第九章 策略学习高级技巧

本章介绍策略学习的高级技巧。第 9.1 介绍置信域策略优化 (TRPO)，它是一种策略学习方法，可以代替策略梯度方法。第 9.2 介绍熵正则，可以用在所有的策略学习方法中。

9.1 Trust Region Policy Optimization (TRPO)

置信域策略优化 (Trust Region Policy Optimization, TRPO) 是一种策略学习方法，跟以前学的策略梯度有很多相似之处。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（即智能体收集到的状态、动作、奖励）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)。置信域方法不是 TRPO 的论文提出的，而是数值最优化领域中一类经典的算法，历史至少可以追溯到 1970 年。TRPO 论文的贡献在于巧妙地把置信域方法应用到强化学习中，取得非常好的效果。

本节分以下 4 小节讲解 TRPO：第 9.1.1 小节介绍置信域方法，第9.1.2节回顾策略学习，第9.1.3节推导 TRPO，第9.1.4讲解 TRPO 的算法流程。

9.1.1 置信域方法

有这样一个优化问题：$\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$。这里的 $J(\boldsymbol{\theta})$ 是目标函数，$\boldsymbol{\theta}$ 是优化变量。求解这个优化问题的目的是找到一个变量 $\boldsymbol{\theta}$ 使得目标函数 $J(\boldsymbol{\theta})$ 取得最大值。有各种各样的优化算法用于解决这个问题。几乎所有的数值优化算法都是做这样的迭代：

$$\boldsymbol{\theta}_{\text{new}} \;\leftarrow\; \text{Update}\Big(\text{Data};\ \boldsymbol{\theta}_{\text{now}}\Big).$$

此处的 $\boldsymbol{\theta}_{\text{now}}$ 和 $\boldsymbol{\theta}_{\text{new}}$ 分别是优化变量当前的值和新的值。不同算法的区别在于具体怎么样利用数据更新优化变量。

置信域方法用到一个概念——**置信域**。下面介绍置信域。给定变量当前的值 $\boldsymbol{\theta}_{\text{now}}$，用 $\mathcal{N}(\boldsymbol{\theta}_{\text{now}})$ 表示 $\boldsymbol{\theta}_{\text{now}}$ 的一个邻域。举个例子：

$$\mathcal{N}(\boldsymbol{\theta}_{\text{now}}) \;=\; \Big\{\boldsymbol{\theta}\;\Big|\;\big\|\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{now}}\big\|_2 \le \Delta\Big\}. \qquad (9.1)$$

这个例子中，集合 $\mathcal{N}(\boldsymbol{\theta}_{\text{now}})$ 是以 $\boldsymbol{\theta}_{\text{now}}$ 为球心、以 Δ 为半径的球；见右图。球中的点都足够接近 $\boldsymbol{\theta}_{\text{now}}$。

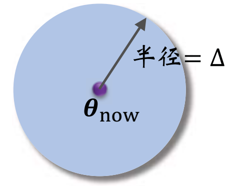


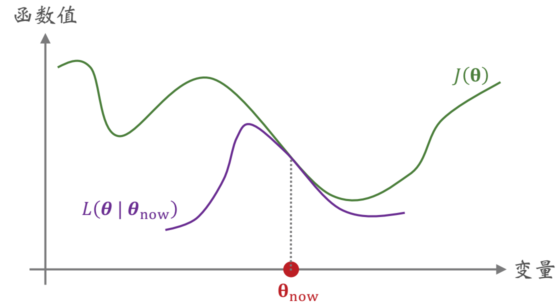
图 9.1: 公式(9.1)中的邻域 $\mathcal{N}(\boldsymbol{\theta}_{\text{now}})$。

