模拟卷 A：中文理解讲义

这份讲义怎么用

题号对应之前的英文模拟卷 A1-A10、B1-B5、C1-C3、D1-D2、E1，共 21 道大题。每页先解释“它到底在问什么”，再给直觉、公式和代入过程。可以把英文卷放旁边，按题号查阅。

先理解基础题和计算题，再读深度 RL 简答。推荐顺序：A1-A6 → C1、C3、D1、D2 → C2、E1 → B1-B2 → A7-A10、B3-B5。不要要求自己一次记住整本；先读懂一个小主题，再合上答案重算一次。

先把五个词变成具体的东西

英文	走迷宫时对应什么	你要回答什么
State 状态 s	现在在哪个格子，以及影响未来的必要信息	我现在处于什么情况?
Action 动作 a	上、下、左、右	这一步做什么?
Reward 奖励 r	走一步扣 1 分，到目标可能得分	刚刚这一步得到多少?
Policy 策略 π	每个格子如何选方向的规则	遇到这种情况，通常怎么选?
Value 价值 V/Q	从这里继续走，预计总共得到多少	长期来看好不好?

最容易混淆：奖励是“这一次收到的分数”；价值是“从现在起往后的预计总分”。价值可以是负数，-1 也比 -10 好。

整门课围绕三个问题

① 给定一种走法，它到底好不好？这叫 policy evaluation / prediction，策略评估或预测。

② 有没有更好的走法？这叫 policy improvement / control，策略改进或控制。

③ 环境太大或规则未知，如何从经验近似学出来？这引出 MC、TD、函数近似、DQN 和策略梯度。

看到公式时，先把它读成人话

$$\text{new} = \text{old} + \alpha(\text{target} - \text{old})$$

旧估计是 2，目标值是 6，学习率 α=0.5：先算差距 6-2=4，再走差距的一半，得到 2+0.5×4=4。不要误算成“新值直接等于目标值”。不同学习算法最重要的区别之一，是 target 从哪里来。

本讲义用于理解之前的原创模拟题，不是官方试卷。计算题沿用原卷的奖励时序、终止条件、初始值和独立更新约定。

Markov：当前状态够不够用?

英文题到底在问什么

题目问：一个状态要满足什么条件，才能叫 Markov state（马尔可夫状态）？核心不是背“memoryless”，而是判断：已知当前状态和动作后，还需不需要过去的信息，才能预测下一步？

用两个迷宫例子理解

例一：地图固定，门永远开着。知道现在在哪个格子、准备往哪个方向走，就足以判断下一步到哪里。之前绕了多少圈不影响这个判断，因此位置可以是合适的状态。

例二：开门必须有钥匙。如果只记录位置，同一个门口可能因为“以前有没有拿钥匙”而产生不同结果。只记位置就不够；把“是否持有钥匙”也放进状态，才可能恢复所需的 Markov 信息。

不是把所有过去都删除，而是把过去对未来有用的信息总结在当前状态里。

$$P(S_{t+1}, R_{t+1} \mid H_t, A_t) = P(S_{t+1}, R_{t+1} \mid S_t, A_t)$$

左边：知道全部历史 H_t 与当前动作时，预测下一结果。右边：只知道当前状态 S_t 与动作时，预测下一结果。两者相同，表示更早历史不再提供额外预测信息。

为什么选 B，其他选项错在哪里

选项	判断理由
A：下一状态必须确定	错。Markov 也允许随机转移，例如向右走有 80% 成功、20% 滑倒。
B：给定状态和动作，不再需要更早历史	对。这就是要求当前状态信息足够。
C：状态必须复制全部历史	错。一个“有无钥匙”标记就可能代替很长的历史。
D：奖励必须与动作无关	错。不同动作完全可以带来不同奖励。

最短记忆句

“Given the current state and action, the earlier history provides no additional information about the next outcome.”

会判断是否缺少“钥匙、速度、剩余资源”等信息，比单纯记住“无记忆”更有用。

Return：怎样把未来奖励加起来?

题目翻译

依次收到奖励 2、-1、4，然后回合结束。折扣因子 γ=0.5，问起点的 return（回报）是多少。

为什么不能直接算 2-1+4?

因为题目规定未来奖励要折扣。站在当前时刻，下一次收到的奖励直接算；再晚一次的奖励乘 γ；再晚两次的奖励乘 γ^2。γ=0.5 表示每往后推迟一步，权重再减半。

从起点看	实际奖励	折扣权重	计入总回报
第一次收到	2	1	2
第二次收到	-1	0.5	-0.5
第三次收到	4	0.5^2=0.25	1

$$G_0 = 2 + 0.5(-1) + 0.5^2(4) = 2.5$$

所以选择 C。未折扣的总分是 5，但本题问的是折扣回报。

下标是什么意思

G_0 表示“从第 0 个时刻往后看”的回报。奖励通常写成 R_1、R_2、R_3，因为先在时刻 0 做动作，之后才收到 R_1。

$$G_t = R_{t+1} + \gamma G_{t+1}$$

这条式子读成：“现在往后的总回报 = 下一次收到的奖励 + 后面所有回报打一次折扣。”它就是后续 Bellman 方程的出发点。

从后往前算也可以

$$G_2 = 4, \qquad G_1 = -1 + 0.5(4) = 1$$
$$G_0 = 2 + 0.5G_1 = 2 + 0.5(1) = 2.5$$

向后换一个起点，折扣次数就重新数。不能把 G_0 中的折扣权重原样搬到 G_1。

常见错误与自检

不要把第一个奖励也乘 γ；到达终点的那一步奖励仍然计入，只是终点之后没有额外奖励。

若 γ=1，本题回报为 5；若 γ=0，只看第一个奖励，回报为 2。用这两个极端值帮助理解折扣的作用。

V 和 Q：差的是第一步是否已经选定

先区分三件事

π(a|s)：在状态 s 选动作 a 的概率，描述“怎么选”。$q_\pi(s,a)$：第一步先选定 a，之后按 π 行动，预计能得到多少总回报。$v_\pi(s)$：第一步也按 π 选动作，预计能得到多少总回报。

Q 问：“先做这个动作，后面照原策略走，会怎样？” V 问：“站在这里，直接按原策略走，会怎样？” 两者都看长期结果。

把公式换成一次加权平均

假设状态 s 有 Left、Right 两个动作。策略以 25% 概率选 Left、75% 概率选 Right；对应 Q 值分别是 8 和 4。

动作	被策略选中的概率	动作价值	对平均价值的贡献
Left	0.25	8	0.25×8=2
Right	0.75	4	0.75×4=3

$$v_\pi(s) = 0.25(8) + 0.75(4) = 5$$

把任意数量的动作都写进去，就是：

$$v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a)$$

Σ 的意思是“对所有动作求和”。每项是“这个动作出现的概率 × 这个动作的长期价值”。所以 A 正确。

为什么不是 max?

