）**进行**自动特征提取**。

CNN 主要的组件有卷积层（**Convolution Layer**）、归一化层（**Normalization**）、激活函数（ReLU）、池化层（**Pooling Layer**）、随机失活（**Dropout**）、全连接层（Fully connected Layer），下面将对其进行分别介绍。

卷积层（卷：滑动，积：累积）是关键的自动提取特征的组件，例如，对于一张 32×32×3(长/宽/通道数) 的图片，可以通过一个 $5 \times 5 \times 3$ 的**滤波器**在空间上划过，对于每个块输出一个 $W^T \cdot X + b$（注意这里是内积）形式的**标量**，最终形成一个 $28 \times 28 \times 1$ 的**卷积核**（在实际操作中，为了不使图像尺寸快速减小/不丢失边缘信息，常常使用**补零**操作，在原图边缘加若干圈 0 以仍然得到 32×32 的图片）。（用不同参数的滤波器）重复上述操作 6 次，得到 $28 \times 28 \times 6$ 的输出，这就完成了一次卷积操作。其中的最后一个参数（**通道数**）会随卷积次数的增加不断升高，这代表了自动提取的**特征种类数**。CNN 就是由 [卷积层 + 激活函数] 为主体组件搭建的。

归一化层大致将各特征统一到相同尺度，从而利于优化。最常见的归一化操作是**批归一化**（batch normalization），对每一个通道：

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

其中 μ_j 表示该通道的均值，σ_j 表示方差，ϵ 是一个极小数，防止方差过小导致梯度爆炸，最终得到近似**均值为 0，方差为 1** 的数据组，完成归一化。**接下来同样重要的是自主学习分布**，即自主学习出一个 $y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$（在训练阶段进行参数训练，在测试阶段即可以将它当成一个简单的线性层），给予网络**恢复表达能力的自由度**，提高灵活性. 还有其他一些归一化，如层归一化，实例归一化等。BN 层通常插在**卷积层和激活函数层之间**。

池化层（实际上 pooling 意指集中/合并）是一种下采样操作，通过缩小数据规模，简化后续计算。其将提取到的特征划分成若干 2×2 的 block，在窗口中保留**最大值（最大池化）**或平均值（平均池化），与卷积层不同，其没有可学习参数。

随机失活（dropout）是为了防止特征过多造成的过拟合/增强面对微扰的鲁棒性而设计的，其核心是在训练过程的每一次迭代中，以 p 的概率随机丢弃一些神经元。需要注意**保留的神经元需要除以 1-p**，以保持一层的期望不变。在测试过程，使用全部的神经元。

最后再将多维特征**展平**，再经过若干全连接层（与一般神经网络中一致），得到分类结果。

CNN 的思想在 1998 年（LeNet-5）就已经引入，在 2012 年（AlexNet）通过扩大规模增加训练完成了在实际任务上的实现，此后人们发现可以通过堆叠小卷积层（3*3*3<7*7）实现更轻量的计算和更好的表达能力（VGGNet，2014）。ResNet（2015）引入了**残差连接结构**，通过直接将输入加到输出上，使网络能够专注于学习输入和输出之间的差异（残差），从而缓解梯度消失，并有效构建更深的神经网络（152 Layers）。

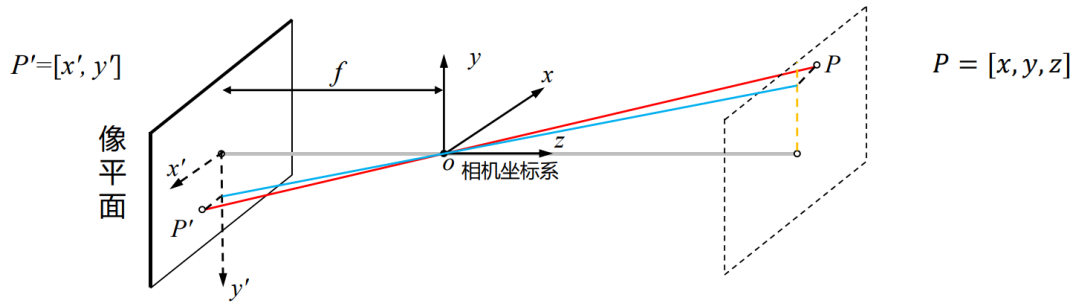
最后，针对数据不足的问题，引入**图像数据增强**的方法，防止过拟合，增强泛化性，如翻转，旋转，裁剪，变形，缩放（几何变换）或噪声、模糊、颜色变换、擦除、填充（颜色变换）等。

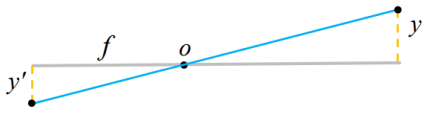
4.2　三维重建

计算机处理图像的另一个难点是，从三维世界到计算机二维图像的转换需要相对复杂的过程，其涉及多个坐标系之间的相互转化：

首先，从**世界坐标系**到**相机坐标系**，公式为 $R\begin{pmatrix}X\\Y\\Z\end{pmatrix}+T=\begin{pmatrix}x\\y\\z\end{pmatrix}$，其中 R 为旋转矩阵，T 为平移矩阵，也可写作 $\begin{pmatrix}R & T\end{pmatrix}\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}=\begin{pmatrix}x\\y\\z\end{pmatrix}$，这个 $\begin{pmatrix}R & T\end{pmatrix}$ 被称为**相机外参**。

其次是从**相机坐标系**到**成像坐标系**，也即小孔成像（如下图），可以写作 $\begin{pmatrix}x'\\y'\\z\end{pmatrix}=\begin{pmatrix}f&0&0&0\\0&f&0&0\\0&0&1&0\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}$，需要注意**这里（为了方便写成矩阵乘法的形式）用的是“齐次坐标”！**即 $(u,v,w)^T$（齐次）$=w(\frac{u}{w},\frac{v}{w},1)^T\sim(\frac{u}{w},\frac{v}{w})^T$（普通形式），事实上可以理解为 $\begin{pmatrix}x'\\y'\end{pmatrix}=\begin{pmatrix}\frac{f}{z}&0&0\\0&\frac{f}{z}&0\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}$。





$\frac{z}{f}=\frac{y}{y'}\Rightarrow y'=f\frac{y}{z}$

三角形相似

最后是从**成像坐标系**转化为**图像坐标系**，即将像挪到视角中央，表示为 $\begin{pmatrix}u\\v\end{pmatrix}=\begin{pmatrix}x'\\y'\end{pmatrix}+\begin{pmatrix}c_x\\c_y\end{pmatrix}$，这里就看出齐次坐标的优越性，因为如此其可以与上一步合并为 $\begin{pmatrix}u'\\v'\\z\end{pmatrix}=\begin{pmatrix}f&0&c_x&0\\0&f&c_y&0\\0&0&1&0\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}$，将

矩阵 $K=\begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix}$ 称作相机内参，于是得到最终成像公式：

$$K\begin{pmatrix} R & T \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}=z\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