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

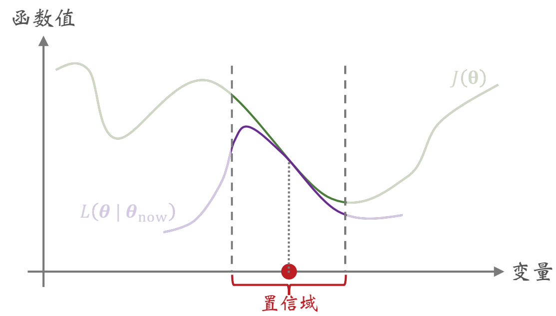
$$L\big(\boldsymbol{\theta}\,\big|\,\boldsymbol{\theta}_{\text{now}}\big)\ \text{很接近}\ J(\boldsymbol{\theta}), \qquad \forall\, \boldsymbol{\theta} \in \mathcal{N}(\boldsymbol{\theta}_{\text{now}}),$$

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

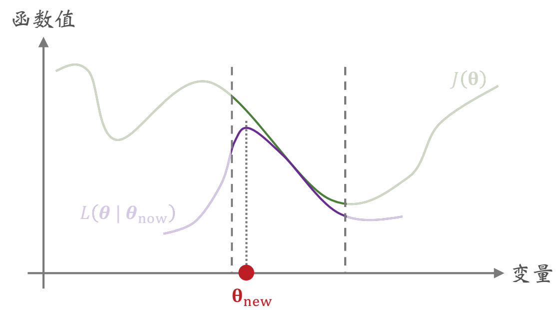
图 9.2 用一个一元函数的例子解释 $J(\boldsymbol{\theta})$ 和 $L(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}_{\text{now}})$ 的关系。图中横轴是优化变量 $\boldsymbol{\theta}$，纵轴是函数值。如图 9.2(a) 所示，函数 $L(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}_{\text{now}})$ 未必在整个定义域上都接近 $J(\boldsymbol{\theta})$，而只是在 $\boldsymbol{\theta}_{\text{now}}$ 的领域里接近 $J(\boldsymbol{\theta})$。$\boldsymbol{\theta}_{\text{now}}$ 的邻域就叫做置信域。



(a) 构造 $L(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}_{\text{now}})$ 作为 $J(\boldsymbol{\theta})$ 在点 $\boldsymbol{\theta}_{\text{now}}$ 附近的近似。



(b) L 在点 $\boldsymbol{\theta}_{\text{now}}$ 的邻域内接近 J；这个领域就叫置信域。



(c) 在置信域内寻找最大化 L 的解，记作 $\boldsymbol{\theta}_{\text{new}}$。

图 9.2: 一元函数的例子解释置信域和置信域算法。

通常来说，J 是个很复杂的函数，我们甚至可能不知道 J 的解析表达式（比如 J 是某个函数的期望）。而我们人为构造出的函数 L 相对较为简单，比如 L 是 J 的蒙特卡洛近似，或者是 J 在 $\boldsymbol{\theta}_{\text{now}}$ 这个点的二阶泰勒展开。既然可以信任 L，那么不妨用 L 代替复杂的函数 J，然后对 L 做最大化。这样比直接优化 J 要容易得多。这就是**置信域方法**的思想。具体来说，置信域方法做下面这两个步骤，一直重复下去，当无法让 J 的值增大的时候终止算法。

第一步——做近似： 给定 $\boldsymbol{\theta}_{\text{now}}$，构造函数 $L(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}_{\text{now}})$，使得对于所有的 $\boldsymbol{\theta}\in\mathcal{N}(\boldsymbol{\theta}_{\text{now}})$，函数值 $L(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}_{\text{now}})$ 与 $J(\boldsymbol{\theta})$ 足够接近。图 9.2(b) 解释了做近似这一步。

第二步——最大化： 在置信域 $\mathcal{N}(\boldsymbol{\theta}_{\text{now}})$ 中寻找变量 $\boldsymbol{\theta}$ 的值，使得函数 L 的值最大化。把找到的值记作

$$\boldsymbol{\theta}_{\text{new}} \;=\; \underset{\boldsymbol{\theta}\in\mathcal{N}(\boldsymbol{\theta}_{\text{now}})}{\operatorname{argmax}}\; L\big(\boldsymbol{\theta}\,\big|\,\boldsymbol{\theta}_{\text{now}}\big).$$