本例 max(8,4)=8，但原策略有 75% 概率选到价值为 4 的动作，所以原策略的平均表现不能算成 8。

取 max 表示“现在允许改成选最好的动作”。题目若在评估 fixed policy（固定策略），就应该按它的概率平均；题目若在求最优或贪心改进，才考虑 max。

其他错误选项

直接把所有 Q 加起来，没有考虑概率，不能表示期望；取最小值表示总选最差动作，也不是题目给定策略的表现。

读题抓词：given/fixed policy → 按 π 加权；greedy/optimal → 比较动作、取最大。

k 和同步更新：先分清“哪一张表”

k 不是智能体走了几步

V_0 是算法开始时的初始估计表。用它计算一次所有状态，得到 V_1；再用 V_1 算一次所有状态，得到 V_2。这里的 k 是“整张价值表更新了多少轮”。

同步 synchronous 的含义是：算新一轮时，一直读旧表；等新表全部算完，再整体换掉旧表。

一个两格小例子

每次更新都使用下面两条规则：A 的新估计是 1+0.5×B 的旧估计；B 的新估计是 2+0.5×A 的旧估计。两格初始都是 0。

当前读哪张表	新 A 怎么算	新 B 怎么算	得到的新表
$V_0=(0,0)$	1+0.5×0=1	2+0.5×0=2	$V_1=(1,2)$
$V_1=(1,2)$	1+0.5×2=2	2+0.5×1=2.5	$V_2=(2,2.5)$

算 $V_2(B)$ 时使用的是 $V_1(A)=1$。刚算出的 $V_2(A)=2$ 属于新表，这一轮还不能读它。

原地更新为什么会得到不同数字

如果从同样的旧表 (1,2) 出发，只把这一轮改成 in-place（原地更新），而且先算 A 再算 B，那么 A 先变为 2，B 可以立即使用这个新 A，算成 2+0.5×2=3。

这不是哪种算法算错，而是更新约定不同。原模拟卷明确规定同步 DP，所以 A4 选择 D；选项 B 描述的是另一种更新方式。

三个下标不要混淆

记号	本卷中的含义
t	一条实际或模拟轨迹里的时间位置。
k	DP 价值表的更新轮数。
s 或格子坐标	你正在计算哪个状态的价值。

你之前问 k=2、k=3 时，正确操作都是：确定后继位置 → 从上一轮抄数 → 代入。后面的 C3 会用真实题目再走一遍。

ε-greedy：两条选动作的途径

先区分 exploitation 和 exploration

Exploitation（利用）：相信当前估计，挑现在看起来最好的动作。Exploration（探索）：给其他动作机会，避免一直相信可能不准的早期估计。

ε-greedy 的规则是：以 1-ε 的概率走利用分支；以 ε 的概率走随机探索分支。原卷规定探索时从所有动作均匀选择，最优动作也在里面。

把 ε=0.2 想成选择 100 次时的平均次数

有 4 个动作，其中只有一个价值最大。大约 80 次进入利用分支，这 80 次都选最优动作；大约 20 次进入随机分支，每个动作平均分到 20/4=5 次。

途径	最优动作得到的概率
利用分支	0.8
探索分支也抽中它	0.2×1/4=0.05
合计	0.85

$$P(a_*) = (1-\varepsilon) + \frac{\varepsilon}{m} = 0.8 + 0.05 = 0.85$$

其他三个动作各 0.05，总和 0.85+3×0.05=1。因此选 B。

“探索”并不是“必须选非最优动作”。本卷的随机池包含全部动作，所以探索也可能抽中最优动作。

为什么 0.80 不对

它只算到了利用分支，漏掉了随机分支抽中同一个动作的机会。0.20 则是“进入随机分支”的概率，不是最终选中最优动作的总概率。

如果有并列最大动作

本卷约定利用概率在并列最优动作之间均分。若 4 个动作中有 2 个并列最优，ε=0.2，则每个并列动作概率为 0.8/2+0.2/4=0.45；其余两个各 0.05。

上面的“100 次”是在讲期望频率，不是要求真实随机试验每 100 次都恰好出现这些次数。GLIE 如何安排长期探索，见 B2。

SARSA 与 Q-learning：后面准备怎么走?

先看它们共同在做什么

两者都要学习 Q(s,a)：在状态 s 做动作 a，长期看有多好。观察到一次 (s,a,r,s′) 后，利用下一状态的估计，修正当前 Q(s,a)。它们不要求事先知道环境转移概率。

$$Q_{\text{new}}(s,a) = Q_{\text{old}}(s,a) + \alpha[Y - Q_{\text{old}}(s,a)]$$

区别集中在目标 Y：到了下一状态 s′，用哪个动作的价值来代表未来?

SARSA：使用实际选中的下一动作

假设下一状态有两个动作，Q 值是 10 和 2；探索策略这次实际选了价值为 2 的那个动作。SARSA 就用这个 2。

$$Y_{\text{SARSA}} = r + \gamma Q(s', a'_{\text{actual}})$$

它在评估“自己正在执行的这套行为，包括探索的后果”，因此标准 SARSA 是 on-policy（同策略）。

Q-learning：目标里假设下一步选最好的

同一条经验下，Q-learning 不使用实际选中的 2，而使用 max(10,2)=10。

$$Y_{\text{QL}} = r + \gamma \max_b Q(s', b)$$

它可以一边用探索策略收集经验，一边学习贪心目标的价值，因此是 off-policy（离策略）。

离策略不是“只学别人给的数据”。即使数据都是自己刚走出来的，只要行为策略和学习目标策略不同，也可以是离策略。

回到选择题

A 错：SARSA 不总取最大值。B 错：Q-learning 的行为策略仍可以探索。C 对：实际下一动作与贪心下一动作正是两者区别。D 错：二者都能直接用经验学习，不要求转移模型。

遇到计算题怎么选公式

先圈出题目里的 actually selected next action（实际下一动作）。SARSA 需要它；Q-learning 的目标需要下一状态整行 Q 中的最大值。完整数值计算见 D2。

价值函数和环境模型不是同一种东西

A7：为什么不能永远用一张 Q 表?

