一、 策略学习

策略学习的意思是通过求解一个优化问题，学出最优策略函数或它的近似。

1.1 策略网络

对于离散动作空间来说，策略函数 π 是个条件概率质量函数:

$$\pi(a|s) \triangleq \mathbb{P}(A = a \mid S = s)$$

为了得到这样要给策略函数，当前最有效的方法是通过神经网络 $\pi(a|s;\theta)$ 近似策略函数 $\pi(a|s)$。神经网络 $\pi(a|s;\theta)$ 被称为策略网络。

1.2 策略学习的目标函数

我们有动作值函数

$$Q_\pi(s_t, a_t) = \mathbb{E}[U_t \mid S_t = s_t, A_t = a_t]$$

则也有状态值函数

$$V_\pi(s_t) = \mathbb{E}_{A_t \sim \pi(\cdot|s_t;\theta)}[Q_\pi(s_t, A_t)]$$

如果一个策略很好，那么对所有的状态 S，状态价值 $V_\pi(S)$ 的均值应该很大，因此我们可以定义目标函数

$$J(\theta) = \mathbb{E}_S[V_\pi(S)]$$

这个目标函数排除了状态 S 的影响，只依赖于策略网络 π 的参数 θ，因此策略学习可以描述为优化问题

$$\max_\theta J(\theta)$$

我们使用梯度上升可得

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} + \beta \cdot \nabla_\theta J(\theta_{\text{now}})$$

其中梯度

$$\nabla_\theta J(\theta_{\text{now}}) \triangleq \frac{\partial J(\theta)}{\partial \theta}\Big|_{\theta=\theta_{\text{now}}}$$

被称为策略梯度，我们可以使用 **策略梯度定理**:

策略梯度定理（不严谨表述）

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\theta)}\left[\frac{\partial \ln \pi(A|S;\theta)}{\partial \theta}\cdot Q_\pi(S,A)\right]\right]$$

更严格的表述是

策略梯度定理（完整表述）

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1-\gamma^n}{1-\gamma}\mathbb{E}_{S\sim d(\cdot)}\left[\mathbb{E}_{A\sim\pi(\cdot|S;\theta)}\left[\frac{\partial \ln \pi(A|S;\theta)}{\partial \theta}\cdot Q_\pi(S,A)\right]\right]$$

其中 $d(\cdot)$ 是马尔科夫链的稳态分布，而系数 $\frac{1-\gamma^n}{1-\gamma}$ 无关紧要，因为会被学习率 β 吸收掉。

1.3 近似策略梯度

为了进行梯度上升：

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} + \beta\cdot\nabla_\theta J(\theta_{\text{now}})$$

策略梯度定理 证明：

$$\nabla_\theta J(\theta) = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\theta)}\left[\frac{\partial \ln \pi(A|S;\theta)}{\partial \theta}\cdot Q_\pi(S,A)\right]\right]$$

解析求出这个期望是不可能的，因为我们并不知道状态 S 概率密度函数，即使我们知道，我们也不愿意这样做，因为连加或定积分的计算量非常大。

我们可以使用期望的蒙特卡洛近似，用于近似策略梯度中的期望。每次从环境中观测一个状态 s，它相当于随机变量 S 的观测值，然后再根据当前最新的策略网络随机抽样得出一个动作：

$$a\sim\pi(\cdot\,|s;\theta)$$

计算得到随机梯度

$$g(s,a;\theta) \triangleq Q_\pi(s,a)\cdot\nabla_\theta \ln \pi(a|s;\theta)$$

很显然，$g(s,a;\theta)$ 是策略梯度 $\nabla_\theta J(\theta)$ 的无偏估计，于是我们有结论

结论

随机梯度 $g(s,a;\theta) \triangleq Q_\pi(s,a)\cdot\nabla_\theta \ln \pi(a|s;\theta)$ 是策略梯度 $\nabla_\theta J(\theta)$ 的无偏估计。

应用上述结论，则可以通过随机梯度上升来更新 θ:

$$\theta \leftarrow \theta + \beta \cdot g(s, a; \theta)$$

但是这种方法仍然不可行，因为我们不知道动作价值函数 $Q_\pi(s, a)$。在后面，我们可以有两种方法对 $Q_\pi(s, a)$ 近似，一种是 REINFORCE，用实际观测的回报 u 近似 $Q_\pi(s, a)$；另一种方法是 Actor-Critic，用神经网络 $q(s, a; w)$ 近似 $Q_\pi(s, a)$。