图 9.2(c) 解释了最大化这一步。

置信域方法其实是一类算法框架，而非一个具体的算法。有很多种方式实现实现置信域方法。第一步需要做近似，而做近似的方法有多种多样，比如蒙特卡洛、二阶泰勒展开。第二步需要解一个带约束的最大化问题；求解这个问题又需要单独的数值优化算法，比如梯度投影算法、拉格朗日法。除此之外，置信域 $\mathcal{N}(\boldsymbol{\theta}_{\text{now}})$ 也有多种多样的选择，既可以是球，也可以是两个概率分布的 KL 散度 (KL Divergence)，稍后会介绍。

9.1.2 策略学习

首先复习策略学习的基础知识。策略网络记作 $\pi(a|s;\boldsymbol{\theta})$，它是个概率质量函数。动作价值函数记作 $Q_\pi(s,a)$，它是回报的条件期望。状态价值函数记作

$$V_\pi(s) \;=\; \mathbb{E}_{A\sim\pi(\cdot|s;\boldsymbol{\theta})}\big[Q_\pi(s,A)\big] \;=\; \sum_{a\in\mathcal{A}}\pi(a|s;\boldsymbol{\theta})\cdot Q_\pi(s,a). \tag{9.2}$$

注意，$V_\pi(s)$ 依赖于策略网络 π，所以依赖于 π 的参数 $\boldsymbol{\theta}$。策略学习的目标函数是

$$J(\boldsymbol{\theta}) \;=\; \mathbb{E}_S\big[V_\pi(S)\big]. \tag{9.3}$$

$J(\boldsymbol{\theta})$ 只依赖于 $\boldsymbol{\theta}$，不依赖于状态 S 和动作 A。第 7 章介绍的策略梯度方法（包括 REINFORCE 和 Actor-Critic）用蒙特卡洛近似梯度 $\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta})$，得到随机梯度，然后做随机梯度上升更新 $\boldsymbol{\theta}$，使得目标函数 $J(\boldsymbol{\theta})$ 增大。

下面我们要把目标函数 $J(\boldsymbol{\theta})$ 变换成一种等价形式。从等式(9.2)出发，把状态价值写成

$$\begin{aligned} V_\pi(s) &= \sum_{a\in\mathcal{A}}\pi\big(a\,\big|\,s;\,\boldsymbol{\theta}_{\text{now}}\big)\cdot\frac{\pi(a\,|\,s;\,\boldsymbol{\theta})}{\pi(a\,|\,s;\,\boldsymbol{\theta}_{\text{now}})}\cdot Q_\pi(s,a) \\ &= \mathbb{E}_{A\sim\pi(\cdot|s;\boldsymbol{\theta}_{\text{now}})}\left[\frac{\pi(A\,|\,s;\,\boldsymbol{\theta})}{\pi(A\,|\,s;\,\boldsymbol{\theta}_{\text{now}})}\cdot Q_\pi(s,A)\right]. \end{aligned} \tag{9.4}$$

第一个等式很显然，因为连加中的第一项可以消掉第二项的分母。第二个等式把策略网络 $\pi(A|s;\boldsymbol{\theta}_{\text{now}})$ 看做动作 A 的概率质量函数，所以可以把连加写成期望。由公式 (9.3) 与 (9.4) 可得定理 9.1。定理 9.1 是 TRPO 的关键所在，甚至可以说 TRPO 就是从这个公式推出的。

定理 9.1. 目标函数的等价形式

目标函数 $J(\boldsymbol{\theta})$ 可以等价写成：

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

♡

公式中的期望是关于状态 S 和动作 A 求的。状态 S 的概率密度函数只有环境知道，而我们并不知道，但是我们可以从环境中获取 S 的观测值。动作 A 的概率质量函数是策略网络 $\pi(A\,|\,S;\boldsymbol{\theta}_{\text{now}})$；注意，策略网络的参数是旧的值 $\boldsymbol{\theta}_{\text{now}}$.

