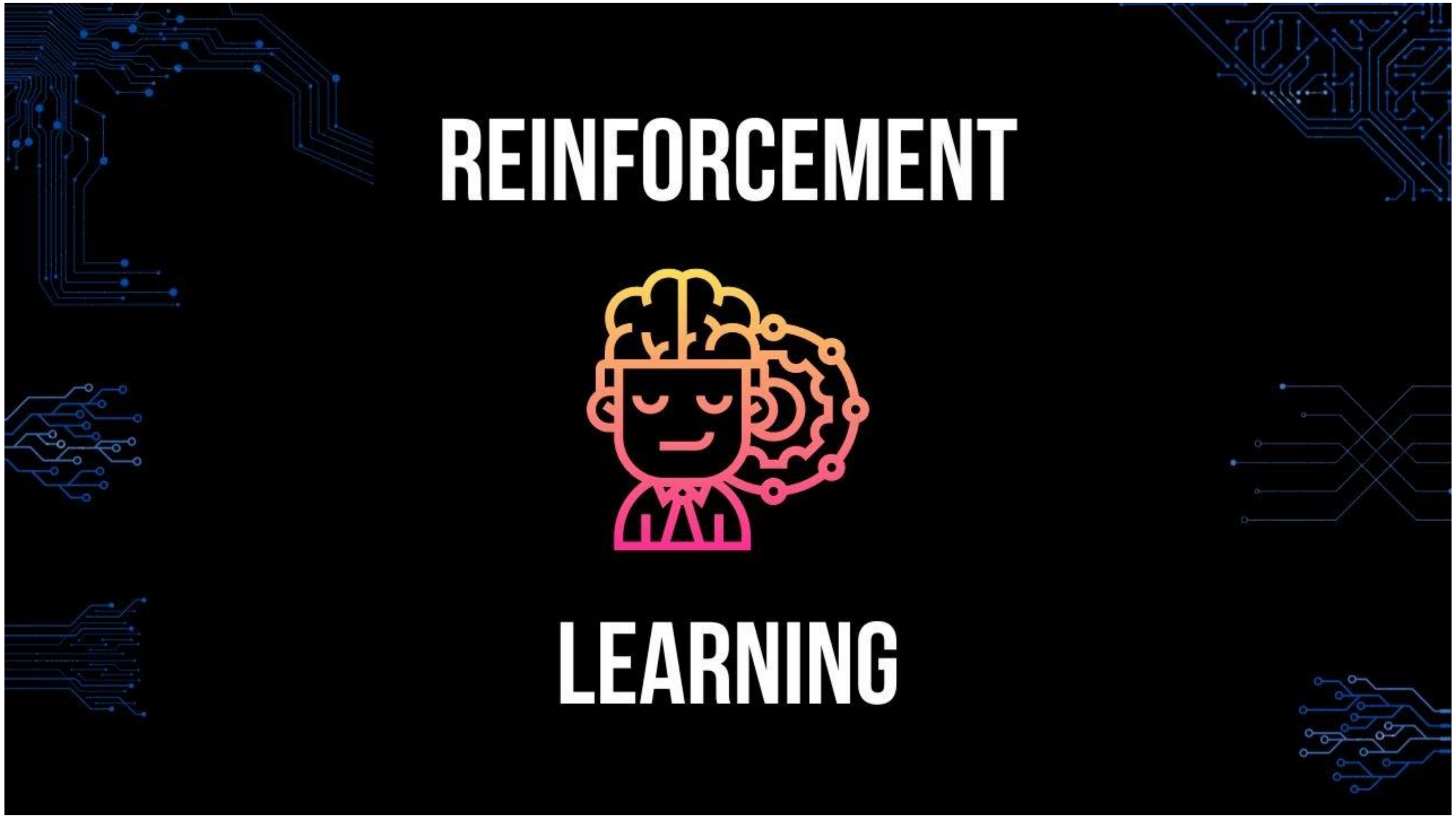
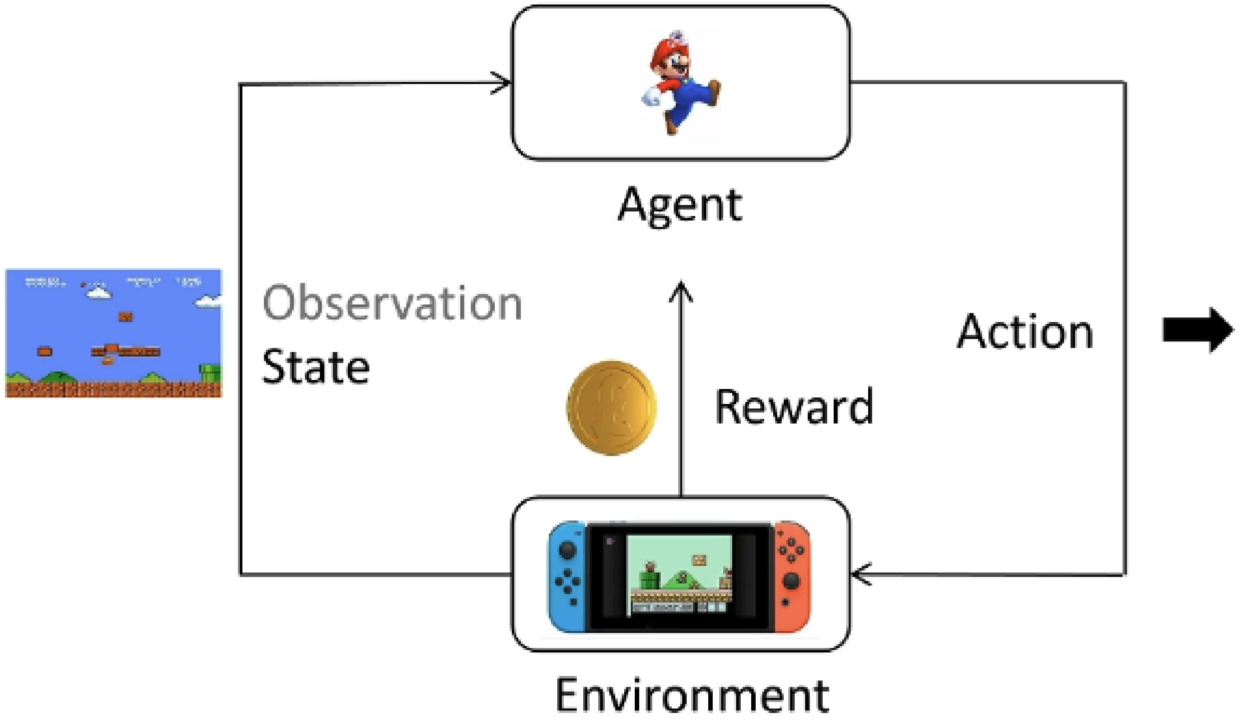