1.4 REINFORCE

由于动作价值定义为 U_t 的条件期望

$$Q_\pi(s_t, a_t) = \mathbb{E}[U_t \mid S_t = s_t, A_t = a_t]$$

让智能体完成一局游戏，观测到所有奖励，然后计算出 $u_t = \sum_{k=t}^{n} \gamma^{k-t} \cdot r_k$。由于 u_t 是 U_t 的观测值，所以是该公式的蒙特卡洛近似。实践中，可以使用 u_t 代替 $Q_\pi(s_t, a_t)$，那么随机梯度 $g(s_t, a_t; \theta)$ 可以近似成

$$\tilde{g}(s_t, a_t; \theta) = u_t \cdot \nabla_\theta \ln \pi(a_t | s_t; \theta)$$

$\tilde{g}$ 是 g 的无偏估计，所以也是策略梯度 $\nabla_\theta J(\theta)$ 的无偏估计；$\tilde{g}$ 也是一种随机梯度。

这种方法就叫 REINFORCE。

训练流程:

1. 用策略网络 θ_{now} 控制智能体从头玩一局游戏，得到一条轨迹：

$$s_1, a_1, r_1, \quad , s_2, a_2, r_2, \quad \cdots, \quad , s_n, a_n, r_n$$

2. 计算所有的回报：

$$u_t = \sum_{k=t}^{n} \gamma^{k-t} \cdot r_k, \quad \forall t = 1, \cdots, n$$

3. 用 $\{(s_t, a_t)\}_{t=1}^{n}$ 作为数据，做反向传播计算：

$$\nabla_\theta \ln \pi(a_t | s_t; \theta_{\text{now}}), \quad \forall t = 1, \cdots, n$$

4. 做随机梯度上升更新策略网络参数：

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} + \beta \cdot \sum_{t=1}^{n} \gamma^{t-1} \cdot \underbrace{u_t \cdot \nabla_\theta \ln \pi(a_t | s_t; \theta_{\text{now}})}_{\text{即随机梯度 } \tilde{g}(s_t, a_t; \theta_{\text{now}})}$$

注意，REINFORCE 是一种 **同策略** 方法，要求行为策略与目标策略相同，因此不适用经验回放。

1.5 Actor-Critic

Actor-Critic 方法使用神经网络近似动作价值函数 $Q_\pi(s,a)$，这个神经网络叫做价值网络，记作 $q(s,a;w)$。

价值网络与 DQN 的区别：

- 价值网络是对动作价值函数 $Q_\pi(s,a)$ 的近似，而 DQN 则是对最优动作价值函数 $Q_\star(s,a)$ 的近似。
- 对价值网络的训练使用的是 SARSA 算法，是同策略算法，不能用经验回放；而 DQN 训练使用的是 Q 学习算法，属于异策略，可以用经验回放。

Actor-Critic 翻译成「演员—评委」方法。策略网络 $\pi(a|s;\theta)$ 相当于演员，它基于状态 s 做出动作 a。价值网络 $q(s,a;w)$ 相当于评委，它给演员打分，量化在状态 s 下做动作 a 的好坏程度。

为什么我们不能直接将当前奖励 R 反馈给策略网络（演员），而是要用价值网络（评委）这一个中介呢？这是因为策略学习的目标函数 $J(\theta)$ 是回报 U 的期望，而不是奖励 R 的期望。虽然我们能观测到当前奖励 R，但是它对策略网络毫无意义。而我们使用 R 帮忙训练价值网络，价值网络经过训练后能够估算出回报 U 的期望。

而加入中介价值网络（评委）的好处是不再需要像 REINFORCE 一样要完成一整局游戏后才能进行一次更新。

将之前的策略梯度无偏估计：

$$g(s,a;\theta) \triangleq Q_\pi(s,a) \cdot \nabla_\theta \ln \pi(a|s;\theta)$$

里的动作价值函数 $Q_\pi(s,a)$ 替换成价值网络即可得到近似策略梯度：

$$\hat{g}(s,a;\theta) \triangleq q(s,a;w) \cdot \nabla_\theta \ln \pi(a|s;\theta)$$

