

# 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： $\max_{\theta} J(\theta)$ 。这里的  $J(\theta)$  是目标函数， $\theta$  是优化变量。求解这个优化问题的目的是找到一个变量  $\theta$  使得目标函数  $J(\theta)$  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：

$$\theta_{\text{new}} \leftarrow \text{Update} \left( \text{Data}; \theta_{\text{now}} \right).$$

此处的  $\theta_{\text{now}}$  和  $\theta_{\text{new}}$  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  $\theta_{\text{now}}$ ，用  $\mathcal{N}(\theta_{\text{now}})$  表示  $\theta_{\text{now}}$  的一个邻域。举个例子：

$$\mathcal{N}(\theta_{\text{now}}) = \left\{ \theta \mid \|\theta - \theta_{\text{now}}\|_2 \leq \Delta \right\}. \quad (9.1)$$

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



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

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

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

那么集合  $\mathcal{N}(\theta_{\text{now}})$  就被称作置信域。顾名思义，在  $\theta_{\text{now}}$  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  $J(\theta)$ 。

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

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  $\theta_{\text{now}}$  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  $\theta_{\text{now}}$ , 构造函数  $L(\theta | \theta_{\text{now}})$ , 使得对于所有的  $\theta \in \mathcal{N}(\theta_{\text{now}})$ , 函数值  $L(\theta | \theta_{\text{now}})$  与  $J(\theta)$  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  $\mathcal{N}(\theta_{\text{now}})$  中寻找变量  $\theta$  的值, 使得函数  $L$  的值最大化。把找到的值记作

$$\theta_{\text{new}} = \underset{\theta \in \mathcal{N}(\theta_{\text{now}})}{\operatorname{argmax}} L(\theta | \theta_{\text{now}}).$$

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  $\mathcal{N}(\theta_{\text{now}})$  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  $\pi(a|s; \theta)$ ，它是个概率密度函数。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

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

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

$$J(\theta) = \mathbb{E}_S [V_\pi(S)]. \quad (9.3)$$

$J(\theta)$  只依赖于  $\theta$ ，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  $\nabla_\theta J(\theta)$ ，得到随机梯度，然后做随机梯度上升更新  $\theta$ ，使得目标函数  $J(\theta)$  增大。

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

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

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  $\pi(A|s; \theta_{\text{now}})$  看做动作  $A$  的概率密度函数，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  $J(\theta)$  可以等价写成：

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



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  $A$  的概率密度函数是策略网络  $\pi(A|S; \theta_{\text{now}})$ ；注意，策略网络的参数是旧的值  $\theta_{\text{now}}$ 。

### 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  $J(\theta)$ ，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  $J(\theta)$  写成了期望的形式。我们无法直接算出期望，无法得到  $J(\theta)$  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  $\pi(A | S; \theta_{\text{now}})$  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：

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

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  $\pi(\cdot | s_t; \theta_{\text{now}})$  抽取的样本。所以，

$$\frac{\pi(a_t | s_t; \theta)}{\pi(a_t | s_t; \theta_{\text{now}})} \cdot Q_\pi(s_t, a_t) \quad (9.5)$$

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：

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

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

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

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

$$\tilde{L}(\theta | \theta_{\text{now}}) = \sum_{t=1}^n \frac{\pi(a_t | s_t; \theta)}{\pi(a_t | s_t; \theta_{\text{now}})} \cdot u_t. \quad (9.7)$$

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  $\tilde{L}(\theta | \theta_{\text{now}})$  作为对目标函数  $J(\theta)$  的近似，然后求解这个带约束的最大化问题：

$$\max_{\theta} \tilde{L}(\theta | \theta_{\text{now}}); \quad \text{s.t. } \theta \in \mathcal{N}(\theta_{\text{now}}). \quad (9.8)$$

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

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

$$\max_{\theta} \tilde{L}(\theta | \theta_{\text{now}}); \quad \text{s.t. } \|\theta - \theta_{\text{now}}\|_2 \leq \Delta. \quad (9.9)$$

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

$$\max_{\theta} \tilde{L}(\theta | \theta_{\text{now}}); \quad \text{s.t. } \frac{1}{t} \sum_{i=1}^t \text{KL}\left[\pi(\cdot | s_i; \theta_{\text{now}}) \parallel \pi(\cdot | s_i; \theta)\right] \leq \Delta. \quad (9.10)$$

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  $J(\theta)$ ：

- (a). 设当前策略网络参数是  $\theta_{\text{now}}$ 。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，记录下轨迹：

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

- (b). 对于所有的  $t = 1, \dots, n$ ，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：