强化学习

Never Settle

Chapter One
Five Elements

RL Architecture



五大核心要素

1. Agent： 理解成需要培训的人
2. Observation(State)： 观测(状态)
3. Environment： 环境
4. Reward： 奖励机制
5. Action： 动作

RL Architecture

Action Space：可选择的动作，比如 {left, up, right}

Policy：策略函数，输入State，输出Action的概率分布。一般用π表示

$$\pi(left|s_t) = 0.1$$
$$\pi(up|s_t) = 0.2$$
$$\pi(right|s_t) = 0.7$$

Trajectory：轨迹，用τ表示，一连串状态和动作的序列。Episode，Rollout。$\{s_0, a_0, s_1, a_1, \ldots\}$

确定的状态转移 $s_{t+1} = f(s_t, a_t)$确定

随机的状态转移 $s_{t+1} = P(\cdot\, |s_t, a_t)$确定

Return：回报，从当前时间点到游戏结束的Reward的累积和

期望的定义 Expect

老师告诉你小明考试情况：
 20%考80分
 80%考90分

小明的考试期望结果
0.2 * 80 + 0.8 * 90 = 88
方法2：再考n次，统计平均成绩

期望：每个可能结果的概率与其结果值的乘积之和

$$E(x)_{x\sim p(x)} = \sum_{x} x * p(x) \approx \frac{1}{n}\sum_{i=1}^{n} x_{\quad x\sim p(x)}$$

p(x)：分布p
x：从分布p(x)取x，n代表取了n个，当n趋于无穷大，等式约等于

目的：训练一个**Policy**神经网络π,在所有状态S下，给出相对应的**Action**，得到**Return**的期望最大

通俗说法：训练一个**Policy**神经网络π,在所有可能的**Trajectory**中，得到的**Return**的期望最大

Math

$$E\big(R(\tau)\big)_{\tau\sim P_\theta(\tau)} = \sum_\tau R(\tau)P_\theta(\tau)$$

τ:符合分布$P_\theta(\tau)$，其中θ是我们需要训练的神经网络参数(Parameters)

τ:是决策网络来决定的

含义：**在由参数为θ的策略所产生的轨迹分布中，我们能获得的期望总回报**

符合分布：在决策每一步的概率分布中随机抽的一个样本

怎么让期望尽可能大?