最后做梯度上升即可。

训练流程（使用 SARSA 训练，含目标网络）：

1. 观测到当前状态 s_t，根据策略网络做决策：$a_t \sim \pi(\cdot\,|s_t;\theta_{\text{now}})$，并让智能体执行动作 a_t。
2. 从环境中观测到奖励 r_t 和新状态 s_{t+1}。
3. 根据策略网络做决策：$\tilde{a}_{t+1} \sim \pi(\cdot\,|s_{t+1};\theta_{\text{now}})$，但不让智能体执行动作 $\tilde{a}_{t+1}$。
4. 让价值网络给 s_t, a_t 打分：

$$\hat{q}_t = q(s_t,a_t;w_{\text{now}}).$$

5. 让目标网络给 $s_{t+1}, \tilde{a}_{t+1}$ 打分:

$$\hat{q}_{t+1}^{-} = q(s_{t+1}, \tilde{a}_{t+1}; w_{\text{now}}^{-}).$$

6. 计算 TD 目标与 TD 误差:

$$\hat{y}_t^{-} = r_t + \gamma \cdot \hat{q}_{t+1}^{-} \quad \text{和} \quad \delta_t = \hat{q}_t - \hat{y}_t^{-}.$$

7. 更新价值网络:

$$w_{\text{new}} \leftarrow w_{\text{now}} - \alpha \cdot \delta_t \cdot \nabla_w q(s_t, a_t; w_{\text{now}}).$$

8. 更新策略网络:

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} + \beta \cdot \hat{q}_t \cdot \nabla_\theta \ln \pi(a_t | s_t; \theta_{\text{now}}).$$

9. 设 $\tau \in (0,1)$ 是需要手动调的超参数。做加权平均更新目标网络的参数:

$$w_{\text{new}}^{-} \leftarrow \tau \cdot w_{\text{new}} + (1-\tau) \cdot w_{\text{now}}^{-}.$$

二、 带基线的策略梯度方法

带基线的策略梯度（Policy Gradient with Baseline）可以大幅提升策略梯度方法的表现。使用基线后，REINFORCE 变成 REINFORCE with Baseline，Actor-Critic 变成 Advantage Actor-Critic（A2C）。

2.1 策略梯度中的基线

策略梯度定理 证明:

$$\nabla_\theta J(\theta) = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\theta)}[Q_\pi(S,A) \cdot \nabla_\theta \ln \pi(A|S;\theta)]\right]$$

我们只需要做一个小改动，就能大幅提升表现：把 b 作为动作价值函数 $Q_\pi(S,A)$ 的基线（Baseline），用 $Q_\pi(S,A) - b$ 替换掉 Q_π。设 b 是任意的函数，只要不依赖于动作 A 即可；例如 b 可以是状态价值函数 $V_\pi(S)$。

带基线的策略梯度定理

$$\nabla_\theta J(\theta) = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\theta)}[(Q_\pi(S,A) - b) \cdot \nabla_\theta \ln \pi(A|S;\theta)]\right]$$

其原因在于

$$\mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|S;\theta)}[b \cdot \nabla_\theta \ln \pi(A|S;\theta)]\right] = 0$$

证明

$$\begin{aligned}
\mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[b\cdot\frac{\partial\ln\pi(A|s;\theta)}{\partial\theta}\right] &= b\cdot\mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left[\frac{\partial\ln\pi(A|s;\theta)}{\partial\theta}\right] \\
&= b\cdot\sum_{a\in\mathcal{A}}\pi(a|s;\theta)\cdot\frac{\partial\ln\pi(a|s;\theta)}{\partial\theta} \\
&= b\cdot\sum_{a\in\mathcal{A}}\pi(a|s;\theta)\cdot\frac{1}{\pi(a|s;\theta)}\cdot\frac{\partial\pi(a|s;\theta)}{\partial\theta} \\
&= b\cdot\sum_{a\in\mathcal{A}}\frac{\partial}{\partial\theta}\pi(a|s;\theta) \\
&= b\cdot\frac{\partial}{\partial\theta}\sum_{a\in\mathcal{A}}\pi(a|s;\theta) \\
&= b\cdot\frac{\partial 1}{\partial\theta} = 0
\end{aligned}$$