小迷宫只有几十个格子，每个格子 4 个动作，存一张表很方便。但如果状态是一张游戏截图，不同图像的数量极大，不可能每张图都存一行、每个状态都单独练到很多次。

函数近似（function approximation）用共享参数 w 构造一个函数，把状态或状态-动作输入进去，输出价值估计。

$$\hat{v}(s,w) \approx v_\pi(s), \qquad \hat{q}(s,a,w) \approx q_\pi(s,a)$$

例如用位置、速度、与目标距离等特征估值。学到“离目标近通常更有利”这类规律，就可能对没有遇到过的相似情况也给出估计。这叫 generalization（泛化）。

表格：一个状态记一个数。函数近似：用同一套参数计算很多状态的数。所以更新一次参数，可能同时改变多个状态的估计。

A7 选 A。B 的“一次更新保证精确”、C 的“不需要经验和奖励”、D 的“所有训练算法都收敛”，都不是函数近似能保证的事。

A10：环境模型回答的是什么?

价值函数回答：“这样做长期好不好？”环境模型回答：“这样做，下一步会发生什么？”它们的输出不同。

对象	输入	典型输出
Q 价值函数	状态、动作	预计长期回报，例如 8.5。
环境模型 Model	状态、动作	下一状态和即时奖励，或它们的概率分布。

Dyna 如何把它们一起用

① 真走一步：得到状态、动作、奖励、下一状态。② 用真实经验更新 Q，同时学习模型。③ 从学到的模型模拟额外经验，再更新 Q。

比如机器人刚撞过一次墙，学会“这里向右会留在原地并扣分”。之后它可以在模型中练习这种结果，增加学习次数，不必每次真的去撞。

所以 A10 选 C：模型提供模拟转移，供额外价值更新使用。模型需要经验才能学好；如果模型猜错，规划也可能受影响。

replay 是重用已经存下来的转移；model-based planning 是用模型生成转移。两者都能增加更新次数，但数据来源不同。

Dueling：先算平均优势，再逐个代入

题目提供了什么

网络的状态价值输出 V(s)=5，三个动作的原始优势输出 A=(2,-1,0)。题目明确要求 mean-centered（减均值）的聚合方法。

先把它看成两种信息：V 表示“这个状态整体的价值水平”；各 A 输出表示不同动作之间的相对差别。网络把这两部分组合成每个动作的 Q。

$$Q(s,a) = V(s) + A(s,a) - \bar{A}(s)$$

$$\bar{A}(s) = \frac{1}{m}\sum_{b} A(s,b)$$

第一步：求原始优势的平均值

$$\bar{A} = \frac{2 + (-1) + 0}{3} = \frac{1}{3}$$

注意：这里三个数都是原始网络输出，平均值不一定已经为 0。公式负责把它们中心化。

第二步：每个动作都减同一个平均值

动作	代入	Q 值
1	5+2-1/3	20/3≈6.6667
2	5-1-1/3	11/3≈3.6667
3	5+0-1/3	14/3≈4.6667

所以答案是 D。最好的动作仍然是动作 1，因为统一减去同一个数，不改变大小顺序。

为什么要减平均值

只写 Q=V+A 时，V 增加 10、所有 A 各减少 10，最终 Q 仍不变，V 和 A 的分工不明确。减均值让有效优势的平均值为 0，使输出 V 与各 Q 的算术平均对应。

本题可检查：(20/3+11/3+14/3)/3=5，正好回到 V。这里的 V 是网络聚合分支，不自动等于任意行为策略下的 v_π。

为什么其他选项不对

A=(7,4,5) 漏了减均值。B=(5,2,3) 相当于减了最大优势 2，属于另一种聚合规则，不是本题指定规则。C 只抄了优势输出，连 V 都没加。

Dueling 是“网络怎么表示 Q”；Double DQN 是“学习目标怎么选动作、评价动作”。名字相似，但解决的是不同问题。

随机策略：为什么有时不能让对手猜中？

确定性与随机性先分清

确定性策略（deterministic policy）：给定相同状态，总选择同一个动作。随机策略（stochastic policy）：给定相同状态，按照一个概率分布选择动作。

例如在同一个状态，出石头概率 1，是确定性策略；石头、剪刀、布各 1/3，是随机策略。

石头剪刀布中的问题

如果你每次都出石头，对手知道以后就会一直出布，你会持续被针对。如果你每轮独立地等概率随机选择，对手无法根据一个固定的动作规律稳定地克制你。

因此，“对手会利用你的可预测性”是直接学习随机策略的合理理由，A9 选 B。

这里随机不是因为你忘了哪个动作好，而是策略本身就需要保持不可预测。它和 ε-greedy 中为获取信息而探索，是不同的理由。

不要推出一个过强的结论

不能因此说“随机策略在所有 MDP 中都严格更好”。对于标准的有限、完全可观测、折扣 MDP，存在确定性的平稳最优策略。

也不能说随机策略会让奖励变成确定，或不需要优化目标。策略是随机的，训练仍然需要知道什么结果算好、希望最大化什么。

另一个直觉：看起来一样的两个位置

如果两个真实位置在观察上完全相同，但各自应该向不同方向走，一个只根据当前观察做决定的确定性策略可能卡住。允许随机选择，有时可以避开这种陷阱。

这个例子中关键信息被观察隐藏了，所以不能拿它直接反驳完全可观测 MDP 的结论。

与 Policy Gradient 的连接

以前的价值方法先学 Q，再按 Q 选动作。策略梯度直接学 $\pi_\theta(a|s)$，也就是动作概率如何随状态变化。比如学到石头概率 0.33、剪刀 0.34、布 0.33，而不是必须先对三个动作各算一个 Q 再选最大。

英文核心句：A predictable deterministic policy can be exploited by an adaptive opponent.

DP、MC、TD：目标值从哪里来?

先用“预计几点到家”理解

你正在回家，想估计从当前位置到家的剩余时间。MC 等真的到家后，回看这段路实际用了多久。TD 走完一小段后，用“刚走的时间 + 下一地点预计还需多久”更新当前预测。DP 则已知各种路况、转移概率和耗时，用模型计算所有可能后续的加权平均。

这只是直觉类比。RL 最大化回报；若用“时间越短越好”的任务，可把时间成本写成负奖励。

两个关键词必须单独判断

Sampling（采样）：用实际或模拟抽到的一条结果，代替枚举所有可能结果。Bootstrapping（自举）：目标中用到了另一个尚在学习的价值估计。

$$\text{MC target} = G_t$$

$$\text{TD target} = R_{t+1} + \gamma V(S_{t+1})$$

$$\text{DP target} = \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s]$$

方法	需要环境模型?	采样?	自举?
标准完整期望 DP	需要	否	是，用 V(s')
完整回合 MC	不需要	是	否，用真实完整回报
TD(0)	不需要	是	是，用 V(s')

Bias 与 variance 怎么理解

假设真实水平是 10。某估计每次都报 8，很稳定，却系统性偏低，这叫偏差；另一个估计有时 0、有时 20，平均是 10，但波动大，这叫方差。

固定策略下，完整 MC 回报的条件期望是正确价值，但长途中的许多随机事件通常使其波动较大。TD 只观察一步，通常波动较小，但若下一状态估计 V 不准确，目标可能被带偏。