9.1.3 TRPO 数学推导

前面介绍了数值优化的基础和价值学习的基础，终于可以开始推导 TRPO。TRPO 是置信域方法在策略学习中的应用，所以 TRPO 也遵循置信域方法的框架，重复**做近似**和**最大化**这两个步骤，直到算法收敛。收敛指的是无法增大目标函数 $J(\boldsymbol{\theta})$，即无法增大期望回报。

第一步——做近似：我们从定理 9.1 出发。定理把目标函数 $J(\boldsymbol{\theta})$ 写成了期望的形式。我们无法直接算出期望，无法得到 $J(\boldsymbol{\theta})$ 的解析表达式；原因在于只有环境知道状态 S 的概率密度函数，而我们不知道。我们可以对期望做蒙特卡洛近似，从而把函数 J 近

似成函数 L。用策略网络 $\pi(A\,|\,S;\,\boldsymbol{\theta}_{\text{now}})$ 控制智能体跟环境交互，从头到尾玩完一局游戏，观测到一条轨迹：

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

其中的状态 $\{s_t\}_{t=1}^n$ 都是从环境中观测到的，其中的动作 $\{a_t\}_{t=1}^n$ 都是根据策略网络 $\pi(\cdot\,|\,s_t;\,\boldsymbol{\theta}_{\text{now}})$ 抽取的样本。所以，

$$\frac{\pi(a_t\,|\,s_t;\,\boldsymbol{\theta})}{\pi(a_t\,|\,s_t;\,\boldsymbol{\theta}_{\text{now}})}\cdot Q_\pi\big(s_t,a_t\big) \tag{9.5}$$

是对定理 9.1 中期望的无偏估计。我们观测到了 n 组状态和动作，于是应该对公式 (9.5) 求平均，把得到均值记作：

$$L\big(\boldsymbol{\theta}\,\big|\,\boldsymbol{\theta}_{\text{now}}\big) \;=\; \frac{1}{n}\sum_{t=1}^{n}\underbrace{\frac{\pi(a_t\,|\,s_t;\,\boldsymbol{\theta})}{\pi(a_t\,|\,s_t;\,\boldsymbol{\theta}_{\text{now}})}\cdot Q_\pi\big(s_t,a_t\big)}_{\text{定理 9.1 中期望的无偏估计}}. \tag{9.6}$$

既然连加里每一项都是期望的无偏估计，那么 n 项的均值 L 也是无偏估计。所以可以拿 L 作为目标函数 J 的蒙特卡洛近似。

公式 (9.6) 中的 $L(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}_{\text{now}})$ 是对目标函数 $J(\boldsymbol{\theta})$ 的近似。可惜我们还无法直接对 L 求最大化，原因是我们不知道动作价值 $Q_\pi(s_t,a_t)$。解决方法是把 $Q_\pi(s_t,a_t)$ 近似成观测到的折扣回报：

$$u_t \;=\; r_t+\gamma\cdot r_{t+1}+\gamma^2\cdot r_{t+2}+\cdots+\gamma^{n-t}\cdot r_n.$$

拿 u_t 替代 $Q_\pi(s_t,a_t)$，那么公式 (9.6) 中的 $L(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}_{\text{now}})$ 变成了

$$\boxed{\tilde{L}\big(\boldsymbol{\theta}\,\big|\,\boldsymbol{\theta}_{\text{now}}\big) \;=\; \sum_{t=1}^{n}\frac{\pi(a_t\,|\,s_t;\,\boldsymbol{\theta})}{\pi(a_t\,|\,s_t;\,\boldsymbol{\theta}_{\text{now}})}\cdot u_t.} \tag{9.7}$$