考虑三**维重建**过程，就是计算上述摄像机成像的**逆变换**：已知某**同一点**在**两张图**中的位置 $(u_1,v_1),(u_2,v_2)$，且已知相机外参 $\begin{pmatrix} R & T \end{pmatrix}$ 和内参 K，设 $K\begin{pmatrix} R & T \end{pmatrix}=A$ 则列出

$$A_1(X,Y,Z,1)^T=w_1(u_1,v_1,1)^T, A_2(X,Y,Z,1)^T=w_2(u_2,v_2,1)^T$$

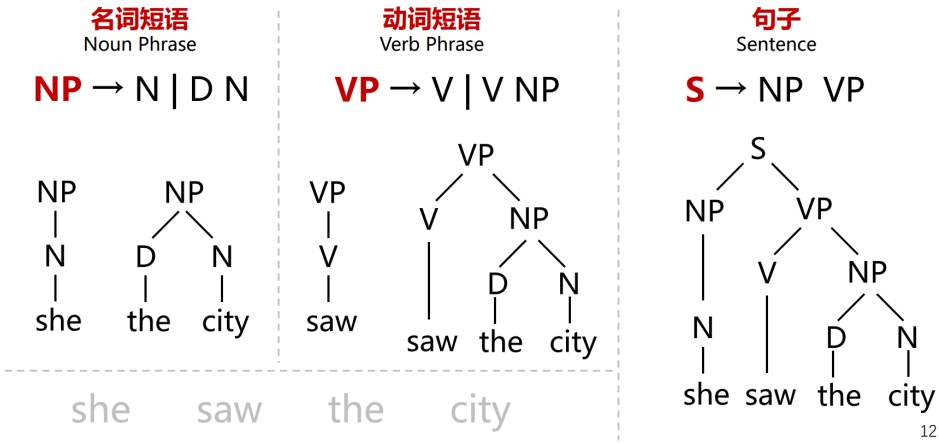
即可反解出原坐标 (X,Y,Z) 以及齐次参数 w_1,w_2（理论上是五个未知数，六个方程，必定有解）

于是问题转化为获得**对应点**，这就需要用到上一节中讲的特征点算子（最经典的就是 SIFT 算子），于是提取特征点，使用算子进行描述，再进行匹配即可。需要指出，如此的简单匹配在实际应用中会遇到遮挡现象、纹理缺乏、光照变化等阻碍，需要开发更完善的算法进行解决，这也是计算机视觉的一个**前沿问题**。

第五章　自然语言处理

5.1　句法分析与分词

自然语言处理是困难的，因为其除了形式语言也具有的**句法**外，还有复杂的**语义**。我们首先考虑更简单的情况——**上下文无关的文法**（CFG），即如下图所示将句子通过一定的文法转化为树结构（其进而可以改写成概率上下文无关的文法）：



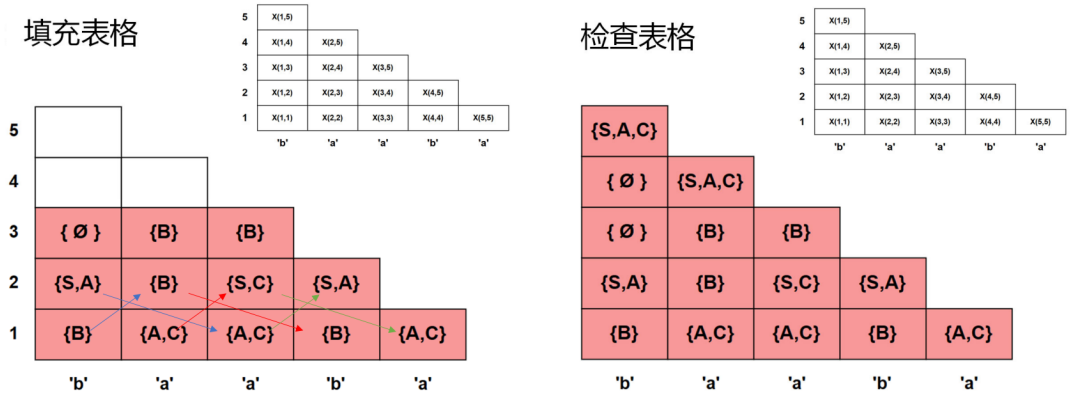
下面对其进行**句法分析**（根据文法规则分析单词以获得其短语结构）一个自然地想法是自顶向下地进行递归拆分，但其效率极低且**易陷入死循环**，因此尝试改进为自底向上从单词逐步归纳回句子结构，特别地，此处介绍 **CYK 算法**：

首先将 CFG 转化成更易求解的**乔姆斯基范式**（CNF），即所有文法都只是 $A \to BC, A \to a, S \to \emptyset$ 三种中的一种，其中大写字母表示**非终止符**，即短语/句子，小写字母表示**终止符**，即（不可分割的）单词等结构。这种转化是容易的，例如 $A \to BCD \sim A \to BX\ X \to CD$.

接下来用 dp 的思想解决此问题：设 $f[i][j]$ 表示可以拆分为长度为 i，从 j 号位置开始的子串的**非终止符集合**（长度为 i 也代表经过了 $i-1$ 次 $A \to BC$ 操作和 i 次 $A \to a$ 操作），则可将较长子串拆分成已经解决的较短字串的组合：

$$f[i][j] = \cup_{k=j+1}^{j+i-1} F(f[k-j][j] \oplus f[i-k+j][k])$$

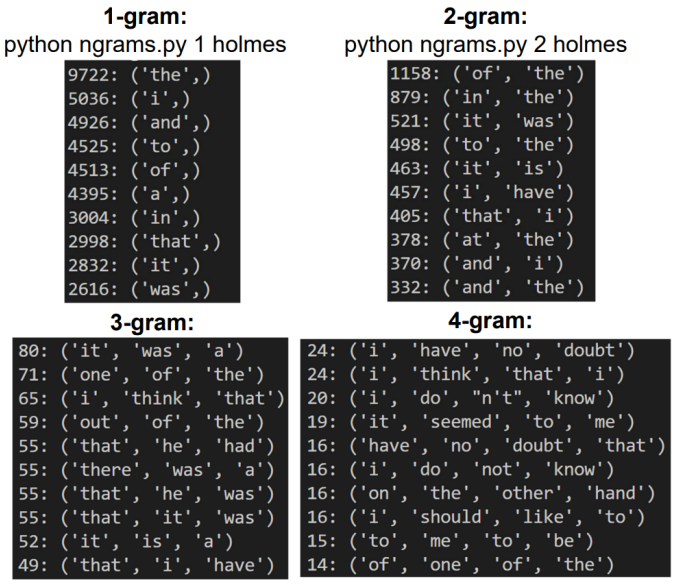
其中 $a \oplus b$ 代表两集合元素分别组合，如 $\{A, C\} \oplus \{B, C\} = \{AB, CB, AC, CC\}$. 而 $F(A)$ 代表对于 A 集合中每一个形如 AB 的组合，可以拆分为该组合的非终止符集合的并集。



