MIRA Book for Reinforcement Learning [Draft 2020/12/10]

作者：强化学习组 @MIRA

组织：MIRA Lab

目录

第一部分

总更新记录

2020/11/25	更新：更新目录和内容格式（王治海、刘骐源）
2020/11/17	更新：将公众号文章《一种基于动作采样的简单高效的正则化强化学习方法》更新至此书（匡宇飞）
2020/11/11	更新：将公众号文章《DQN 中目标网络两种常见更新方法对比》更新至此书（石志皓）
2020/11/11	更新：将公众号文章《基于值的强化学习方法中的高估现象》更新至此书（周祺）
2020/11/10	更新：梳理目录（王治海、周祺、刘骐源）

第二部分

强化学习基础

第一章 强化学习简介（内容待补充）

内容待补充。

第二章 马尔可夫决策过程（内容待补充）

内容待补充。

第三章 动态规划（内容待补充）

内容待补充。

第 四 章　无模型预测（内容待补充）

内容待补充。

第 五 章　无模型控制

5.1　基于值的强化学习方法中的高估现象（周祺-2020/11/11）

强化学习算法基于智能体与环境的交互，以最大化累积奖励为目标，学习状态到动作的映射（即策略）。基于值（value-based）的强化学习方法首先学习最优值函数进而学习最优策略。过往的研究发现，此类算法在学习值函数的过程中会出现高估（overestimation）现象。本文结合理论与示例对值函数的高估现象进行了详尽的分析。

文中所有的证明在最后一部分统一给出。

5.1.1　背景

5.1.1.1　强化学习（Reinforcement Learning）

强化学习问题往往通过马尔科夫决策过程（Markov Decision Process）进行建模。该过程可以通过六元组 $(\mathcal{S}, \mathcal{A}, \rho, P, R, \gamma)$ 表示。其中

- $\mathcal{S}$ 表示状态空间。我们假设其是有限的，即 $|\mathcal{S}| < \infty$。
- $\mathcal{A}$ 表示动作空间，我们同样假设 $|\mathcal{A}| < \infty$。
- $\rho : \mathcal{S} \to [0, 1]$ 是初始状态分布的概率函数。即 $\rho(s)$ 表示初始状态为 s 的概率。
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ 是状态转移的概率分布。即 $P(s'|s, a)$ 表示在状态 s 执行动作 a 后下一个状态是 s' 的概率。
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{R} \to [0, 1]$ 是奖励的概率分布。即 $R(x|s, a)$ 表示在状态 s 执行动作 a 后所获奖励为 x 的概率。其中，集合 $\mathcal{R}$ 包含所有可能的奖励取值。我们假设 $\mathcal{R}$ 有限且有界。令 r_{max} 与 r_{min} 分别表示 $\mathcal{R}$ 的上确界和下确界。同时，我们令函数 $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ 表示奖励的期望，即 $r(s, a) \triangleq \mathbb{E}_{X \sim R(\cdot|s,a)}[X]$。
- $\gamma \in (0, 1)$ 表示折扣因子，是一个常数。

我们使用 $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ 表示策略。具体地，$\pi(a|s)$ 表示在状态 s 下执行动作 a 的概率。给定策略 π，我们按以下方式定义值函数 Q^π：

$$Q^\pi(s, a) \triangleq \mathbb{E}_{(X_0, S_1, A_1, X_1, \cdots)} \left[\sum_{t=0}^{\infty} \gamma^t X_t \middle| S_0 = s, A_0 = a \right]$$

其中，A_t 以概率 $\pi(A_t|S_t)$ 采样，X_t 以概率 $R(X_t|S_t, A_t)$ 采样，S_{t+1} 以概率 $P(S_{t+1}|S_t, A_t)$ 采样。本文中，我们认为值函数是 $|\mathcal{S}| \times |\mathcal{A}|$ 维的向量。值函数（向量）之间的等式或不等关系对每一维均成立。强化学习的目标是找到一个最优策略 π^* 使得 $Q^{\pi^*} \geq Q^\pi$。我们将 Q^{π^*} 简记为 Q^*。

5.1.1.2 值迭代 (value iteration)

值迭代是一类动态规划算法。此类方法首先求解最优值函数 Q^*。具体地，首先按以下方式定义算子 $\mathcal{T}^*$

$$(\mathcal{T}^*Q)(s,a) \triangleq r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{a' \in \mathcal{A}} Q(s',a')$$

基于此算子，我们构建函数序列 $(Q^n)_{n=1}^{\infty}$ 其满足 $Q^{n+1} = \mathcal{T}^*Q^n$。因为 $\mathcal{T}^*$ 是压缩映射且有 $Q^* = \mathcal{T}^*Q^*$，所以该函数列收敛到最优值函数 Q^*。求解出 Q^* 后，在每个状态 s 选择使 $Q^*(s,\cdot)$ 最大的动作，便能得到一个最优策略。

5.1.1.3 算法流程

值迭代难以直接应用于实际问题。一方面，值迭代需要函数 P, R 已知，但这在真实任务中往往是不可行的。另一方面，实际问题中的状态空间往往很大，因此难以使用表格准确地存储价值函数 Q^n。为了解决以上困难，前人提出了以下改进方案：

1. **数据采样：** 从环境中采样与奖励和状态转移函数相关的数据用于值函数的更新
2. **函数近似：** 使用神经网络等函数近似器替代表格近似地储存值函数。

为了统一讨论以上两种改进方案对值函数学习的影响，本文考虑如下算法流程。则执行算法一时，必然存在正数 N，使得 $\mathbb{E}[Q^n] \geq Q^* + \epsilon e$ 对所有的 $n > N$ 成立。

算法 5.1

1. 随机初始化值函数 Q^0 并设 n 为 1
2. 选取随机算子 $\mathcal{T}^n$ 作为 $\mathcal{T}^*$ 的近似
3. 计算 $Q^n = \mathcal{T}^n Q^{n-1}$
4. $n \leftarrow n+1$
5. 重复步骤 2 到 4

♡

为了便于理解，我们认为 $\mathcal{T}^{n+1}Q^n = \mathcal{T}(W^n, Q^n)$。其中 $\mathcal{T} : \mathcal{W} \times \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ 是某个固定的函数，W^n 是取值范围为 $\mathcal{W}$ 的随机变量。我们假设，$\mathbb{E}[Q^0]$ 存在且有界。

5.1.2 值函数的高估现象及其成因

值函数的高估现象指，将 Q^n 作为 Q^* 的估计量时往往存在正偏差 (positive biases)。本文中，我们将高估现象严格定义为，存在 N 使 $\mathbb{E}[Q^n] > Q^*$ 对所有的 $n > N$ 成立。为了对该现象建立基本理解，我们引入以下定理。

引理 5.1

考虑多个随机变量 $X_1, X_2, \cdots, X_n, Y$。定义 $M_1 \triangleq \{j \,|\, \mathbb{E}[X_j \,|\, Y] = \max_i \mathbb{E}[X_i \,|\, Y]\}$，$M_2 \triangleq \{j \,|\, X_j = \max_i X_i)\}$。此时以下关系成立

$$\mathbb{E}\left[\max_{1\leq i\leq n} X_i\right] \geq \mathbb{E}\left[\max_{1\leq i\leq n} \mathbb{E}\left[X_i \,|\, Y\right]\right]$$

且等号成立当且仅当 $P(M_1 \subset M_2) = 1$。

♡

该定理的一个特例是，$\mathbb{E}\left[\max_{1\le i\le n} X_i\right] \ge \max_{1\le i\le n} \mathbb{E}\left[X_i\right]$。因此，如果算子 $\mathcal{T}$ 需要计算 $\max_{a\in\mathcal{A}} Q^n(s,a)$，则必然存在 $\mathbb{E}[\max_{a\in\mathcal{A}} Q^n(s,a)] \ge \max_{a\in\mathcal{A}} \mathbb{E}[Q^n(s,a)]$ 的不等关系。此不等关系可能导致高估现象。比如，考虑 $\mathcal{T}^n = \mathcal{T}^*$，有 $\mathbb{E}[Q^{n+1}] = \mathbb{E}[\mathcal{T}^* Q^n] \ge \mathcal{T}^* \mathbb{E}[Q^n]$。如果 $\mathbb{E}[Q^0] = Q^*$ 且存在 Q^n 使得 $\mathbb{E}[\mathcal{T}^* Q^n]$ 与 $\mathcal{T}^* \mathbb{E}[Q^n]$ 的取等条件不成立，则易证高估现象存在（利用算子 $\mathcal{T}^*$ 的单调性）。但同时，我们也注意到，即使 $\mathbb{E}\left[\mathcal{T}^n Q^n\right] > \mathcal{T}^n \mathbb{E}[Q^n]$ 对所有的 n 都成立，依然无法保证高估现象的必然发生，甚至可能出现 $\mathbb{E}[Q^n] < Q^*$ 对所有 n 成立的情况（见后文例子一）。为了更深入地理解高估现象，我们给出了一个充分条件。

定理 5.1

考虑算法一，如果存在 $\epsilon > 0$ 和一个不动点为 Q^* 的压缩映射 $\widetilde{\mathcal{T}}$，使得对任意 n 满足以下条件