总结一下，我们把目标函数 J 近似成 L，然后又把 L 近似成 $\tilde{L}$。

第二步——最大化： TRPO 把公式(9.7)中的 $\tilde{L}(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}_{\text{now}})$ 作为对目标函数 $J(\boldsymbol{\theta})$ 的近似，然后求解这个带约束的最大化问题：

$$\boxed{\max_{\boldsymbol{\theta}}\;\tilde{L}\big(\boldsymbol{\theta}\,\big|\,\boldsymbol{\theta}_{\text{now}}\big);\qquad \text{s.t.}\;\;\boldsymbol{\theta}\in\mathcal{N}\big(\boldsymbol{\theta}_{\text{now}}\big).} \tag{9.8}$$

公式中的 $\mathcal{N}(\boldsymbol{\theta}_{\text{now}})$ 是置信域，即 θ_{now} 的一个邻域。该用什么样的置信域呢？

- 一种方法是用以 $\boldsymbol{\theta}_{\text{now}}$ 为球心、以 Δ 为半径的球作为置信域。这样的话，公式(9.8)就变成

$$\max_{\boldsymbol{\theta}}\;\tilde{L}\big(\boldsymbol{\theta}\,\big|\,\boldsymbol{\theta}_{\text{now}}\big);\qquad \text{s.t.}\;\;\big\|\boldsymbol{\theta}-\boldsymbol{\theta}_{\text{now}}\big\|_2\leq\Delta. \tag{9.9}$$

- 另一种方法是用 KL 散度衡量两个概率质量函数——$\pi(\cdot|s_i;\boldsymbol{\theta}_{\text{now}})$ 和 $\pi(\cdot|s_i;\boldsymbol{\theta})$——的距离。两个概率质量函数区别越大，它们的 KL 散度就越大。反之，如果 $\boldsymbol{\theta}$ 很接近 $\boldsymbol{\theta}_{\text{now}}$，那么两个概率质量函数就越接近。用 KL 散度的话，公式(9.8)就变成

$$\max_{\boldsymbol{\theta}}\;\tilde{L}\big(\boldsymbol{\theta}\,\big|\,\boldsymbol{\theta}_{\text{now}}\big);\qquad \text{s.t.}\;\;\frac{1}{t}\sum_{i=1}^{t}\text{KL}\Big[\pi\big(\,\cdot\,\big|\,s_i;\,\boldsymbol{\theta}_{\text{now}}\big)\,\Big\|\,\pi\big(\,\cdot\,\big|\,s_i;\,\boldsymbol{\theta}\big)\Big]\;\leq\;\Delta. \tag{9.10}$$

用球作为置信域的好处是置信域是简单的形状，求解最大化问题比较容易，但是用球做

置信域的实际效果不如用 KL 散度。

TRPO 的第二步——最大化——需要求解带约束的最大化问题 (9.9) 或者 (9.10)。注意，这种问题的求解并不容易；简单的梯度上升算法并不能解带约束的最大化问题。数值优化教材里面通常会有章节介绍这类问题的求解，有兴趣的话自己去阅读数值优化教材。这里就不详细解释如何求解问题 (9.9) 或者 (9.10)。读者可以这样看待优化问题：只要你能把一个优化问题的目标函数和约束条件写出来，通常会有数值算法能解决这个问题。

9.1.4 训练流程

在本节的最后，我们总结一下用 TRPO 训练策略网络的流程。TRPO 需要重复**做近似**和**最大化**这两个步骤：

1. **做近似**——构造函数 $\tilde{L}$ 近似目标函数 $J(\boldsymbol{\theta})$：
   (a). 设当前策略网络参数是 $\boldsymbol{\theta}_{\text{now}}$。用策略网络 $\pi(a \,|\, s;\, \boldsymbol{\theta}_{\text{now}})$ 控制智能体与环境交互，玩完一局游戏，记录下轨迹：
   $$s_1, a_1, r_1, \quad s_2, a_2, r_2, \quad \cdots, \quad s_n, a_n, r_n.$$
   (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). 得出近似函数：
   $$\tilde{L}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_{\text{now}}) \;=\; \sum_{t=1}^{n} \frac{\pi(a_t \,|\, s_t;\, \boldsymbol{\theta})}{\pi(a_t \,|\, s_t; \boldsymbol{\theta}_{\text{now}})} \cdot u_t.$$
2. **最大化**——用某种数值算法求解带约束的最大化问题：
   $$\boldsymbol{\theta}_{\text{new}} \;=\; \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\; \tilde{L}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}_{\text{now}}); \qquad \text{s.t.}\;\; \big\| \boldsymbol{\theta} - \boldsymbol{\theta}_{\text{now}} \big\|_2 \;\leq\; \Delta.$$
   此处的约束条件是二范数距离。可以把它替换成 KL 散度，即公式 (9.10)。