（示例：$S \to AB|BC, A \to BA|a, B \to CC|b, C \to AB|a, w = baaba$）如上左图箭头所示逐层填充表格，直至长度为 5，检查 $f[5][0]$ 是否有整个句子 S，若有，则该句子对此文法合法。

在概率上下文无关的文法中，自然也可以使用 CYK 算法，但同时由于概率这一参数的引入，事实上还可以使用**自顶向下**的 A* 算法，将目前为止应用的规则所定义的概率的倒数取作代价函数，其效率高于 CYK 算法（效率为 $O(n^3m)$）。

另一个重要的任务是文本生成，我们首先介绍一种简单的生成策略——基于**马尔可夫模型**的文本生成（其某种意义上是 next-token-prediction 的雏形），即通过前 $n-1$ 个词预测第 n 个词的最有可能的情况（上下文范围相对窄，**手机输入法**的联想输入就是一种典例），这需要构建如下图的 n **元模型**（连着的 n 个词组成词的词频库）：



而进一步地又需要对文本进行**分词**，这在英文是容易的（按空格分词即可），但中文需要一些技巧，常见的有基于字符串的（暴力词典）匹配，或是机器学习的部分基于上下文的匹配（如隐马尔可夫模型（HMM）、条件随机场（CRF）等）。

5.2　统计语言模型和词表示

下面正式进入自然语言处理的任务解决。面临的最大问题显然是如何表示句子。一个最朴素的想法是构建一个包含所有已知单词的字典（例如 10000 维）并统计该句子中每个单词出现的频率形成一个 10000 维的向量，称为**词袋模型**。这显然是一种很劣的策略，因为它未考虑文本中词与词之间的上下文关系，且字典可能极大、文本向量稀疏、关键词的重要性未体现。但将它进行优化应用，可以解决一些简单的任务：

对于文本情感分类（如判断好评等），可以使用**朴素贝叶斯模型**，其将每个词视为独立，并通过组合每个词的“情感倾向”获得整个句子的情感标签。例如，对于“My grandma loves it”，其为好评（G）/差评（B）的概率可以通过下述推导计算：

$$\begin{aligned}\frac{P(G|\text{“my grandma loves it”})}{P(B|\text{“my grandma loves it”})}&=(\text{上下文无关})\ \frac{P(G|\text{“my”, “grandma”, “loves”, “it”})}{P(B|\text{“my”, “grandma”, “loves”, “it”})}\\&=(\text{贝叶斯公式})\ \frac{P(\text{“my”, “grandma”, “loves”, “it”}|G)P(G)}{P(\text{“my”, “grandma”, “loves”, “it”}|B)P(B)}\\&=(\text{独立性假设})\ \frac{P(\text{“my”}|G)\cdots P(\text{“it”}|G)P(G)}{P(\text{“my”}|B)\cdots P(\text{“it”}|B)P(B)}\end{aligned}$$

而上述右式每个参数都可以通过对数据集的分析得到，于是再经过归一化，就得到了该句子的情感标签。（还有一个小问题，独立性假设推出的乘法公式可能使某个没出现的词将概率清零，于是需要进行**加法平滑**：$P(t|G)=\frac{|D(t)|+\alpha}{|D(S)|+\alpha V}$，其中 V 为词汇类别数，$\alpha=1$ 时称为**拉普拉斯平滑**）

对于信息检索（根据输入找到相关文档/找到给定文档的主题和关键词），可以通过 **tf-idf**（词频-逆文档频率）策略：对于某一文档，去除（a,the 等）**功能词**后，得到实义词列表，此时词频 $tf=\log_{10}(n+1)$ 衡量词语在文档中的重要程度，而逆文档频率 $idf=\log_{10}\frac{D}{d+1}$（其中 D 为总文档数，d 为含有该词的文档数）衡量在语料库中的罕见程度，于是 $tf\times idf$ 较高的就是该文档的**关键词**候选项。

为解决词袋模型文本向量稀疏的问题，提出**词表示**方法，即一个词用一个稠密向量表示：若用**独热表示**（e_i），则向量仍旧稀疏且没有考虑上下文；因此考虑**分布式表示**，其中心思想是词的语义由上下文决定，其分为基于矩阵的分布表示（每个词由其上下文词语的“**共现频率**”表示，如 dog 常与 bark 共现）；基于聚类的分布表示（统计共现后分若干类别赋予类别标签）；**基于神经网络的分布表示**，其经典代表就是 **word2vec**。

word2vec 分为 **CBOW 模型**（根据上下文推断中心词）和 **skip-gram** 模型（根据中心词推断上下文），下面介绍 CBOW 模型过程，而 skip-gram 模型只需将过程反向即可：

将当前词若干上下文词语的独热编码 $(1\times V)$ 输入到输入层；将这些词向量乘以同一个**周围词向量矩阵** $W(V\times N)$ 后取平均得到上下文表示向量$(1\times N)$，将此向量乘**中心词向量矩阵** $W'(N\times V)$，得到预测标签，softmax 分类后（生成概率分布）与中心词真实标签作交叉熵损失，每个词训练完反向传播更新 W,W'，最终得到基于上下文的词表示（即著名的 King-man=Queen-woman）。这种标签来自于输入数据本身而非人工标注的训练方式称为**自监督学习**。

5.3　Transformer

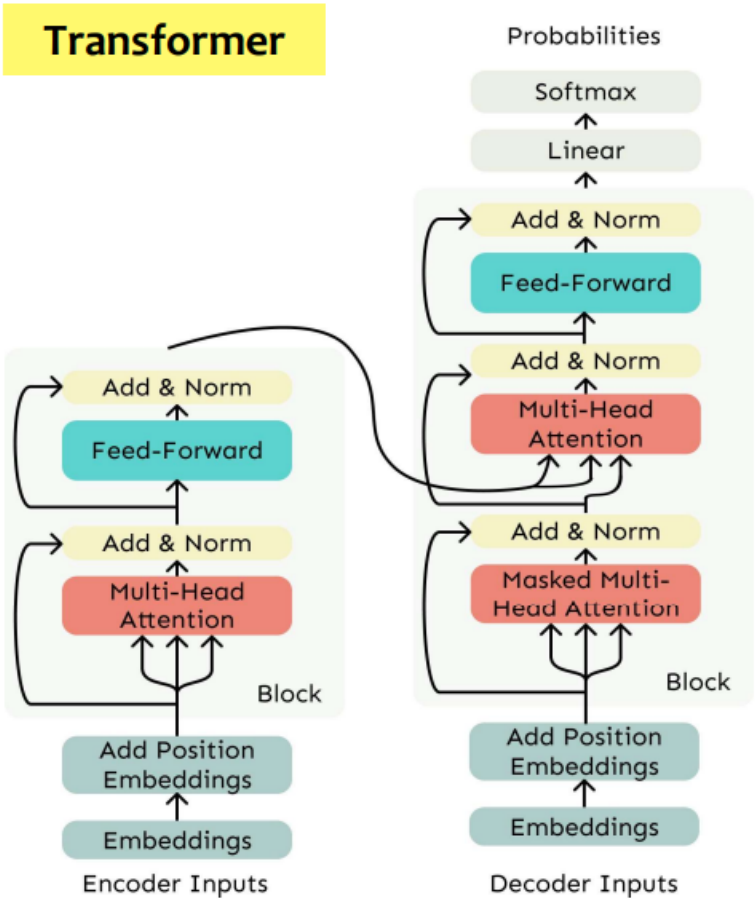
上一节中多为基于统计的简单模型，能力有限，本节介绍基于**神经网络**的语言模型，在学习中完成语言处理任务。