1. 对任意 $w \in \mathcal{W}$ 和随机变量 Y^n，有 $\mathbb{E}\left[\mathcal{T}(w, Q^n)\right] \ge \mathbb{E}\left[\mathcal{T}(w, \mathbb{E}[Q^n \mid Y^n])\right]$；
2. 对任意给定的值函数 Q，有 $\mathbb{E}\left[\mathcal{T}^n Q\right] = \mathbb{E}\left[\mathcal{T}(W^n, Q)\right] = \widetilde{\mathcal{T}} Q$；
3. 对任意 $w \in \mathcal{W}$，给定的值函数 Q 和标量 $r > 0$，$\mathcal{T}(w, Q - re) \ge \mathcal{T}(w, Q) - re$，其中 e 是恒为 1 的函数；
4. 对任意给定的两个值函数 Q_1 和 Q_2，如果 $Q_1 \ge Q_2$，则 $\mathcal{T}(w, Q_1) \ge \mathcal{T}(w, Q_2)$；
5. $\mathbb{E}\left[\mathcal{T}^{n+1}\mathcal{T}^n Q^*\right] = \mathbb{E}[\mathcal{T}\left(W^{n+1}, \mathcal{T}(W^n, Q^*)\right)] \ge Q^* + 2\epsilon e$；

则执行算法一时，必然存在正数 N，使得 $\mathbb{E}[Q^n] \ge Q^* + \epsilon e$ 对所有的 $n > N$ 成立。♡

取算子 $\mathcal{T}^n$ 和 $\widetilde{\mathcal{T}}$ 恒为 $\mathcal{T}^*$，则前 4 个条件成立。但此时，不一定出现高估现象（见后文例子一）。所以，我们添加了条件 5。其含义是，对最优值函数 Q^* 连续使用两次随机算子，必会出现高估现象。此外，该定理还为高估现象中正偏差的大小提供了一个下界。

结合定理一，我们可以讨论上一章节中两种改进方案对值函数学习的影响。

5.1.2.1 由数据采样的随机性导致的高估

为了讨论数据采样的影响，我们考虑如下式的 Q 值更新方式

$$\mathcal{T}(W^n, Q^n)(s,a) = Q^n(s,a) + \alpha\left(X^n(s,a) + \gamma \max_{a\in\mathcal{A}} Q^n(S^n(s,a),a) - Q^n(s,a)\right)$$

其中 $\alpha \in (0,1]$ 是学习率，$W^n = (X^n, S^n)$，$X^n(s,a)$ 从分布 $R(\cdot|s,a)$ 中随机采样，$S^n(s,a)$ 从分布 $P(\cdot|s,a)$ 中随机采样。我们假设，$Q^0, W^1, W^2, \cdots$ 之间相互独立。

对任意给定的值函数 Q，我们定义 $\widetilde{\mathcal{T}} Q \triangleq (1-\alpha)Q + \alpha\mathcal{T}^* Q$。易验证 $\widetilde{\mathcal{T}}$ 是不动点为 Q^* 的压缩映射，且定理一中的前 4 个条件成立。而条件 5 的成立依赖于 X^n 或 S^n 的随机性。当其随机性较强，第二次使用随机算子时会导致严格的正偏差，即有条件 5 满足。此时，无论我们使用何种初始化方式，都会出现高估现象。我们在后文提供了一个简单的例子进行说明（例子二）。

5.1.2.2 由函数近似的误差导致的高估

为了讨论函数近似的影响，我们考虑如下式的 Q 值更新方式

$$\mathcal{T}(W^n, Q^n) = F(W^n, \mathcal{T}^* Q^n)$$

其中 F 会将值函数投影到由所有函数近似器组成的集合 $\mathcal{Q}$ 内，如 $F(w, Q) = \arg\min_{Q' \in \mathcal{Q}} \|Q' - Q\|_2$。当采用随机梯度下降算法时，我们可以认为，$W^n$ 表示此次函数优化中多个数据批次 (batch) 组成的序列。

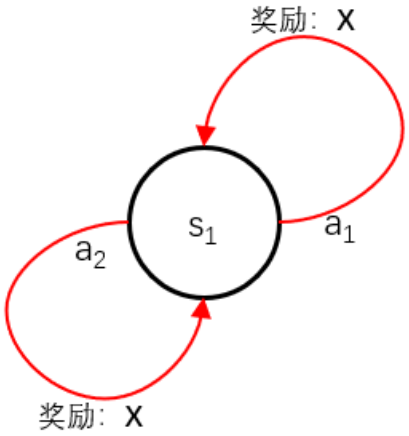
在真实的优化过程中，定理一中的条件往往很难满足，也难以保证高估现象一定出现。一个简单的例子是，当 $Q^* > 0$ 且任意 $Q \in \mathcal{Q}$ 均有 $Q < 0$ 时，必然不会出现高估现象。之前的工作假设 $F(W^n, \mathcal{T}^* Q^n) = \mathcal{T}^* Q^n + U^n$，其中 U^n 的期望为 0 且独立于 Q^n 。该假设下，如果 U^n 的随机性足够强，可以满足定理一的所有条件，进而必然导致高估现象。相关现象的实证，可以参考近年来的相关论文 [2−5]。

5.1.3 几个例子

本章节中，我们通过几个例子加深对高估现象的理解。首先，我们举例说明，取最大值的操作不一定导致高估现象。其次，我们给出一个一定会出现高估现象的例子。然后，我们结合例子说明，高估现象是如何影响策略学习的。

5.1.3.1 例子一：不出现高估现象的情况

我们考虑一个单状态双动作的马尔可夫决策过程（如下图所示）。



该过程中，奖励 X 采样于 $[-c, c]$ 上的均匀分布。在这里，我们用二维向量表示值函数。向量的第一维对应动作 a_1 的价值，第二维对应 a_2 的价值。易知 $Q^* = (0, 0)$。同时，我们定义 $\mathcal{T}^n Q \triangleq \widetilde{\mathcal{T}} Q \triangleq (1-\alpha)Q + \alpha \mathcal{T}^* Q$，并按以下方式初始化 Q^0

$$P\left(Q^0 = (-1, 0)\right) = 0.5 P\left(Q^0 = (0, -1)\right) = 0.5$$

此时，易有

$$P\left(Q^n = (-(1-\alpha)^n, 0)\right) = 0.5 P\left(Q^n = (0, -(1-\alpha)^n)\right) = 0.5$$

则有

$$\mathbb{E}[Q^{n+1}] = -0.5 \times (1-\alpha) \times ((1-\alpha)^n, (1-\alpha)^n) \tag{5.1}$$

$$\mathcal{T}^n\mathbb{E}[Q^n] = -0.5 \times (1-\alpha+\alpha\gamma) \times ((1-\alpha)^n, (1-\alpha)^n) \tag{5.2}$$

故 $\mathbb{E}\left[Q^{n+1}\right] > \mathcal{T}^n\mathbb{E}[Q^n]$ 对任意 $n > 0$ 恒成立。且容易验证定理一中的前 4 个条件均成立。但 $\mathbb{E}\left[Q^n\right] < Q^*$ 同样对任意 $n > 0$ 恒成立，未出现高估现象。

5.1.3.2 例子二：一定出现高估现象的情况

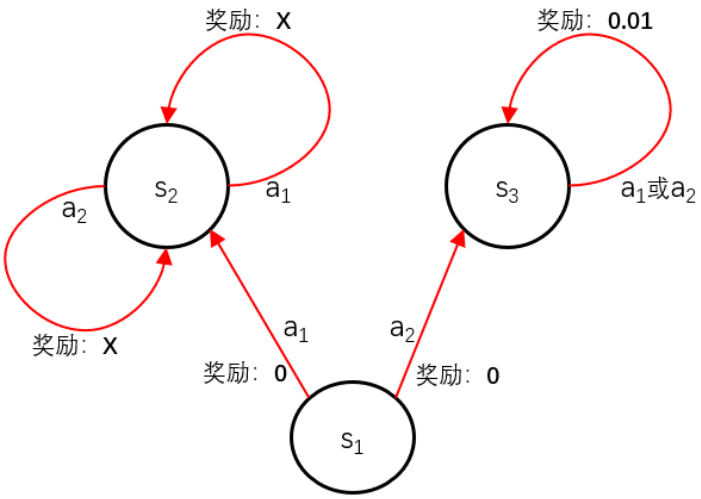
我们提供一个满足定理一的例子。同样考虑如例子一中的马尔可夫过程。奖励同样采样于 $[-c.c]$ 的均匀分布。我们采用前文讨论数据采样影响时定义的 $\mathcal{T}$，并取 $\alpha = 1, \gamma = 0.7$。此时，对任意 $a \in \{a_1, a_2\}$，有

$$\mathbb{E}[\mathcal{T}^{n+1}\mathcal{T}^nQ^*](s_1, a) = 0.7 \times \int_{-1}^{1} x\frac{1+x}{2}\mathbf{d}x = \frac{7}{30}$$

所有，取任意 $\epsilon < \frac{7}{60}$ 时，均有条件 5 成立。不同于例子一中 Q^n 的随机性随着 n 的增加而逐渐减小，例子二中 Q^n 一直存在较大的随机性，这也是例子二中必然出现高估现象的原因。类似地，当状态转移存在随机性时，也存在必然导致高估现象的例子. 相应的例子读者可以自己尝试构造。