TRPO 中有两个需要调的超参数：一个是置信域的半径 Δ，另一个是求解最大化问题的数值算法的学习率。通常来说，Δ 在算法的运行过程中要逐渐缩小。虽然 TRPO 需要调参，但是 TRPO 对超参数的设置并不敏感。即使超参数设置不够好，TRPO 的表现也不会太差。相比之下，策略梯度算法对超参数更敏感。

TRPO 算法真正实现起来并不容易，主要难点在于第二步——**最大化**。不建议读者自己去实现 TRPO。

9.2 熵正则 (Entropy Regularization)

策略学习的目的是学出一个策略网络 $\pi(a|s;\boldsymbol{\theta})$ 用于控制智能体。每当智能体观测到当前状态 s，策略网络输出一个概率分布，智能体依据概率分布抽样一个动作，并执行这个动作。举个例子，在超级玛丽游戏中，动作空间是 $\mathcal{A}=\{左,右,上\}$。基于当前状态 s，策略网络的输出是

$$\begin{aligned} p_1 &= \pi\big(左\,\big|\,s;\boldsymbol{\theta}\big) = 0.03, \\ p_2 &= \pi\big(右\,\big|\,s;\boldsymbol{\theta}\big) = 0.96, \\ p_3 &= \pi\big(上\,\big|\,s;\boldsymbol{\theta}\big) = 0.01. \end{aligned}$$

那么超级玛丽做的动作可能是左、右、上三者中的任何一个，概率分别是 0.03, 0.96, 0.01。概率都集中在“向右”的动作上，接近确定性的决策。确定性大的好处在于不容易选中很差的动作，比较安全。但是确定性大也有缺点。假如策略网络的输出总是这样确定性很大的概率分布，那么智能体就会安于现状，不去尝试没做过的动作，不去探索更多的状态，无法找到更好的策略。

我们希望策略网络的输出的概率不要集中在一个动作上，至少要给其他的动作一些非零的概率，让这些动作能被探索到。可以用熵 (Entropy) 来衡量概率分布的不确定性。对于上述离散概率分布 $\boldsymbol{p}=[p_1,p_2,p_3]$，熵等于

$$\text{Entropy}(\boldsymbol{p}) = -\sum_{i=1}^{3} p_i \cdot \ln p_i.$$

熵小说明概率质量很集中，熵大说明随机性很大；见图 9.3 的解释。

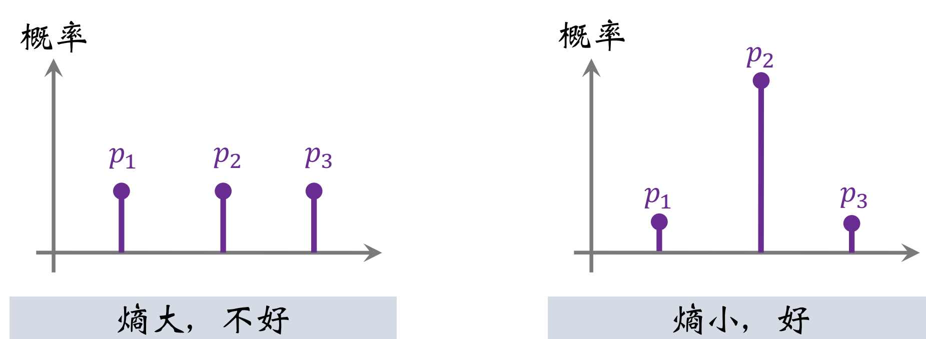


图 9.3: 两张图中分别描述两个离散概率分布。左边的概率比较均匀，这种情况熵很大。右边的概率集中在 p_2 上，这种情况的熵较小。

策略学习中的熵正则： 我们希望策略网络输出的概率分布的熵不要太小。我们不妨把熵作为正则项，放到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，它表示定义在动作空间上的离散概率分布。这个概率分布的熵定义为：