使用蒙特卡洛近似，从环境中观测一个状态 s，然后根据策略网络抽样得到 $a\sim\pi(\cdot\,|s;\theta)$。那么策略梯度 $\nabla_\theta J(\theta)$ 可以近似为下面的随机梯度：

$$g_{b(s,a;\theta)} = [Q_\pi(s,a)-b]\cdot\nabla_\theta\ln\pi(a|s;\theta)$$

得到的随机梯度 $g_{b(s,a;\theta)}$ 是 $\nabla_\theta J(\theta)$ 的无偏估计：

$$\text{Bias} = \mathbb{E}_{S,A}\left[g_{b(S,A;\theta)}\right]-\nabla_\theta J(\theta) = 0$$

但是 b 的取值会对方差有影响，方差为

$$\text{Var} = \mathbb{E}_{S,A}\left[\left\|g_b(S,A;\theta)-\nabla_\theta J(\theta)\right\|^2\right].$$

如果 b 很接近 $Q_\pi(s,a)$ 关于 a 的均值，那么方差会比较小，所以 $b=V_\pi(s)$ 是很好的基线。

基线解释

为什么我们能减去基线呢？这是因为我们在意的是动作价值函数各个动作对应值的相对大小，如果都减去基线，相对大小是不变的，这是我们能够使用基线的原因。

至于为什么使用了基线能提高策略梯度算法效果？我还得再看看。

2.2 带基线的 REINFORCE 算法

REINFORCE 使用实际观测的回报 u 来代替动作价值 $Q_\pi(s,a)$。为了使用基线，我们还需要一个神经网络 $v(s;w)$ 近似状态价值函数 $V_\pi(s)$。这样一来，$g(s,a;\theta)$ 就被近似成了：

$$\tilde{g}(s,a;\theta) = [u - v(s;w)] \cdot \nabla_\theta \ln \pi(a|s;\theta)$$

2.2.1 策略网络和价值网络

带基线的 REINFORCE 需要两个神经网络：策略网络 $\pi(a|s;\theta)$ 和价值网络 $v(s;w)$；$v(s;w)$ 的输入是状态 s，输出是一个实数。策略网络和价值网络输入都是状态 s，因此可以让两个神经网络共享 **卷积网络** 的参数，这是编程实现中常用的技巧。

注意，这里的价值网络并不是用作评委，这里与 A2C 是不同的。

2.2.2 训练流程

1. 用策略网络 θ_{now} 控制智能体从头玩一局游戏，得到一条轨迹：

$$s_1, a_1, r_1, \quad , s_2, a_2, r_2, \quad \cdots, \quad , s_n, a_n, r_n$$

2. 计算所有的回报：

$$u_t = \sum_{k=t}^{n} \gamma^{k-t} \cdot r_k, \quad \forall t = 1, \cdots, n$$

3. 让价值网络做预测：

$$\hat{v}_t = v(s_t; w_{\text{now}}), \quad \forall t = 1, \cdots, n.$$

4. 计算误差 $\delta_t = \hat{v}_t - u_t, \quad \forall t = 1, \cdots, n$。

5. 用 $\{s_t\}_{t=1}^n$ 作为价值网络输入，做反向传播计算：

$$\nabla_w v(s_t; w_{\text{now}}), \quad \forall t = 1, \cdots, n$$

6. 用 $\{(s_t, a_t)\}_{t=1}^n$ 作为数据，做反向传播计算：

$$\nabla_\theta \ln \pi(a_t|s_t; \theta_{\text{now}}), \quad \forall t = 1, \cdots, n$$

7. 做随机梯度上升更新策略网络参数：

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} - \beta \cdot \sum_{t=1}^{n} \gamma^{t-1} \cdot \underbrace{\delta_t \cdot \nabla_\theta \ln \pi(a_t|s_t; \theta_{\text{now}})}_{\text{负的近似梯度 } -\tilde{g}(s_t, a_t; \theta_{\text{now}})}$$

2.3 Advantage Actor-Critic（A2C）

我们之前得到的策略梯度的无偏估计

$$g(s,a;\theta)=\Bigg[\underbrace{Q_\pi(s,a)-V_\pi(s)}_{\text{优势函数}}\Bigg]\cdot\nabla_\theta\ln\pi(a|s;\theta)$$

公式中的 $Q_\pi - V_\pi$ 被称为优势函数（Advantage Function）。因此，基于上面公式得到的 Actor-Critic 方法被称为 Advantage Actor-Critic，缩写 A2C。