“TD 的中间目标可能有偏差”不等于“TD 最后一定学错”。合适条件下，表格 TD 仍可收敛到正确价值。MC/TD 的方差大小也是通常的比较，不是任何例子都严格成立。

简答题怎样组织

先画表回答模型、采样、自举三项；再用两句解释“MC 完整回报通常高方差；TD 借用近似估计可能有自举偏差”。这比直接背六个 yes/no 更不容易混。

探索必须足够，最后又要趋向贪心

GLIE 不是一个新更新公式

全称是 Greedy in the Limit with Infinite Exploration。它描述学习过程中一系列策略应满足的两个性质，不是一种叫 GLIE 的动作选择算法。

条件	人话解释	解决什么问题
Infinite exploration	相关状态-动作对在无限训练中都被尝试无穷多次。	不要因为早期几次运气差，就永远放弃一个其实很好的动作。
Greedy in the limit	训练越来越久时，选非贪心动作的概率趋向 0。	知识逐渐充分以后，不要永远保持固定比例的随机损失。

一个条件保证“该学的能学到”，另一个保证“最后越来越多地选当前认为最好的”。二者必须同时满足。

为什么固定 ε=0.1 不行

假设 4 个动作且只有 1 个最优。非最优动作的总概率为 3×0.1/4=0.075。无论训练多久，这个 7.5% 都不趋向 0，因此不满足极限贪心。

这里还需要注意：即使每次选动作都有随机探索，也不能凭这一点就证明所有状态都能被充分访问。某些状态可能很难到达。

为什么仅 ε 趋向 0 也不够

如果很早就停止探索，或者探索概率下降得过快，你可能根本没足够尝试一个重要动作。之后一直选一个自以为好的动作，虽然已经非常贪心，却不满足充分探索。

因此检查 ε 的变化只是在看一部分条件，还要看各状态-动作对是否实际被持续覆盖。

把你之前混淆的三个词分层

名词	回答什么问题
MC	价值怎么学？用完整回报更新。
ε-greedy	当前这一步怎么选动作？大概率贪心，小概率随机。
GLIE	这些策略长期变化时，是否同时满足充分探索和极限贪心？

“MC + ε-greedy”可以构成一种控制方法，但不会自动满足 GLIE。GLIE 配合适当学习率、覆盖等其他假设，支持合适表格控制算法的收敛结论；它单独不是万能保证。

两个网络，分别负责什么?

DQN 先解决“大表存不下”的问题

Q-learning 原本在表中保存 Q(s,a)。DQN 改用神经网络输出这些 Q 值。输入状态，网络给出各动作的价值，例如 (4,1,3)，表示第一个动作目前估计最好。

这些输出不是概率，不要求加起来等于 1，也可以为负。网络参数 θ 是用来计算 Q 值的一组可学习数字。

经验回放为什么有用

把经历过的 (s,a,r,s′) 保存起来，再随机抽小批量训练。这样可以重复利用较少的真实经验，也减少连续相邻时刻样本过于相似的问题。

它不是把旧经验当成绝对正确答案；学习时仍需要重新算目标和误差。随机抽样也不表示一切相关性和分布变化都消失了。

目标网络为什么有用

Online network（在线网络）参数 θ 不断更新。Target network（目标网络）参数 θ^- 更新较慢，负责提供相对稳定的下一状态目标。

可以把目标网络理解为在线网络的一份“较慢更新的参考版本”。如果参考也每一步随当前预测大幅变化，学习就容易追着一个不断移动的目标跑。

标准 DQN 的目标

下面的目标针对非终止转移；如果下一状态已经终止，目标只使用即时奖励 r。

$$Y_{\text{DQN}} = r + \gamma \max_a Q(s', a; \theta^-)$$

在目标网络中直接找最大的下一动作值，再加即时奖励。注意这里不是用在线网络的最大输出。

Double DQN 改了哪一步

$$a^* = \arg\max_a Q(s', a; \theta)$$

$$Y_{\text{Double}} = r + \gamma Q(s', a^*; \theta^-)$$

第一步，在线网络选动作编号；第二步，目标网络评价这个动作。不要在第二步又重新选一次目标网络自己的最优动作。

“为什么要分开？”若用同一套带噪声的估计既选最高的又评价它，容易选中恰好高估的动作。选择与评价分开可以减轻最大化导致的过估计，不保证每个样本都更准确。

记忆句：Double = online 选，target 评。具体数字的箭头对应见 D2。

一个改抽样，一个改表示

优先经验回放：哪些经验值得多练?

TD error 表示“本次目标和当前预测差多少”。误差绝对值很大，说明这条经验当前还没有被预测好，因此可以让它更容易被抽中。

$$\delta_i = Y_i - Q(s_i, a_i), \qquad p_i = |\delta_i| + \eta$$

这里 η 是很小的正数，确保误差恰好为 0 的样本仍有抽样机会。因为网络会改变，以前误差为 0，不代表以后永远不值得再看。

$$P(i) = \frac{p_i^\kappa}{\sum_j p_j^\kappa}$$

κ 控制优先程度：κ=0 时均匀抽样；κ>0 时优先级较大的一般更常被抽中。

一个两条经验的例子

假设加完偏移后的优先级是 1 和 3，取 κ=1。它们被抽中的概率就是 1/(1+3)=0.25 和 3/(1+3)=0.75。第二条经验平均被多练。

但这样会改变训练样本分布：第二条被看得更多，对更新的影响也可能过重。因此引入 importance-sampling weights（重要性采样权重），修正相对于均匀回放目标的抽样偏差。

$$w_i = \left(\frac{1}{NP(i)}\right)^\beta$$

β 控制校正强度。N=2、β=1 时，两个原始权重为 2 与 2/3；被多抽中的第二条，单次更新权重反而较小。实践中常再统一归一化这些权重。

不是“多抽某样本，同时再给它更大校正权重”。抽样概率和校正权重的作用方向不同。

Dueling 部分怎样答

它有状态价值 V 与动作优势 A 两条输出流，用下面公式组合：

$$Q(s, a) = V(s) + A(s, a) - \frac{1}{m}\sum_b A(s, b)$$

数字计算已在 A8 逐项展示。PER 关注“抽哪条经验训练”；Dueling 关注“网络怎样表示 Q”。二者可以同时使用。

本讲义用 κ 表示优先指数、β 表示校正指数、η 表示小偏移，避免和学习率 α、探索概率 ε 混淆。遇到题目换符号，先看它的定义。

Policy Gradient：直接调选动作的概率

为什么突然出现 θ 和梯度?

