2021 01 22 Offline Reinforcement Learning Review

Chen Gong

22 Jan 2021

目录

Offline 强化学习在 2019 年由 UC Berkeley 的大佬开出来的坑。最近是非常的火，组会上一听到师兄介绍 Offline RL 的思想，小编就觉得非常有意思。Offline RL 舍弃了和环境的交互，让 agent 在一个固定的数据集（batch）上进行训练，从而得到想要的策略。这样不就可以直接解决强化学习采样效率低下，采样昂贵的问题。而限制强化学习大规模应用的主要原因之一，是需要和环境进行实时交互，来实时收集数据，以提高策略。但是，在很多应用中，样本收集很困难，或者很危险。所以，实时的和环境进行交互是不太可能的，所以发掘一种可以仅利用之前收集的数据来训练的方法非常重要。所以，这个思想吸引了我，觉得这或许是强化学习未来发展重要的方向。

这篇文章是 Sergey 大佬写的 Offline RL 入门的资料，为了帮助同学们更好的理解强化学习，明白强化学习的优缺点，希望可以激发读者解决 offline RL 的较好想法。文章中会列举大量的 Offline 强化学习算法，并描述其算法思想，面临的挑战性问题等。最后，会讲一些 Offline RL 的应用，以及其面临的开放性挑战问题。文章中有部分介绍强化学习基础的部分，我将直接跳过，默认读此文章的同学基本具备强化学习的基础。

1 Paper name

Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems

2 Offline RL 的问题描述和大致概述

2.1 什么是 Offline 强化学习？

图一中给出的三幅图非常详细的解释了 Offline RL，online RL 和 Off-policy 强化学习之间的区别。大家可以自己看看，也非常的好懂。

Offline RL 可以被定义为 data-driven 形式的强化学习问题。在不和环境交互的情况下，来使目标最大化：

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{H} \gamma r(s_t, a_t) \right] \tag{1}$$

我们给算法提供一个静态的数据集 $\{(s_t^i, a_t^i, s_{t+1}^i, r_t^i)\}$，并且通过数据集来学到最好的策略。其本质上，Offline RL 需要学习算法从固定的数据集中获得对 MDP 的充分的理解，并构造一个策略 $\pi(a|s)$，以在实际交互中获得最多的累计奖励。本文中用 π_β 表示数据集 $\mathcal{D}$ 中的状态和动作分布，比如 $(s, a) \in \mathcal{D}$，并且 $s \sim d^{\pi_\beta}(s)$，而动作是根据行为策略采样而来，$a \sim \pi_\beta(a|s)$。

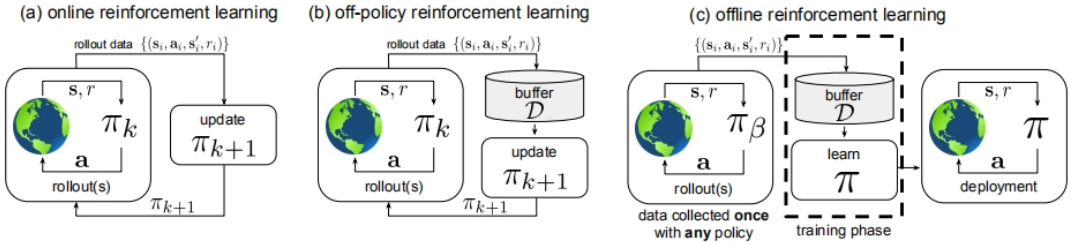

Figure 1: Pictorial illustration of classic online reinforcement learning (a), classic off-policy reinforcement learning (b), and offline reinforcement learning (c). In online reinforcement learning (a), the policy π_k is updated with streaming data collected by π_k itself. In the classic off-policy setting (b), the agent's experience is appended to a data buffer (also called a replay buffer) $\mathcal{D}$, and each new policy π_k collects additional data, such that $\mathcal{D}$ is composed of samples from $\pi_0, \pi_1, \ldots, \pi_k$, and all of this data is used to train an updated new policy π_{k+1}. In contrast, offline reinforcement learning employs a dataset $\mathcal{D}$ collected by some (potentially unknown) behavior policy π_β. The dataset is collected once, and is not altered during training, which makes it feasible to use large previous collected datasets. The training process does not interact with the MDP at all, and the policy is only deployed after being fully trained.

图 1: Offline 强化学习示意图

2.2 Offline RL 中的困难

造成 Offline RL 学习困难的原因有很多。其中最主要的困难是：学习算法需要完全的依赖于静态数据集 $\mathcal{D}$，但是没有办法提高探索，因为不和环境进行交互，就无法知道探索得到的数据是否有效，所以 Offline RL 不可能通过探索发现高奖励的区域。而且，并没有办法解决此问题，所以，假设数据集 $\mathcal{D}$ 可以充分的覆盖高奖励的转移对。

另外一个非常微妙，并且实际中更加重要的挑战是，Offline RL 需要从观测到的数据集 $\mathcal{D}$ 中，学习到一个超越 $\mathcal{D}$ 中观测到的数据的策略。大家是不是在这里感觉问题来了？在之前的监督学习中，我们希望从训练数据集中学到网络可以在测试数据集上获得较好的性能，且测试数据和训练数据是独立同分布的。而 Offline RL 中，我们希望学到的策略和在数据集 $\mathcal{D}$ 上观测的不一样，所以会造成非常严重的**分布偏移**的问题。Offline RL 中训练目标和最终想得到的目标并不一样。那么，我们的函数模拟器（策略，值函数，模型）需要在一个分布下训练，而将在不同的分布下评估，为了最大化累计收益，对于同一个状态，新策略也会给出不同的动作。

目前数据分布偏移问题，有很多的解决方式，最简单的一种即为在训练过程中限制一些东西，比如，分布偏移的下界。比如，限制目标策略 $\pi(a|s)$ 和行为策略 $\pi_\beta(a|s)$ 的差异程度，我们可以约束状态的分布位移。

接下来将简要的描述分布偏移在 MDP 中对策略造成的不利影响。假设，我们对于每一个 $s \in \mathcal{D}$ 都提供了最优的动作标签 a^*。那么，我们期望如果没有被提供这些最优的动作标签 a^*，我们的策略依然可以获得同样好的性能。那么，损失函数可以定义为：

$$\ell(\pi) = \mathbb{E}_{p_\pi(\tau)}\left[\sum_{t=0}^{H} \delta\left(\mathbf{a}_t \neq \mathbf{a}_t^\star\right)\right] \tag{2}$$

如果使用监督学习的思想（标准经验误差最小化）来学习，可以得到如下结论：

这意味着，考虑时间步为 H 的情况下，在离线强化学习中误差上界和时间步之间是平方关系，而在在线强化学习中是线性关系。直觉上感觉，造成这样的原因是因为，学习策略 $\pi(a|s)$ 可能会进入和

训练分布差距很远的状态，这将导致 $d^{\pi}(s)$ 和 $d^{\pi_\beta}(s)$ 差距非常大。那么，当策略在 t 时刻遇到了分布之外（数据集中没见过）的状态，那么策略在之后的 $H-t$ 个时刻就有可能不断的犯错，所以累计误差为 $O(H)$，而且每一个时间步，都有可能进入分布外的状态，所以整体误差为 $O(H^2)$。但是在在线学习中，不会遇到分布外的状态。这个现象提示我们，如果不注意尽量的减少不利的影响，分布偏移可能会对从任何一个固定数据集中学习到的策略，都造成非常大的影响。

3 基于重要性采样的 Offline RL 与离线策略评估

事实上所有的 off-policy 算法都可以改为 offline 算法，最直接的方法之一就是重要性采样，利用从 $\pi_\beta(\tau)$ 中采样出的轨迹来估计 $J(\pi)$。这部分，我们将回顾 off-policy 中使用重要性采样来进行策略评估的方法，并且讨论如何将重要性采样运用到 offline RL 中。

3.1 Off-policy 中通过重要性采样估计

可以通过重要性采样来获得 $J(\pi)$ 的无偏估计：

$$\begin{aligned} J\left(\pi_\theta\right) &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)}\sum_{t=0}^{H}\gamma^t r(\mathbf{s},\mathbf{a})\right] \\ &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\left(\prod_{t=0}^{H}\frac{\pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}{\pi_\beta\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}\right)\sum_{t=0}^{H}\gamma^t r(\mathbf{s},\mathbf{a})\right] \approx \frac{1}{n}\sum_{i=1}^{n} w_H^i \sum_{t=0}^{H}\gamma^t r_t^i \end{aligned} \tag{3}$$

其中，$w_t^i = \frac{1}{n}\prod_{t'=0}^{t}\frac{\pi_\theta(a_{t'}^i|s_{t'}^i)}{\pi_\beta(a_{t'}^i|s_{t'}^i)}$，并且 $\{s_0^i, a_0^i, r_0^i, s_1^i, \cdots\}$ 是从 $\pi_\beta(\tau)$ 中采样出的 n 条轨迹。但是不幸的是，由于重要性权重的连乘这样的估计方法可能会有很高的方差，CS 598 note 6 中给出了证明，其方差为 $r^2(K-1)$，其中 K 等于的动作空间的大小。简单的处理方法是将权重进行归一化，除以 $\sum_{i=1}^{n} w_H^i$。这样使得重要性采样不再是无偏估计，但是可以使得方差减小，其效果依然非常好。统计里面评估参数估计的好坏有三个性质：无偏性，有效性，一致性，方差一高，有效性就低。