$$H(s;\boldsymbol{\theta}) \triangleq \text{Entropy}\Big[\pi\big(\,\cdot\,\big|\,s;\boldsymbol{\theta}\big)\Big] = -\sum_{a\in\mathcal{A}} \pi\big(a\,\big|\,s;\boldsymbol{\theta}\big)\cdot \pi\big(a\,\big|\,s;\boldsymbol{\theta}\big). \tag{9.11}$$

熵 $H(s;\boldsymbol{\theta})$ 只依赖于状态 s 与策略网络参数 $\boldsymbol{\theta}$。我们希望对于大多数的状态 s，熵都会比

较大，也就是让 $\mathbb{E}_S[H(S;\boldsymbol{\theta})]$ 比较大。

回忆一下，$V_\pi(s)$ 是状态价值函数，衡量在状态 s 的情况下，策略网络 π 表现的好坏程度。策略学习的目标函数是 $J(\boldsymbol{\theta}) = \mathbb{E}_S[V_\pi(S)]$。策略学习的目的是寻找参数 $\boldsymbol{\theta}$ 使得 $J(\boldsymbol{\theta})$ 最大化。同时，我们还希望让熵比较大，所以把熵作为正则项，放到目标函数里。使用熵正则的策略学习可以写作这样的最大化问题：

$$\boxed{\max_{\theta} \quad J(\boldsymbol{\theta}) \;+\; \lambda \cdot \mathbb{E}_S\Big[H\big(S;\,\boldsymbol{\theta}\big)\Big].} \tag{9.12}$$

此处的 λ 是个超参数，需要手动调。

优化： 带熵正则的最大化问题 (9.12) 可以用各种方法求解，比如策略梯度方法（包括 REINFORCE 和 Actor-Critic）、TRPO 等。此处只讲解策略梯度方法。公式 (9.12) 中目标函数关于 $\boldsymbol{\theta}$ 的梯度是：

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

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

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

是梯度 $\boldsymbol{g}(\boldsymbol{\theta})$ 的无偏估计（见定理 9.2）。因此可以用 $\tilde{\boldsymbol{g}}\big(s,a;\boldsymbol{\theta}\big)$ 更新策略网络的参数：

$$\boldsymbol{\theta} \;\leftarrow\; \boldsymbol{\theta} + \beta \cdot \tilde{\boldsymbol{g}}(s,a;\boldsymbol{\theta}).$$

此处的 β 是学习率。

定理 9.2. 带熵正则的策略梯度

$$\nabla_{\boldsymbol{\theta}}\Big[\,J(\boldsymbol{\theta}) \;+\; \lambda \cdot \mathbb{E}_S\big[H\big(S;\,\boldsymbol{\theta}\big)\big]\,\Big] \;\;=\;\; \mathbb{E}_S\Big[\mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\Big[\tilde{\boldsymbol{g}}\big(S,A;\boldsymbol{\theta}\big)\Big]\Big].$$

♡

证明 首先推导熵 $H(S;\boldsymbol{\theta})$ 关于 $\boldsymbol{\theta}$ 的梯度。由公式 (9.11) 中 $H(S;\boldsymbol{\theta})$ 的定义可得

$$\begin{aligned}
\frac{\partial\, H(s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}} \;&=\; -\sum_{a\in\mathcal{A}} \frac{\partial\, \pi(a|s;\boldsymbol{\theta}) \cdot \ln \pi(a|s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}} \\
&=\; -\sum_{a\in\mathcal{A}} \bigg[\ln \pi(a|s;\boldsymbol{\theta}) \cdot \frac{\partial\, \pi(a|s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}} + \pi(a|s;\boldsymbol{\theta}) \cdot \frac{\partial\, \ln \pi(a|s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}}\bigg].
\end{aligned}$$