前面的价值方法先学 Q，再用 Q 决定动作。策略梯度直接建立一个策略 $\pi_\theta(a|s)$。θ 是策略模型的参数；改变 θ，会改变不同动作被选中的概率。

Gradient（梯度）告诉我们：参数朝哪个方向微调，会让目标增长得更快。目标是增加预期回报，所以用梯度上升，也就是更新公式中的加号。

REINFORCE 的想法

先按当前策略完整走完一回合，算出每个时刻后面的回报 G_t。结果好的动作，在类似状态下以后倾向于更常被选；结果差的动作则相反。具体怎么调参数，由 log π 的梯度决定。

本题明确无折扣，即 γ=1。带状态基线的更新是：

$$\theta \leftarrow \theta + \alpha[G_t - b(S_t)]\nabla_\theta \log \pi_\theta(A_t \mid S_t)$$

公式里的部分	含义
α	本次改动多大。
G_t	这次实际得到的后续总回报。
$b(S_t)$	处于该状态时的参考表现，常用估计 V。
G_t-$b(S_t)$	这次比参考水平好多少或差多少。
∇ log π	参数怎样变化才能提高所选动作的概率。

先把“表现评价”和“参数改变方向”分开理解。不要把整条公式误读成“直接给动作概率加上 αG”。真正更新的是 θ。

为什么要减参考水平

某动作拿到 3 分，是好还是不好？如果这个状态通常只有 1 分，3 分很好；如果通常能拿 8 分，3 分就偏差。baseline 提供这种参照，合适时还能降低梯度估计的波动。

一个最小数值例子

假设在某状态选 A 的概率为 sigmoid(θ)，开始 θ=0，对应概率 0.5。选 A 后得到回报 3，baseline 为 1，α=0.1；此时 log π(A) 对 θ 的导数为 1-0.5=0.5。

$$\theta_{\text{new}} = 0 + 0.1(3 - 1)(0.5) = 0.1$$

更新后的动作 A 概率约为 0.525。你不必先会神经网络反向传播，也能检查：正优势乘上增加 A 概率的方向，确实让 A 变得更常被选。

Baseline 为什么不改变期望？Critic 做什么？

先看 baseline 的条件

基线 b(s) 可以依赖当前状态，但在同一个状态下，它不能随这次选中了哪个动作而任意变化。计算 actor 梯度时，把基线当常量；它自己的参数可以单独训练。

把证明拆成三步

第一步：固定 s，对动作按策略概率求平均，看看“减去 baseline”带来什么额外梯度。

$$\sum_a \pi_\theta(a \mid s) b(s) \nabla_\theta \log \pi_\theta(a \mid s)$$

第二步：利用 π∇logπ=∇π，并把与动作无关的 b(s) 提出来。

$$= b(s) \sum_a \nabla_\theta \pi_\theta(a \mid s)$$

第三步：所有动作概率相加永远等于 1，常数 1 的导数是 0。

$$= b(s) \nabla_\theta \left[\sum_a \pi_\theta(a \mid s) \right] = b(s) \nabla_\theta 1 = 0$$

因此减去基线不改变“平均的梯度方向”，但会改变每个样本的更新。适当基线可以降低方差，不是任何基线都一定更好。

Actor 和 Critic 的分工

Actor 是策略模型，负责选动作并改进策略。Critic 是评价模型，负责估计价值，为 actor 提供“这次表现比预期好多少”的信号。

Advantage（优势）定义为 $q_\pi(s,a)$-$v_\pi(s)$：指定动作的长期表现，减去该状态按原策略行动的平均表现。

用一步 TD error 近似优势

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

本题 γ=1。若即时奖励是 1、当前 V(s)=4、下一状态 V(s′)=5，那么目标是 1+5=6，TD error=6-4=2：结果好于当前预测。

若下一状态终止，则下一价值为 0。即使奖励为正，TD error 也可能是负数，例如 r=3、当前 V=4，误差就是 -1。

只有 critic 恰好等于真实 v_π 时，给定状态和动作，TD error 的条件期望才等于真实优势。实际用近似 critic，得到的是近似学习信号。

为什么一个样本回报不同于状态价值?

把题目信息重新读一遍

γ=0.8。从 A 出发，本步奖励一定是 1，下一状态以一半概率到 A、一半到 B。从 B 出发，本步奖励一定是 2，下一状态以 0.25 概率到 A、0.75 概率到终点 T。终点之后的价值为 0。

(a) 这一次实际走出的路线

样本为 A → A → B → T，实际收到奖励 1、1、2。既然题目给的是已经发生的一条路线，算这条路线的回报时，不再乘它发生的转移概率。

$$G_0 = 1 + 0.8(1) + 0.8^2(2) = 3.08$$

$$G_1 = 1 + 0.8(2) = 2.6$$

0.5、0.25 等转移概率用来平均“所有可能结果”；已经给定一条实际轨迹时，直接折扣它实际收到的奖励即可。

(b) 如果问所有可能未来的平均表现

从 A 开始：本步先得 1，之后可能到 A 或 B，未来价值按各一半平均，再乘一次 γ。

$$v_A = 1 + 0.8(0.5v_A + 0.5v_B) = 1 + 0.4v_A + 0.4v_B$$

从 B 开始：本步先得 2，之后 0.25 概率回 A，0.75 概率结束。

$$v_B = 2 + 0.8(0.25v_A + 0.75 \times 0) = 2 + 0.2v_A$$

(c) 解这两个互相依赖的未知数

$$v_A = 1 + 0.4v_A + 0.4(2 + 0.2v_A) = 1.8 + 0.48v_A$$

$$v_A - 0.48v_A = 1.8 \quad \Rightarrow \quad v_A = 1.8/0.52 = 45/13$$

$$v_B = 2 + 0.2(45/13) = 35/13$$

所以 v(A)≈3.4615，v(B)≈2.6923。3.08 是一个样本；3.4615 是从 A 出发所有可能未来的期望，两者不必相等。

(d) 不直接解方程，先迭代两轮

轮次	A	B
V_0	0	0
V_1	1+0.4×0+0.4×0=1	2+0.2×0=2
V_2	1+0.4×1+0.4×2=2.2	2+0.2×1=2.2

计算 V_2(B) 时用旧 A=1，不用刚算出的新 A=2.2。两轮结果是近似值，不需要已经等于联立方程的精确解。

先固定策略，再合成一个概率模型

这题比 C1 多了什么?