考虑到当 $t' > t$ 时，r_t 和 $s_{t'}$ 和 $a_{t'}$ 无关，所以，可以将 t 时刻之后的重要性权重都舍弃掉，就得到了 pre-decision 重要性采样估计的一般形式：

$$J\left(\pi_\theta\right) = \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\sum_{t=0}^{H}\left(\prod_{t'=0}^{t}\frac{\pi_\theta\left(\mathbf{a}_{t'} \mid \mathbf{s}_{t'}\right)}{\pi_\beta\left(\mathbf{a}_{t'} \mid \mathbf{s}_{t'}\right)}\right)\gamma^t r(\mathbf{s},\mathbf{a})\right] \approx \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{H} w_t^i \gamma^t r_t^i \tag{4}$$

这样虽然可以通过减小重要性权重来减小方差。实际上，这样的做法仍然会导致很大的方差。如果，我们使用近似的模型 $\hat{Q}^{\pi}(s_t, a_t)$ 来模拟 Q 函数，并将其合并到公式 (4) 中，就得到了大名鼎鼎的 doubly robust estimator。其思想为：之前分析了重要性采样的方差为 $r^2(K-1)$，由于 K 是不能改变的，所以自然的想到减去一个常数，使方差为 $(r-c)^2(K-1)$，当 $c=r$ 时，方差降到 0，所以用 $\hat{Q}^{\pi}(s_t, a_t)$ 来近似 r。而且小编觉得文章中这里写错了，第二个减号应该为加号。并且，双重鲁棒重要性采样一定是无偏估计。（此部分来自于 CS 598 note 6，有兴趣的同学可以自行详细阅读）。

$$J\left(\pi_\theta\right) \approx \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{H}\gamma^t\left(w_t^i\left(r_t^i - \hat{Q}^{\pi_\theta}\left(\mathbf{s}_t, \mathbf{a}_t\right)\right) + w_{t-1}^i \mathbb{E}_{\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s}_t)}\left[\hat{Q}^{\pi_\theta}\left(\mathbf{s}_t, \mathbf{a}\right)\right]\right) \tag{5}$$

当然还有很多很多的采样方法，这些采样方法都可以用来做策略提升。在 offline RL 中，我们希望可以改进行为策略，以保证算法的性能有很高的概率不低于一个边界。

3.2 Off-Policy 梯度下降

大家看到这第一眼是不是感觉有点奇怪，通常我们研究 policy gradient 都是在 on-policy 算法中，这是因为策略不能用其他策略产生的数据更新。而重要性采样可以直接被用于估计策略梯度，而不是先求值函数在得到策略。策略梯度方法则是通过优化 $J(\pi)$，直接对策略函数参数的梯度进行估计，来优化策略函数。而我们在这部分将策略梯度的方法扩展到 Offline RL 中。

Off-Policy 算法中，轨迹 τ 是从行为策略中采样出的，并且 $\pi_\beta(a|s) \neq \pi(a|s)$。接下来将从 Off-Policy 算法出发，讲述 offline RL 中的扩展和挑战。策略梯度可定义为：

$$\begin{aligned}\nabla_\theta J\left(\pi_\theta\right) &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)}\sum_{t=0}^{H}\gamma^t\nabla_\theta\log\pi_\theta\left(\mathbf{a}_t\mid\mathbf{s}_t\right)\hat{A}\left(\mathbf{s}_t,\mathbf{a}_t\right)\right]\\ &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\left(\prod_{t=0}^{H}\frac{\pi_\theta\left(\mathbf{a}_t\mid\mathbf{s}_t\right)}{\pi_\beta\left(\mathbf{a}_t\mid\mathbf{s}_t\right)}\right)\sum_{t=0}^{H}\gamma^t\nabla_\theta\log\pi_\theta\left(\mathbf{a}_t\mid\mathbf{s}_t\right)\hat{A}\left(\mathbf{s}_t,\mathbf{a}_t\right)\right]\\ &\approx \frac{1}{n}\sum_{i=1}^{n}w_H^i\sum_{t=0}^{H}\gamma^t\nabla_\theta\log\pi_\theta\left(\mathbf{a}_t^i\mid\mathbf{s}_t^i\right)\hat{A}\left(\mathbf{s}_t^i,\mathbf{a}_t^i\right)\end{aligned} \tag{6}$$

其中 $\{s_0^i, a_0^i, r_0^i, s_1^i, \cdots\}_{i=1}^n$ 是从 $\pi_\beta(\tau)$ 中采样出的 n 条轨迹。和前面分析的一样，当 $t' > t$ 时，r_t 不依赖于 s_t, a_t，所以在计算 t 时刻的奖励时，可以不考虑 t' 时刻的动作和状态，所以可以舍弃 t' 时刻的重要性权重，于是得到了 per-decision 重要性策略梯度估计：

$$\nabla_\theta J\left(\pi_\theta\right) \approx \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{H}w_t^i\gamma^t\left(\sum_{t'=t}^{H}\gamma^{t'-t}\frac{w_{t'}^i}{w_t^i}r_{t'} - b\left(\mathbf{s}_t^i\right)\right)\nabla_\theta\log\pi_\theta\left(\mathbf{a}_t^i\mid\mathbf{s}_t^i\right) \tag{7}$$

实际应用中，策略梯度估计的方差非常大，到时很难 work。为了降低方差，和前面重要性采样估计一样，有对重要性权重进行归一化的方法，同时也有双重鲁棒估计的方法。而且，Off-Policy 算法中经常将重要性权重作为正则化项，以保证学习策略 $\pi_\theta(a|s)$ 和采样策略 $\pi_\beta(a|s)$ 之间差距不会太大。

$$\nabla_\theta\bar{J}\left(\pi_\theta\right) \approx \left(\sum_{i=1}^{n}w_H^i\sum_{t=0}^{H}\gamma^t\nabla_\theta\log\pi_\theta\left(\mathbf{a}_t^i\mid\mathbf{s}_t^i\right)\hat{A}\left(\mathbf{s}_t^i,\mathbf{a}_t^i\right)\right) + \lambda\log\left(\sum_{i=1}^{n}w_H^i\right) \tag{8}$$

通过 softmax 函数，使得 $\sum_i w_H^i \to 1, n\to\infty$。在样本有限的情况下，这样的做法会自动的调整策略 π_θ，使得最少有一个样本的重要性权重较大。目前，基于重要性采样的深度强化学习算法，通常采用基于样本的 KL 散度正则化项，比如大名鼎鼎的 TRPO。

3.3 近似 Off-Policy 策略梯度

重要性采样的目标函数，需要将每个时刻的重要性权重连乘起来，所以会导致非常大的方差，这或许就是重要性采样方差较大的本质原因。所以，我们使用行为策略得到的状态分布 $d^{\pi_\beta}(s)$，代替当前策略 $d^\pi(s)$，推导出近似的基于重要性采样的梯度。近似估计的结果是有偏估计，$d^{\pi_\beta}(s) \neq d^\pi(s)$，但是在实际应用中效果还不错。这里用 $J_\beta(\pi_\theta)$ 来表示目标函数，以强调对行为策略的依赖，写作：

$$J_\beta(\pi_\theta) = \mathbb{E}_{s\sim d^{\pi_\beta}(s)}[V^\pi(s)]. \tag{9}$$

注意，$J_\beta(\pi_\theta)$ 和 $J_(\pi_\theta)$ 中的状态分布不一样，导致 $J_\beta(\pi_\theta)$ 是对 $J_(\pi_\theta)$ 的有偏估计。很显然，在 Offline RL 中，在 $d^{\pi_\beta}(s)$ 下估计期刊，可以简单的直接从数据集 $\mathcal{D}$ 中进行采样，不需要进行重要性采样。而

且，大家发现这个偏差在实际应用中是可接受的。所以，策略梯度被写成如下所示，并且此近似梯度被大规模的应用于深度强化学习算法中。

$$\begin{aligned}\nabla_\theta J_\beta\left(\pi_\theta\right) &= \mathbb{E}_{\mathbf{s}\sim d^\beta(\mathbf{s}),\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}\left[Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\nabla_\theta \log \pi_\theta(\mathbf{a}\mid\mathbf{s}) + \nabla_\theta Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\right] \\ &\approx \mathbb{E}_{\mathbf{s}\sim d^\beta(\mathbf{s}),\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}\left[Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\nabla_\theta \log \pi_\theta(\mathbf{a}\mid\mathbf{s})\right].\end{aligned} \tag{10}$$

其中，为了估计策略梯度的近似值，我们需要估计 $Q^{\pi_\theta}(\mathbf{s},\mathbf{a})$。之后，将详细的讨论在 Offline RL 中如何估计价值函数。

3.4 边缘化重要性采样

本小节，边缘化重要性采样比较的难懂，我参考了 CS 598 的 Marginalized Importance Sampling，里面讲的比较详细。前文中我们详细的描述了重要性采样有较大的方差，而且用 Off-Policy 的策略梯度估计是有偏估计。如果，我们想避免因为错误的状态分布而导致偏差，还有对每个动作计算重要性采样率并连乘导致的较大的方差。所以，找到了一个取代的方法，直接估计边缘状态的重要性：$\rho^{\pi_\theta}(s)=\frac{d^{\pi_\theta}(s)}{d^\beta(s)}$。这个思想是怎么来的呢？下面将进行详细的分析。

对于一个策略，我们可以采用定义为按照策略获得的规范化的预期奖励来评价：

$$J(\pi) := (1-\gamma)\cdot\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_t \mid s_0\sim\beta, \forall t, a_t\sim\pi\left(s_t\right), r_t\sim R\left(s_t,a_t\right), s_{t+1}\sim T\left(s_t,a_t\right)\right] \tag{11}$$