这种方法也有一个价值网络 $v(s;w)$，在这里是作为评委，他的评分可以帮助策略网络（演员）改进技术。这里的神经网络结构与上一节相同，但是训练方法不同。

可以注意到，这里也和不带基线的 Actor-Critic 方法不同，这里没有用到动作价值网络 $q(s,a;w)$，而是使用了价值网络 $v(s;w)$，而前者可以通过贝尔曼公式得到。

2.3.1 算法推导

训练价值网络:

根据贝尔曼公式:

$$V_\pi(s_t)=\mathbb{E}_{A_t\sim\pi(\cdot|s_t;\theta)}\Big[\mathbb{E}_{S_{t+1}\sim p(\cdot|s_t,A_t)}\big[R_t+\gamma\cdot V_\pi(S_{t+1})\big]\Big].$$

我们对两边做近似:

- 方程左边的 $V_\pi(s_t)$ 近似成 $\hat{v}_t \triangleq v(s_t;w)$，这是价值网络在 t 时刻对 $V_\pi(s_t)$ 做出的估计。。
- 方程右边的期望，给定 s_t，智能体执行动作 a_t，环境给出奖励 r_t 和新状态 s_{t+1}，然后蒙特卡洛近似得到 TD 目标:

$$\hat{y}_t \triangleq r_t+\gamma\cdot v(s_{t+1};w)$$

这是价值网络在 $t+1$ 时刻对 $V_\pi(s_t)$ 做出的估计。

之后便是照常地定义均方损失函数与求梯度，然后做梯度下降。

训练策略网络:

为了对 $g(s,a;\theta)$ 做近似，我们继续使用贝尔曼公式:

$$Q_\pi(s_t,a_t)=\mathbb{E}_{S_{t+1}\sim p(\cdot|s_t,a_t)}\big[R_t+\gamma\cdot V_\pi(S_{t+1})\big].$$

因此使用蒙特卡洛近似可得

$$\tilde{g}(s_t, a_t; \theta) \triangleq \left[\underbrace{r_t + \gamma \cdot v(s_{t+1}; w)}_{\text{TD 误差 } \hat{y}_t} - v(s_t; w) \right] \cdot \nabla_\theta \ln \pi(a_t | s_t; \theta)$$

再根据 TD 目标和 TD 误差:

$$\hat{y}_t \triangleq r_t + \gamma \cdot v(s_{t+1}; w) \quad \text{和} \quad \delta_t \triangleq v(s_t; w) - \hat{y}_t$$

可得

$$\tilde{g}(s_t, a_t; \theta) \triangleq -\delta_t \cdot \nabla_\theta \ln \pi(a_t | s_t; \theta)$$

为什么价值网络不显示包含 a_t 也可以帮助策略网络（演员）改进演技?

价值网络 v 告诉策略网络 π 的唯一信息是 δ_t，根据其定义

$$-\delta_t = \underbrace{r_t + \gamma \cdot v(s_{t+1}; w)}_{\text{TD 目标 } \hat{y}_t} - \underbrace{v(s_t; w)}_{\text{基线}}$$

有基线 $v(s_t; w)$ 是价值网络在 t 时刻对 $\mathbb{E}[U_t]$ 的估计，此时未执行动作 a_t；其 TD 目标 $\hat{y}_t$ 是价值网络在 $t+1$ 时刻对 $\mathbb{E}[U_t]$ 的估计，此时已执行动作 a_t。

- 如果 $\hat{y}_t > v(s_t; w)$, 说明动作 a_t 很好, 奖励 r_t 超出预期, 这时候应该更新 θ 使 $\pi(a_t|s_t;\theta)$ 变大。
- 如果 $\hat{y}_t < v(s_t; w)$, 说明动作 a_t 不好, 奖励 r_t 不及预期, 这时候应该更新 θ 使 $\pi(a_t|s_t;\theta)$ 减小。