对于文本情感分类任务，可以通过人工提取特征（如正面词个数，“no”的个数等）进行训练，也可通过**池化表示**（即提取每个词的表示并进行池化操作）进行训练进而给出情感分类。

对于文本生成任务，传统的前馈语言模型通常基于固定大小的上下文窗口来预测下一个词，但如此完全抛弃了距离较远词语的信息，于是给出无限窗口大小的优化：**RNN**（循环神经网络），其核心公式是 $h_t = f_W(h_{t-1}, x_t)$，即不断用当前状态更新 h_t（**隐状态**），最终输出即为 $y = f_{W'}(h_t)$，由此，其可处理长上下文信息，但其劣势在于**无法并行计算**，且由于**梯度消失，远距离上文信息大量丢失**。可以对其进行改进以一定程度解决劣势：

LSTM（长短期记忆网络，不考）通过构建**遗忘门**和**输入门**（sigmoid 函数表征遗忘/写入程度）$f_t, i_t = \sigma(W(h_{t-1}, x_t) + b)$，以及候选记忆值（tanh 函数使数据**压缩**且因拥有**负向**表达能力更强）$C'_t = \tanh(W(h_{t-1}, x_t) + b)$，选择性地选取上文记忆更新 $C_t = f_t \odot C_{t-1} + i_t \odot C'_t$，最终再通过**输出门** $o_t = \sigma(W(h_{t-1}, x_t) + b)$ 得到输出（同时也是下一层参数）$h_t = o_t \odot \tanh(C_t)$，从而实现上文信息保留。

而另一种更有效的优化是 **Transformer** 模型，先给出其架构图：



下面将分别对其核心机制进行阐述：

其整体上使用 **Encoder-Decoder 架构**（编码器-解码器），需要指出，这是因为其初始目标任务是文本翻译（类似 seq2seq 处理），而在当前的大模型文本生成任务中，用的是 Decoder-only 架构。Encoder 对源语言**输入**进行编码，生成语义表示；而 Decoder 则基于该表示以及**先前生成**的目标词序列，逐步预测下一个目标词。

Encoder 和 Decoder 都由多个几乎相同的层（block）构成，简单的层由 **Attention 层**和**前馈层**构成，前馈层本质就是一个**两层的感知机**，是为了进行非线性处理而添加的，下面重点介绍 Attention 层：

注意力机制是 Transformer 架构的最核心机制，其通过 **Q-K-V** 矩阵对词表示进行处理，得到注意力词表示 q_i, k_i, v_i。q_i 代表提问者，表示当前词语需要分配自己的注意力以**寻找和自己相关的内容**，而 k_i 则扮演回应者，表示其他词语需要**放多少注意力**在自己身上（对外界查询的响应特征），由此 $q_i \cdot k_j$（内积）的相对大小即表示第 i 个词需要分配多少注意力在第 j 个词上（例如，设有代词 A 与其代指的名词 B，则 $q_A \cdot k_B$ 应当较大，而 $q_B \cdot k_A$ 则不见得较大）。而 v_i 本质还是原来的词表示，只是为了配合 Attention 层的维数改变而进行的转换。

例如，对于输入的 2×4 词表示矩阵 X（2 个词，4 维特征），则可训练出 4×3 的权重矩阵 W^Q, W^K, W^V，作矩阵乘法即可得到 $Q, K, V \in M_{2\times 3}$ 作为新的表示矩阵。接下来，计算 $QK^T \in M_{2\times 2}$, 对照注意力机制可得 $(QK^T)_{i,j}$ 即为第 i 个词需要分配多少注意力在第 j 个词上。

接下来需要对 QK^T 的每一行（表示同一个词分配的注意力）进行 **softmax 归一化**。但为了**防止点积过大**导致 softmax 饱和，需要先对于数据统一除以 $\sqrt{d_k}$, 其中 d_k 是 q_i, k_i, v_i 的维数，这个数字的来源是：假设 q_i, k_j 均值为 0，方差为 1，则 $q_i \cdot k_j$ 均值为 0，方差为 d_k，除以 $\sqrt{d_k}$ 可以使数据标准差变为 1.

归一化后得到 2×2 “注意力分布” 矩阵 T，下面将 V 的行向量对分布系数作线性组合（$z_1 = T_{11}v_1 + T_{12}v_2$），也即 $Z = TV$, 综上，得到核心公式：

$$Z = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

上述机制彻底解决了 RNN 的两个弊端——不可并行化和长距离上文限制。可以进一步优化注意力层，改为**多头注意力**（顾名思义，并行地放置多个注意力层），这可以使模型从不同子空间中获得多层次的词间关系（如主谓/指代等），只需要将上述得到的 $Z_i \in M_{2\times 3}$ 拼接成大矩阵（以 8 头注意力为例）$Z \in M_{2\times 24}$, 再训练一个线性映射 $W^O \in M_{24\times 4}$, 即可得到最终表示矩阵 $Z_0 \in M_{2\times 4}$.

为了仍旧给予词表示一部分位置信息，使模型能够感知词序与相对位置（例如，临近词之间往往具有更强语义关联），对输入的词表示进一步进行**位置编码**，即对原嵌入额外加上一个位置相关的矩阵，原论文中采取了固定的位置编码矩阵 $PE(pos, 2i) = \sin \frac{pos}{10000^{2i/d_{model}}}, PE(pos, 2i+1) = \cos \frac{pos}{10000^{2i/d_{model}}}$，也可以将其作为一个参数进行训练。

为了防止梯度消失，保留原始信息，在 Attention 层和前馈层中都引入了**残差连接**结构，其公式是 output=LayerNorm$(x + f(x))$, 即将输入直接引入到最终的输出中（因为这个因素，Attention

层和前馈层**最终**的输出维度应与输入维度一致）

上述完整地表述了 Encoder 部分的结构，Decoder 部分结构大体相同，仅有几处微小之处不同：

首先，（虽然训练时已经给出了完整的输出，）但为了与测试保持一致，模型还是应当屏蔽未来词，从而构建 **Masked Self-Attention** 层（掩码自注意力），通过在 softmax 之前将 $QK^T_{i,j}(i<j)$ 全部赋为 $-\infty$, 则通过 softmax 后即全为 0，实现屏蔽未来词的效果（这个机制也解决了 Decoder 训练并行化的问题）。