而 off-policy 估计的目标是用一个确定的数据集 $\mathcal{D}$ 中的转移对 (s,a,r,s') 来估计 $J(\pi)$。而其中 $\mathcal{D}$ 可能来自于一个单一的行为策略 (之前的工作中收集的数据)，多个的行为策略，或者一个数据库。在 $\mathcal{D}$ 是由已知行为策略 π_β 收集的完整轨迹的特殊情况下，可以使用重要性采样 (IS) 来估计 $J(\pi)$。具体来说，对于从 π_β 中采样得到的一个有限长度的轨迹，$\tau=(s_0,a_0,r_0,\cdots,s_H)$，使用重要性估计 $J(\pi)$ 为：

$$J(\pi) = (1-\gamma)\left(\prod_{t=0}^{H-1}\frac{\pi\left(a_t\mid s_t\right)}{\pi_\beta\left(a_t\mid s_t\right)}\right)\left(\sum_{t=0}^{H-1}\gamma^t r_t\right) \tag{12}$$

尽管，已有很多方法来改进重要性采样中方差较大的问题，但是实际中仍然会存在较大的问题。**其解决方法的思路，主要是跳开连乘的操作，计算多个，每个都有偏差然后承在一起会造成很大的误差。但是，如何先进行连乘操作，然后估计连乘的结果，方差是不是会减小很多呢？**首先，可以将对策略的评估重写为：

$$J(\pi) = \mathbb{E}_{(s,a)\sim d^\pi, r\sim R(s,a)}[r] \tag{13}$$

其中，

$$d^\pi(s,a) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr\left[s_t=s, a_t=s \mid s_0\sim\mu_0, a_t\sim\pi\left(s_t\right), s_{t+1}\sim T\left(s_t,a_t\right) \text{ for } t\geq 0\right]$$

观测可得，也就是将公式 (11) 中的概率转移那部分用 d^π 来表示了，其为关于 π 的状态动作对的标准化折扣平稳分布。同理，我们可以类似的定义状态的折现平稳分布 $d^\pi(s)$，其中，$d^\pi(s,a)=d^\pi(s)\pi(a|s)$。如果，$\mathcal{D}$ 包含行为策略 β 收集的轨迹，那么策略评估值可以被记为：

$$J(\pi) = \mathbb{E}_{(s,a)\sim d^\beta, r\sim R(s,a)}\left[\frac{d^\pi(s,a)}{d^\beta(s,a)}r\right] = \mathbb{E}_{(s,a)\sim d^\beta, r\sim R(s,a)}\left[\rho^\pi(s,a)r\right] \tag{14}$$

这样做就直接跳开了连乘的操作。研究中证明了，**边缘状态重要性的方差一定不大于每个动作的重要性权重的乘积**。但是，直接计算边缘状态重要性基本不可能。直到最近才出现一些基于动态规范的对 ρ^π 估计方法。根据所使用的底层 Bellman 方程的形式，我们可以将它们分为两类，

3.4.1 基于前向贝尔曼方程的方法

基于前向贝尔曼方程的方法：意思是，$\rho(\pi)$ 满足前向贝尔曼方程：

$$\forall \mathbf{s}', \underbrace{d^{\pi_\beta}(\mathbf{s}')\,\rho^{\pi}(\mathbf{s}')}_{(d^{\pi_\beta}\circ\rho^{\pi})(s,a)} = \underbrace{(1-\gamma)d_0(\mathbf{s}') + \gamma\sum_{\mathbf{s},\mathbf{a}} d^{\pi_\beta}(\mathbf{s})\rho^{\pi}(\mathbf{s})\pi(\mathbf{a}\mid\mathbf{s})T(\mathbf{s}'\mid\mathbf{s},\mathbf{a})}_{\left(\overline{\mathcal{B}}^{\pi}\circ\rho^{\pi}\right)(\mathbf{s},\mathbf{a})} \tag{15}$$

这个公式理解起来并不难，左边拆开就是，$d^{\pi_\theta}(\mathbf{s}')$。这个方程可用于使用时间差分更新来估计当前策略下的 $\rho(\pi)$。

比如，当使用随机梯度更新，Gelada 和 Bellemare 等人提出只有如下方法来在线更新 $\rho(\pi)$：

$$\hat{\rho}^{\pi}(\mathbf{s}') \leftarrow \hat{\rho}^{\pi}(\mathbf{s}') + \alpha\left[(1-\gamma) + \gamma\frac{\pi(\mathbf{a}\mid\mathbf{s})}{\pi_\beta(\mathbf{a}\mid\mathbf{s})}\hat{\rho}^{\pi}(\mathbf{s}) - \hat{\rho}^{\pi}(\mathbf{s}')\right] \tag{16}$$

其中，$\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}')$，$\mathbf{a}\sim\pi_\beta(\mathbf{a}\mid\mathbf{s})$，$\mathbf{s}'\sim T(\mathbf{s}'\mid\mathbf{s},\mathbf{a})$。其他额外的技术，包括 $TD(\lambda)$ 估计，自动调节特征维度，也可以用来稳定学习过程。

而 Liu 等人提出了用对抗的方法来获得 $\rho(\pi)$。从公式 (16) 中，可以推断出公式：

$$L(\rho,f) = \gamma\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(\rho(\mathbf{s})\frac{\pi(\mathbf{a}\mid\mathbf{s})}{\pi_\beta(\mathbf{a}\mid\mathbf{s})} - \rho(\mathbf{s}')\right)f(\mathbf{s}')\right] + (1-\gamma)\mathbb{E}_{\mathbf{s}_0\sim d_0}[(1-\rho(\mathbf{s}))f(\mathbf{s})] \tag{17}$$

对于 $\forall f$，当且仅当 $\rho=\rho^{\pi}$ 时有 $L(\rho,f)=0$。其中，ρ 为生成器，f 为判别器。那么，目标函数可以写为一个求鞍点的问题，

$$\rho^{*} = \arg\min_{\rho}\max_{f} L(\rho,f)^2 \tag{18}$$

计算出 ρ^{*} 就可以用来估计 Off-Policy 梯度了。同时 Zhang 等人提出了另一种使用贝尔曼前进方程直接优化贝尔曼残差的方法，这就是 Bo Dai 他们的那一套方法，直接优化两个分布间的 f 散度。此方法优化的是公式 (11) 的 f 散度，并且增加约束对 ρ^{π} 进行归一化，来防止模型坍缩。

$$\min_{\rho^{\pi}} D_f\left(\left(\overline{\mathcal{B}}^{\pi}\circ\rho^{\pi}\right)(\mathbf{s},\mathbf{a}), (d^{\pi_\beta}\circ\rho^{\pi})(\mathbf{s},\mathbf{a})\right) \quad \text{s.t.} \quad E_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\rho^{\pi}(\mathbf{s},\mathbf{a})\right] = 1 \tag{19}$$

然后，Bo Dai 使用了凸共轭的技巧，在对偶空间求解此目标函数，并且可以避免由于采样估计造成的偏差。读者可以详细阅读 Gendice: Generalized offline estimation of stationary values.

3.4.2 基于凸共轭方法的后向 Bellman 方程

此小节最后，我们将讨论使用后向 Bellman 方程（**后向 Bellman 方程使用凸共轭得到了类似泛函的价值函数，其中泛函的意思为函数的函数，个人看到这理解的是建立以一个关于 ρ^{π} 的函数**）对 off-policy 的策略估计和提升。此类方法中用到了成熟的凸优化技巧。最先是 Lee and He (2018) 等人提出了将应用于凸优化的工作扩展到 Off-Policy 策略优化中。他们证明了 Off-Policy 的采样复杂度下界，然而，将凸优化中的技术扩展到 RL 中很具挑战性。

Nachum 等人用类似的方法来进行 Off-Policy 评估。其主要思想是将其转换为一个优化问题，**读原文才知道，此处采用的是凸共轭，凸优化问题为** $\min_x J(x) := \frac{1}{2}mx^2 - nx$**，其中** $x^{*}(s,a) = \frac{m}{n} = \frac{d^{\pi}(s,a)}{d^{\mathcal{D}}(s,a)}$，

$$\rho^{\pi} = \arg\min_{x:\mathcal{S}\times\mathcal{A}\to\mathbb{R}} \frac{1}{2}E_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[x(\mathbf{s},\mathbf{a})^2\right] - E_{\mathbf{s}\sim d^{\pi}(\mathbf{s}),\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}[x(\mathbf{s},\mathbf{a})] \tag{20}$$

其中，令 $\nu : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ 为任意的价值函数，并满足：

$$\nu(s,a) := x(s,a) + \gamma \mathbb{E}_{s' \sim T(s,a), a' \sim \pi(s')} \left[\nu\left(s', a'\right) \right], \forall (s,a) \in S \times A \tag{21}$$