第二个等式由链式法则得到。由于 $\frac{\partial\,\pi(a|s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}} = \pi(a|s;\boldsymbol{\theta}) \cdot \frac{\partial\,\ln\pi(a|s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}}$，上面的公式可以写成：

$$\begin{aligned}
\frac{\partial\, H(s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}} \;&=\; -\sum_{a\in\mathcal{A}} \bigg[\ln \pi(a|s;\boldsymbol{\theta}) \cdot \pi(a|s;\boldsymbol{\theta}) \cdot \frac{\partial\, \ln \pi(a|s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}} + \pi(a|s;\boldsymbol{\theta}) \cdot \frac{\partial\, \ln \pi(a|s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}}\bigg] \\
&=\; -\sum_{a\in\mathcal{A}} \pi(a|s;\boldsymbol{\theta}) \cdot \Big[\ln \pi(a|s;\boldsymbol{\theta}) + 1\Big] \cdot \frac{\partial\, \ln \pi(a|s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}} \\
&=\; -\mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\bigg[\Big[\ln \pi(A|s;\boldsymbol{\theta}) + 1\Big] \cdot \frac{\partial\, \ln \pi(A|s;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}}\bigg].
\end{aligned} \tag{9.13}$$

应用第 7 章推导的策略梯度定理，可以把 $J(\boldsymbol{\theta})$ 关于 $\boldsymbol{\theta}$ 的梯度写作

$$\frac{\partial\, J(\boldsymbol{\theta})}{\partial\, \boldsymbol{\theta}} \;=\; \mathbb{E}_S\left\{\mathbb{E}_{A\sim\pi(\cdot|S;\boldsymbol{\theta})}\left[Q_\pi(S,A)\cdot\frac{\partial\,\ln\pi(A|S;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}}\right]\right\}. \tag{9.14}$$

由公式 (9.13) 与 (9.14) 可得：

$$\begin{aligned}
&\frac{\partial}{\partial\,\boldsymbol{\theta}}\Big[J(\boldsymbol{\theta})+\lambda\cdot\mathbb{E}_S\big[H(S;\boldsymbol{\theta})\big]\Big] \\
&= \mathbb{E}_S\left\{\mathbb{E}_{A\sim\pi(\cdot|S;\boldsymbol{\theta})}\left[\Big(Q_\pi(S,A)-\lambda\cdot\ln\pi(A|S;\boldsymbol{\theta})-\lambda\Big)\cdot\frac{\partial\,\ln\pi(A|S;\boldsymbol{\theta})}{\partial\,\boldsymbol{\theta}}\right]\right\} \\
&= \mathbb{E}_S\Big\{\mathbb{E}_{A\sim\pi(\cdot|S;\boldsymbol{\theta})}\Big[\tilde{\boldsymbol{g}}\big(S,A;\boldsymbol{\theta}\big)\Big]\Big\}.
\end{aligned}$$

上面第二个等式由 $\tilde{\boldsymbol{g}}$ 的定义得到。 □

第九章 相关文献

TRPO 由 John Schulman 等学者在 2015 年提出 [88]。TRPO 是置信域方法在强化学习中的成功应用。置信域是经典的数值优化算法，对此感兴趣的读者可以阅读这些教材：[76, 30]。TRPO 每一轮循环都需要求解带约束的最大化问题；这类问题的求解可以参考这些教材：[13, 19]。

熵正则是策略学习中常见的方法，在很多论文中有使用，比如 [120, 69, 77, 3, 44, 89]。虽然熵正则能鼓励探索，但是增大决策的不确定性是有风险的：很差的动作可能也有非零的概率。一个号的办法是用 Tsallis Entropy [104] 做正则，让离散概率具有稀疏性，每次决策只给少部分动作非零的概率，“过滤掉”很差的动作。有兴趣的读者可以阅读这些论文：[29, 59, 121]。