5.1.3.3 例子三：高估现象影响策略学习的例子

我们考虑一个三状态双动作的马尔可夫决策过程（如下图所示）。



该过程中奖励 X 采样于 $[-c, c]$ 上的均匀分布。当采用例子二中的算法设置，对于状态 s_2，其价值会被高估，而对于状态 s_3，不会出现高估现象。因此，当 c 取值较大时，s_2 价值的期望必然会超过 s_3，此时有很大概率导致非最优的策略，即在 s_1 状态选择动作 a_1。

5.1.4 实验

Q-learning 算法是强化学习中最经典的算法之一，其算法的详细介绍可以参见 Sutton 的教材 [1]。Q-learning 同样可以使用算法一的框架统一表示。即

$$\mathcal{T}(W^n, Q^n)(s,a) = \begin{cases} Q^n(s,a) + \alpha(N^n)\Delta^n, & \text{if } (s,a) = (S^n, A^n), \\ Q^n(s,a), & \text{otherwise}, \end{cases}$$

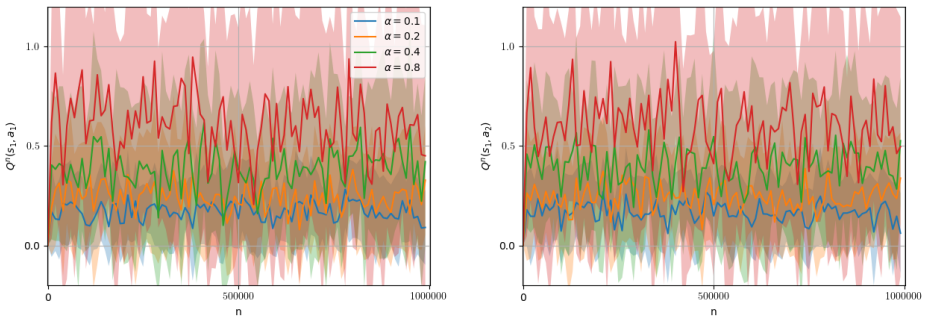
其中 $\Delta^n = X^n + \gamma \max_{a\in\mathcal{A}} Q^n(S^{n+1}, a) - Q^n(S^n, A^n)$；$S^n, A^n, X^n$ 表示依某个策略进行探索时，在第 n 个时刻访问到的状态、动作和奖励；N^n 表示在第 n 个时刻，已访问 (S^n, A^n) 的次数；α 是访问次数到学习率的函数；$W^n = (S^n, A^n, X^n, S^{n+1}, N^n)$。

本文的实验考虑例子一中单状态的马尔可夫决策过程。我们采用不同的设定，绘制训练中的Q值，以讨论不同设定下的高估现象。以下实验中，如未特殊说明，我们取 $\gamma = 0.7, c = 1, Q^0 = 0, r_{mean} = 0$。所有的曲线均采用 20 个随机种子的实验结果进行绘制。图中的实线部分表示不同随机种子下值函数的均值，阴影部分表示方差；每张图中，左侧的子图表示 $Q(s_1, a_1)$ 的变化趋势，右侧子图表示 $Q(s_1, a_2)$ 的变化趋势。

直接使用定理一讨论 Q-learning 的高估现象是困难的。主要原因是，$W^1, W^2, \cdots, W^n$ 相互之间并不独立。但当学习率固定时，直觉上可以认为，前文中数据采样导致的高估现象与 Q-learning 算法中的高估现象存在类似的性质。本章节中，如未特殊说明，我们默认取学习率为 $\alpha = 0.2$。

不同的学习率:

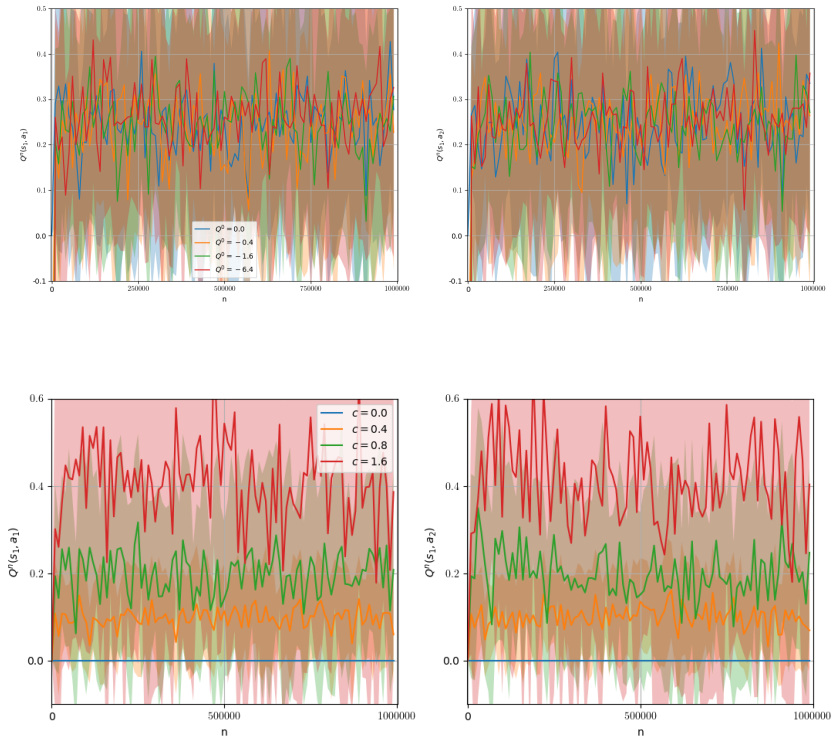
5.1.4.1 学习率固定为常数



本实验中，我们固定不同的 α，观察学习率对高估现象的影响。由图可知，当环境奖励随机时，学习率越高，高估现象越严重。我们可以通过定理一对该现象建立起初步的理解。具体地，可以认为学习率较大时，ϵ 可以有较大的取值，导致了更严重的高估现象。

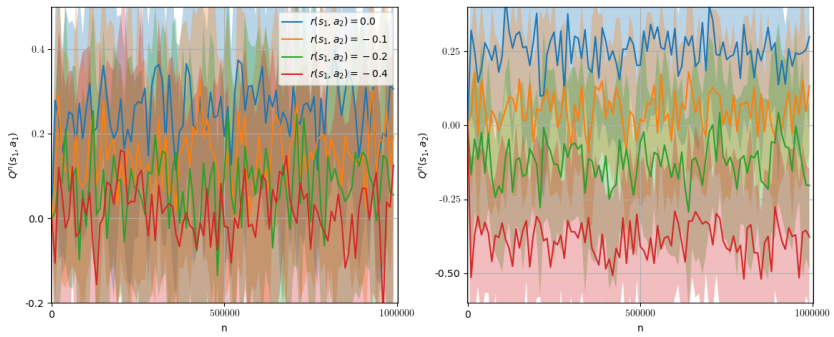
不同的初始价值: 本实验中，我们使用不同的固定的值初始化 Q^0。由图可见，高估现象的出现与值函数的初始化无关，不同的初始化方法最终都会出现相同程度的高估现象。这与定理一是基本一致的。

奖励随机性不同: 采用例子一中的环境时，参数 c 的大小决定了奖励随机性的大小。本



实验中，我们考虑不同的 c，研究奖励随机性对高估现象的影响。当环境不存在随机性时 $(c = 0)$，算法快速收敛到 Q^*，不存在高估现象。否则，存在高估现象，且奖励的随机性越大，高估现象越严重。该现象同样可以通过定理一进行解释。当奖励随机性越大，定理一中可取到的 ϵ 越大，进而有越大的高估现象。

动作之间价值的差异不同： 本实验中，每次执行动作 a_1 和 a_2 后，智能体均会收到采样



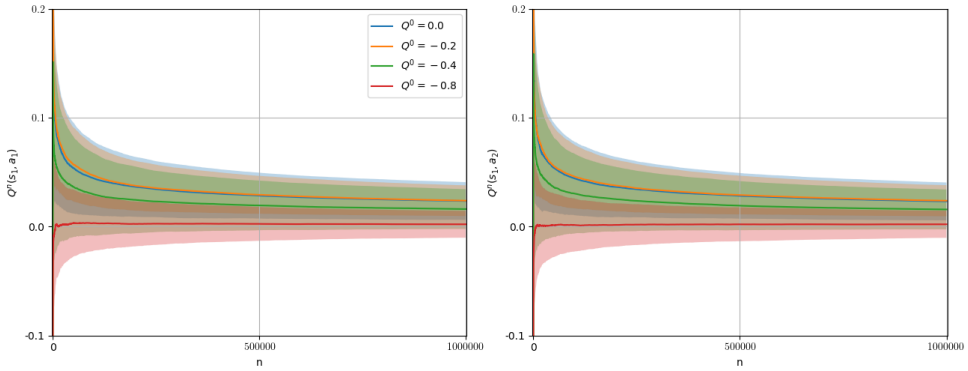
于 $[-1, 1]$ 均匀分布的奖励。此外，若执行动作 a_2，智能体会收到额外的、值为 $r(s_1, a_2)$ 的负奖励。此时，显然有 $Q^*(s_1, a_1) = 0$。$r(s_1, a_2)$ 越小，$Q^*(s_1, a_2)$ 越小，两个动作最优价值的差异就越大。

由图可知，当 $Q^*(s, a_1)$ 和 $Q^*(s, a_2)$ 的价值存在较大差异时，高估现象较弱。我们可以使用定理一理解该现象。当两动作 Q^* 差异较大时，ϵ 取值较小，高估现象相对较弱。

5.1.4.2 学习率随着训练递减:

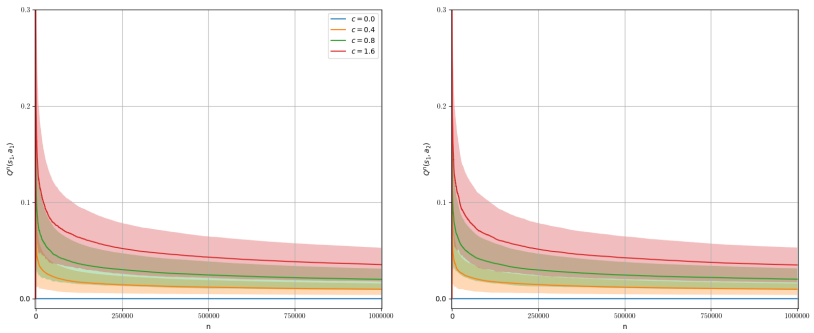
由前文的实验可见，固定学习率时，算法不一定收敛，其值函数可能一直存在较大的正偏差。而当学习率依照特定的方式递减时，大量工作证明了 Q-learning 的收敛性。此时，值函数会收敛到真实的 Q^*，不可能保持较大的正偏差。另一个角度来讲，当学习率递减时，定理一往往无法成立，因此难以得出正偏差的下界。所以，学习率递减时的高估现象与固定学习率时的高估现象存在一定差别。本章节中，除了学习率之外，我们均采用与前文相同的实验设定。此外，我们默认取学习率为 $\alpha = \frac{1}{N}$。

不同的初始价值: 由图可知，由于正偏差的存在，初始化 Q^0 为 Q^* 并不能较快地收敛。



而略低于 Q^* 的初始值反而更有利于收敛。不同于固定学习率，使用不同初始值函数，会导致不同程度的正偏差。

奖励随机性不同: 从图中可以看出，与使用固定的学习率类似，奖励随机性越大，正偏

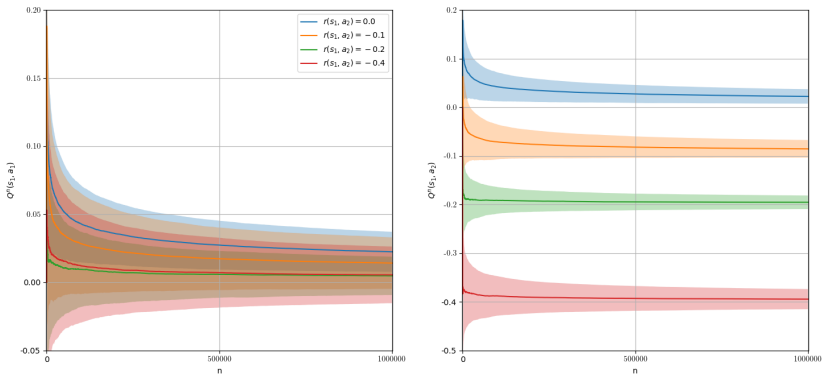


差越大，进而导致 Q 值收敛越慢。

动作之间价值的差异不同: 由图可知，当 $Q^*(s, a_1)$ 和 $Q^*(s, a_2)$ 的价值存在较大差异时，正偏差较大，进而导致较快的收敛速度。

5.1.5 总结

本文首先给出了高估现象存在的一个充分条件，然后结合示例讨论了不同因素对高估现象的影响情况。一般认为，高估现象会影响学习效率。作为对比，使用"Optimism in the face of uncertainty" 的探索策略时不确定的状态会有偏高的估值，但该估值会随访问次



数增加趋近于真实值。然而，高估现象不一定会出现随访问次数增加而减弱的趋势（如固定学习率的 Q-learning）。一些被频繁访问的状态也可能出现高估现象。本文的一个简单启示是，降低单次更新引入的随机性可以有效降低高估现象。具体到算法层面，可以在每次更新时使用更多的数据。

5.1.6 证明

引理一

我们考虑任意的函数 f, g，使对任意 $M \subset \{1, 2, \cdots, n\}$，均有 $f(M) \in M$ 且 $g(M) \in M$。我们定义随机变量 $I = f(M_1), J = g(M_2)$。则有

$$\begin{aligned}\mathbb{E}\left[\max_{1\le i\le n} X_i\right] &= \mathbb{E}\left[X_J\right] \\ &= P(I \in M_2)\mathbb{E}\left[X_J | I \in M_2\right] + P(I \notin M_2)\mathbb{E}\left[X_J | I \notin M_2\right] \\ &= P(I \in M_2)\mathbb{E}\left[X_I | I \in M_2\right] + P(I \notin M_2)\mathbb{E}\left[X_J | I \notin M_2\right]\end{aligned}$$

$$\begin{aligned}\mathbb{E}\left[\max_{1\le i\le n} \mathbb{E}\left[X_i \mid Y\right]\right] &= \mathbb{E}\left[\mathbb{E}\left[X_I \mid Y\right]\right] \\ &= \mathbb{E}\left[X_I\right] \\ &= P(I \in M_2)\mathbb{E}\left[X_I | I \in M_2\right] + P(I \notin M_2)\mathbb{E}\left[X_I | I \notin M_2\right]\end{aligned}$$

对比以上两式子，因 $\mathbb{E}\left[X_J | I \notin M_2\right] > \mathbb{E}\left[X_J | I \notin M_2\right]$ 恒成立，故 $\mathbb{E}\left[\max_{1\le i\le n} X_i\right] \ge \mathbb{E}\left[\max_{1\le i\le n} \mathbb{E}\left[X_i \mid Y\right]\right]$ 恒成立。此外，等号成立当且仅当 $P(I \notin M_2) = 0$，即 $P(I \in M_2) = 1$。由于函数 f 的任意性，我们可以得到，等号成立当且仅当 $P(M_1 \in M_2) = 1$。

定理一我们首先证明，必存在 N_1，使得 $\mathbb{E}[Q^n] \ge Q^* - \epsilon e$ 对所有 $n > N_1$ 均成立。我们令 $L^0 = \mathbb{E}[Q^0]$，$L^{n+1} = \widetilde{\mathcal{T}} L^n$。则由于 Q^* 是 $\widetilde{\mathcal{T}}$ 的不动点，故序列 $(L^n)_{n=0}^{\infty}$ 收敛至 Q^*。故必存在 N_1，使得 $L^n \ge Q^* - \epsilon e$ 对所有的 $n > N_1$ 均成立。又 $\mathbb{E}[Q^0] \ge L^0$。同时，如果 $\mathbb{E}[Q^n] \ge L^n$ 成立，则有

$$
\begin{aligned}
\mathbb{E}[Q^{n+1}] &= \mathbb{E}[\mathcal{T}(W^n, Q^n)] \\
&\geq \mathbb{E}\left[\mathcal{T}\left(W^n, \mathbb{E}\left[Q^n\right]\right)\right] && 1 \\
&\geq \mathbb{E}\left[\mathcal{T}\left(W^n, L^n\right)\right] && 4 \\
&= \widetilde{\mathcal{T}} L^n && 2 \\
&= L^{n+1}
\end{aligned}
$$

进而，通过数学归纳法可知，$\mathbb{E}[Q^n] \geq L^n$ 对所有 $n > 0$ 成立。此时，$\mathbb{E}[Q^n] \geq L^n \geq Q^* - \epsilon e$ 对所有 $n > N_1$ 成立。

再证明，$\mathbb{E}\left[\mathcal{T}^{n+1}\mathcal{T}^n(Q^* - \epsilon e)\right] \geq Q^* + \epsilon e$。我们有

$$
\begin{aligned}
\mathbb{E}\left[\mathcal{T}^{n+1}\mathcal{T}^n(Q^* - \epsilon e)\right] &\geq \mathbb{E}\left[\mathcal{T}^{n+1}(\mathcal{T}^n Q^* - \epsilon e)\right] && 3 \\
&\geq \mathbb{E}\left[\mathcal{T}^{n+1}\mathcal{T}^n Q^* - \epsilon e\right] && 3 \\
&= \mathbb{E}\left[\mathcal{T}^{n+1}\mathcal{T}^n Q^*\right] - \epsilon e \\
&\geq Q^* + 2\epsilon e - \epsilon e && 5 \\
&= Q^* + \epsilon e
\end{aligned}
$$

接着，我们证明，取 $N = N_1 + 2$ 时，必有 $\mathbb{E}[Q^n] \geq Q^* - \epsilon e$ 对任意 $n > N$ 成立。我们有

$$
\begin{aligned}
\mathbb{E}[Q^n] &= \mathbb{E}[\mathcal{T}^n\mathcal{T}^{n-1}Q^{n-2}] \\
&= \mathbb{E}[\mathcal{T}^n\mathcal{T}(W^{n-2}, Q^{n-2})] \\
&\geq \mathbb{E}\left[\mathcal{T}^n\mathbb{E}\left[\mathcal{T}\left(W^{n-2}, Q^{n-2}\right) \mid W^{n-2}\right]\right] && 1 \\
&\geq \mathbb{E}\left[\mathcal{T}^n\mathbb{E}\left[\mathcal{T}\left(W^{n-2}, \mathbb{E}\left[Q^{n-2}\right]\right) \mid W^{n-2}\right]\right] && 1 \\
&= \mathbb{E}\left[\mathcal{T}^n\mathcal{T}\left(W^{n-2}, \mathbb{E}\left[Q^{n-2}\right]\right)\right] \\
&= \mathbb{E}\left[\mathcal{T}^n\mathcal{T}^{n-2}\mathbb{E}\left[Q^{n-2}\right]\right] \\
&\geq \mathbb{E}\left[\mathcal{T}^n\mathcal{T}^{n-2}(Q^* - \epsilon e)\right] && 4 \\
&\geq Q^* + \epsilon e
\end{aligned}
$$

综上，我们有定理一成立。

[1] Sutton, Richard S., and Andrew G. Barto. "Reinforcement learning: An introduction". MIT press, 2018.