$x(s,a) = \nu(\mathbf{s},\mathbf{a}) - \gamma \mathbb{E}_{\mathbf{s}' \sim T(\mathbf{s},\mathbf{a}), \mathbf{a}' \sim \pi(\mathbf{a}'|\mathbf{s}')} \left[\nu\left(\mathbf{s}', \mathbf{a}'\right) \right]$。此处可以定义一个修正的 Bellman 算子，$\tilde{\mathcal{B}}^{\pi} \nu(\mathbf{s},\mathbf{a}) := \gamma \mathbb{E}_{\mathbf{s}' \sim T(\mathbf{s},\mathbf{a}), \mathbf{a}' \sim \pi(\mathbf{a}'|\mathbf{s}')} \left[\nu\left(\mathbf{s}', \mathbf{a}'\right) \right]$，这和没有 $r(s,a)$ 项的贝尔曼算子 $\tilde{\mathcal{B}}^{\pi}$ 很类似。在 DualDICE: Behavior-Agnostic Estimation of Discounted Stationary Distribution Corrections 的 3.2 中详细的推导了，公式 (20) 的优化问题，可以写为：

$$\min_{\nu: \mathcal{S} \times \mathcal{A} \to \mathbb{R}} \frac{1}{2} \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}' \sim \mathcal{D}} \left[\left(\nu(\mathbf{s},\mathbf{a}) - \tilde{\mathcal{B}}^{\pi} \nu(\mathbf{s},\mathbf{a}) \right)^2 \right] - (1-\gamma) \mathbb{E}_{\mathbf{s}_0 \sim d_0(\mathbf{s}_0), \mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s}_0)} \left[\nu\left(\mathbf{s}_0, \mathbf{a}\right) \right] \tag{22}$$

公式 (22) 的最优解为 ν^*，那么就可以利用 $x(s,a) = \nu(\mathbf{s},\mathbf{a}) - \tilde{\mathcal{B}}^{\pi} \nu(\mathbf{s},\mathbf{a})$ 计算得到 $\rho^{\pi}(\mathbf{s},\mathbf{a})$，可用于 off-Policy 评估和提升。

类似的在 Algaedice: Policy gradient from arbitrary experience 中用类似的方法设计了一个 Off-Policy RL 算法。**其核心思想是通过求解一个优化问题得到 On-Policy 问题的策略梯度估计**。公式表达为，

$$\max_{\pi} \mathbb{E}_{\mathbf{s} \sim d^{\pi}(\mathbf{s}), \mathbf{a} \sim \pi(\cdot|\mathbf{s})} [r(\mathbf{s},\mathbf{a})] - \alpha D_f \left(d^{\pi}(\mathbf{s},\mathbf{a}), d^{\pi_\beta}(\mathbf{s},\mathbf{a}) \right) \tag{23}$$

其中，

$$d^{\pi}(s,a) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \Pr\left[s_t = s, a_t = s \mid s_0 \sim \mu_0, a_t \sim \pi\left(s_t\right), s_{t+1} \sim T\left(s_t, a_t\right) \text{ for } t \geq 0 \right]$$

通过使用对 f 散度的对偶可以得到：

$$D_f(p,q) = \max_{x: \mathcal{S} \times \mathcal{A} \to \mathbb{R}} \left(\mathbb{E}_{\mathbf{y} \sim p(\mathbf{y})} [x(\mathbf{y})] - \mathbb{E}_{\mathbf{y} \sim q(\mathbf{y})} \left[f^*(x(\mathbf{y})) \right] \right) \tag{24}$$

利用改变变量的技巧，将公式 (24) 变成一个鞍点优化问题，

$$\begin{aligned} \max_{\pi} \min_{Q} L\left(Q, \pi_\beta, \pi\right) &:= \mathbb{E}_{\mathbf{s}_0 \sim d_0(\mathbf{s}_0), \mathbf{a} \sim \pi(\cdot|\mathbf{s}_0)} \left[Q\left(\mathbf{s}_0, \mathbf{a}\right) \right] \\ &+ \alpha \mathbb{E}_{\mathbf{s},\mathbf{a} \sim d^{\pi_\beta}(\mathbf{s},\mathbf{a})} \left[f^* \left(\frac{r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim T(\mathbf{s},\mathbf{a}), \mathbf{a}' \sim \pi(\mathbf{a}'|\mathbf{s}')} \left[Q\left(\mathbf{s}', \mathbf{a}'\right) \right] - Q(\mathbf{s},\mathbf{a})}{\alpha} \right) \right] \end{aligned} \tag{25}$$

其中 Q 为 Q 函数，满足：$Q(\mathbf{s},\mathbf{a}) = \mathbb{E}_{\mathbf{s}' \sim T(\mathbf{s},\mathbf{a}), \mathbf{a}' \sim \pi(\mathbf{a}'|\mathbf{s}')} \left[r(\mathbf{s},\mathbf{a}) - \alpha x(\mathbf{s},\mathbf{a}) + \gamma Q\left(\mathbf{s}', \mathbf{a}'\right) \right]$。其中，$f(x) = x^2$，$f^*(x) = x^2$，这一目标简化为应用常规的 AC 算法就可求解，但是注意附加了条件，优化在初始状态 s_0 时刻的 Q 值。假设在最优解 Q^* 的情况下，函数 $L(Q^*, \pi_\beta, \pi)$ 策略梯度为正规化政策梯度问题中的 on-policy 的策略梯度：（具体过程需要阅读 Algaedice: Policy gradient from arbitrary experience）

$$\frac{\partial}{\partial \pi} L\left(Q^*, \pi_\beta, \pi\right) = \mathbb{E}_{\mathbf{s} \sim d^{\pi}(\mathbf{s}), \mathbf{a} \sim \pi(\cdot|\mathbf{s})} \left[\tilde{Q}^{\pi}(\mathbf{s},\mathbf{a}) \cdot \nabla_{\pi} \log \pi(\mathbf{a} \mid \mathbf{s}) \right] \tag{26}$$

其中 $\tilde{Q}^{\pi}$ 是对应于正则化 RL 问题的行为-价值函数。**这篇文章小编大概读过，他的核心创新点在于使用对偶方式构建的 Off-Policy RL 算法，计算梯度时只依赖 Off-Policy 数据，并且这些数据可以从任意行为策略收集而来。计算出的对偶函数的梯度就是原来的 On-Policy 的策略梯度，所以他们的做法有效的避免了求重要性权重**。

3.5 挑战和开放性问题

第三部分中，我们讨论了不同形式的重要性采样估计当前策略 π_θ 的累积回报期望或者估计策略梯度的方法。本节中讨论的策略提升的方法主要是针对经典的 Off-Policy，其中附加的数据是在线收集的，但重用之前的数据以提高效率。据我们所知，这些方法还没有普遍应用于离线环境中。

将这些方法运用到完全 offline 的 RL 中，将会主要遇到下列一些问题。

1. 重要性采样通常会陷入高方差的问题，这个方差在连续设置中将急剧增加，因为在连续的时间步长的重要性权重是相乘的（比如公式 (6)），这将导致误差指数级的累积。近似和边缘重要性抽样方法在一定程度上缓解了这一问题，避免了多个时间步长的重要性权重相乘，但根本问题仍然存在：当行为策略 π_β 与当前学习的策略 π_θ 差异过大时，其重要性权重会发生退化，任何对 return 或梯度的估计都有较大的方差而无法使用，特别是在高维状态和行为空间中，或者对于采样轨迹很长的问题。在策略 π_θ 只与行为策略 π_β 偏差较小的情况下，重要性抽样估计是最合适的。在经典的 off-policy 设置中，使用重要性采样是因为，其问题本身通常是符合这种情况的，因为使用最新的策略收集到新的轨迹会重复收集并添加到数据集，但是在 offline 中，通常不是这种情况。这样在 offline 中，非常容易产生行为策略 π_β 与当前学习的策略 π_θ 差异过大的情况。所以，限制重要性采样在 offline RL 中的应用的主要困难是，**1. 策略 π_θ 与行为策略 π_β 偏差较大；2. 状态和动作空间维度过大；3. 采样的轨迹过长（有效视界过长）**。

2. 第二个挑战为：在 Off-Policy 估计梯度的方法中，最有效的方法要么需要估计值函数，要么需要通过动态规划估计状态边缘密度比率。动态规划方法在 offline 中存在分布漂移的问题（策略得到的动作和原数据中不一样），这使得在不进行额外修正的情况下很难稳定地学习值函数。在经典的 Off-Policy 中，这个问题没有那么严重，因为 off-policy 可以额外的收集最新数据。

4 动态规划下的 Offline RL

动态规划方法，比如 Q-learning 算法，与策略梯度法相比，原则上可以为离线强化学习提供更有吸引力的选择。在第 4.1 节中，我们将讨论基于线性函数近似的一种较老的方法，它们也被有效地用于深度神经网络函数近似器。比如，kalashnikov 等人提出了描述了一种名为 QT-Opt 的 Q-learning 算法，该算法能够从之前实验过程中记录的约 50 万次抓取试验中学习有效的基于视觉的机器人抓取策略，但观察到额外的 online 微调，和单纯根据记录的数据训练的策略相比，仍然大大提高了策略的性能。实际上一些关于 offline RL 的工作也注意到，对于某些类型的数据集，传统的动态规划算法，如 deep Q-learning 或确定性 AC 算法，实际上也可取得较好的效果（An optimistic perspective on offline reinforcement learning）。

但是，实际使用中，这样的方法在 offline RL 中会遇到很多的问题。其中，最重要的是我们之前提到的分布偏移问题，解决分布偏移问题的方法可以大致分成两类，**1. 4.3 小节中将描述的策略约束方法，以减小学习策略 π_θ 和行为策略 π_β 间的某种距离，来减小分布偏移。2. 4.4 小节中将描述的非确定性方法，此类方法试图估计对 Q 值的不确定性，然后利用这种不确定性来发现分布位移**。在 4.5 小节中，将总结所有类型的分布偏移修正方法，并总结其开放性问题和面临的挑战。