C1 没有动作选择。C2 中，先由策略决定选哪个动作，再由环境决定动作的结果。需要把这两层随机性合起来。

γ=0.5。X 的策略：wait 与 go 各一半。Y 的策略：retry 概率 0.25，finish 概率 0.75。

(a) 从 X 出发

选 wait 的概率 0.5，之后一定留在 X，奖励 0；选 go 的概率 0.5，之后一定到 Y，奖励 2。

所以 X→X 概率 0.5，X→Y 概率 0.5，X→T 概率 0；平均本步奖励是 0.5×0+0.5×2=1。

从 Y 出发，要展开分支

事件	发生概率	最终去哪里
选 retry，再回 X	0.25×0.5=0.125	X
选 retry，再到终点	0.25×0.5=0.125	T
选 finish	0.75×1=0.75	T

因此去 T 的总概率是 0.125+0.75=0.875；平均奖励是 0.25×0+0.75×4=3。

当前状态	到 X	到 Y	到 T	平均奖励
X	0.5	0.5	0	1
Y	0.125	0	0.875	3
T	0	0	1	0

转移矩阵每一行加起来应为 1。这是检查概率有没有漏项的最快方法。终点行写吸收状态，只是矩阵表示；它不再产生奖励。

(b) 写成 Bellman 方程

$$v_X = 1 + 0.5(0.5v_X + 0.5v_Y) = 1 + 0.25v_X + 0.25v_Y$$
$$v_Y = 3 + 0.5(0.125v_X + 0.875 \times 0) = 3 + 0.0625v_X$$

第二式代入第一式:

$$v_X = 1 + 0.25v_X + 0.25(3 + 0.0625v_X) = 1.75 + 0.265625v_X$$
$$v_X = 1.75/0.734375 = 112/47, \qquad v_Y = 3 + v_X/16 = 148/47$$

0.0625 中有折扣 0.5、选 retry 的概率 0.25、retry 回 X 的概率 0.5。三者作用不同，不能漏掉一个，也不能重复乘。

算 Q 时先固定动作，改策略后重新估值

(c) 当前动作已指定，不再乘它被选中的概率

现在已知原策略的 v_X=112/47、v_Y=148/47。计算 q_π(X,go) 的含义是：“第一步确定选 go，此后才按照原策略 π 行动。”

所以 go 的即时奖励 2 必须完整计入，不能再乘“原策略选 go 的概率 0.5”。环境结果本身若随机，仍要平均。

先指定的动作	本步奖励	下一步以后	q_π
X: wait	0	$\gamma \times v_X$	56/47≈1.1915
X: go	2	$\gamma \times v_Y$	168/47≈3.5745
Y: retry	0	$\gamma \times (0.5 v_X + 0.5 \times 0)$	28/47≈0.5957
Y: finish	4	后续为 0	4

重点检查 retry

$$q_\pi(Y, \text{retry}) = 0 + 0.5\,[0.5 v_X + 0.5(0)] = 28/47$$

外面的 0.5 是折扣 γ；括号里的 0.5 是环境回到 X 的概率。这里没有再乘原策略的 retry 概率 0.25，因为动作已经被题目指定。

V：第一步也按策略选，所以要乘动作概率。Q：第一步已指定，所以不乘当前动作的策略概率。这条区别贯穿 A3、C2、E1。

(d) 按动作价值挑更好的动作

X：go 的 3.5745 大于 wait 的 1.1915，所以改成总选 go。Y：finish 的 4 大于 retry 的 0.5957，所以改成总选 finish。

新策略是 π′(X)=go，π′(Y)=finish。

为什么新策略的 X 价值是 4，而不是 3.5745?

新策略下，从 X 走 go 得到 2，到 Y 后一定 finish，再得到 4。因此：

$$v_{\pi'}(Y) = 4, \qquad v_{\pi'}(X) = 2 + 0.5(4) = 4$$

原来的 q_π(X,go)=3.5745，假设到 Y 后仍按旧策略，有 25% 概率 retry。新策略到 Y 后已经改变，未来变得更好，所以不能直接把旧 q 当成新策略的最终价值。

policy improvement 是换动作；policy evaluation 是算换了以后整条后续行为的价值。两步有关，但不是同一步。

3×3 格子：从 k=0 算到 k=2

这是模拟卷的 3×3，不是课件的 4×4

左上角 (1,1) 和右下角 (3,3) 是终点。非终点每次走一步奖励 -1；上、下、左、右概率各 1/4；撞墙留在原地，也扣 1 分；γ=1。所有初值为 0。

(a) 把每个动作的结果算出来，再取平均

$$V_{k+1}(s) = \frac{1}{4} \sum_{a \in \{U,D,L,R\}} [-1 + V_k(s'_a)]$$

也可以先把共同的 -1 提出来：

$$V_{k+1}(s) = -1 + \frac{V_k(s'_U) + V_k(s'_D) + V_k(s'_L) + V_k(s'_R)}{4}$$

相同后继位置出现两次，也要分别计入：我们平均的是四个动作的结果，不是把重复格子去掉后平均。

(b) k=1：所有非终点都是 -1

因为 V_0 中所有后继值都是 0，新值就是 -1+0。终点保持 0，不再走动作扣分。

V_1	第 1 列	第 2 列	第 3 列
第 1 行	0	-1	-1
第 2 行	-1	-1	-1
第 3 行	-1	-1	0

(c) k=2：以格子 (1,2) 为例

往上撞墙，仍在 (1,2)，旧值 -1；往下到 (2,2)，旧值 -1；往左进终点，旧值 0；往右到 (1,3)，旧值 -1。

$$V_2(1,2) = -1 + \frac{-1-1+0-1}{4} = -1.75$$

若四个动作都没有到终点，四个旧值都是 -1，新值就是 -2。

V_2	第 1 列	第 2 列	第 3 列
第 1 行	0	-1.75	-2
第 2 行	-1.75	-2	-1.75
第 3 行	-2	-1.75	0

目前所有动作仍然随机，各 1/4。不要因为左边是终点，就擅自把策略改成总往左。改策略是后面 (e) 才问的事。

k=3 用 V_2；贪心选择看四个动作各自的值

(d) 先计算 $V_3(1,2)$

请看上一页的 V_2 表。动作顺序按上、下、左、右：

动作	下一位置	从 V_2 抄来的值
Up	(1,2)，撞墙原地	-1.75
Down	(2,2)	-2
Left	(1,1)，终点	0
Right	(1,3)	-2

$$V_3(1,2) = -1 + \frac{-1.75 - 2 + 0 - 2}{4} = -2.4375$$

再计算 $V_3(2,2)$

3×3 的中心格子，四个邻居分别是 (1,2)、(3,2)、(2,1)、(2,3)。在 V_2 表中，它们全都是 -1.75。

$$V_3(2,2) = -1 + \frac{4(-1.75)}{4} = -2.75$$

4×4 课件中 (2,2) 的邻居结构不同，得到的数可能是 -2.875。不要只记坐标和答案；一定回到本题网格数邻居。

(e) 此时才问“如果选最好动作，该选谁”

题目指定用 V_2 作为后续价值。现在第一步分别固定成某个动作，各算一个“本步奖励 + 后续价值”。

固定动作	算式	动作候选值
Up	-1+(-1.75)	-2.75
Down	-1+(-2)	-3
Left	-1+0	-1
Right	-1+(-2)	-3

$$\max(-2.75, -3, -1, -3) = -1, \qquad \arg\max = \text{Left}$$

这些数都是负数，所以 -1 最大。max 给最大数值 -1；argmax 给实现它的动作 Left。

本题最关键的分界

(a)-(d) 评估原来的随机策略，所以四个动作取平均；(e) 比较第一步该选什么，所以四个动作各算一个值后取最大。这是 policy evaluation 和 policy improvement 的区别。

同一回合，first-visit 和 every-visit 用哪些样本?