其次，其注意力不仅需要分配到上文输出中，还要放到**输入**中，因此在 Masked Self-Attention 层之后，需要额外的 **Cross-Attention** 层，其 Q 矩阵仍旧通过前面掩码层表示构建 $Q=X_{dec}W^Q$, 但 **K,V 矩阵**通过 Encoder 层的最终输出表示构建 $K=X_{enc}W^K, V=X_{enc}W^V$，其余正常构建即可。

最后，在 Decoder 最后一层输出后，加入一个全连接层，将输出表示转化为一个维数等于单词数的向量，每个位置表示该单词的可能倾向值，再通过 softmax 转化为概率分布，计算交叉熵损失，进行反向传播，训练所有参数（包括词嵌入矩阵、各层 W^Q, W^K, W^V、多头注意力投影矩阵 W^O、前馈层参数、最后线性层参数等），即完成 Transformer 构建。

第六章　知识图谱

知识图谱（KG）是一种旨在通过**图结构**刻画事物间关系、沉淀领域知识的大规模语义网络。在图结构中，节点代表**实体**，即事物/概念，而边代表实体之间的**关系**，常用三元组表示，例如⟨我，喜欢，猫⟩。其曾在 60 年代的专家系统中被大量应用，也随着专家系统的式微而沉寂，进入 2010 年代，随着互联网大数据的发展，Google 重新提出了知识图谱，借助海量数据，构建出更高质量的图谱系统，推动了其广泛应用。

知识图谱的构建与应用主要包括三个步骤：知识抽取、知识表示和知识推理。

知识抽取显然包括实体抽取和关系抽取，它们都可以通过人工设计模板进行匹配，还可以通过机器学习进行抽取：**实体抽取**中，可以学习出如 B-per（人名实体开头），I-per（人名实体内部），O（非实体）等标签；**关系抽取**中，可以基于抽取好的实体词构建多种学习方法，其中**特征函数 + 最大熵模型**是重要的方法：我们希望最大化条件熵（**以保持在已有知识下对未知部分最少假设**）：

$$H(P) = -\sum_{x} \tilde{P}(x) \sum_{y} P(y \mid x) \log P(y \mid x)$$

其中，$\tilde{P}(x)$ 为经验分布，可通过数学推导（约束模型下的期望与经验分布下的期望一致，进而运用拉格朗日乘子法），得到最大熵的条件概率分布具有如下形式：

$$P(y \mid x) = \frac{1}{Z(x)} \exp\left(\sum_{i} \lambda_i f_i(x, y)\right)$$

对参数 λ_i 进行训练即可。特别地还有**事件抽取**（结构化地呈现事件的时间地点人物等信息），其是通过事件触发（如遇到关键词“成立”触发“设立单位”事件类型）和事件要素提取两步构建的。

知识表示通过构建人可理解、计算机易处理的语言描述知识，最经典的方法是**逻辑表示法**，即运用此前学的**一阶谓词逻辑**（用否定 $\neg$, 析取 $\vee$, 合取 $\wedge$, 蕴含 $\rightarrow$, 等价 $\longleftrightarrow$ 等**联结词**将（无法分解的）**原子命题**组合为**复合命题**）表示人类的自然语言。

一阶谓词逻辑的基本表示方法是：以实体（在此也称为**个体词**）为单位，通过**函数**构建新的实体（例如：Principal(PKU) 构建了实体“北京大学校长”），通过**谓词**描述其性质或多个体之间的关系（例如：North(PKU,China) 表示北大在中国北方）（**注意，想要句子可判断一定要有谓词**），通过**量词**（$\forall, \exists$）对个体数量进行约束，例如：“北京大学五月举办校庆”可以这么表示（**注意，不是“仅五月”**）：

$$(\forall \text{Month})(\text{May}(\text{Month}) \rightarrow \text{Hold}(\text{PKU},\text{Anniversary}))$$

常见的表示法还有**产生式表示法**（类似数学推理：给定公理（规则库），前提（事实库），通过推理链得到结论），**框架表示法**（构建不同的“类”，建立类之间的联系）。

但上述三种都依赖于人工构建，在当前数据爆炸的时代，倾向于使用**向量化表示**，拓宽表达空间。其主要思想是：对于三元组 $\langle A, B, C\rangle$, 建立知识表示使得 $A+B=C$, 这就是最朴素的 **TransE** 算法，此后又发展出多种优化（不考）：**TransH** 对每个关系构建超平面，将实体投影到超平面上，从而更好地处理多对多的关系（$A_1^{\perp}+B=A_2^{\perp}+B$）;**TransR** 彻底将实体与关系分离，将实体空间向量映射至关系空间进行处理；**TransD** 进一步将映射矩阵改为向量表示组合出的低秩矩阵 $v_B^T v_A$ 降低计算量；**TransA** 通过将“距离函数”设为可训练参数提高灵活性；**TransG** 构建了关系词的一词多义；**KG2E** 将词从点表示转向高维球计算重叠程度……这是一个不断在发展优化的课题（比如，以上方法全都无法解决“我和你是朋友”这样的对称表示）。

最后，**知识推理**基于上述构建的特征信息，辅助推理出新的事实/关系等。主要形式是依赖逻辑表示法的**逻辑表示推理**。基本推理规则为此前提到的德摩根定律、分配律，以及 $P\to Q \iff \neg P\vee Q$,$P\leftrightarrow Q \iff (P\wedge Q)\vee(\neg P\wedge\neg Q)$。演绎方法包括**自然演绎推理**（类似产生式表示，通过为真的规则尝试能否推出结论）和**归结演绎推理**：其核心是反证法，**将结论取反**，和条件一起组成子句集，使用 2.3 节提到的 DPLL 等算法尝试能否推出矛盾，**能推出矛盾则原命题为真**。

最后介绍使用 GNN（**图神经网络**，本身是解决图数据（如社交网络、城市网络等）的工具）在知识图谱上学习的方法（不考）。

在一张图上学习最平凡的方法是直接将**邻接矩阵** $A(A_{ij}=\delta_{ij}$表示是否有边) 与节点的特征矩阵 X 拼接在一起，输入进一个 MLP，但如此参数量为 $|V|$ 规模，同时对节点顺序敏感（无法将仅更换标注节点顺序的图识别为同一个）。作为改进，图神经网络通过**聚合邻居节点**（这就实现了顺序无关）的特征学习中心节点表示：例如，单层 GNN 表示可以简单地被设计为 AX（如此参数规模减小为 d），也可以通过增加**可学习参数**，设计更为完善的聚合方式。