答案：训练参数θ，让决策网络进行决策修改

$$\nabla E\big(R(\tau)\big)_{\tau\sim P_\theta(\tau)} = \nabla\sum_\tau R(\tau)P_\theta(\tau)$$

$$= \sum_\tau R(\tau)\nabla P_\theta(\tau)$$

$$= \sum_\tau R(\tau)\nabla P_\theta(\tau)\frac{P_\theta(\tau)}{P_\theta(\tau)}$$

$$= \sum_\tau P_\theta(\tau)R(\tau)\frac{\nabla P_\theta(\tau)}{P_\theta(\tau)}$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} R(\tau^n)\frac{\nabla P_\theta(\tau^n)}{P_\theta(\tau^n)} \qquad \nabla\log f(x) = \frac{\nabla f(x)}{f(x)}$$

$$= \frac{1}{N}\sum_{n=1}^{N} R(\tau^n)\nabla\log P_\theta(\tau^n)$$

Policy Gradient 策略梯度

$$= \frac{1}{N}\sum_{n=1}^{N} R(\tau^n)\,\nabla log P_\theta(\tau^n)$$

当前状态 + 当前动作

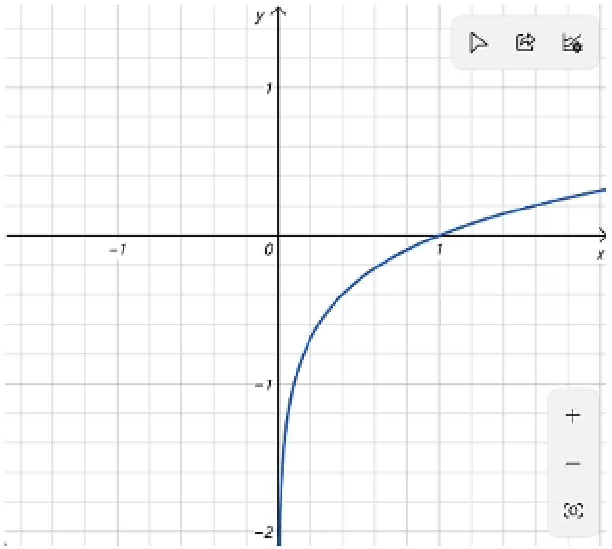
$$= \frac{1}{N}\sum_{n=1}^{N} R(\tau^n)\,\nabla log \prod_{t=1}^{T_n} P_\theta(a_n^t|s_n^t)$$

当前Trajectory给出的连乘

$$= \frac{1}{N}\sum_{n=1}^{N} R(\tau^n)\sum_{t=1}^{T_n} \nabla log P_\theta(a_n^t|s_n^t)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla log P_\theta(a_n^t|s_n^t)$$

更新参数：学习率+梯度



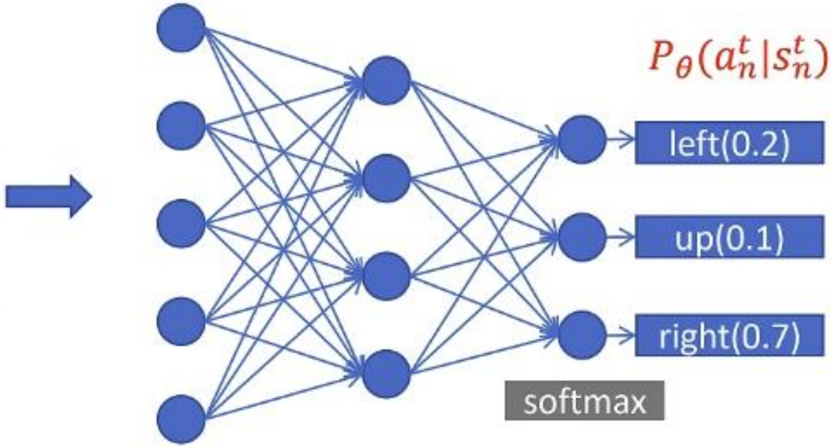
$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n) log P_\theta(a_n^t|s_n^t)$$

部分1：$R(\tau^n)$ 轨迹得到的Return
部分2：$log P_\theta(a_n^t|s_n^t)$：每一步根据State做出Action的概率

总结： 如果Return > 0，则增大$log P_\theta(a_n^t|s_n^t)$
如果Return < 0，则减小$log P_\theta(a_n^t|s_n^t)$

Policy Network训练

$$Loss = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n) log P_\theta(a_n^t|s_n^t)$$



$R(\tau^n)$

$\tau^1 -> R(\tau^1)$

$\tau^2 -> R(\tau^2)$

$\vdots$

$\tau^n -> R(\tau^n)$

负号：优化器最小化（最大化目标函数）

2. 优化器的工作方式是什么？

绝大多数深度学习框架（如TensorFlow, PyTorch）中的优化器，比如**梯度下降（Gradient Descent）及其变种（Adam, SGD等）**，**它们的设计目标是最小化一个损失函数（Loss Function）**。

它们的工作流程是：

1. 计算损失函数 `Loss` 相对于网络参数 `θ` 的梯度 `∇Loss`。
2. 将参数 `θ` 沿着梯度的**反方向**更新一小步，从而让 `Loss` 下降。

