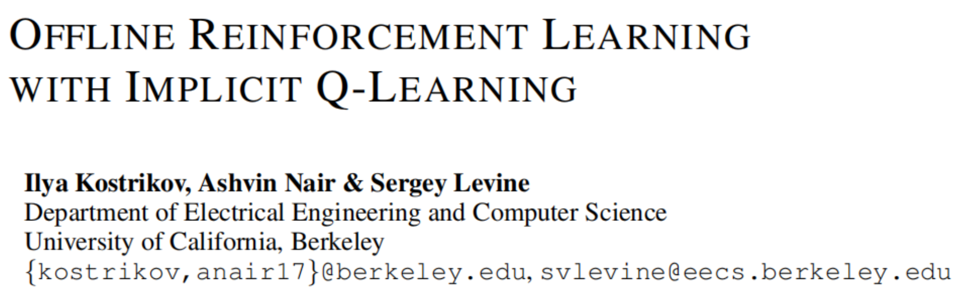
離線強化學習(Offline RL)系列3: (演算法篇) IQL(Implicit Q-learning)演算法詳解與實現

知
zhuanlan.zhihu.com /p/497358947

Jensen Wang 在讀Ph.D，研究深度強化學習


OFFLINE REINFORCEMENT LEARNING WITH IMPLICIT Q-LEARNING

Ilya Kostrikov, Ashvin Nair & Sergey Levine
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{kostrikov,anair17}@berkeley.edu, svlevine@eecs.berkeley.edu

[更新記錄]

論文資訊：Ilya Kostrikov, Ashvin Nair, Sergey Levine: “Offline Reinforcement Learning with Implicit Q-Learning”, 2021; **arXiv:2110.06169** .

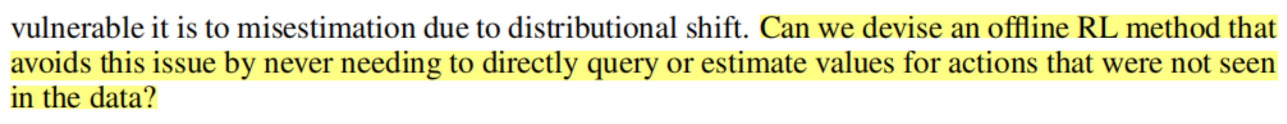
本論文由柏克萊Sergey Levine團隊的Ilya Kostrikov以第一作者提出，發表在ICLR2022頂會上,並被確定為**Poster**，接收意見是：「*This paper proposes a new paradigm --- called in-sample Q learning --- to tackle offline reinforcement learning. Based on the novel idea of using expectile regression, the proposed algorithm enjoys stable performance by focusing on in-sample actions and avreing querying the actions appicion semion squerying the sstm querying the sm. , outperforming existing baselines on several tasks. The paper is also well written.* ”

> **摘要**：目前基於策略約束和正規化的離線強化學習演算法非常廣泛，它們直接正面去解決OOD以外的動作分佈，使得Learned policy能夠很好的逼近行為策略，本文提出的**IQL(Implicit Q-learning)**直接沒有去學習OOD以外的動作，而是用已知的state-action進行學習，透過使用SARSA style的方式重構策略和值函數（引入Expectile Regression）L，在策略的抽取方面採用了AWR(Advantage Weighted Regression)方式抽取，直接確定Q值如何隨著不同的動作而變化，並藉助隨機動態對未來結果進行平均，而不是確定Q值如何隨著不同的未來結果而變化.結果表明，該算法可以達到SOTA的效果。

1. 問題及背景簡介

在前幾篇部落格文章[**BCQ**、**BEAR**、**BRAC**、**TD3+BC**、**CQL**、**REM**]中，這些方法都面臨了OOD問題，為了解決這些問題，作者們都從station-action的軌跡優化著手，透過使用函數近似，行為克隆、支撐集、隨機Q值函數等方面不斷讓學習策略（learned policy）和行為策略（behavior policy）距離變小或者處在一個範圍內，不管從Policy Constraint還是regulization方面都達到了不錯的效果。

於是，本文的作者就發出疑問：


vulnerable it is to misestimation due to distributional shift. Can we devise an offline RL method that avoids this issue by never needing to directly query or estimate values for actions that were not seen in the data?