4.1 基于线性价值函数的 Off-Policy 价值函数估计

文章中首先讨论了基于线性函数近似的价值函数和策略估计的标准 offline RL 算法，算法本身并不能减轻分布偏移的影响，但当有好的线性特征时，算法依然效果不错。现代深度强化学习方法通常避开线性特征，倾向于非线性的神经网络函数逼近器，而大量的文献中线性方法是离线强化学习算法的重要组成部分（Batch reinforcement learning）。首先介绍的是，使用线性函数来近似 Q 函数，$Q_\phi \approx f(s,a)^{\mathrm{T}}\phi$，其中 $f(s,a)^{\mathrm{T}} \in \mathbb{R}^d$ 表示为状态动作对 (s,a) 的线性特征。而对学习策略 $\pi(a|s)$ 的 Q 函数估计，是通过对行为策略 $\pi_\beta(a|s)$，而状态分布为 $d^{\pi_\beta}(s)$。并且，贝尔曼算子表示为：$\overrightarrow{Q^\pi} = \mathcal{B}^\pi \overrightarrow{Q^\pi}$。

由于用线性近似来表示 Q 函数，那么 Q 函数可以看成是线性函数集合中的一个解，并可以通过最小化均方误差进行求解。这看上去很简单，下面我们将介绍不同的求解 Q^π 的方法。再强调一遍 Q 函数是关于特征 $f(s,a)$ 的线性函数，而线性特征 $f(s,a)$ 的表格形式为 $\mathbf{F} \in \mathbb{R}^{|S||A|\times d}$，这样可得到 $\vec{Q}_\phi = \mathbf{F}\phi$。下面主要介绍两种求解线性近似器的方法。

4.1.1 贝尔曼残差最小化

核心思想为寻找合适的 ϕ 使贝尔曼残差最小化。首先，将 Bellman 方程写成 Bellman 算子形式，并将其扩展为使用奖励函数的向量化表达式，$\overrightarrow{R}$，和一个线性算子 P^π 来表示动力学转移过程，于是有 $\left(P^\pi \vec{Q}\right)(\mathbf{s},\mathbf{a}) = \mathbb{E}_{\mathbf{s}'\sim T(\mathbf{s}'|\mathbf{s},\mathbf{a}),\mathbf{a}'\sim\pi(\mathbf{a}'|\mathbf{s}')}\left[Q\left(\mathbf{s}',\mathbf{a}'\right)\right]$。而是，贝尔曼方程可以表示为：

$$\mathbf{F}\phi \approx \mathcal{B}^\pi \mathbf{F}\phi = \vec{R} + \gamma P^\pi \mathbf{F}\phi \Longrightarrow \left(\mathbf{F} - \gamma P^\pi \mathbf{F}\right)\phi \approx \vec{R}$$

求解此方程写出最小二乘解，我们得到：

$$\phi = \left(\left(\mathbf{F} - \gamma P^\pi \mathbf{F}\right)^T \left(\mathbf{F} - \gamma P^\pi \mathbf{F}\right)\right)^{-1} \left(\mathbf{F} - \gamma P^\pi \mathbf{F}\right)^T \vec{R}$$

表达式中求解的是**伪逆**，此表达式使 Bellman 残差 (Bellman 方程左、右两边的差的平方) 的 2 范数最小化，并被称为 Bellman 残差最小化解。

4.1.2 最小二乘不动点近似

4.1.1 中描述的方法，很显然存在着求逆的问题，当求解空间较大时，直接求解出最优解非常困难。另一种方法是使用投影不动点迭代，而不是直接最小化贝尔曼误差，来得到最小二乘不动点近似解。实际上，对强化学习有所了解的同学知道，贝尔曼方程是符合压缩映射定理的。最小二乘不动点近似方法中，迭代 Bellman 算子直到收敛，即为 $\vec{Q}_{k+1} \leftarrow \mathcal{B}^\pi \vec{Q}_k$，当 $k \to \infty$ 时，收敛到唯一的解 $\vec{Q}^\pi$。此方法中，使用函数来近似表示 $\vec{Q}$，但是，并不要严格的使 $\vec{Q}_{k+1} = \mathcal{B}^\pi \vec{Q}_k$，因为可能没有 ϕ 可以非常准确的表示 $\mathcal{B}^\pi \vec{Q}_k$。需要明确的是，我们求解的是 ϕ，而 $\vec{Q}_k = \mathbf{F}\phi_k$，那么可以将 Bellman 方程做如下改写：

$$\begin{aligned}
\vec{Q}_{k+1} &= \mathbf{F}\left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T\left(\vec{R} + \gamma P^\pi \vec{Q}_k\right) \\
\mathbf{F}\phi_{k+1} &= \mathbf{F}\left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T\left(\vec{R} + \gamma P^\pi \mathbf{F}\phi_k\right) \\
\phi_{k+1} &= \left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T\left(\vec{R} + \gamma P^\pi \mathbf{F}\phi_k\right)
\end{aligned}$$

通过，迭代的计算 $\phi_{k+1} = \left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T\left(\vec{R} + \gamma P^\pi \mathbf{F}\phi_k\right)$ 可以求解出 Bellman 算子的不动点。

但是，当真正的 Q 函数不在 $\mathbf{F}$ 张成的空间中时 (比如：Q 函数不能线性表示)，不同的方法通常会得到不同的解。首先，我们可以猜测，一个好的线性拟合方法需要找到 Q 函数：$\mathbf{F}\phi$，它对应于真实

的 $\vec{Q}^{\pi}$ 在 F 定义的超平面上的最小二乘投影。可惜，Bellman 残差最小化和最小二乘不动点一般都不能得到这个解。然而，通过 Bellman 残差最小化得到的解可能更接近真实 Q 函数，因为它是通过直接最小化 Bellman 残差得到的。最小二乘不动点迭代得到了一个 Q 函数，它是投影 Bellman 算子的不动点，但可能是任意次优的。然而，根据经验，最小二乘不动点比 Bellman 残差最小化更加具有计算可行性。**实际上，并没有理论来讨论，哪个算法一定就比较好**。在实践中，与 Bellman 残差最小化相比，最小二乘不动点迭代可以产生更有效的策略，而 Bellman 残差最小化方法求解过程更稳定，结果比较唯一。

4.1.3 最小二乘时间差分 Q-Learning (LSTD-Q)

接下来描述的 LSTD-Q，是一种从静态数据集直接采样来估计 Q^{π} 的方法。如果，采用增量式更新的方法，

$$\begin{aligned}
\phi_{k+1} &= \left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T\left(\vec{R}+\gamma P^{\pi}\mathbf{F}\phi_k\right) \\
\left(\mathbf{F}^T\mathbf{F}\right)\phi_{k+1} &= \mathbf{F}^T\left(\vec{R}+\gamma P^{\pi}\mathbf{F}\phi_k\right) \\
\phi_{k+1} &= \phi_k+\epsilon\left(\left(\mathbf{F}^T\mathbf{F}\right)\phi_k-\mathbf{F}^T\left(\vec{R}+\gamma P^{\pi}\mathbf{F}\phi_k\right)\right) \\
\phi_{k+1} &= \phi_k+\epsilon\left(\left(\mathbf{F}^T\mathbf{F}-\gamma\mathbf{F}^TP^{\pi}\mathbf{F}\right)\phi_k-\mathbf{F}^T\vec{R}\right)
\end{aligned} \tag{27}$$

所以，通过计算 $\left(\mathbf{F}^T\mathbf{F}-\gamma\mathbf{F}^TP^{\pi}\mathbf{F}\right)$ 和 $\mathbf{F}^T\vec{R}$ 就可以计算出 ϕ 了。注意，LSTD-Q 算法不直接适用于估计最优 Q 函数 Q^*，因为 Q^* 的最优 Bellman 方程由于存在最大化操作而不是线性的，因此不能以封闭形式求解。

4.1.4 最小二乘策略迭代 (LSPI)

最后，我们讨论最小二乘策略迭代 (LSPI)，这是一种经典的离线强化学习算法，它使用对 Q 函数的线性近似来执行近似的策略迭代 (见 2.1 节的讨论)。LSPI 使用 LSTD-Q 来近似策略评估，于是就能得到对 Q^{π} 的估计，记为 Q_{ϕ}。然后，根据 Q_{ϕ} 使用贪心策略来进行策略迭代，比如，$\pi_{k+1}(\mathbf{a}\mid\mathbf{s})=\delta\left(\mathbf{a}=\arg\max_{\mathbf{a}}Q_{\phi}(\mathbf{s},\mathbf{a})\right)$。LSPI 的一个重要的特性是，当动作是离散的时，它不需要单独对策略近似表示，因此消除了 AC 算法中由于 Actor 的函数近似而产生的误差。

4.2 动态规划在 Offline 强化学习中产生的分布偏移问题

分布偏移在训练阶段和测试阶段都会对使用动态规划方法的 offline RL 产生影响。因为 Q 函数是通过 π_{β} 获得的数据集来学的，他的 state distribution 和 action distribution 和自己用学出来的 π 所采样出来的 state 和 action 的分布是不一样的。在测试阶段，offline RL 训练策略的状态访问概率分布 $d^{\pi}(s)$ 与训练数据的状态访问概率分布 $d^{\pi_{\beta}}(s)$ 存在系统差异。这意味着，策略梯度算法中，对于状态 $s\sim d^{\pi_{\beta}}(s)$ 通过 AC 算法可能会产生意想不到的错误动作。减小分布偏移的方法是限制学习策略和行为策略之间的散度。比如，通过限制 $D_{\mathrm{KL}}(\pi(a|s)\|\pi_{\beta}(a|s))\leq\epsilon$，可以得到 $D_{\mathrm{KL}}(d^{\pi}(s)\|d^{\pi_{\beta}}(s))$ 的下界为 δ，且 $\delta\propto O(\epsilon/(1-\gamma)^2)$。这个下界在实际中是非常松散的，但仍然表明，通过限定学习到的策略与收集离线训练数据的行为策略的偏离程度，可以减轻状态分布转移的影响。然而，这样的做法可能使最终性能不那么好，因为行为策略 (以及与之接近的任何策略) 可能比从 offline 数据中学到的最佳策略差得多。