3. "负号"如何连接"目标"和"工具"？

现在我们面临一个矛盾：

- 我们的**目标**：最大化 `Σ R * log P`。
- 我们的**工具**（优化器）：只能最小化 `Loss`。

解决方案非常简单：利用一个数学技巧。
最大化一个函数 `J(θ)` 等价于 最小化它的相反数 `-J(θ)`。

想象一座山，找到它的最高点（最大化海拔），就等同于把这座山完全翻转过来变成一个山谷，然后去寻找这个山谷的最低点（最小化 `-海拔`）。

On Policy

Policy Network训练

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla log P_\theta(a_n^t|s_n^t)$$

优化方向1：Action只会对当前Reward
和之后有影响，而不是从头开始
优化方向2：Action的影响会逐步衰减

$$R(\tau^n) \rightarrow \sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n = R_t^n$$

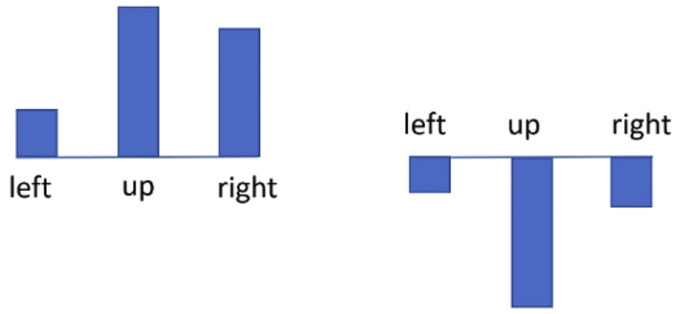
γ: 衰减因子

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R_t^n \nabla log P_\theta(a_n^t|s_n^t)$$

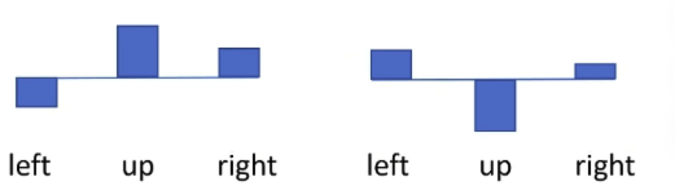
$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} (R_t^n - B(s_n^t))\nabla log P_\theta(a_n^t|s_n^t)$$

Actor – Critic 算法

Actor：用来做动作的神经网络
Critic：对动作进行打分



好的局势　　坏的局势



单一的局势会导致训练变慢：

额外的概念

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(R_t^n - B(s_n^t))\nabla logP_\theta(a_n^t|s_n^t)$$

Action-Value Function

R_t^n每次都是一次随机采样，方差很大，训练不稳定（具体采样）

$Q_\theta(s,a)$在State s下，做出Action a，期望的回报。动作价值函数

State-Value Function

$V_\theta(s)$在State s下，期望的回报。状态价值函数

Advantage Function

$A_\theta(s,a)=Q_\theta(s,a)-V_\theta(s)$ 在State S下，做出Action a,比其他动作能带来多少优势

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}A_\theta(s_n^t,a_n^t)\nabla logP_\theta(a_n^t|s_n^t)$$

额外的概念

$$A_\theta(s,a) = Q_\theta(s,a) - V_\theta(s)$$

$Q_\theta(s,a)$在State S下，做出Action a, 期望的回报。动作价值函数。

动作价值函数和状态价值函数的关系

$Q_\theta(s,a) = r_t + \gamma * V_\theta(s_{t+1})$ # 当前的Reward + 这个动作的衰减状态价值

$A_\theta(s_t,a) = r_t + \gamma * V_\theta(s_{t+1}) - V_\theta(s_t)$ # 只需要训练状态价值函数

采样状态价值函数

$V_\theta(s_{t+1}) \approx r_{t+1} + \gamma * V_\theta(s_{t+2})$

$$A_\theta^1(s_t,a) = r_t + \gamma * V_\theta(s_{t+1}) - V_\theta(s_t)$$

$$A_\theta^2(s_t,a) = r_t + \gamma * r_{t+1} + \gamma^2 * V_\theta(s_{t+2}) - V_\theta(s_t)$$

$$A_\theta^3(s_t,a) = r_t + \gamma * r_{t+1} + \gamma^2 * r_{t+2} + \gamma^3 V_\theta(s_{t+3}) - V_\theta(s_t)$$

$$\vdots$$

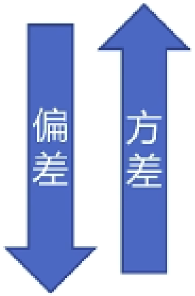
$$A_\theta^T(s_t,a) = r_t + \gamma * r_{t+1} + \gamma^2 * r_{t+2} + \gamma^3 * r_{t+3} + \dots + \gamma^T * r_T - V_\theta(s_t)$$