先把出现时间标出来

轨迹是 A → B → A → T，三次奖励依次为 1、2、3，γ=0.9。A 在 t=0 出现一次，在 t=2 又出现一次，但这是同一个状态的两次访问。

时间位置	当前状态	从这里往后收到的奖励
t=0	A	1、2、3
t=1	B	2、3
t=2	A	3

(a) 从后往前求三个回报

$$G_2 = 3$$
$$G_1 = 2 + 0.9(3) = 4.7$$
$$G_0 = 1 + 0.9(4.7) = 5.23$$

同一个状态 A 在不同时间出现，后面还剩多少奖励不同，所以本次轨迹提供了两个不同回报：5.23 和 3。

(b) First-visit MC：每回合只取该状态第一次出现

本小题初值 V(A)=1、V(B)=2，α=0.5。A 只用 t=0 的回报 5.23，t=2 再次出现不触发第二次更新。B 用 t=1 的 4.7。

$$V(A) = 1 + 0.5(5.23 - 1) = 3.115$$
$$V(B) = 2 + 0.5(4.7 - 2) = 3.35$$

这里“第一次”指本回合内第一次，不是整个训练历史只允许更新一次。

(c) Every-visit sample mean：另一套独立估计

题目另开一个估计器，明确“没有以前的样本，只对本回合全部访问的回报取算术平均”。因此 A 用两个回报，B 用一个回报。

$$V(A) = \frac{5.23 + 3}{2} = 4.115, \qquad V(B) = 4.7$$

这一问没有让你用旧值 1、2，也没有让你用 α=0.5。它问的是样本均值，所以不要沿用上一问的更新结果。

为什么两个 MC 结果不同?

一是选样本不同：首次访问只取 A 的第一份回报，每次访问用两份。二是题目指定的估计方式不同：(b) 从已有估计向目标走一半；(c) 重新从本回合样本求平均。

看到 first-visit/every-visit，先确定选哪些回报；再看题目要求固定步长更新还是样本平均。这是两个独立判断。

在线 TD：走一步就修正一次预测

开始前必须重置

这部分与 MC 小题独立，重新使用 V(A)=1、V(B)=2、V(T)=0。α=0.5，γ=0.9。不能把前一页 MC 算出的值带过来。

TD 每次只用“一步真实奖励 + 下一状态当前估值”构造目标：

$$Y = r + \gamma V(s'), \qquad V(s) \leftarrow V(s) + 0.5[Y - V(s)]$$

第一步：A → B，收到奖励 1

当前 A 的旧值是 1，下一状态 B 的当前值是 2。目标不是完整回报 5.23，而是 1+0.9×2=2.8。

$$\delta = 2.8 - 1 = 1.8, \qquad V(A) = 1 + 0.5(1.8) = 1.9$$

此时表已经变为 A=1.9、B=2。

第二步：B → A，收到奖励 2

当前 B 的旧值是 2；下一状态 A 现在已经是 1.9，不再是初始值 1。

$$Y = 2 + 0.9(1.9) = 3.71$$

$$\delta = 3.71 - 2 = 1.71, \qquad V(B) = 2 + 0.5(1.71) = 2.855$$

此时表变为 A=1.9、B=2.855。

第三步：A → T，收到奖励 3

当前 A 的旧值是 1.9。终点之后没有剩余价值，所以目标就是 3。

$$Y = 3 + 0.9(0) = 3$$

$$V(A) = 1.9 + 0.5(3 - 1.9) = 2.45$$

最终结果为 V(A)=2.45、V(B)=2.855。

(e) 为什么和 MC 不一样?

MC 等整段结果出来，目标用实际完整回报。TD 中途就更新，目标中的未来部分还是当前估计。加上本题的访问与更新规则不同，训练一次后结果当然可能不同；不是说某个算法必须算错了。

这里是 online TD，所以新值立刻生效。前面的 C1、C3 明确是 synchronous DP，所以整轮只读旧表。看到“更新”两字，必须再确认是哪一种顺序。

先找到 target，再代入更新式

把已知量摆在一起

旧 Q(s,a)=2，r=1，γ=0.9，α=0.5。下一状态在线/当前 Q 值为 (4,1,3)，对应 (a1,a2,a3)。实际下一动作是 a2，ε=0.3。

(a) 行为策略概率

利用分支 0.7 全给 a1；探索分支 0.3 均分成三个 0.1。

$$P(a_1) = 0.8, \qquad P(a_2) = 0.1, \qquad P(a_3) = 0.1$$

虽然 a2 只有 10% 概率，它仍然可能被选中。题目告诉我们，这次恰好就选了它。

(b) SARSA 用实际 a2 的值 1

$$Y = 1 + 0.9(1) = 1.9, \qquad Q_{\text{new}} = 2 + 0.5(1.9 - 2) = 1.95$$

目标略低于旧值 2，所以新值略微下降。这是一个很方便的方向检查。

(c) Q-learning 用最大值 4，重新从旧值 2 开始

$$Y = 1 + 0.9(4) = 4.6, \qquad Q_{\text{new}} = 2 + 0.5(4.6 - 2) = 3.3$$

目标大于旧值，因此新值上升。本题不需要乘 a1 被行为策略选中的概率 0.8，因为更新目标已经指定取最大下一动作价值。

(d) 深度网络题：只算目标，不算参数更新

网络	a1	a2	a3
Online 在线网络	4	1	3
Target 目标网络	2	5	3

标准 DQN：在目标网络这一行找最大值 5，对应 a2。因此 Y=1+0.9×5=5.5。

Double DQN：先在在线网络找最大值 4，记住它的动作编号 a1；再到目标网络读取同一列 a1 的值 2。因此 Y=1+0.9×2=2.8。

Double 要带过去的是“动作编号 a1”，不是把在线网络的数值 4 带过去，也不是换到目标网络后重新取最大值 5。

(e) 如果下一状态是终点

$$Y = r = 1, \qquad Q_{\text{new}} = 2 + 0.5(1 - 2) = 1.5$$

终点后续价值为零；不能再把表中的下一动作值加进去。进入终点收到的即时奖励 1 仍然保留。

Policy iteration：先评价，再改变动作

题目有哪些选择

γ=0.5。S：a 得 1 并回 S；b 得 0 并去 U。U：c 得 3 并终止；d 得 1 并去 S。初始策略在 S 总选 a、在 U 总选 d。

(a) 按初始策略走，到底会发生什么?