這也是本篇論文要解決的核心問題。在開始前我們先說兩個概念**Multi-step DP**和**Single-step DP**，以及前序工作。

1.1 Multi-step DP

最近提出的離線RL 方法的很大一部分是基於約束或正則化的近似動態規劃（例如，Q-learning 或actor-critic 方法），constraint或regulizaion用於限制與行為策略的偏差。 我們將這些方法稱為**多步驟動態規劃(Multi-step DP)**演算法，因為它們對多次迭代執行真正的動態規劃，因此如果提供高覆蓋率數據，原則上可以恢復最佳策略。通常Multi-step DP問題也可以分為：

- 顯式密度模型(explicit density model)：BRAC，BCQ，BEAR等
- 隱性差異約束（implicit divergence constraints）：AWAC，CRR，AWR等

1.2 Single-step DP

與Multi-step DP相比，有一種方法依賴單步策略迭代的方法，即對行為策略的價值函數或Q函數進行擬合，然後提取相應的貪心策略，或者完全避免價值函數並利用行為克隆目標。 我們將這些統稱為**單步（Single-step DP）**方法。 這些方法也避免了query看不見的station-action，**因為它們要不是根本不使用價值函數，就是學習行為策略的價值函數**。

下圖是一個關於one-step和multi-step的比較：

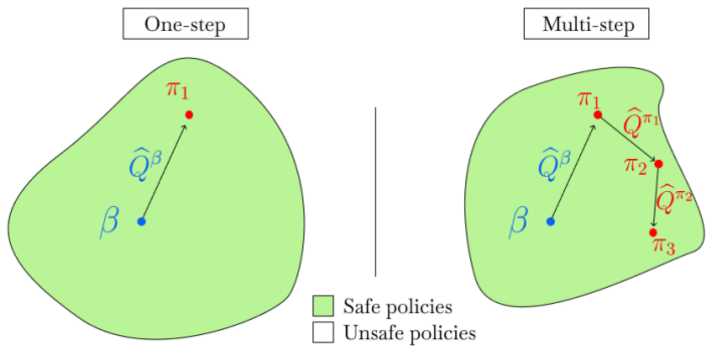


Figure 1: A cartoon illustration of the difference between one-step and multi-step methods. All algorithms constrain themselves to a neighborhood of "safe" policies around β. A one-step approach (left) only uses the on-policy $\widehat{Q}^{\beta}$, while a multi-step approach (right) repeatedly uses off-policy $\widehat{Q}^{\pi_i}$.

1.3 Offline RL 優化目標

我們知道對於離線強化學習來說，最佳化的TD目標為：

L_{TD}(\theta)=\mathbb{E}_{\left(s, a, s^{\prime}\right) \sim \mathcal{D}}\left[\left(r(s, a)+\gamma \max _{a^{\prime}} Q_{\hat{\theta}}\left(s^{\prime}, a^{\prime}\right)-Q_{\theta} (s, a)\right)^{2}\right] \\

其中公式中的Q_{\hat{\theta}}(\cdot)表示target network的值函數，且\pi(s)= \arg \max_{a} Q_{\theta}(s,a)，那麼就可以發現對於處於OOD之外的動作對a^{'}，就會計算得到一個錯誤的Q_{\hat{\theta}}(s^{'},a^{'})，從而通過\ max導致了**Overestimate**，且這個誤差會不斷地增大而導致learned策略變廢。

於是作者根據**SARSA** style重新建構了一個損失函數：

L(\theta)=\mathbb{E}_{\left(s, a, s^{\prime}, a^{\prime}\right) \sim \mathcal{D}}\left[\left(r(s, a)+\gamma Q_{\hat{\theta}}\left(s^{\prime}, a^{\prime}\right)-Q_{\theta}(s, a)\right)^{2}\right] \\

$$L_{TD}(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma \boxed{\max_{a'}} Q_{\hat{\theta}}(s',a') - Q_\theta(s,a))^2]$$

$$L(\theta) = \mathbb{E}_{(s,a,s',a')\sim\mathcal{D}}[(r(s,a) + \gamma Q_{\hat{\theta}}(s',a') - Q_\theta(s,a))^2].$$