额外的概念

$$A_\theta^1(s_t, a) = r_t + \gamma * V_\theta(s_{t+1}) - V_\theta(s_t)$$

$$A_\theta^2(s_t, a) = r_t + \gamma * r_{t+1} + \gamma^2 * V_\theta(s_{t+2}) - V_\theta(s_t)$$

$$A_\theta^3(s_t, a) = r_t + \gamma * r_{t+1} + \gamma^2 * r_{t+2} + \gamma^3 V_\theta(s_{t+3}) - V_\theta(s_t)$$



$$\vdots$$

$$A_\theta^T(s_t, a) = r_t + \gamma * r_{t+1} + \gamma^2 * r_{t+2} + \gamma^3 * r_{t+3} + \cdots + \gamma^T * r_T - V_\theta(s_t)$$

替换法

$$\delta_t^V = r_t + \gamma * V_\theta(s_{t+1}) - V_\theta(s_t)$$

$$\delta_{t+1}^V = r_{t+1} + \gamma * V_\theta(s_{t+2}) - V_\theta(s_{t+1})$$

$$A_\theta^1(s_t, a) = \delta_t^V$$

$$A_\theta^2(s_t, a) = \delta_t^V + \gamma\delta_{t+1}^V$$

$$A_\theta^3(s_t, a) = \delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V$$

优势函数Generalized Advantage Estimation

替换法

TD-Error (时间差分误差Temporal Difference Error)

$$\delta_t^V = r_t + \gamma * V_\theta(s_{t+1}) - V_\theta(s_t)$$

$A_\theta^1(s_t, a) = \delta_t^V$ ： 一步优势(1-step Advantage) 方差低、偏差高

$A_\theta^2(s_t, a) = \delta_t^V + \gamma\delta_{t+1}^V$

$A_\theta^3(s_t, a) = \delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V$ ： 多步优势： 偏差-方差权衡

GAE优势函数 Advantage Function

1． 短期准确性

2． 长期远见

λ:超参(0~1)之间

$$A_\theta^{GAE}(s_t, a) = (1 - \lambda)(A_\theta^1 + \lambda A_\theta^2 + \ldots)$$

如果$\lambda = 0$． 回退到一步TD-Error （高偏差， 低方差）

如果$\lambda = 1$. $A_\theta^{GAE}(s_t, a) = A_\theta^1 + A_\theta^2 + \ldots$（蒙特卡洛估计）． 从头到尾的所有步数奖励

如果$\lambda = 0.9$． $A_\theta^{GAE}(s_t, a) = 0.1A_\theta^1 + 0.09A_\theta^2 + \ldots$： 我们主要相信短期的优势估计

优势函数Generalized Advantage Estimation

GAE优势函数 Advantage Function

λ:超参(0~1)之间

$$A_\theta^{GAE}(s_t,a) = (1-\lambda)\left(A_\theta^1 + \lambda A_\theta^2 + \ldots\right)$$

$$= (1-\lambda)(\delta_t^v + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V)\ldots)$$

$$= (1-\lambda)\left(\delta_t^v(1+\lambda+\lambda^2+\ldots) + \gamma\delta_{t+1}^v(\lambda+\lambda^2+\ldots)\right)$$

$$= (1-\lambda)\left(\delta_t^v\frac{1}{1-\lambda} + \gamma\delta_{t+1}^v\frac{\lambda}{1-\lambda} + \ldots\right)$$

$$= \sum_{b=0}^{\infty}(\lambda\gamma)^b\delta_{t+b}^v$$

总结：在状态s_t时候，做动作a的优势，并且平和采样不同步导致的偏差和方差的问题

总结数学公式

GAE优势函数 Advantage Function

$$A_\theta^{GAE}(s_t, a) = \sum_{b=0}^{\infty} (\lambda\gamma)^b \delta_{t+b}^{v}$$