这说明 δ_t 可以间接反映 a_t 的好坏。

2.3.2 训练流程（含目标网络）

1. 观测到当前状态 s_t，根据策略网络做决策：$a_t \sim \pi(\cdot\,|s_t; \theta_{\text{now}})$，并让智能体执行动作 a_t。
2. 从环境中观测到奖励 r_t 和新状态 s_{t+1}。
3. 让价值网络给 s_t 打分:

$$\hat{v}_t = v(s_t; w_{\text{now}}).$$

4. 让目标网络给 s_{t+1} 打分:

$$\hat{v}_{t+1}^- = v(s_{t+1}; w_{\text{now}}^-).$$

5. 计算 TD 目标与 TD 误差:

$$\widehat{y}_t^- = r_t + \gamma \cdot \widehat{v}_{t+1}^- \quad 和 \quad \delta_t = \widehat{v}_t - \widehat{y}_t^-.$$

6. 更新价值网络:

$$w_{\text{new}} \leftarrow w_{\text{now}} - \alpha \cdot \delta_t \cdot \nabla_w v(s_t; w_{\text{now}}).$$

7. 更新策略网络:

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} - \beta \cdot \delta_t \cdot \nabla_\theta \ln \pi(a_t | s_t; \theta_{\text{now}}).$$

8. 设 $\tau \in (0,1)$ 是需要手动调的超参数。做加权平均更新目标网络的参数:

$$w_{\text{new}}^- \leftarrow \tau \cdot w_{\text{new}} + (1-\tau) \cdot w_{\text{now}}^-.$$

三、 策略学习高级技巧

3.1 Trust Region Policy Optimization（TRPO）

置信域策略优化是一种策略学习方法，可以代替策略梯度方法。跟策略梯度方法相比，TRPO 有两个优势: 第一, TRPO 表现更稳定, 收敛曲线不会剧烈波动, 而且对学习率不敏感; 第二, TRPO 用更少的经验就能达到与策略梯度方法相同的表现。

学习 TRPO 的关键在于理解置信域方法（Trust Region Methods）。

3.1.1 置信域方法

置信域方法需要构造一个函数 $L(\theta|\theta_{\text{now}})$，这个函数要满足条件:

$$L(\theta|\theta_{\text{now}}) \text{ 很接近 } J(\theta), \quad \forall \theta \in \mathcal{N}(\theta_{\text{now}}),$$

那么集合 $\mathcal{N}(\theta_{\text{now}})$ 就被称作 **置信域**。顾名思义，在 θ_{now} 的邻域上，我们可以信任 $L(\theta|\theta_{\text{now}})$，可以拿 $L(\theta|\theta_{\text{now}})$ 来替代目标函数 $J(\theta)$。

因此置信域方法主要分两步:

- **第一步——做近似**: 给定 θ_{now}，构造函数 $L(\theta|\theta_{\text{now}})$，使得对于所有 $\theta \in \mathcal{N}(\theta_{\text{now}})$，函数值 $L(\theta|\theta_{\text{now}})$ 与 $J(\theta)$ 足够接近。
  - 近似的方法多种多样，如蒙特卡洛、二阶泰勒展开等。
- **第二步——最大化**: 在置信域 $\mathcal{N}(\theta_{\text{now}})$ 中寻找变量 θ 的值，使得函数 L 的值最大化。把找到的值记作

$$\theta_{\text{new}} - \underset{\theta \in \mathcal{N}(\theta_{\text{now}})}{\operatorname{argmax}} L(\theta \mid \theta_{\text{now}})$$

  - 需要求解一个带约束的最大化问题，求解这个问题的数值优化方法很多，如梯度投影算法、拉格朗日法等。

- 置信域 $\mathcal{N}(\theta_{\text{now}})$ 也有很多选择，既可以是球，也可以是两个概率分布的 KL 散度（KL Divergence）。

3.1.2 策略学习

状态价值函数可以转化为等价形式：

$$
\begin{aligned}
V_\pi(s) &= \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}[Q_\pi(s,A)] = \sum_{a\in\mathcal{A}} \pi(a|s;\theta)\cdot Q_\pi(s,a) \\
&= \sum_{a\in\mathcal{A}} \pi(a|s;\theta_{\text{now}})\cdot \frac{\pi(a|s;\theta)}{\pi(a|s;\theta_{\text{now}})}\cdot Q_\pi(s,a) \\
&= \mathbb{E}_{A\sim\pi(\cdot|s;\theta_{\text{now}})}\left[\frac{\pi(A|s;\theta)}{\pi(A|s;\theta_{\text{now}})}\cdot Q_\pi(s,A)\right]
\end{aligned}
$$