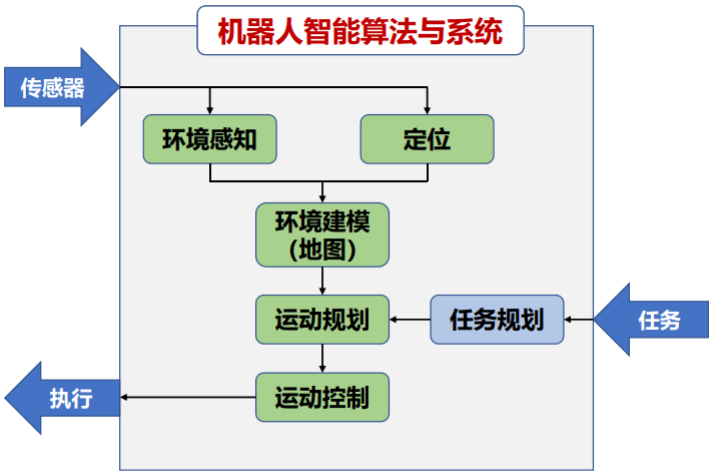
进一步地，为了产生更大的感受野，可以构建多层 GNN。每一层中，每个节点产生一个**消息** $m_u^{(l)}=\mathrm{MSG}^{(l)}(h_u^{(l-1)})$(MSG 为消息函数，如线性层)，并传递给邻居节点. 节点聚合邻居节点传递来的消息，**并与自身的消息** $m_v^{(l)}=B^{(l)}h_v^{(l-1)}$ 聚合：

$$h_v^{(l)}=\mathrm{concat}(\mathrm{AGG}(\{m_u^{(l)},u\in N(v)\}),m_v^{(l)})$$

其中 AGG 表示某种聚合方式，可以是求和/平均/最大值/Attention 等，经由多层消息传递后，经过一个全局 pooling 层学到全图表示，即完成多层 GNN 构建。

第七章　智能机器人

机器人是通过**传感器**（如测距仪等）感知环境，通过**效应器**在真实世界中作出反馈，完成任务的实体智能体，其中涉及的智能算法可以由下图概括：



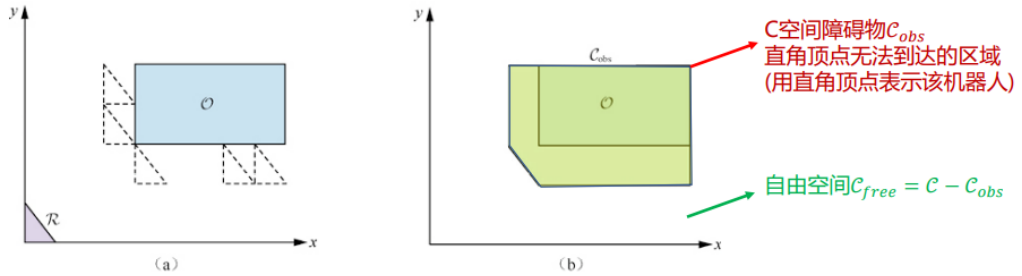
首先，机器人通过传感器感知到的数据，进行**定位与地图创建**，定位的方法很多，最经典的是基于**粒子滤波器**的蒙特卡洛定位。粒子滤波器的思想是通过一组加权后的随机样本计算随机事件（此处是物体位置）的后验概率分布，从而估计状态（实现定位），其在定位中的应用可用以下公式表示：

$$P(X_{t+1}|z_{1:t+1},a_{1:t}) = \alpha P(z_{t+1}|X_{t+1}) \int P(X_{t+1}|X_t,a_t)P(X_t|z_{1:t},a_{1:t-1})dX_t$$

其中 X_t 表示 t 时刻的位置，z_t 表示该时刻进行的**观测**，a_t 表示采取的动作，于是 $P(X_t|z_{1:t},a_{1:t-1})$ 表示 t 时刻的**“信念”**，也即预测定位概率分布，(a) 将其输入进**运动模型** $P(X_{t+1}|X_t,a_t)$（受噪声影响，采取某动作后的位置并非确定，而是满足高斯分布），得到**先验概率分布**；对 X_t 积分（全概率公式）后 (b) 再根据 $t+1$ 的观测 z_{t+1} 构建**传感器模型** $P(z_{t+1}|X_{t+1})$ 对定位进行加权，进而归一化（也就是公式中的 α） (c) 得到后验概率分布 $P(X_{t+1}|z_{1:t+1},a_{1:t})$，也即新一轮的“信念”状态。

实际的蒙特卡洛定位算法就是按照此思想实现的：通过在地图上均匀撒粒子初始化，将其作为首轮样本，此后重复执行 [（a）**预测**，将每个粒子输入进运动模型并加上随机噪声表示下一步位置；（b）**更新**，通过传感器模型更新粒子权重（归一化）；（c）**重采样**，根据权重从当前例子中**有放回地重新采样**一组粒子作为下一层输入]，最终粒子位置收敛，即得到预测定位，同时还可以通过观测完成地图绘制。

有了环境数据后，就可以进行规划，分为任务规划和运动规划（以拧魔方为例，任务规划是采取什么样的抽象步骤，运动规划是怎么将魔方拧一步），任务规划可以用其他领域的算法解决，此处重点讨论**运动规划**——在**不与障碍物碰撞**的前提下，找出使机器人在构形之间转变的规划（在此处简化问题，仅考虑几何路径）。从真实世界中的**工作空间**出发，我们将机器人简化为其上的一个点，则障碍物**“膨胀”**为所有该点不可达的区域（参考下图），形成**构形空间**，在构形空间中机器人点可达的区域称为**自由空间**。



一个可以平移的简单机器人$\mathcal{R}$，它需要躲避矩形障碍物$\mathcal{O}$。（a）工作空间，（b）构形空间或C空间

于是，问题被转化为找到贯穿自由空间的连续路径。为解决这一问题，有如下方法：**单元分解**（将空间进行网格化划分，在网格间跑最短路，相对依赖网格设计）；**能见度图**（起点、终点、障碍物顶点间若可达则建边，跑最短路，是最优解但太靠近障碍不够安全）；**沃洛诺伊图**（取所有障碍物顶点连线**中垂线**相交构成骨架，在上面跑最短路，安全但复杂度高且不一定优）；**随机运动规划**（随机采样点建图跑最短路）等。

上述方法都依赖提前建图，对**动态环境**适应性较差。下介绍适合实时路径规划的**快速探索随机树**（RRT）算法。其步骤如下：重复 [随机采样一个点 X_{rand}，找到 RRT 树上离 X_{rand} 最近的点 X_{near}，从 X_{near} 出发尝试向 X_{rand} 走一定距离至 X_{new}，若途中未遇到障碍，则将 X_{new} 加入 RRT 树] 直到离终点足够近。如此便实现了向终点路径的实时规划。针对其不适应大量障碍/狭窄通道的问题，可以进行双向搜索以优化算法。

运动规划做完后，需要将电流传送给电动机来产生合适的扭矩，真正使得机器人动起来。在理想环境中，我们可以构建**动力学模型** $x'' = f(x, x', u)$ 表示在 x 构形，速度为 x'，施加扭矩 u 后会产生加速度 x''，进而通过 $u(t) = f^{-1}(x(t), x'(t), x''(t))$ 控制扭矩即可。但现实世界中存在误差，累积后会变得不可接受，于是引入**运动控制**过程，控制误差的累积。