TD-Error 时间差分误差Temporal Difference Error

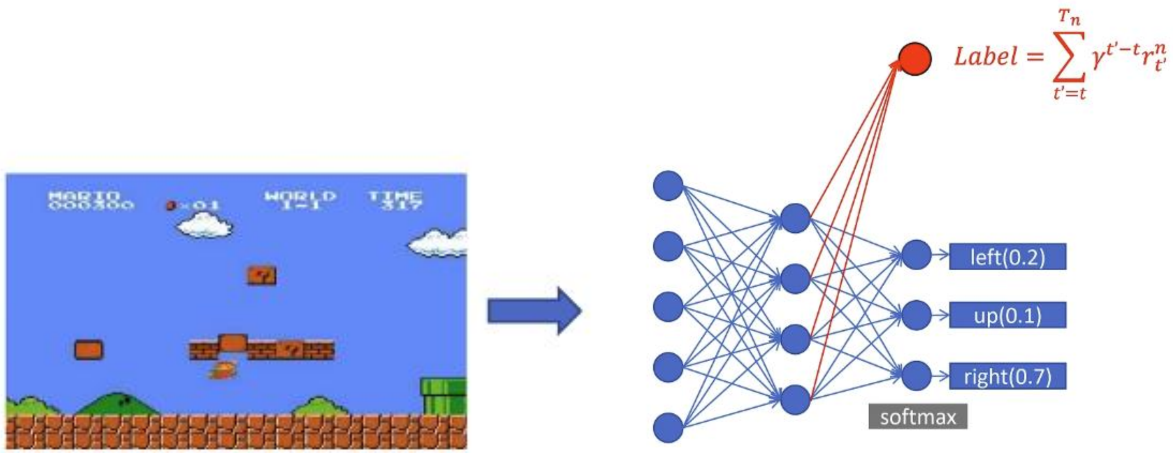
$$\delta_t^V = r_t + \gamma * V_\theta(s_{t+1}) - \boxed{V_\theta(s_t)}$$

梯度策略(Policy Gradient)算法更新规则

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_\theta(s_n^t, a_n^t) \nabla log P_\theta(a_n^t | s_n^t)$$

$$\underbrace{A_\theta^{GAE}(s_n^t, a_n^t)}_{\text{这步动作有多好?（幅度）}} \times \underbrace{\nabla \log P_\theta(a_n^t | s_n^t)}_{\text{如何让这步动作更多/少出现?（方向）}}$$

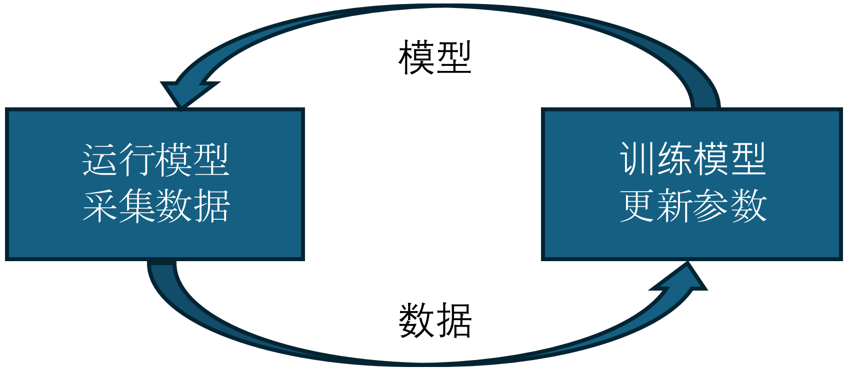
状态价值函数



关键点：这个 Label 必须在一局游戏结束后才能计算出来，因为它需要用到未来的所有奖励信息

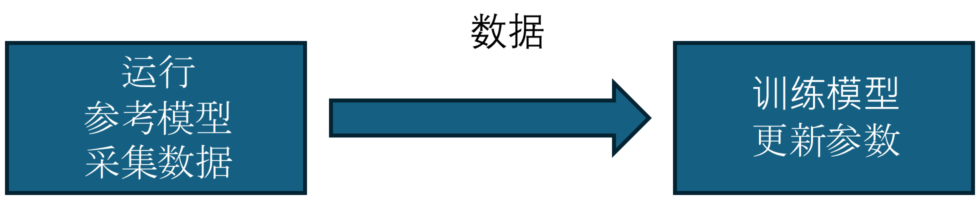
Proximal Policy Optimization
邻近策略优化

为什么要用PPO



On Policy

在线策略



Off Policy

离线策略（模仿学习）

- **On-Policy (自己开车，边开边学)**
  - 你亲自坐上驾驶座，手握方向盘，开始在路上练习。
  - 你转弯转得太急了，车子差点撞到马路牙子。你感受到了这次失误，于是**立刻调整自己的驾驶策略**："下次转弯要慢一点，方向盘打得柔和一点"。
  - 你只能从**你当前正在使用的、独一无二的驾驶技术**所产生的经验中学习。你不能用你朋友昨天开车的录像来指导你今天的操作。
  - **核心思想**：学习者和行动者是同一个人，从自己当前的行动中学习。
- **Off-Policy (看别人开车的视频学习)**
  - 你坐在家里，观看大量专业赛车手（或者你朋友）开车的视频录像。这些录像就是"经验"。
  - 你观察到赛车手在某个弯道用了某种漂亮的漂移技巧。你**根据他的行为和结果，来更新你脑海中"理想的"驾驶策略**。
  - 重要的是，你学习所用的数据（视频），并不是由你当前的驾驶技术产生的。你可能还是个新手，但你正在从专家的经验中学习。
  - **核心思想**：学习者和行动者可以不是同一个人。你可以利用**任何来源的经验**（过去的自己、其他智能体、人类专家等）来优化你自己的策略。