[2] Thrun, Sebastian, and Anton Schwartz. "Issues in using function approximation for reinforcement learning." *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*. 1993.

[3] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with

double q-learning." *arXiv preprint arXiv:1509.06461* (2015).

[4] Fujimoto, Scott, Herke Van Hoof, and David Meger. "Addressing function approximation error in actor-critic methods." *arXiv preprint arXiv:1802.09477* (2018).

5.2 DQN 中目标网络两种常见更新方法对比（石志皓-2020/11/11）

近年来，传统强化学习方法与深度学习相结合的研究在许多人工智能任务中取得突破性进展。大多数深度强化学习算法都需要一个值估计模块，来评估某一状态或者某一状态-动作对的价值。深度强化学习为了实现稳定的值估计，通常会引入目标网络 (Target Network) 这一技术。目标网络的更新方式一般分为两种：DQN[1] 算法提出的硬更新 (Hard Update) 和 DDPG[2] 算法提出的软更新 (Soft Update) 。在 OpenAI Baselines 和 RLkit 的算法实现中，DQN 算法 [1]、DDQN 算法 [3] 这些基于离散动作空间的算法，使用硬更新技术；DDPG[2]、TD3[4]、SAC[2] 这些基于连续动作空间的算法，使用软更新技术。目前，缺少足够的理论和实验比较两种更新方式的性能。

本文将通过实验，从收敛速度方面，评估两种更新方式下的 DQN 算法 [1] 的实际性能。实验结果表明，当两种更新方式的超参数满足一定条件的时候，两种更新方式下的 DQN 算法 [1] 的性能是相当的。

5.2.1 研究的实际问题

强化学习算法主要用来解决序列决策问题，而现有的深度强化学习算法常常不稳定。最常见的深度强化学习算法 DQN 增强稳定性的关键技术之一是目标网络技术。本次公众号投稿的主题是研究目标网络技术，以进一步增强深度强化学习算法的稳定性。

5.2.2 DQN 简介

强化学习的经典算法，Q-learning，通过直接学习最优 Q 函数

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s, a_t = a, \pi],$$

进而得到相应的最优策略。最优 Q 函数的目标是最大化一个累积折扣奖励，其中 γ 为折扣因子，奖励序列 $(r_t, r_{t+1}, \dots)$ 通过从状态 $s_t = s$ 出发，第一步执行动作 $a_t = a$，之后执行策略 π 的过程获得。

标准的 Q-learning 算法主要是解决状态和动作空间离散且空间不大的问题。在这样的简单问题中，可以使用表格来存储每个状态-动作对对应的价值 (Q 值)。然而，对于状态空间极大或者状态空间是连续的时候，使用 Q 表来存储每个状态-动作对的价值 (Q 值) 就不现实了。

一个解决思路是使用神经网络来拟合最优 Q 函数。对于状态连续、动作空间离散的问题，这一神经网络的输入是状态，输出是分别执行每个动作的价值 (Q 值) 。之后本文把这一网络叫做 Q 网络。

5.2.3 贝尔曼最优算子

了解了如何用 Q 网络来拟合 Q 函数，还需要一个方法来训练 Q 网络。一个常用方法是通过求解贝尔曼最优方程

$$\begin{aligned} Q^*(s,a) &= r(s,a) + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)}[\max_{a'} Q^*(s',a')] \\ &= (\mathcal{T}Q^*)(s,a), \end{aligned}$$

来找到最优 Q 函数，其中 $\mathcal{T}$ 是贝尔曼最优算子;$p(s'|s,a)$ 代表环境的状态转移函数，即在状态 s，执行动作 a 后，转移状态的概率分布。

贝尔曼最优算子 $\mathcal{T}$ 是压缩映射。对任意初始化的 Q 函数，可以通过不动点迭代求解最优 Q 函数，即 $\mathcal{T}^n Q \to Q^*$。训练 Q 网络的核心技术难题是如何在连续的状态空间中有效近似贝尔曼最优算子 $\mathcal{T}$。

5.2.3.1 基于贝尔曼最优算子的 Q 函数训练

DQN[1] 提出用监督学习的训练方法来近似贝尔曼最优算子 $\mathcal{T}$，进而通过近似的不动点迭代寻找满足贝尔曼最优方程的 Q 网络。DQN 算法 [1] 引入目标网络 (target network) 技术来计算目标标签，然后把近似贝尔曼最优算子问题转化为回归问题。这样的监督学习训练方法能使得算法在实践中表现更加稳定。

DQN 算法 [1] 会不断与环境交互获得一个不断更新的经验回放池 $D_i = \{(s_n, a_n, r_n, s_{n+1})\}_{n=1}^{i}$，用 $(s,a,r,s') \sim U(D_i)$ 表示从 D_i 中均匀采样，用 $Q(s,a;\theta)$ 表示 Q 网络，其中 θ 是神经网络参数。在第 i 次迭代的 Q-learning 更新中，算法先利用目标网络 $\hat{Q}(s',a';\theta_i^-)$ 计算目标标签 $y(r,s') = r + \gamma \max_{a'} \hat{Q}(s',a';\theta_i^-)$，然后基于动态的经验回放池 D_i 使用如下损失函数

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D_i)}[y(r,s') - Q(s,a;\theta_i))^2]$$

来进行随机梯度更新，θ_i^- 是目标网络的参数，它会每隔固定的 C 步用 Q 网络参数 θ_i 同步更新一次。如果神经网络泛化性能足够好，DQN 算法 [1] 的每 C 步迭代就相当于使用近似贝尔曼算子 $\mathcal{T}$ 更新一次 Q 函数，即 $Q(\cdot,\cdot;\theta_{i+C}) \approx \mathcal{T}Q(\cdot,\cdot;\theta_i)$。

DQN[1] 的伪代码如下图所示

5.2.4 关于目标网络的简单分析

本章节讨论 DQN[1] 中的关键技术之一，即目标网络。

5.2.4.1 目标网络的硬更新

DQN 算法 [1] 引入目标网络的目的是为了增加算法稳定性。目标网络的引入使得计算的目标标签 $y(r,s') = r + \gamma \max_{a'} \hat{Q}(s',a';\theta_i^-)$ 在每 C 步的更新中都保持相对稳定。目标网络更新间隔 C 越大，算法就会越稳定，目标网络更新频率越慢，算法收敛速度会越

Algorithm 1: deep Q-learning with Experience Replay

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with random weights $\theta^- = \theta$
for episode $= 1, \ldots, M$ **do**
  for $t = 1, \ldots, T$ **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
    Execute action a_t in emulator and observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ with respect to the network parameter θ
    Every C steps reset $\hat{Q} = Q$

图 5.1: DQN 伪代码。

慢。一个合适的目标网络更新间隔 C，能让 DQN 算法 [1] 训练既稳定又快速。这种将网络参数整体替换的更新方式被称为硬更新，在 DQN[1] 及其之后的改进算法被经常使用。

5.2.4.2 目标网络的软更新

DDPG[2] 提出了另一种软更新思路，保证目标网络在每次迭代中都会更新，相当于网络更新间隔为 1。软更新利用当前网络参数与目标网络参数的凸组合来更新网络，即目标网络的参数 θ_i^- 会用当前网络参数 θ_i 按照如下公式更新

$$\begin{aligned} \theta_{i+1}^- &\leftarrow (1-\epsilon)\theta_i^- + \epsilon\theta_i \\ &= \theta_i^- + \epsilon(\theta_i - \theta_i^-) \end{aligned}$$

其中 $0 < \epsilon << 1$。这样目标网络的参数变化较小，计算的目标标签值 $y(r, s')$ 变化也会相对平缓。这样即使每次迭代都更新目标网络，算法也能保持一定的稳定性。相似的，软间隔更新系数 ϵ 越小，算法会越稳定，目标网络参数变化越小，算法收敛速度会越慢。一个合适的软间隔更新系数 ϵ，也能让 DQN 算法 [1] 训练既稳定又快速。

基于 DQN[1] 的改进算法经常使用硬更新，如 DDQN[3]；基于 DDPG[2] 的改进算法经常使用软更新，如 TD3[4]、SAC[2]。本文的实验部分发现当超参数满足一定条件的时候，DQN 算法 [1] 使用这两种目标网路更新方式的性能是相当的。

5.2.5 实验验证

在实验部分，本章节用实验验证前面分析中提出的几点论述。

1. 在硬更新中，目标网络更新间隔 C 越大，算法收敛速度会越慢。2. 在软更新中，软间隔更新系数 ϵ 越小，算法收敛速度会越慢。3. 两种更新方式的性能是相当的。