自然的想法是施加一个正比于（目标状态 $\xi(t)$ 和实际状态 q_t）观测误差的扭矩，即 **P 控制器**：

$$u(t) = K_P(\xi(t) - q_t)$$

但这样可能无法解决系统性的长期误差（例如，误差来源是一个稳定将无人机向下吹 $1m/s$ 的风，如果 P 控制器计算得调整扭矩亦为 $1m/s$，无人机将永远无法移动）于是，引入**积分控制器**，添加与过去所有误差累积成正比的项，形成 PI 控制器：

$$u(t) = K_P(\xi(t) - q_t) + K_I \int_0^t (\xi(s) - q_s)ds$$

如此基本可以保证机器人达到目标位置，但震荡过于严重使控制不够精细（例如，目标是悬停在天花板下 $0.1m$, 那么很可能冲上天花板），于是引入**微分控制器**，添加与误差增长率负相关的项（若误差正在减小，则抑制扭矩），形成 **PID 控制器**：

$$u(t) = K_P(\xi(t) - q_t) + K_I \int_0^t (\xi(s) - q_s)ds + K_D(\xi'(t) - q_t)$$

如此得到了针对误差的反馈分量（需要乘上质量得到驱动力），再与前述逆动力学前馈分量合并，即得到了最终的扭矩控制公式：

$$u(t) = f^{-1}(\xi(t), \xi'(t), \xi''(t)) + m(\xi(t))(K_P(\xi(t) - q_t) + K_I \int_0^t (\xi(s) - q_s)ds + K_D(\xi'(t) - q_t))$$

第八章　强化学习

强化学习（**Reinforcement Learning**）是一种在与环境交互中学习的方法，其主要解决马尔可夫决策问题（未来状态仅有由当前状态、动作决定），目标是寻找在当前**状态** $s \in S$, 应该选择何种**动作** $a \in A$, 以在给定**转移模型** $s' \sim p(s'|s,a)$ 和**奖励函数** $r(s,a,s')$ 的情况下，获得最大回报。最终选择的行动被称为**策略** π, 其可以是确定的 $a = \pi(s)$, 也可以是随机的 $a \sim \pi(a|s)$.

我们的目标是找出最优策略，但在这之前，先要明确如何判断策略的优越与否。定义**轨迹** $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \cdots)$ 表示采取某策略后状态、行动、奖励的序列，进而可以定义**累积收益**:

$$G(\tau) = \sum_{i=0}^{T} \gamma^i r_i$$

其中**折扣因子** γ 表示看重当下/未来收益程度（$\gamma = 1$ 表示将来每步收益均一样重要；$\gamma = 0$ 表示只看下一步收益）

于是可以定义从状态 s 出发，采取策略 π 能获得的收益，即**状态价值** $V_\pi(s)$:

$$V_\pi(s) = E_{\tau\sim\pi}(G(\tau)|s_0 = s)$$

以及从状态 s 出发，**做动作** a, 再采取策略 π 能获得的收益，即**动作价值** $Q_\pi(s,a)$:

$$Q_\pi(s,a) = E_{\tau\sim\pi}(G(\tau)|s_0 = s, a_0 = a)$$

为计算上述式子，引入基于动态规划思想的 **Bellman** 方程, 其推导如下:

$$\begin{aligned} Q_\pi(s,a) &= E_{\tau\sim\pi}(G(\tau)|s_0 = s, a_0 = a) = E_{\tau\sim\pi}(r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots | s_0 = s, a_0 = a) \\ &= E_{(r_0,s_1)\sim p(\cdot|s,a)}(r_0 + \gamma V_\pi(s_1)) = \sum_{s',r} p(s', r|s, a)[r + \gamma V_\pi(s')] \\ \Rightarrow V_\pi(s) &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V_\pi(s')] \end{aligned}$$

以上公式称为 **Bellman 期望方程**，进而从中选择最大价值的策略，即得到 **Bellman 最优方程**的解 $V^*(s), Q^*(s,a)$.

利用上述公式，可以设计计算最优策略的方法，主要分为两种:

策略迭代的核心是不断寻求更优的策略，其分为两步，**策略估值**通过迭代逼近当前策略的状态价值 $V_\pi(s)$:

$$\text{Do}\{\ V_{\pi,k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V_{\pi,k}(s')]\ \}\text{until} \max_s |V_{\pi,k+1}(s) - V_{\pi,k}(s)| < \epsilon$$

策略提升利用计算出的价值函数构造一个更优策略：$\pi'(s) = \text{argmax}_a Q_\pi(s,a)$. 可以证明，当 $\pi(s) = \pi'(s)$ 时，即找到了最优策略。如此，每轮都可得到一个完整策略，但每轮评估可能要迭代很多次。

值迭代不显式维护策略，而是直接利用 **Bellman 最优方程**迭代**价值函数**：

$$V_{k+1}(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V_k(s')]$$

仅在收敛后在最后确定一次策略（如此，其效率较高但无法维护中间策略）：

$$\pi^*(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V^*(s')]$$

但是在实际情况中，通常没有显式的 $p(s',r|s,a)$ 表达式, 或是状态极多无法有效遍历，此时需要利用类似蒙特卡洛的**采样**方式解决，这就是 **Q-Learning**，其步骤如下：

每次采样跑一条从起点到终点的路径，在路径选取中，每次**对下一步动作**的选择利用**行为策略**（包含探索，例如 ϵ-greedy），然后将本次采样的“收益”定为下一步奖励加上**预期后续最大奖励**（此后的步骤选取利用**目标策略**，即直接贪心）并据此以一定的学习率 α 更新 $Q(s,a)$：

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(r_t + \gamma \max_{a'_{t+1}} Q(s_{t+1},a'_{t+1}) - Q(s_t,a_t))$$

其中 $r_t + \gamma \max_{a'} Q(s_{t+1},a')$ 被称为 **TD-target**（时序差分目标），是对“真实期望值”的**采样近似**，依据此不断采样即可。

当状态空间 S 特别大，无法以查表方式构建 $Q(s,a)$, 于是可以利用深度学习建立近似函数，也即**深度强化学习**。但此方法极其黑盒，难以稳定训练，因此成为当前亟需解决的前沿课题。

例如，对于此前的 Q-Learning，可将其优化为 **Deep Q-Network**（DQN），其学习一个以 θ 为参数的 $Q_\theta(s,a)$. 在此基础上，DQN 作者对其作了两点至关重要的优化，可以从下面的优化目标式子中看出：

$$\theta^* \leftarrow \arg\min_\theta \frac{1}{2} \sum_{(s_t,a_t)\in D} \left[Q_\theta(s_t,a_t) - (r + \gamma \max_{a'} Q^{\text{target}}(s_{t+1},a')) \right]^2$$

第一点是建立 **replay buffer**（D），储存一定数量的 (s_t,a_t), 每次随机取一批用于训练，从而打破数据时序上的相关性；第二点更为重要，即储存冻结的目标网络 Q^{target}，每 N 步才将 Q_θ 更新进 Q^{target} 内，防止 TD-target 随参数更新而更新（否则学习将变得混乱），提高数值稳定性。

第九章　多智能体

研究多智能体问题最关键的基础就是**博弈论**（每个参与者都是理性的）。从各个维度，博弈可分为合作/非合作，静态/动态（行动有先后且后者能看到前者行动），完全信息/不完全信息，零和/一般和，例如著名的囚徒困境就是**完全信息静态非合作博弈**，双人博弈的结果可写成**收益矩阵**。