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

2. **最大化**——用某种数值算法求解带约束的最大化问题：

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

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  $\Delta$ ，另一个是求解最大化问题的数值算法的学习率。通常来说， $\Delta$  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  $\pi(a|s; \theta)$  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  $\mathcal{A} = \{\text{左}, \text{右}, \text{上}\}$ 。基于当前状态  $s$ , 策略网络的输出是

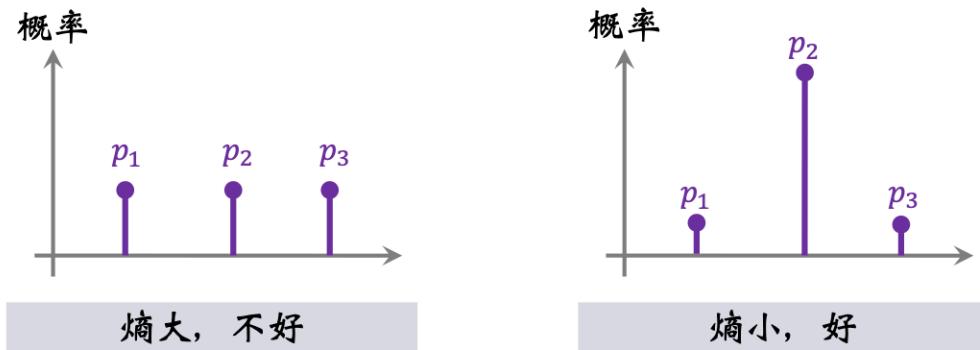
$$\begin{aligned} p_1 &= \pi(\text{左} | s; \theta) = 0.03, \\ p_2 &= \pi(\text{右} | s; \theta) = 0.96, \\ p_3 &= \pi(\text{上} | s; \theta) = 0.01. \end{aligned}$$

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。概率密度都集中在“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密度不要集中在一个动作上, 至少要给其他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  $\mathbf{p} = [p_1, p_2, p_3]$ , 熵等于

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

熵小说明概率密度很集中, 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。左边的概率密度很均匀, 这种情况熵很大。右边的概率密度集中在  $p_2$  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:

$$H(s; \theta) \triangleq \text{Entropy} [\pi(\cdot | s; \theta)] = - \sum_{a \in \mathcal{A}} \pi(a | s; \theta) \cdot \ln \pi(a | s; \theta). \quad (9.11)$$

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

较大，也就是让  $\mathbb{E}_S[H(S; \theta)]$  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  $\pi$  表现的好坏程度。策略学习的目标函数是  $J(\theta) = \mathbb{E}_S[V_\pi(S)]$ 。策略学习的目的是寻找参数  $\theta$  使得  $J(\theta)$  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：

$$\max_{\theta} J(\theta) + \lambda \cdot \mathbb{E}_S[H(S; \theta)]. \quad (9.12)$$

此处的  $\lambda$  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  $\theta$  的梯度是：

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

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

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

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

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

此处的  $\beta$  是学习率。

### 定理 9.2. 带熵正则的策略梯度

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



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

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

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

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

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

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

由公式 (9.13) 与 (9.14) 可得：

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

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

## 相关文献

TRPO 由 John Schulman 等学者在 2015 年提出 [49]。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：[41, 20]。TRPO 每一轮循环都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：[8, 12]。

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 [70, 37, 42, 2, 26, 50]。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 [58] 做正则，让离散概率密度具有稀疏性，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：[19, 33]。