从 S 开始会一直选 a，每一步都得 1。虽然永不终止，但折扣后的总回报有限。

$$v_{\pi_0}(S) = 1 + 0.5 v_{\pi_0}(S) \quad \Rightarrow \quad v_{\pi_0}(S) = 2$$

从 U 开始先选 d，得 1 并到 S；之后沿用刚算出的 S 价值：

$$v_{\pi_0}(U) = 1 + 0.5(2) = 2$$

无限步不一定产生无限回报。这里 1+0.5+0.25+...=2。需要看是否折扣及奖励是否有界。

(b) 四个第一步动作都试着算一下

注意：即使原策略不选某个动作，也要计算“假如第一步选它，之后恢复原策略”会怎样。

指定第一动作	本步奖励 + 折扣后续	$q_{\pi 0}$
S 选 a	1+0.5×2	2
S 选 b	0+0.5×2	1
U 选 c	3+0	3
U 选 d	1+0.5×2	2

在 S，a 优于 b；在 U，c 优于 d。因此改进策略为 π_1=(a,c)。

(c) 换策略后，重新评估

S 还是一直选 a，价值仍为 2；U 改成直接选 c 到终点，价值变成 3。于是新价值是 (2,3)。

再比较：S 的 a=1+0.5×2=2，b=0+0.5×3=1.5；U 的 c=3，d=1+0.5×2=2。最佳动作仍是 (a,c)，策略稳定，并满足 Bellman 最优性方程。

$$v_* = (2, 3), \qquad \pi_* = (a, c)$$

不要只看“U 可以拿 3，比 S 每步拿 1 多”。从 S 去 U 要先经历一个奖励为 0 的步骤，再折扣 3，只值 1.5；留在 S 的持续奖励值 2。

Value iteration：每轮直接比较最佳候选

与上一页的策略迭代有什么不同

策略迭代先固定一套动作规则，评估它，再换规则。价值迭代每一轮都直接比较动作候选值，不等某个固定策略被完全评估完。

$$V_{k+1}(S) = \max\{1 + 0.5V_k(S),\ 0.5V_k(U)\}$$
$$V_{k+1}(U) = \max\{3,\ 1 + 0.5V_k(S)\}$$

大括号里的两个数分别对应两个动作。先各自算完，再选较大者；不是把它们平均。

第 1 轮：只用 $V_0=(0,0)$

S：a 候选值为 1，b 为 0，选 1。U：c 为 3，d 为 1，选 3。

$$V_1 = (1, 3)$$

第 2 轮：只用 $V_1=(1,3)$

S：a=1+0.5×1=1.5；b=0.5×3=1.5，最大值是 1.5。U：c=3；d=1+0.5×1=1.5，最大值是 3。

$$V_2 = (1.5, 3)$$

第 3 轮：只用 $V_2=(1.5,3)$

S：a=1+0.5×1.5=1.75；b=0.5×3=1.5，最大值是 1.75。U：c=3；d=1+0.5×1.5=1.75，最大值是 3。

$$V_3 = (1.75, 3)$$

第二轮的平局为什么不代表两个动作都最优?

当时用的是不精确的中间估计。后续 S 的估计继续上升，最终达到 2；U 保持 3。最终比较中，S 选 a 得到 2，选 b 只有 1.5，已经不是平局。

“对当前 V_k 贪心”是基于目前估计选动作；“对真实最优 v_* 贪心”才给出最优动作。有限轮的中间数字可能还没反映完整远期价值。

把三种 DP 任务放回原位

Policy evaluation：固定 π，对动作按 π 加权。Policy improvement：用已估价值比较动作，改变策略。Value iteration：反复做取 max 的最优性更新。

哪一步不懂，就从哪一组题重读

基础概念先过这一关

你卡住的地方	先看题号 / 本讲义页码
状态是什么、Markov 是什么	A1 / 第 2 页
奖励和回报有什么区别、γ 放哪里	A2 / 第 3 页；C1 / 第 17 页
V 和 Q 为什么不是同一个数	A3 / 第 4 页；C2 / 第 18-19 页
k=2、k=3 到底用哪张表	A4 / 第 5 页；C3 / 第 20-21 页
ε-greedy 的概率、GLIE 的作用	A5 / 第 6 页；B2 / 第 12 页
价值函数、环境模型和 Dyna 有何不同	A7、A10 / 第 8 页
MC 和 TD 的 target 为什么不同	B1 / 第 11 页；D1 / 第 22-23 页
SARSA 与 Q-learning 怎么选下一值	A6 / 第 7 页；D2 / 第 24 页
Double、Dueling、PER 名字混淆	A8 / 第 9 页；B3-B4 / 第 13-14 页
REINFORCE、baseline、critic 看不懂	A9 / 第 10 页；B5 / 第 15-16 页
什么时候用 π 加权、什么时候取 max	C2、C3、E1 / 第 18-21、25-26 页

重做计算题时，只写这四行

① 我现在估计的是哪个状态或动作？② 本题规定的目标是什么？③ 目标里的每个数字来自哪里？④ 新值 = 旧值 + α×差距，或者题目要求的完整 Bellman 更新。

如果第②行写不出来，先回看算法含义；如果第③行写不出来，先找旧表、当前策略和终止条件；最后才检查算术。

做简答题时，按“问题 → 方法 → 作用”说清楚

例如目标网络：目标随参数频繁变化，训练可能不稳定 → 用一份较慢更新的参考网络产生目标 → 让学习目标更稳定。

例如 GLIE：过早贪心可能错过好动作，永久随机又会持续浪费 → 要求无限探索与极限贪心同时成立 → 为适当控制算法的收敛提供策略条件。

建议先完成的五个检查

能解释 V 与 Q 的第一步区别；能独立算 C3 的 V_2；能算 D1 的第二次 TD 更新；能从 D2 两行网络值找到 Double 的目标；能说出 E1 为何策略稳定。

先做到“能解释一题为什么这么算”，再追求速度。看过答案觉得熟悉，和合上答案后自己能决定公式，是两种不同的掌握程度。

本册题号与之前模拟卷保持一致。第 20-21 页严格使用该卷的 3×3 网格；不要与原课件的 4×4 示例混用。原试卷、原答案未作改动。