在博弈中，有一些重要的平衡点：对于某参与者的**占优策略**表示无论其他参与者策略如何，这都是最优策略之一；所有参与者都达成了占优策略的组合成为**占优策略均衡**；

若对于某策略组合，**不存在**另一个使每个人都**不变差**的策略组合（即不存在**帕累托改善**），则称该组合为**帕累托最优**；

若对于某策略组合，任何人都无法仅通过改变自己的策略以获得更高的收益，则称为**纳什均衡点**（可能有多个，也可能没有），但在允许使用混合策略（随机性的策略，相对于纯策略）的情况下，**存在至少一个纳什均衡**。

计算纯策略纳什均衡可以使用穷举法，在策略组合较多时可以采用**短视最优响应**——若某参与者可以单方面改善，则改善（可能不收敛）。而对于双人零和博弈，可利用**极小极大理论**计算：

$$V_{\text{maximin}} = \max_{p_1} \min_{p_2} p_1^T M p_2 = \min_{p_2} \max_{p_1} p_1^T M p_2 = V_{\text{minimax}}$$

其中 p_1, p_2 分别表示采取的混合策略，而 M 表示收益矩阵，选择其中一个求解即可。

对于合作博弈，最重要的概念是**联盟**，即对于参与者的集合划分。参与者需要决定加入哪个联盟，并分配联盟整体获得的价值。一个理想的分配方案 $\{\phi_1(G), \cdots, \phi_n(G)\}$ 应满足**公平性公理**：分配所有收益；无贡献者无收益；**对称参与者**（加入任何联盟二者贡献都相同）收益相同；分配方案（对于博弈问题）线性可加。可以证明，满足公平性公理的解是唯一的，且为**沙普利值**：

$$\phi_i(G) = \sum_{C \subseteq N | \{i\}} \frac{|C|!(n - |C| - 1)!}{n!} \text{mc}_i(C)$$

其中，$\text{mc}_i(C) = v(C \cup \{i\}) - v(c)$, 即 i 加入联盟 C 带来的**边际效应**，该公式通过列举 i 加入联盟时，联盟的成员，计算出**所有可能加入联盟顺序**的边际贡献的平均值。

应用强化学习的思想，就得到了**多智能体强化学习**（不考）：同样可以定义对某智能体的价值函数：

$$V^i_{\pi^i, \pi^{-i}}(s) = E_{\tau_{s_0 = s} \sim \pi^i, \pi^{-i}} \left[\sum_t \gamma^t r^i_t \right]$$

以及改进后的多智能体 Minimax-Q-Learning：

$$Q_i(s_t, a^i_t, a^{-i}_t) \mathrel{+}= \alpha(r_t + \gamma V_i(s_{t+1}) - Q_i(s_t, a^i_t, a^{-i}_t)), \quad V_i(s_{t+1}) = \max_{\pi_i} \min_{\pi^{-i}} E_{a^i_1, a^{-i}_1 \sim \pi^i, \pi^{-i}} \left[Q_i(s_{t+1}, a^i_1, a^{-i}_1) \right]$$

第十章　仿真

仿真主要分为三个环节，外观的仿真、现象的仿真和行动的仿真。

外观的仿真也就是让物体看起来像,分为构形和着色。表示一个形状分为**隐式表示**($x^2+y^2=r^2$，容易判断点与形状的关系，不易列举点）和**显式表示**（$\begin{cases} x=r\cos\theta \\ y=r\sin\theta \end{cases}$，容易列举点，不易判断点与形状的关系)。隐式表示除了通过代数方程外，还可以使用**有符号距离函数**——计算每个点到表面的距离——表示更复杂的形状。显式表示方法有点云表示（采样大量点)、体素表示（用离散化网格是否包含表面表示)、多边形网格（用三角形/四边形拟合表面形状)。配合三维视觉即可完成重建。

着色环节分为两种思路：**光栅化绘制**（将形状投影到平面，网格化着色，较近的物体覆盖较远的物体）和**光线追踪**，即从点阵发出若干射线（分为**正交投影**，即平行发出射线，保持物体真实尺寸；以及**透视投影**，即均从视点发出，放射发布，更符合近大远小的视觉效果)，再根据射线是否与物体相交选择着物体/背景颜色（此处也可以对于射线上每个点通过神经网络计算颜色及透明度，进而得到像素颜色)。

着色过程中应考虑**光照**。在实际应用中，一般遵循简单的**朗伯模型**，即假设满足理想漫反射(粗糙的表面均匀地将光反射到各个方向）以及朗伯余弦定律（表面接收到光强与入射角相关)，最终形成的经验公式为:

$$I=k_aI_a+k_dI_l\cos\theta$$

其中 I_a,I_l 分别为环境光、入射光光强，k_a,k_d 表示环境光反射系数、漫反射系数。

最后，在有反射、折射的情况下，须考虑所有从光源到视点的光路，将每一条光路得到的颜色叠加形成最终的颜色。

现象的仿真即模拟真实的物理现象，由于计算机无法模拟连续的时空，因此须将时空**离散化**。时间的离散化是容易的，空间的离散化可以分为两种视角：**欧拉视角**（对空间离散化，常用于描述连续介质）和**拉格朗日视角**（对物体本身离散化，常用于描述粒子运动)。计算物理仿真的方式主

要是依据运动方程 $Ma = f(x, v, t)$ 使用数值积分求解的，数值积分分为以下三种：

显式欧拉积分: $\begin{cases} v_{n+1} = v_n + M_n^{-1} f(x_n, v_n, t_n)\Delta t \\ x_{n+1} = x_n + v_n \Delta t \end{cases}$ （纯粹的马尔可夫过程）

隐式欧拉积分: $\begin{cases} v_{n+1} = v_n + M_{n+1}^{-1} f(x_{n+1}, v_{n+1}, t_{n+1})\Delta t \\ x_{n+1} = x_n + v_{n+1} \Delta t \end{cases}$ （解一个向量的二元一次方程组，消元即可）

半隐式欧拉积分: $\begin{cases} v_{n+1} = v_n + M_{n+1}^{-1} f(x_n, v_n, t_n)\Delta t \\ x_{n+1} = x_n + v_{n+1} \Delta t \end{cases}$ （先求 a，再求 v，再代入求 x）

其中隐式积分数值稳定性较高，结果不易发散（通常用于复杂系统），但计算成本较高；而半隐式/显式计算成本较低，但**通常需要较小的** Δt 才能保证数值稳定。

动作的仿真，即动画，通常通过动作捕捉关键**关节**在**关键帧**的动作实现（再经过一次**蒙皮**展示角色外形）。

我们希望仿真环境下训练的智能体能直接应用于真实环境，但仿真与真实环境间往往存在**虚实鸿沟**（建模的系统误差，或真实传感器的获取局限），于是就需要更准确地建模；建立仿真数据与真实数据的映射实现**域自适应**；或是加入模拟虚实鸿沟的噪音以进行应对。