本文在 Openai Gym[5] 的两个环境：Acrobot 和 CartPole，进行实验。在训练中，环境

超参数	取值
经验回放池大小	50000
mini-batch 大小	32
学习率	0.0005
折扣因子	0.99

表 5.1: 其他超参数设置。

如果在交互 200 次后没有重置，会主动重置环境。本实验使用 3 层的的全连接网络，并且其中的 2 个隐藏层都有 64 个神经元，都使用 Tanh 作为激活函数，使用 Adam 优化器进行优化。探索部分使用 ϵ-greedy 算法进行探索，初始 $\epsilon = 0.1$，在与环境交互 10 万次的过程中线性下降到 $\epsilon = 0.02$，之后保持 $\epsilon = 0.02$ 进行探索。其他的超参数在下面的表格列出。

每个实验使用 5 个随机种子进行 5 次实验，每个实验总共与环境交互 30 万步。每当与环境交互 3000 步，就会评估一次当前策略的性能。评估方法是利用当前策略与测试环境交互得到 100 条轨迹，计算每条轨迹的总奖励。测试环境每条轨迹的最大步数设置为 1000。

本实验绘制了交互次数-平均总奖励曲线。为了方便比较，作图时每个点取了附近的 30 个点进行均值平滑处理。图中的实线是 5 个随机种子得到的结果的均值，阴影区域的上界表示均值加相应的标准差，阴影区域的下界表示均值减相应的标准差。

5.2.5.1 硬更新中目标网络更新间隔与算法性能的关系

本实验设置目标网络更新间隔 C 分别为 1000、500、200。实验结果如下图所示。

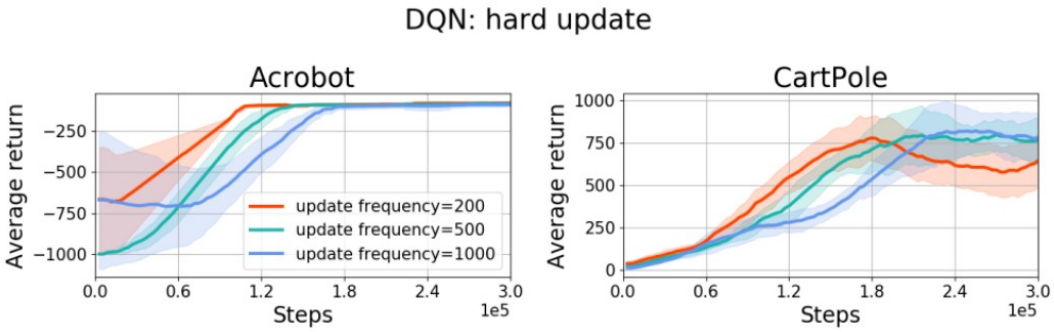


图 5.2: DQN:hard update

在两个实验环境中，随着目标网络更新间隔 C 增大，算法收敛速度会越慢，算法稳定性会提高。

5.2.5.2 软更新中目标网络更新间隔与算法性能的关系

本实验设置软间隔更新系数 ϵ 分别为 0.005、0.002、0.001。实验结果如下图所示。

在两个实验环境中，随着软间隔更新系数 ϵ 减小，算法收敛速度会越慢。

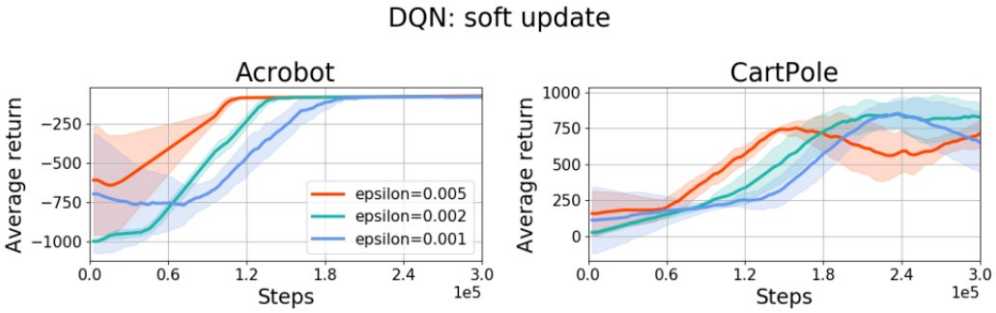


图 5.3: DQN:soft update

5.2.5.3 硬更新与软更新算法性能对比

本文在实验中发现，在 Acrobot 和 CartPole 两个任务中，当目标网络更新间隔 C 和软间隔更新系数 ϵ 满足 $\epsilon C = 1$ 时，硬更新和软更新能得到相似的实验结果。硬更新方式和软更新方式在实验结果上的相似性暗示着，这两种方式可能在优化方面有着更深层次的联系，需要未来的研究进行发掘。

5.2.6 总结

本文在几个简单的实验环境中研究了深度强化学习的价值估计模块中硬更新和软更新算法, 以及两种算法的性能差异。实验结果表明，当目标网络更新间隔 C 和软间隔更新系数 ϵ 满足 $\epsilon C = 1$ 时，DQN 算法 [1] 使用这两种目标网路更新方式的性能是相当的。这结果暗示着这两种方式可能在优化方面有着更深层次的联系，需要未来的研究进行发掘。

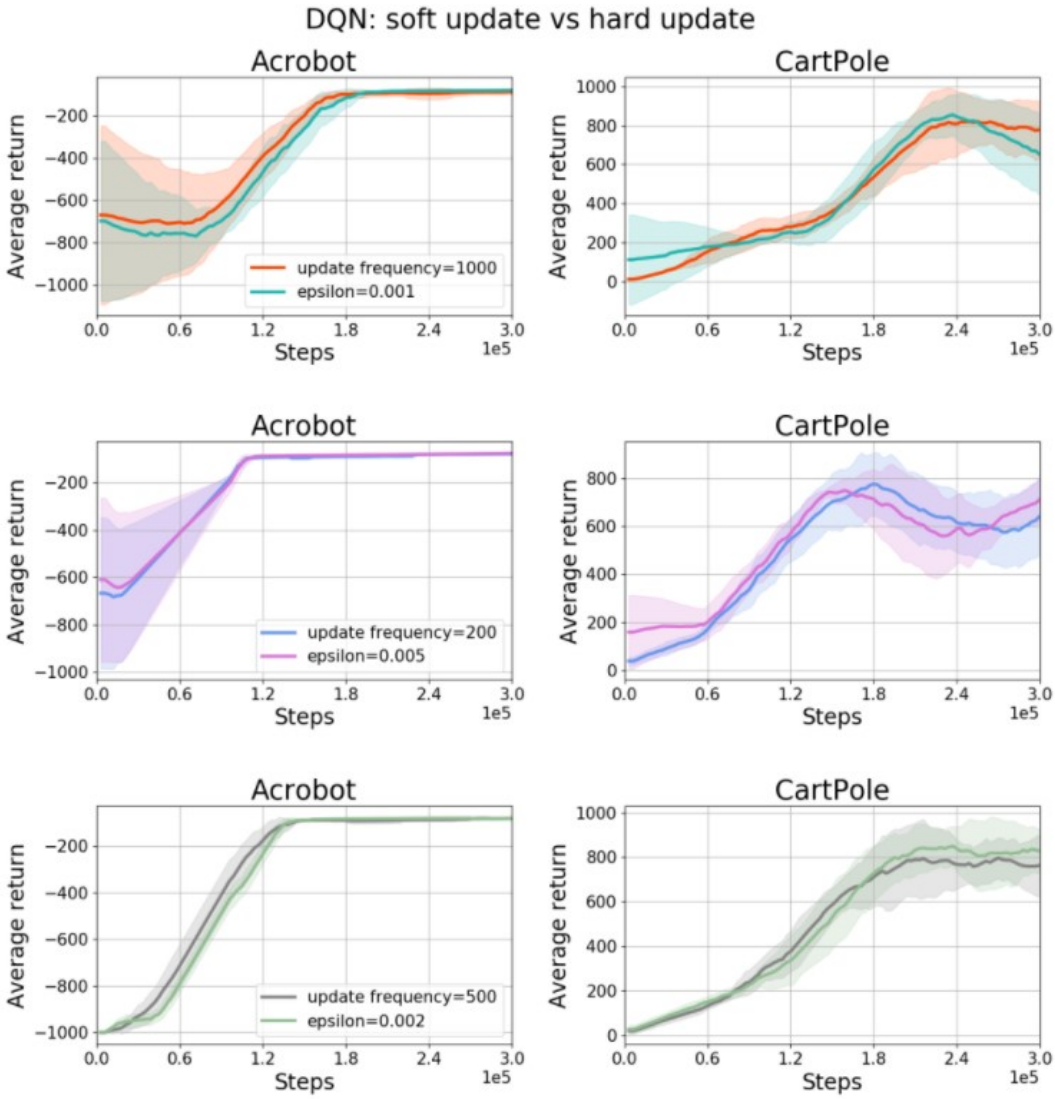


图 5.4: DQN:soft update vs hard update

第六章 策略梯度方法（内容待补充）

第三部分

强化学习进阶

第一章 实际案例（内容待补充）

1.1 游戏 AI——我的世界

1.2 现实应用——智慧交通

第 二 章 强化学习中的安全性（内容待补充）

2.1 概述

2.2 基于模型的强化学习

2.3 鲁棒强化学习

第 三 章 强化学习中的探索

3.1 概述（内容待补充）

3.2 正则强化学习（周祺、匡宇飞-2020/11/17）

NeurIPS 2020 | 一种基于动作采样的简单高效的正则化强化学习方法

3.2.1 引言

近年来，强化学习算法在游戏智能、机器人控制等领域取得了令人瞩目的成果。传统的强化学习方法最大化累积回报的期望，最后习得的策略往往接近于一个确定性策略[6]。然而，相比于确定性策略，随机策略更有利于探索未知环境，且在环境参数发生变化时具有更好的鲁棒性 [7-8]，因此我们更希望训练得到的策略是随机策略。