## 参考文献

- [1] M. S. Abdulla and S. Bhatnagar. Reinforcement learning based algorithms for average cost markov decision processes. *Discrete Event Dynamic Systems*, 17(1):23–52, 2007.
- [2] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning (ICML)*, 2019.
- [3] L. V. Allis et al. *Searching for solutions in games and artificial intelligence*. Ponsen & Looijen Wageningen, 1994.
- [4] L. Baird. Residual algorithms: reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.
- [5] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.
- [6] P. Baudiš and J.-l. Gailly. Pachi: State of the art open source go program. In *Advances in computer games*, pages 24–38. Springer, 2011.
- [7] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.
- [8] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.
- [9] S. Bhatnagar and S. Kumar. A simultaneous perturbation stochastic approximation-based actor-critic algorithm for markov decision processes. *IEEE Transactions on Automatic Control*, 49(4):592–598, 2004.
- [10] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482, 2009.
- [11] B. Bouzy and B. Helmstetter. Monte-Carlo go developments. In *Advances in computer games*, pages 159–174. Springer, 2004.
- [12] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [13] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfs, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.
- [14] M. Buro. From simple features to sophisticated evaluation functions. In *International Conference on Computers and Games*, pages 126–145. Springer, 1998.
- [15] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.
- [16] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-Carlo tree search: A new framework for game AI. In *AIIDE*, 2008.
- [17] G. Chaslot, J.-T. Saito, B. Bouzy, J. Uiterwijk, and H. J. Van Den Herik. Monte-Carlo strategies for computer Go. In *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium*, 2006.
- [18] G. M. J.-B. C. Chaslot. *Monte-Carlo tree search*. Maastricht University, 2010.
- [19] Y. Chow, O. Nachum, and M. Ghavamzadeh. Path consistency learning in Tsallis entropy regularized mdps. In *International Conference on Machine Learning (ICML)*, pages 979–988, 2018.
- [20] A. R. Conn, N. I. Gould, and P. L. Toint. *Trust region methods*. SIAM, 2000.
- [21] R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.
- [22] R. Coulom. Computing “elo ratings” of move patterns in the game of Go. *ICGA journal*, 30(4):198–208, 2007.
- [23] T. Degris, P. M. Pilarski, and R. S. Sutton. Model-free reinforcement learning with continuous action in practice. In *American Control Conference (ACC)*, 2012.
- [24] M. Enzenberger, M. Müller, B. Arneson, and R. Segal. Fuego: an open-source framework for board games and go engine based on monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):259–270, 2010.

- [25] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, et al. Noisy networks for exploration. In *International Conference on Learning Representations (ICLR)*, 2018.
- [26] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning (ICML)*, 2017.
- [27] R. Hafner and M. Riedmiller. Reinforcement learning in feedback control. *Machine learning*, 84(1-2):137–169, 2011.
- [28] M. Hausknecht and P. Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015.
- [29] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [30] T. Jaakkola, M. I. Jordan, and S. P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural computation*, 6(6):1185–1201, 1994.
- [31] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.
- [32] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.
- [33] K. Lee, S. Choi, and S. Oh. Sparse Markov decision processes with causal sparse Tsallis entropy regularization for reinforcement learning. *IEEE Robotics and Automation Letters*, 3(3):1466–1473, 2018.
- [34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.
- [35] L.-J. Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.
- [36] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of Markov reward processes: Implementation issues. In *IEEE Conference on Decision and Control*, 1999.
- [37] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.
- [38] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- [39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [40] M. Müller. Computer go. *Artificial Intelligence*, 134(1-2):145–179, 2002.
- [41] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
- [42] B. O’Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih. Combining policy gradient and Q-learning. In *International Conference on Learning Representations (ICLR)*, 2017.
- [43] D. V. Prokhorov and D. C. Wunsch. Adaptive critic designs. *IEEE transactions on Neural Networks*, 8(5):997–1007, 1997.
- [44] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.
- [45] G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.
- [46] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen. Checkers is solved. *science*, 317(5844):1518–1522, 2007.
- [47] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron. A world championship caliber checkers program. *Artificial Intelligence*, 53(2-3):273–289, 1992.

- [48] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2015.
- [49] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.
- [50] W. Shi, S. Song, and C. Wu. Soft policy gradient method for maximum entropy deep reinforcement learning. *arXiv preprint arXiv:1909.03198*, 2019.
- [51] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- [52] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2014.
- [53] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.
- [54] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems (NIPS)*, 1996.
- [55] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [56] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.
- [57] G. Tesauro and G. R. Galperin. On-line policy improvement using monte-carlo search. In *Advances in Neural Information Processing Systems*, pages 1068–1074, 1997.
- [58] C. Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of statistical physics*, 52(1-2):479–487, 1988.
- [59] J. N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine learning*, 16(3):185–202, 1994.
- [60] J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE transactions on automatic control*, 42(5):674–690, 1997.
- [61] H. J. Van Den Herik, J. W. Uiterwijk, and J. Van Rijswijck. Games solved: Now and in the future. *Artificial Intelligence*, 134(1-2):277–311, 2002.
- [62] H. van Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.
- [63] H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.
- [64] H. van Seijen. Effective multi-step temporal-difference learning for non-linear function approximation. *arXiv preprint arXiv:1608.05151*, 2016.
- [65] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.
- [66] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
- [67] C. J. C. H. Watkins. Learning from delayed rewards. 1989.
- [68] R. J. Williams. *Reinforcement-learning connectionist systems*. College of Computer Science, Northeastern University, 1987.
- [69] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- [70] R. J. Williams and J. Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.
- [71] Z. Yang, Y. Chen, M. Hong, and Z. Wang. Provably global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8353–8365, 2019.