需要注意的是，对于前面讨论的算法，状态分布偏移影响测试阶段的性能，但对训练没有影响。而在训练的时候，state 的分布没偏移，因为用到的 state 数据只有 π_β 采出来的数据 $\mathcal{D}$，无论是策略还是 Q 函数都不会在任何非从 $d^{\pi_\beta}(s)$ 中采样得到的状态下求值。而 action 的分布偏移了，因为训练时自己学的 action 是通过最大化 Q 来得到的，因此可能不在 $\mathcal{D}$ 中，那么 Q 的估计就不准，在不断的训练迭代中累积误差。所以解决 action distribution shift 是非常关键的。详细的说，Bellman 方程依赖于 $a_{t+1} \sim \pi(a_{t+1}|s_{t+1})$，评估 $Q^\pi(s_{t+1}, a_{t+1})$ 时，如果 $\pi(a|s)$ 和 $\pi_\beta(a|s)$ 有较大的差别，就会导致对目标 Q 值的估计有很大的偏差。又由于 $\max \mathbb{E}_{a\sim\pi(a|s)}[Q^\pi(s,a)]$ 的操作是问题进一步加剧，这意味着如果策略产生超出分布的行为，而学习到的 Q 函数会错误地产生过大的值。

分布偏移问题，在实际中将会带来巨大的问题，“unlearning”。

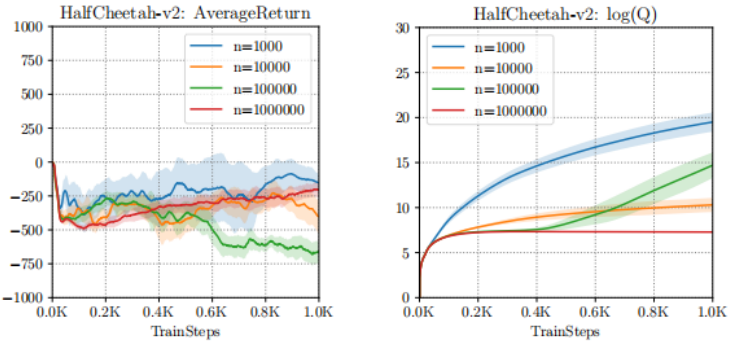

Figure 2: Performance of SAC (Haarnoja et al., 2018), an actor-critic method, on the HalfCheetah-v2 task in the offline setting, showing return as a function of gradient steps (left) and average learned Q-values on a log scale (right), for different numbers of training points (n). Note that an increase the number of samples does not generally prevent the “unlearning effect,” indicating that it is distinct from overfitting. Figure from Kumar et al. (2019a).

图 2: 分布偏移在 offline RL 中带来的问题

通过上述的实验可以观察到，学习曲线类似遇到了过拟合的情况：回报首先提高，然后随着训练的进行急剧下降。然而，即使我们增加数据集的大小，这种“过拟合”效应仍然存在，这表明这与经典的统计过拟合不同。然而，随着 Q 函数训练的时间越来越长，目标值错误越来越大，整个 Q 函数都退化了。

所以，如何解决动作的 out-of-distribution (OOD) 问题是使用动态规划方法求解 offline RL 问题的关键。下一节将会讨论“策略约束”的方法来解决此类问题。

4.3 策略约束在 Off-policy 评估和提升中的运用

通过动态规划进行离线强化学习的策略约束方法背后的基本思想是，使得 $\pi(a'|s')$ 和 $\pi_\beta(a|s)$ 之间靠近。这个操作使得计算 $\mathbb{E}_{a'\sim\pi(a'|s')}[Q(s',a')]$ 时，a' 是 in distribution 的，不会造成错误的累计。当然，在实际中，为了使学习到的策略优于行为策略，策略分布需要偏离，但通过保持这种偏移较小，由

于分布外动作输入而产生的错误可以得到控制。根据概率分布度量方法不一样和约束方法不一样，我们可以沿着这两个方向对方法进行广泛分类。

显式的 f 散度约束：直接增加对学习策略 π 和行为策略 π_β 直接的 f 散度作为约束。

隐式的 f 散度约束：使用对 *actor* 的更新，通过构造来使 π 靠近 π_β。

积分概率度量约束 (IPM)：对于 offline RL，该方法能更好地表达约束条件，具有较好的理论和经验特性。

策略惩罚：不仅可以对 actor 的更新增加约束，也可以对奖励函数或者目标 Q 值增加惩罚。

形式上，我们可以将具有策略约束的策略迭代方法表示为优化以下目标的不动点迭代:

$$\begin{aligned}
&\hat{Q}^{\pi}_{k+1} \leftarrow \arg\min_Q \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi_k(\mathbf{a}'|\mathbf{s}')}\left[\hat{Q}^{\pi}_k\left(\mathbf{s}',\mathbf{a}'\right)\right]\right)\right)^2\right] \\
&\pi_{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\hat{Q}^{\pi}_{k+1}(\mathbf{s},\mathbf{a})\right]\right] \text{ s.t. } D\left(\pi,\pi_\beta\right) \le \epsilon
\end{aligned} \tag{28}$$

一般实际处理中，并不是使得 min 和 max 操作收敛后才停止更新，而是执行一定的步数后停止更新。我们可将其看成是一般的 AC 算法，只不过在策略策略的更新上加上了正则化项 $D(\pi,\pi_\beta) < \epsilon$。对于 D 的选择有很多方法，对于这一类算法，我们称之为策略约束方法。

在**策略惩罚**方法中，对 AC 算法进行了改进，将约束归入 Q 值计算中，迫使策略不仅避免在每一状态下偏离 $\pi_\beta(a|s)$，而且避免在未来时间步长中出现可能导致和 $\pi_\beta(a|s)$ 偏离较大的动作。那么，就可以得到如下的表达:

$$\begin{aligned}
&\hat{Q}^{\pi}_{k+1} \leftarrow \arg\min_Q \\
&\quad \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi_k(\mathbf{a}'|\mathbf{s}')}\left[\hat{Q}^{\pi}_k\left(\mathbf{s}',\mathbf{a}'\right)\right] - \alpha\gamma D\left(\pi_k\left(\cdot \mid \mathbf{s}'\right),\pi_\beta\left(\cdot \mid \mathbf{s}'\right)\right)\right)\right)^2\right] \\
&\pi_{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\hat{Q}^{\pi}_{k+1}(\mathbf{s},\mathbf{a})\right] - \alpha D\left(\pi(\cdot \mid \mathbf{s}),\pi_\beta(\cdot \mid \mathbf{s})\right)\right]
\end{aligned} \tag{29}$$

虽然策略约束和策略惩罚方法的基本出发点是类似的，但如何定义约束和如何执行约束的具体选择在实践中可能产生显著的差异。接下来我们将讨论这些选择以及它们的权衡。

4.3.1 显式的 f 散度

对于任意的凸函数 f，其对应的 f 散度为:

$$D_f\left(\pi(\cdot \mid \mathbf{s}),\pi_\beta(\cdot \mid \mathbf{s})\right) = \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[f\left(\frac{\pi(\mathbf{a} \mid \mathbf{s})}{\pi_\beta(\mathbf{a} \mid \mathbf{s})}\right)\right]. \tag{30}$$

实际上 KL 散度，$\mathcal{X}^2$ 散度都是特殊的 f 散度，只不过是 f 函数不一样。一个 f 散度的变分形式也可以写成，这里是采用了凸共轭的形式，也是公 (30) 的对偶形式，

$$D_f\left(\pi(\cdot \mid \mathbf{s}),\pi_\beta(\cdot \mid \mathbf{s})\right) = \max_{x:S\times A\to\mathbb{R}} \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}[x(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mathbf{a}'\sim\pi_\beta(\cdot|\mathbf{s})}\left[f^*\left(x\left(\mathbf{s},\mathbf{a}'\right)\right)\right]. \tag{31}$$

f 散度的原形式 (30) 和对偶变分形式 (31) 都可以用来做策略约束。在对偶形式中，需要额外训练一个网络来表示函数 x。KL 散度表示为:

$$D_{KL}(\pi,\pi_\beta) = \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[\log\pi(\mathbf{a} \mid \mathbf{s}) - \log\pi_\beta(\mathbf{a} \mid \mathbf{s})\right]$$

可以通过采样动作 $a\sim\pi(\cdot|s)$，然后对期望内的概率进行抽样估计。经常把 f 散度当成正则化项的，这类算法太多了。此外，非对称松弛 f 散度的子族可用于策略约束。

4.3.2 隐式的 f 散度

隐式的 f 散度的基本思想是通过构造来使 π 靠近 π_β。KL-divergence 约束也可以隐式执行，就像在 AWR (Advantage-weighted regression: Simple and scalable off-policy reinforcement learning) 和 ABM (Keep doing what worked: Behavioral modelling priors for offline reinforcement learning) 的算法下一样。这些方法首先求解 KL-divergence 约束下的最优下一个策略迭代：$\bar{\pi}_{k+1}$，其计算过程如下所示，

$$\begin{aligned} \bar{\pi}_{k+1}(\mathbf{a} \mid \mathbf{s}) &\leftarrow \frac{1}{Z}\pi_\beta(\mathbf{a} \mid \mathbf{s}) \exp\left(\frac{1}{\alpha} Q_k^\pi(\mathbf{s}, \mathbf{a})\right) \\ \pi_{k+1} &\leftarrow \arg\min_\pi D_{\mathrm{KL}}\left(\bar{\pi}_{k+1}, \pi\right) \end{aligned} \tag{32}$$