为了促进策略的随机性，过往工作使用了熵正则化方法。该类方法在最大化累积奖励的同时，最大化动作分布的熵。如，soft Q-learning [9] 和 SAC [8, 10] 使用 Shannon 熵作为正则项；sparse PCL [11] 和 TAC [12] 使用 Tsallis 熵作为正则项。

然而，在考虑连续的工作空间时，熵正则化的强化学习方法会陷入表达能力有限的简单策略与复杂低效的训练过程之间的两难选择。例如，SAC 往往使用简单的高斯分布表示策略，而 soft Q-learning 需要复杂低效的采样和推理过程来优化策略。

为解决以上问题，我们提出了一类新的正则化方式。进而，1) 在使用复杂策略时也能高效地估计该正则项的值；2) 该正则项能够广泛兼容一般的策略结构。

3.2.2 背景介绍

我们考虑动作空间连续的马尔可夫决策过程，该过程可用五元组 $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ 表示，其中 $\mathcal{S}, \mathcal{A}$ 为相应的状态空间和动作空间，P 为状态转移函数，r 为奖励函数，γ 为折扣因子。此外，我们用 $\pi(\cdot|s)$ 表示策略在状态 s 下对应的动作分布。

在正则化强化学习框架中，需要在标准的强化学习目标函数的基础上增加一项关于策略的正则项 [13]。此时，其目标函数变为

$$\pi_\alpha^* \triangleq \arg\max_{\pi\in\Pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \alpha \mathcal{F}\left(\pi(\cdot|s_t)\right)\right)\right].$$

这里 α 为正则项所占权重的超参。相应地，该目标函数下的 Q 值函数和 V 值函数定义为

$$Q_\alpha^\pi(s,a) \triangleq \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t) + \alpha\gamma \mathcal{F}\left(\pi(\cdot|s_{t+1})\right)\right) \,\middle|\, s_0 = s, a_0 = a\right],$$

$$V_\alpha^\pi(s) \triangleq \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q_\alpha^\pi(s,a)\right] + \alpha\mathcal{F}(\pi(\cdot|s)).$$

<!– 相应的最优 Q 值函数和 V 值函数为 $Q_\alpha^* \triangleq \sup_{\pi\in\Pi} Q_\alpha^\pi$ 和 $V_\alpha^* \triangleq \sup_{\pi\in\Pi} V_\alpha^\pi$。–>

在基于熵的正则化强化学习方法里，$\mathcal{F}(\pi(\cdot|s_t))$ 通常为策略在该状态下的动作分布的 Shannon 熵或者 Tsallis 熵。

3.2.3 熵正则方法的局限性

在考虑连续的工作空间时，熵正则化的强化学习方法会陷入表达能力有限的简单策略与复杂低效的训练过程之间的两难选择。具体地，熵正则项往往具有

$$\mathcal{F}(\pi(\cdot|s)) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[f(\pi(a|s))\right]$$

的形式。其中 $\pi(a|s)$ 表示动作 a 在给定状态 s 时的概率密度。该形式的正则项会导致

- 熵正则的估计需要计算所选动作的概率密度（probability density），而使用复杂策略时其计算往往低效繁琐。例如，使用标准化流（normalizing flow）表征策略时 [14]，需要额外的串行过程计算概率密度；通过集成多个概率分布来表示策略时，需要计算每个分布的概率密度再进行平均。- 熵正则的定义往往需要动作分布具有连续的累积分布函数，而使用复杂策略时该函数可能并不连续。例如，使用基于狄拉克混合分布（Dirac mixture）表征策略 [15] 时，其动作的累积分布函数是阶梯状的不连续函数；使用噪声网络（noisy network）表征策略时，由于 Relu 激活函数的影响，动作分布的累积分布函数也可能出现不连续的情况。

3.2.4 基于样本的正则化方法

为了解决熵正则项的一系列不足，我们提出了基于样本的正则化方法（Sample Based Regularization：SBR）。本章节中，我们将描述其定义，给出具体的实例，并简单讨论该正则项的性质。

3.2.4.1 正则项的表达式

基于样本的正则项具有如下形式：

$$\mathcal{F}(\pi(\cdot|s)) \triangleq \mathbb{E}_{a\sim\pi(\cdot|s)}\left[f(a)\right] + \mathbb{E}_{a,a'\sim\pi(\cdot|s)}\left[g(a,a')\right].$$

我们之所以将其称为基于样本的正则项，是因为我们可以仅使用动作的采样估计该正则项的取值，而不要求概率密度函数存在或可计算。具体地，我们使用如下的无偏估计

$$\hat{\mathcal{F}}(\pi(\cdot|s)) = \frac{1}{N}\sum_{i=1}^{N}\left(f(a^i) + \frac{1}{N-1}\sum_{j=1,j\neq i}^{N} g(a^i,a^j)\right).$$

这里 N 为从分布 $\pi(\cdot|s)$ 中采样的动作个数，是人为设定的超参。基于样本的正则项中，$\mathbb{E}_{a,a'\sim\pi(\cdot|s)}\left[g(a,a')\right]$ 用于度量动作之间的相互影响。为了促进策略的随机性，我们通过其鼓励动作之间相互远离；而上式中的 $\mathbb{E}_{a\sim\pi(\cdot\|s)}\left[f(a)\right]$ 用于引导动作朝某种先验分布聚集。同时该项可以避免在前一项的作用下所有动作都分布于动作空间的边界。

表 3.1: 函数 ϕ 的部分实例

Function ϕ	Condition	Function ϕ	Condition
$\phi(z)=z^{\kappa}$	$0<\kappa<1$	$\phi(z)=\log(1+\kappa z)$	$0<\kappa$
$\phi(z)=\frac{z}{z+\kappa}$	$0<\kappa$	$\phi(z)=1-e^{-\kappa z}$	$0<\kappa$

表 3.2: 基于广义能量距离导出的 SBR 实例

$\mathcal{F}(q)$	$f(a)$	$g(a,a')$
$-d_\phi^2(q,u)$	$-2\mathbb{E}_{a'\sim u}\left[\phi(\|a-a'\|_2^2)\right]$	$\phi(\|a-a'\|_2^2)$
$-\tilde{d}_\phi^2(q,u)$	$-2\sum_{i=1}^n \mathbb{E}_{a'\sim u}\left[\phi\left(([a]_i-[a']_i)^2\right)\right]$	$\sum_{i=1}^n \phi\left(([a]_i-[a']_i)^2\right)$

3.2.4.2 基于广义能量距离的实例

3.2.4.2.1 广义能量距离 上节中我们给出了基于样本的正则项的表达式，在本节中，我们将基于广义能量距离给出上述正则项的一系列具体实例。

我们首先介绍广义能量距离。广义能量距离（generalized energy distance, GED）在统计推断中常被用来度量两个概率分布之间的一致性 [16]。假设 p,q 为两个概率分布，广义能量距离 d_ϕ 定义为

$$d_\phi^2(q,p)\triangleq 2\mathbb{E}_{x\sim q,y\sim p}\left[\phi(\|x-y\|_2^2)\right]-\mathbb{E}_{x,x'\sim q}\left[\phi(\|x-x'\|_2^2)\right]-\mathbb{E}_{y,y'\sim p}\left[\phi(\|y-y'\|_2^2)\right].$$

这里 ϕ 为满足一定条件的非负函数，其部分实例可参考下表 1：

进一步地，我们可以定义

$$\tilde{d}_\phi^2(q,p)\triangleq\sum_{i=1}^n d_\phi^2([q]_i,[p]_i).$$

这里 $[q]_i,[p]_i$ 为分布 q,p 在其取值空间中第 i 维的边缘分布。同样地，在对动作分布进行一定限制的情况下，$\tilde{d}_\phi$ 是一个描述概率分布之间距离的度量。

3.2.4.2.2 两个实例 我们注意到，分布 q 的 Shannon 熵满足

$$\mathcal{H}(q)=-D_{\mathrm{KL}}(q\,\|\,u)+C$$

其中，D_{KL} 为 KL 散度，u 是一个均匀分布，C 是常数项。也就是说，添加熵正则项实际是鼓励缩小动作分布与均匀分布之间的差异。因此，我们定义正则项

$$\mathcal{F}(\pi(\cdot|s))=-d^2(q,u),$$

其中距离 d 可以取 d_ϕ 或 $\tilde{d}_\phi$。此时，正则项具有 SBR 的形式。具体对应关系如表 2：

上表中的两类正则项实例的详细理论分析，以及其在单状态摇臂机问题中的可视化展现，请参考本论文原文 3.2 3.5 节。

3.2.5 基于能量距离的 actor-critic 算法

基于上文中的正则项实例，我们提出了基于广义能量距离的强化学习算法：Actor Critic with generalized Energy Distance (ACED)。该方法使用

$$\mathcal{F}(\pi(\cdot|s)) = -\tilde{d}^2_\phi(\pi(\cdot|s), u)$$

作为正则项，其算法流程与 SAC 基本一致。

但不同于 SAC 算法，ACED 算法具有如下特点：1) 对动作分布的类型几乎没有任何限制，能够广泛地兼容各种不同的策略结构；2) 正则项值的估计基于动作的采样，不需要概率密度值的显式计算，因此能够显著提升复杂策略下正则项值的计算效率。