重要性采样Off Policy

比喻：在错误的地方钓鱼

想象一下，你的目标是**估算A湖里所有鱼的平均重量**（这是你的目标）。

但是，你因为某些原因，只能在旁边的B湖里钓鱼。你在B湖钓了一整天，钓上来100条鱼，并计算了它们的平均重量。

问题：用B湖鱼的平均重量，来代表A湖鱼的平均重量，这显然是错误的，对吧？可能B湖里都是小鱼，而A湖里都是大鱼。

重要性采样就是修正这个偏差的方法。

你要怎么做呢？你需要一个“修正系数”。如果你知道一些额外信息，比如：

- 在B湖钓到鲤鱼的概率是50%，但在A湖只有10%。
- 在B湖钓到鲈鱼的概率是10%，但在A湖有40%。

当你从B湖钓上一条鲤鱼时，你心里会想：“这条鲤鱼在我的目标A湖里其实是比较稀有的”。所以，在计算平均重量时，你应该**降低**这条鲤鱼的“权重”。
当你从B湖钓上一条鲈鱼时，你会想：“这条鲈鱼在我的目标A湖里很常见”。所以，你应该**提高**这条鲈鱼的“权重”。

这个“权重”就是**重要性采样权重**。它的计算方法是：

$$\text{权重} = \frac{\text{在目标 A 湖中钓到这种鱼的概率}}{\text{在实际 B 湖中钓到这种鱼的概率}}$$

通过给每一条从B湖钓上来的鱼乘以这个权重，你就能更准确地估算出A湖鱼的平均重量了，尽管你一条A湖的鱼都没钓到！

$$
\begin{aligned}
\mathrm{E}(f(x))_{x\sim p(x)} &= \sum_x f(x) * p(x) \\
&= \sum_x f(x) * p(x)\frac{q(x)}{q(x)} \\
&= \sum_x f(x)\frac{p(x)}{q(x)} * q(x) \\
&= \mathrm{E}\left(f(x)\frac{p(x)}{q(x)}\right)_{x\sim q(x)} \\
&\approx \frac{1}{N}\sum_{n=1}^{N} f(x)\frac{p(x)}{q(x)}_{x\sim q(x)}
\end{aligned}
$$

p(x)： A池塘的分布概率

q(x)： B池塘的分布概率

重要性采样Off Policy

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_{\theta}^{GAE}(s_n^t, a_n^t)\nabla \log P_\theta(a_n^t|s_n^t)$$

$$\nabla \log f(x) = \frac{\nabla f(x)}{f(x)}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_{\theta'}^{GAE}(s_n^t, a_n^t)\frac{P_\theta(a_n^t|s_n^t)}{P_{\theta'}(a_n^t|s_n^t)}\nabla \log P_\theta(a_n^t|s_n^t)$$

$A_{\theta'}^{GAE}$是对另一个Agent的优势函数

$P_{\theta'}$ 是另一个Agent做动作的概率值

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_{\theta'}^{GAE}(s_n^t, a_n^t)\frac{\cancel{P_\theta(a_n^t|s_n^t)}}{P_{\theta'}(a_n^t|s_n^t)}\frac{\nabla P_\theta(a_n^t|s_n^t)}{\cancel{P_\theta(a_n^t|s_n^t)}}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_{\theta'}^{GAE}(s_n^t, a_n^t)\frac{\nabla P_\theta(a_n^t|s_n^t)}{P_{\theta'}(a_n^t|s_n^t)}$$

$$\text{Loss} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_{\theta'}^{GAE}(s_n^t, a_n^t)\frac{P_\theta(a_n^t|s_n^t)}{P_{\theta'}(a_n^t|s_n^t)}$$

PPO的缺点

这个学生不能和你差距太大。

不然你很难学到对你有用的经验和教训。

PPO的创新点 – 修改Off Policy造成的影响

$$Loss_{ppo} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_{\theta'}^{GAE}(s_n^t, a_n^t)\frac{P_\theta(a_n^t|s_n^t)}{P_{\theta'}(a_n^t|s_n^t)} + \beta KL(P_\theta, P_{\theta'})$$

β： KL散度约束大小

KL函数： 如果P差异相同则为0,越大则影响越大

$$Loss_{ppo2} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} min(A_{\theta'}^{GAE}(s_n^t, a_n^t)\frac{P_\theta(a_n^t|s_n^t)}{P_{\theta'}(a_n^t|s_n^t)}, clip(\frac{P_\theta(a_n^t|s_n^t)}{P_{\theta'}(a_n^t|s_n^t)}, 1-\varepsilon, 1+\varepsilon)A_{\theta'}^{GAE}(s_n^t, a_n^t))$$