这里用 Q 的指数加权 π_β 来估计一个新的 $\bar{\pi}$，之后让 π_β 和习得的 $\bar{\pi}$ 的距离小。这里 $\bar{\pi}$ 一定程度上代表着 π_β。其中第一步，可以采用从 $\pi_\beta(a|s)$(比如从经验回放池 $\mathcal{D}$ 中) 的重要性采样，而其重要性权重和 $\exp\left(\frac{1}{\alpha} Q_k^\pi(s,a)\right)$ 成正比的。第二步可以通过监督学习的方法来实现。这样，通过重要性采样在策略上有效地实现了 KL-divergence 约束，其中 α 对应拉格朗日乘子。Q 函数 Q_k^π 对应上一次迭代的策略 π_k，这里可以用可以用任意 Q 值或优势估计器进行估计。读者可以参考相关工作 (Advantage-weighted regression: Simple and scalable off-policy reinforcement learning)。

4.3.3 Integral probability metrics (IPMs)

策略约束的另一种选择是积分概率度量 (Integral probability metrics)，它是依赖于函数类 Φ 的函数形式的散度度量：

$$D_\Phi\left(\pi(\cdot \mid \mathbf{s}), \pi_\beta(\cdot \mid \mathbf{s})\right) = \sup_{\phi \in \Phi, \phi: S \times A \to \mathbb{R}} \left| \mathbb{E}_{\mathbf{a} \sim \pi(\cdot|\mathbf{s})}[\phi(\mathbf{s}, \mathbf{a})] - \mathbb{E}_{\mathbf{a}' \sim \pi_\beta(\cdot|\mathbf{s})}\left[\phi\left(\mathbf{s}, \mathbf{a}'\right)\right] \right|. \tag{33}$$

函数类 Φ 的不同选择会产生不同的散度。例如，Φ 由空间 (RKHS) 中可再生核希尔伯特空间 (RKHS) 中等的所有单位 Hilbert 范数的函数组成时，D_ϕ 表示最大均值差异 (MMD) 距离，也可以直接从样本中计算，方法如下:

$$\begin{aligned} \mathrm{MMD}^2\left(\pi(\cdot \mid \mathbf{s}), \pi_\beta(\cdot \mid \mathbf{s})\right) =& \mathbb{E}_{\mathbf{a} \sim \pi(\cdot|\mathbf{s}), \mathbf{a}' \sim \pi(\cdot|\mathbf{s})}\left[k\left(\mathbf{a}, \mathbf{a}'\right)\right] - \\ & 2\mathbb{E}_{\mathbf{a} \sim \pi(\cdot|\mathbf{s}), \mathbf{a}' \sim \pi_\beta(\cdot|\mathbf{s})}\left[k\left(\mathbf{a}, \mathbf{a}'\right)\right] + \mathbb{E}_{\mathbf{a} \sim \pi_\beta(\cdot|\mathbf{s}), \mathbf{a}' \sim \pi_\beta(\cdot|\mathbf{s})}\left[k\left(\mathbf{a}, \mathbf{a}'\right)\right] \end{aligned} \tag{34}$$

实际上也就是平方了一下，而在 RKHS 中，$\sup_{\phi \in \Phi} \phi(\cdot) \cdot \phi(\cdot)$ 就是核函数 $k(\cdot, \cdot)$。而当 ϕ 由单位 Lipschitz 常数组成的话，则 D_ϕ 等价为一阶 Wasserstein 距离：

$$W_1\left(\pi(\cdot \mid \mathbf{s}), \pi_\beta(\cdot \mid \mathbf{s})\right) = \sup_{f, \|f\|_L \leq 1} \left| \mathbb{E}_{\mathbf{a} \sim \pi(\cdot|\mathbf{s})}[f(\mathbf{a})] - \mathbb{E}_{\mathbf{a} \sim \pi_\beta(\cdot|\mathbf{s}}[f(\mathbf{a})] \right| \tag{35}$$

这些方法都很实用，因为它们通常可以用非参数估计器进行估计。读者可以阅读 (On integral probability metrics,ϕ-divergences and binary classification.) 得到详细的对 IPMs 的解读。同样，BEAR 也用 MMD 距离来表示策略约束 (Stabilizing off-policy q-learning via bootstrapping error reduction.)，而 Wu 等人 (Behavior regularized offline reinforcement learning.) 估计了一阶 Wasserstein 距离的实例化。

4.3.4 不同约束间的平衡

KL-divergence 约束，以及其他 f-divergences，代表了一类简单的约束方法，因为它们很容易被用于策略惩罚算法中，只需简单的对奖励函数增加：$\bar{r}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) - \alpha f\left(\pi(\mathbf{a} \mid \mathbf{s})/\pi_\beta(\mathbf{a} \mid \mathbf{s})\right)$，对应到 KL 散度的特殊情况下为：$\bar{r}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \alpha \log \pi_\beta(\mathbf{a} \mid \mathbf{s}) - \alpha \log \pi(\mathbf{a} \mid \mathbf{s})$，当使用最大熵强化学习算法时，最后一项包含在熵正则化器 $\mathcal{H}(\pi(\cdot|s))$ 内。但是，KL -divergence 和其他 f-divergence 对于离线强化学习并不一定是理想的。考虑一个行为策略是均匀分布，在这种情况下，offline 强化学习原则上应该是非常有效的。事实上，即使是没有任何策略约束参与的标准 AC 算法在这种设置下也可以执行得很好，因为当所有动作的概率都相等时，就不存在超出分布的动作。然而，在此设置中的 kl 散度约束将限制学习策略 $\pi(a|s)$ 使其靠近行为策略 $\pi_\beta(a|s)$，而 $\pi_\beta(a|s)$ 是均匀策略，这就使 $\pi(a|s)$ 不会过于集中，**算法产生一个高度随机的策略**。这显然是不合适的，这也表明 KL 散度约束并不一定是理想的选择。

相反，Kumar 等人 (Stabilizing off-policy q-learning via bootstrapping error reduction) 和 Laroche 等人 (Safe policy improvement with baseline bootstrapping.) 认为，只限制学习策略 π 和行为策略 π_β 的 support set，可能足以从理论上和经验上获得一种有效的 offline RL 算法。这部分是来自于，Kumar 等人提出的 Bootstrapping Error Accumulation Reduction (BEAR) 算法，大家可以在 Kumar 的 blog 中看到较好的解释：`https://bair.berkeley.edu/blog/2019/12/05/bear/`。这里有个新问题，什么是 support set？构造集的定义为：

$$\Pi_\epsilon = \{\pi \mid \pi(a \mid s) = 0 \text{ whenever } \pi_\beta(a \mid s) < \epsilon\}$$

实际上就是去掉部分概率值较小的部分。但是，使用支撑集的时候，不仅可以得到确定性和较优的学习策略，而且同时可以避免 OOD 动作。Kumar 举了一个 1 维迷宫的例子，从 S 出发，目标是终点 G。此模型展示在下图中，

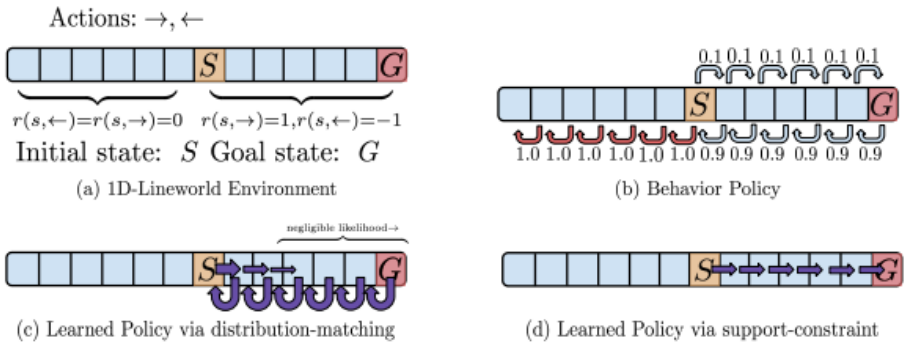

Figure 3: A comparison of support and distribution constraints on a simple 1D lineworld from Kumar (2019). The task requires moving to the goal location (marked as 'G') starting from 'S'. The behavior policy strongly prefers the left action at each state (b), such that distribution constraint is unable to find the optimal policy (c), whereas support-constraint can successfully obtain the optimal policy (d). We refer to Kumar (2019) for further discussion.