3.2.6 实验结果

（注：本节仅选取部分实验结果，更详细的结果请参考本论文原文第 5 节。）

3.2.6.1 算法性能比较

我们在 6 个不同的 MuJoCo 仿真控制任务下比较了 ACED 算法与 SAC[10]、TD3 [4]、DDPG [2] 等算法的性能差异，实验结果如下图。在绝大多数任务中，ACED 算法取得了优于基准算法的性能。

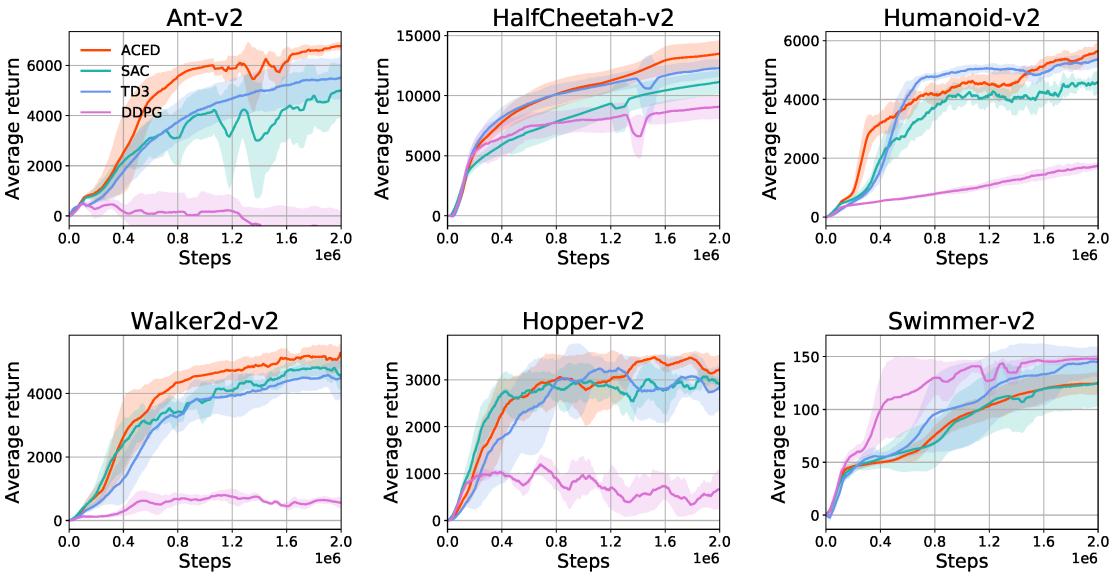


图 3.1: 6 个不同任务下 ACED 算法与 SAC、TD3、DDPG 等算法的性能比较

3.2.6.2 算法效率比较

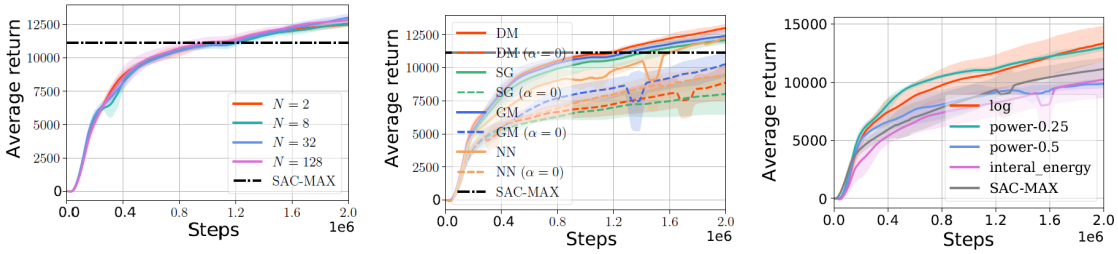
我们同样比较了 ACED 算法（使用基于广义能量距离的正则项）和 SAC 算法（使用基于 Shannon 熵的正则项）在不同策略结构下的计算效率，实验结果如下表 3。可以看出，ACED 算法在计算正则项时增加动作采样数不会明显增加计算开销；且在使用更为复杂的策略结构时，ACED 算法相比 SAC 算法在计算效率上具有明显优势。

表 3.3: 不同策略结构、不同动作采样数下 ACED 算法与 SAC 算法的性能比较。表中所列时间为 $2.0m$ 步训练用时，SG 为基于高斯分布的策略，NF 为基于标准化流网络的策略

Algorithm	**ACED**			**SAC**	
Policy	SG ($N=2$)	SG ($N=32$)	NF ($N=2$)	SG	NF
Time (h)	9.80 ± 0.04	11.14 ± 1.25	12.27 ± 0.80	10.05 ± 0.05	18.75 ± 0.03

3.2.6.3 对比和消融实验

最后，我们对 ACED 算法进行了详细的对比和消融实验。我们首先考虑不同的超参数 N, 即使用不同数量的动作采样估计正则项（如下图 a），我们发现 ACED 算法对动作采样数不敏感，即使在 $N=2$ 的情况下算法的表现仍然很好；接着我们对比了使用/不使用正则项时的性能（如下图 b），实验证明基于样本的正则项在不同策略结构下都有助于提升学习效率；最后我们对比了在正则项中使用不同的函数 ϕ 对性能的影响差异（如下图 c），结果显示选择合适的函数 ϕ 对 ACED 算法的性能较为重要。



(a) Sample number (b) Policy architecture (c) Regularization term

图 3.2: ACED 算法的各项参数的详细的对比和消融实验，所有实验结果均基于 HalfCheetah-v2 任务。图中 SAC-MAX 为 SAC 算法在 $2.0m$ 步训练中的最佳性能；图 b 中 DM、NN、GM 分别对应狄拉克混合策略、基于噪声网络的策略、基于生成模型的策略；图 c 中，log、power-0.25、power-0.5、interal_energy 分别对应正则项使用 $\phi(z) = \log(z+\epsilon)$、$\phi(z) = z^{0.25}$、$\phi(z) = z^{0.5}$ 以及仅包含 $g(a, a')$ 的正则项

3.2.7 总结

在本文中，我们提出了一种基于动作采样的正则项 SBR，并基于广义能量距离（GED）给出了该正则项的一系列实例。SBR 作为熵正则的一种替代方案，能够广泛兼容各种复杂的策略结构，并具备计算高效、样本效率高等诸多优势。然而，“是否还能找到其他更好的 SBR 实例?”，“广义能量距离能否应用于强化学习的其他任务?”，这些问题仍待进一步解决和完善，我们也欢迎大家进行相关研究和讨论。

参考文献

[1] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. nature, 2015, 518(7540):529-533.

[2] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning.[C]// ICLR (Poster). [S.l.: s.n.], 2016.

[3] VAN HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double q-learning[J]. arXiv preprint arXiv:1509.06461, 2015.

[4] FUJIMOTO S, VAN HOOF H, MEGER D. Addressing function approximation error in actor-critic methods [J]. arXiv preprint arXiv:1802.09477, 2018.

[5] BROCKMAN G, CHEUNG V, PETTERSSON L, et al. Openai gym[J]. arXiv preprint arXiv:1606.01540, 2016.

[6] SUTTON R S, BARTO A G. Reinforcement learning: An introduction[M]. [S.l.]: MIT press, 2018.

[7] YANG W, LI X, ZHANG Z. A regularized approach to sparse optimal policy in reinforcement learning[C]// Advances in Neural Information Processing Systems. [S.l.: s.n.], 2019: 5940-5950.

[8] HAARNOJA T, ZHOU A, ABBEEL P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor[C]//International Conference on Machine Learning. [S.l.: s.n.], 2018: 1861-1870.

[9] HAARNOJA T, TANG H, ABBEEL P, et al. Reinforcement learning with deep energy-based policies[C]// International Conference on Machine Learning. [S.l.: s.n.], 2017: 1352-1361.

[10] HAARNOJA T, ZHOU A, HARTIKAINEN K, et al. Soft actor-critic algorithms and applications[J]. arXiv preprint arXiv:1812.05905, 2018.

[11] CHOW Y, NACHUM O, GHAVAMZADEH M. Path consistency learning in tsallis entropy regularized mdps [C]//International Conference on Machine Learning. [S.l.: s.n.], 2018: 979-988.

[12] LEE K, KIM S, LIM S, et al. Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning[J]. arXiv preprint arXiv:1902.00137, 2019.

[13] GEIST M, SCHERRER B, PIETQUIN O. A theory of regularized markov decision processes[C]//International Conference on Machine Learning. [S.l.: s.n.], 2019: 2160-2169.

[14] MAZOURE B, DOAN T, DURAND A, et al. Leveraging exploration in off-policy algorithms via normalizing flows[C]//Conference on Robot Learning. [S.l.]: PMLR, 2020: 430-444.

[15] TANG Y, AGRAWAL S. Discretizing continuous action space for on-policy optimization[C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 34. [S.l.: s.n.], 2020: 5981-5988.

[16] BARINGHAUS L, FRANZ C. Rigid motion invariant two-sample tests[J]. Statistica Sinica, 2010:1333-1361.

3.3 基于内在奖励的探索（内容待补充）

第四章 离线强化学习（内容待补充）