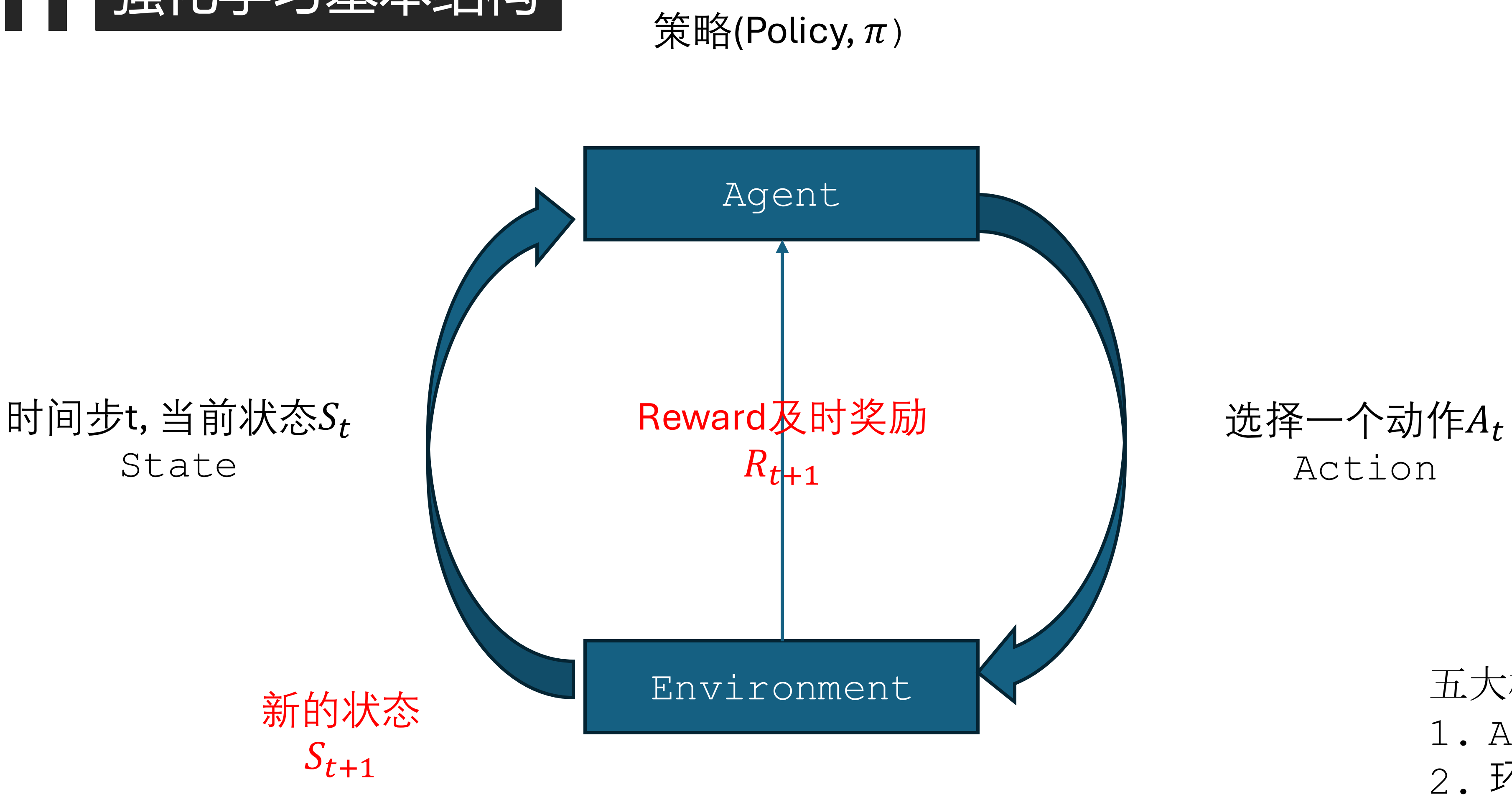
截断函数来替代KL散度函数：

防止训练策略和参考策略偏差过大

clip： 保证P_θ和P_θ' 差距不是过大

总结

强化学习基本结构



五大核心要素

1. Agent
2. 环境Enviroment
3. 状态State
4. 动作Action
5. 奖励Reward

强化学习难点

特性	监督学习(Supervised Learning)	强化学习(Reinforcement Learning)	困难所在
反馈信号	直接、即使、明确(有标准答案)	间接、延迟、稀疏（只有好坏评价)	信用分配问题，不知道哪个动作是关键
数据来源	给定的静态数据集	通过与环境交互态生成	需要平衡探索与利用
数据分布	独立同分布	序列化、高度相关	训练不稳定，优化算法容易失效
学习目标	学习一个映射函数f(x) -> y	学习一个最优策略 π(s) -> a	目标是最大化长期累积奖励，而非单步准确率
评估与调试	简单直接(用测试集评估准确率)	困难复杂 (奖励曲线抖动大，难以判断好坏)	调试困难，失败原因难以定位