因此策略学习的目标函数 $J(\theta) = \mathbb{E}_S[V_\pi(S)]$ 也可以转化为等价形式。

目标函数等价形式

目标函数 $J(\theta)$ 可以等价写成：

$$
J(\theta) = \mathbb{E}_S\left[\mathbb{E}_{A\sim\pi(\cdot|s;\theta_{\text{now}})}\left[\frac{\pi(A|s;\theta)}{\pi(A|s;\theta_{\text{now}})}\cdot Q_\pi(s,A)\right]\right]
$$

上面 Q_π 中的 π 指的是 $\pi(A|S;\theta)$。

3.1.3 训练流程

1. 做近似——构造函数 $\tilde{L}$ 近似目标函数 $J(\theta)$：
   a. 设当前策略网络参数是 θ_{now}。用策略网络 $\pi(a|s;\theta_{\text{now}})$ 控制智能体与环境交互，玩完一局游戏，记下轨迹：

   $$
   s_1, a_1, r_1, \quad s_2, a_2, r_2, \quad \cdots, \quad s_n, a_n, r_n.
   $$

   b. 对于所有的 $t = 1, \cdots, n$，计算折扣回报 $u_t = \sum_{k=t}^{n} \gamma^{k-t}\cdot r_k$。

   c. 得出无偏的蒙特卡洛近似函数：

   $$
   \tilde{L}(\theta \mid \theta_{\text{now}}) = \sum_{t=1}^{n} \frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_{\text{now}})}\cdot u_t.
   $$

2. 最大化——用某种数值算法求解带约束的最大化问题：

   $$
   \theta_{\text{new}} = \underset{\theta}{\operatorname{argmax}}\, \tilde{L}(\theta|\theta_{\text{now}}); \quad \text{s.t.}\ \|\theta-\theta_{\text{now}}\|_2 \le \Delta.
   $$

   此处的约束时二范数距离，也可以替换成 KL 散度，即

$$\theta_{\text{new}} = \underset{\theta}{\operatorname{argmax}}\, \tilde{L}(\theta|\theta_{\text{now}}); \quad \text{s.t.}\ \frac{1}{t}\sum_{i=1}^{t} \mathrm{KL}\big[\pi(\cdot\,|s_i;\theta_{\text{now}}) \,\|\, \pi(\cdot\,|s_i;\theta)\big] \le \Delta.$$

由于 KL 散度能够衡量两个概率质量函数的接近程度，因此效果一般比球置信域的效果要好。

TRPO 有两个要调的超参数：一个是置信域的半径 Δ，另一个是求解最大化问题中数值算法的学习率。一般而言，Δ 在算法运行过程中要逐渐缩小。

3.2 熵正则

策略学习的目标是学习一个策略网络 $\pi(a|s;\theta)$ 用于控制智能体，我们希望策略网络的输出的概率不要集中在一个动作上，至少要给其他动作一些非零概率。我们可以用熵来衡量概率分布的不确定性，熵小代表概率质量集中，熵大说明随机性很大。

因此我们不妨把熵作为正则项，放在策略学习的目标函数中。策略网络的输出是维度等于 $|\mathcal{A}|$ 的向量，这是动作空间上的离散概率分布，这个概率分布的熵定义为：

$$H(s;\theta) \triangleq \mathrm{Entropy}[\pi(\cdot\,|s;\theta)] = -\sum_{a\in\mathcal{A}} \pi(a|s;\theta)\cdot \ln \pi(a|s;\theta).$$

因此加入正则项后最大化问题变为：

$$\max_{\theta} J(\theta) + \lambda \cdot \mathbb{E}_S[H(S;\theta)]$$

带熵正则的最大化问题可以用各种方法求解，其中使用策略梯度方法求关于 θ 的梯度是：

$$g(\theta) \triangleq \nabla_\theta [J(\theta) + \lambda \cdot \mathbb{E}_S[H(S;\theta)]].$$

观测到状态 s，按照策略网络做随机抽样，得到动作 $a \sim \pi(\cdot\,|s;\theta)$。那么

$$\tilde{g}(s,a;\theta) \triangleq [Q_\pi(s,a) - \lambda \cdot \ln \pi(a|s;\theta) - \lambda] \cdot \nabla_\theta \ln \pi(a|s;\theta)$$

是梯度 $g(\theta)$ 的无偏估计。