图 3: 迷宫例子

全约束学习策略和行为策略，会明显更倾向于向左边移动，到达目标点的概率并不高。而支撑集约束可以学习到最优策略，因为当行为策略的概率密度小于某个阈值时，学习策略是学习不到的。**个人理解，通过这样的方式，将硬优化问题转换成了软优化问题，在 Offline RL 中，我们真正希望的不是我们的学习策略和 $\mathcal{D}$ 的行为策略越像越好，而是学习策略在行为策略的支撑集的范围内去进行优化。所以，我们可以把对学习策略的约束变成让学习策略保持在行为策略的支撑集内呢。这样的做法可帮助学习策略，不会陷入到不好的行为策略中**。限制 support set 并不容易。对于离散动作，可以直接让出现在 π_β suport set 之外的概率尽可能小，即为：

$$\arg\min_{\pi} D_{\text{support },\varepsilon}\left(\pi(\cdot \mid s), \pi_\beta(\cdot \mid s)\right) = \arg\min_{\pi} \sum_{\mathbf{a}\in A, \pi_\beta(\mathbf{a}|\mathbf{s})\leq\varepsilon} \pi(\mathbf{a} \mid \mathbf{s}) \tag{36}$$

当 MDP 为一个连续的行动空间并且无法估计精确的支持度时，Kumar 表明一个有限样本的 MMD 距离估计可以用来近似地约束学习策略的支持度。其中，MMD 距离被定义为:

$$\mathrm{MMD}^2(X, Y) = \frac{1}{n^2}\sum_{i,i'} k\left(x_i, x_{i'}\right) - \frac{2}{nm}\sum_{i,j} k\left(x_i, y_j\right) + \frac{1}{m^2}\sum_{j,j'} k\left(y_j, y_{j'}\right) \tag{37}$$

4.4 基于不确定性估计的离线近似动态规划 (Offline approximated dynamic programming with uncertainty estimation)

4.3 小节中描述的策略约束方法可以有效的防止在计算 Q 函数的目标值时出现 out-of-distribution (OOD) 动作。与直接 constraint 的思路不同另一种思路是，我们希望让 Q-function 对 OOD 的 action 其更有适应力。通常不确定性分两种，数据不确定性和模型不确定性。数据不确定性是指 data 有可能出错，而模型不确定性是学到的模型参数有可能出错，进一步导致模型的输出可能出错。这里很容易把模型不确定性和模型输出的概率搞混淆。举例来说，把一张猫的图片输入到一个分类器里，模型以很高的概率判断这是猫，并不代表模型不确定性低。但是如果把一个模型训练 n 次，这 n 个模型都一致判断这是猫，才能说模型对于这张猫图的不确定性低。这里用到的判断模型不确定性的指标就是输出的方差。因此，训练学到的 Q 函数就是模型，那么对于 OOD 的动作作为 Q 函数的输入，Q 函数的模型不确定性就更大。**那么通过减小 Q 函数的不确定性，就可以防止输出 OOD 的动作，达到和策略约束异曲同工的效果**。

这种方法需要从数据集 $\mathcal{D}$ 中学习到 Q 函数的不确定集合或分布，我们可以表示 $P_{\mathcal{D}}(Q^\pi)$。那么，如何来计算 $P_{\mathcal{D}}(Q^\pi)$ 的不确定性呢? 可以用 ensemble 的方式通过衡量输出的方差来估计它，或者对 Q 值分布进行参数化，假设其为已知的分布，如 Gaussian，通过学习 Gaussian 分布参数的形式来实现。如果不确定度可以学习，由此 Q 函数的估计可以替换为：

$$\pi_{k+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{\mathbf{s}\sim\mathcal{D}} \underbrace{\left[\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\mathbb{E}_{Q_{k+1}^\pi\sim\mathcal{P}_{\mathcal{D}}(Q^\pi)}\left[Q_{k+1}^\pi(\mathbf{s}, \mathbf{a})\right] - \alpha\,\mathrm{Unc}\left(\mathcal{P}_{\mathcal{D}}\left(Q^\pi\right)\right)\right]\right]}_{\text{conservative estimate}}, \tag{38}$$

其中，Unc 表示为不确定度，减去它就得到了实际 Q 函数的保守估计结果。不确定度度量 Unc 的选择取决于不确定度估计方法的具体选择。而 bootstrap 集成被用来表示 Q 函数在以前的工作中是很常见的。方法有很多，感兴趣的同学可以自行查看。

4.5 挑战和开放性问题

在 4.2 小节已经讨论了，基本的动态规划近似算法由于分布偏移的问题，表现非常的差。而分布偏移主要是由动作分布的行为策略 $\pi_\beta(a|s)$ 和学习策略 $\pi(a|s)$ 的差距过大导致。此部分我们讨论了策略约束和显式的不确定性估计如何在原则上缓解这一问题，并且讨论这两种方法的一些缺点。

虽然，基于不确定性估计的方法思想看着很有趣，但是实际运用中，估计 $P_{\mathcal{D}}(Q^\pi)$ 和 Unc 是非常困难的。而且，策略约束的方法效果远胜于纯粹的基于非确定性估计的方法。但是，基于非确定性估计的方法广泛的被运用于 **"探索"** 中。在探索问题中，往往会倾向于采用不确定性高的动作，来增加算法的探索性能。但是，在 offline RL 中，我们必须相对于不确定性集采取 conservative 的行动，对不确定性估计精度的要求要高得多。而探索算法中，只需要探索的集合中，有效果好的动作就行，如果有很多效果差的动作也没事。然而，offline RL 中需要设置不确定性来直接度量 Q 函数的可信度，这需要更高的要求。实际上，不完美的不确定性集可能会导致过于保守的估计 (这会阻碍学习)，或者过于松散的估计 (会导致产生 OOD 动作)。**当然，策略约束或基于不确定性方法可能在之后会有很大的改进，研究人员会开发出更好的认知不确定性估计的方法或更好的衡量分布测度算法，并融入到 offline RL 中。**

策略约束的方法在目前取得了较好的性能，也是大家通常使用的方法，但是其还是有很多的缺点。

1. 第一，大部分策略约束，从数据集 $\mathcal{D}$ 拟合一个行为策略 $\pi_\beta(\cdot|s)$ 的估计，并用其来约束学习策略 $\pi(\cdot|s)$。这意味着算法的性能非常依赖于对行为策略估计的准确度。在现实环境中，类似的高度多模态动作的情况下，对行为策略的估计可能非常困难。例如，如果使用单峰高斯策略来建模高维的动作分布，可能无法防止学习策略 π 选择 OOD 动作。**那么这里的开放性问题是，是否可以设计一种策略约束，可以直接从数据集中观察到的样本中计算得到，而不需要拟合一个对行为策略的估计**。其主要思想是，既然估计行为策略很困难，我们能不能直接跳过对行为策略的估计，而直接计算约束呢?

2. 第二，即使行为策略估计准了，函数拟合不准的问题又来了。当用神经网络表示 Q 函数，函数近似的不良影响可能会阻碍 Q 函数的学习，即使在目标值中控制了 OOD 的动作。比如，数据集不够大，在小数据集中近似动态规划算法非常容易过拟合。**在 online 强化学习中，由于函数近似而产生的高估误差可以通过主动数据收集来修正。而 offline RL 中这些误差会累积起来，并影响 offline RL 中之后的迭代过程，而如何解决这个问题仍然是一个开放的问题**。

3. 第三，即使 Q 函数在训练过程中从不估计 OOD 状态，但 Q 函数仍然受训练集状态分布 $d^{\pi_\beta}(s)$ 的影响。比如，例如，在 $d^{\pi_\beta}(s)$ 条件下，函数近似耦合两种概率不同的状态的 Q 值时，使用函数近似的动态规划可能会对概率较低的 $d^{\pi_\beta}(s)$ 状态产生错误的解。这个问题可能会更严重地影响使用函数近似的 offline RL 算法，因为它们根本无法控制数据的分布。

4. offline 近似动态规划的另一个潜在挑战是，行为策略的改进程度受到累积误差的限制。由于在 offline RL 中，整体误差为 $\mathcal{O}(H^2)$，这个在 2.2 小节分析过。每一步与行为策略的小差异就累积起来，最终会导致最终学习策略严重偏离行为分布，性能很差。所以，这要求每一步的策略约束要很强，以确保整体误差比较小。那么，这样的强约束肯定会影响学习策略的提升。最近的研究中所探讨的 (Algaedice: Policy gradient from arbitrary experience.)，另一种选择是直接限制策略的状态动作边缘分布。这部分内容在 3.4 小节的边缘重要性采样中已经详细描述过了，然而对策略表示和使用状态动作边缘分布需要使用贝尔曼方程迭代，而且可能会遇到 OOD 动作。那么

一个开放性问题就来了，研究一种约束以便有效地权衡所学习策略的误差积累和次优性，并且可以在实践中通过标准优化技术建议实现，而不需要额外的函数近似器来拟合行为策略。

5 Model-Based offline 强化学习

小编不太了解 Model-based 的强化学习，这里只简要介绍一下。

Online 的 Model-based RL 已经有许多文章研究，主流方法是从数据集中先学习一个状态转移模型，然后可以直接通过 planning 来生成 action，或者直接训练一个 policy 函数。但其中一个核心问题仍然是，数据集学出来的状态转移模型 $T_\phi(s_{t+1}|s_t, a_t)$ 是对于 π_β 的，并非对应于 plan 的或者学到的策略，所以核心问题仍然是 distributional shift。目前极少工作探讨 offline model-based RL，所以如何解决 model-based RL 中的 distributional shift 是一个公开问题。同时，就算是 online 的 model-based RL 也有自身的挑战，主要是高维的 observation 和 long horizon 的问题。另外，是否 model-based 模型能在理论上帮助提升 model-free DP 算法还是一个公开问题。原因是尽管 DP 方法没有直接学一个动态模型，但是相当于学了一个无参模型。本质上，DP 和 model-based RL 都是在做预测问题，前者预测 future return, 后者预测 future states. 因此，探索 offline RL 对于 non-linear 函数估计的 model-based model 与 DP 方法的 theoretical bounds on the optimal performance 也是一个公开问题。

6 Application and Evaluation

在这个部分，作者讨论了 offline RL 的评估方法，baseline，和应用。首先讨论离线强化学习算法在之前的工作中是如何被评估的，然后讨论这些方法已经产生影响的具体应用领域。这部分比较简单，大家可以自己看看。