> 對兩個公式進行對比發現少了\max操作，也就是說這裡不在query 處於OOD之外的動作，只對分佈內的資料學習（處於fixed Dataset D）。

那麼如果假設D的容量無限，且沒有取樣誤差，則應滿足最優參數（貝爾曼最優方程式）

Q_{\theta^{*}}(s, a) \approx r(s, a)+\gamma \mathbb{E}_{\substack{s^{\prime} \sim p(\cdot \mid s , a) \\ a^{\prime} \sim \pi_{\beta}(\cdot \mid s)}}\left[Q_{\hat{\theta}}\left(s^{\prime}, a^{\prime}\right)\right] \\

於是作者為了估計在支援資料分佈動作上（support of the data distribution）的最大Q值, 在不查詢學習到的Q函數的情況下透過使用** 期望迴歸(Expectile Regression)**，目標定義如下：

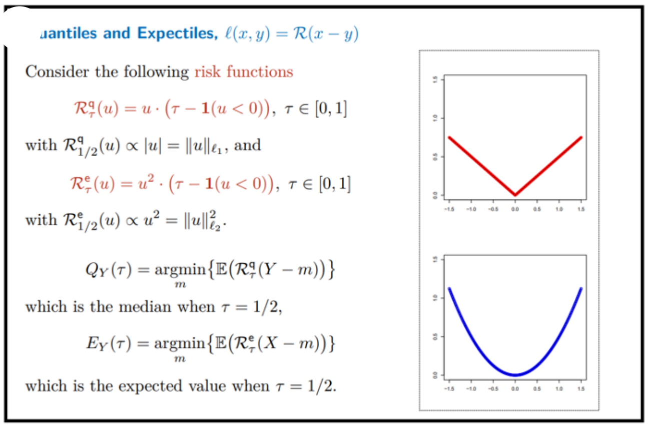
L(\theta)=\mathbb{E}_{\left(s, a, s^{\prime}\right) \sim \mathcal{D}}\left[\left(r(s, a)+ \gamma \max _{\substack{a^{\prime} \in \mathcal{A} \\ \text { st } \pi_{\beta}\left(a^{\prime} \mid s^{\ prime}\right)>0}} Q_{\hat{\theta}}\left(s^{\prime}, a^{\prime}\right)-Q_{\theta}(s, a)\right)^{2}\right] \\

說到這裡，作者只是解釋如何處理OOD之外的數據， 那麼實際的策略、值函數等更新過程到底是啥樣的？

1.4 期望回歸(Expectile Regression)

> 【可跳過】(但建議看一看)

下面是一張關於Expectile Regression和Quantiles Regression之間的差異。



ıantiles and Expectiles, $\ell(x,y)=\mathcal{R}(x-y)$

Consider the following risk functions

$$\mathcal{R}^{\mathrm{q}}_{\tau}(u)=u\cdot\big(\tau-\mathbf{1}(u<0)\big),\ \tau\in[0,1]$$

with $\mathcal{R}^{\mathrm{q}}_{1/2}(u)\propto|u|=\|u\|_{\ell_1}$, and

$$\mathcal{R}^{\mathrm{e}}_{\tau}(u)=u^2\cdot\big(\tau-\mathbf{1}(u<0)\big),\ \tau\in[0,1]$$

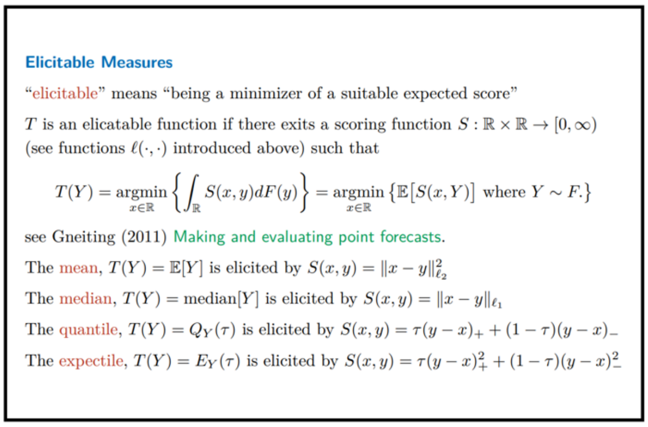
with $\mathcal{R}^{\mathrm{e}}_{1/2}(u)\propto u^2=\|u\|^2_{\ell_2}$.

$$Q_Y(\tau)=\underset{m}{\operatorname{argmin}}\big\{\mathbb{E}\big(\mathcal{R}^{\mathrm{q}}_{\tau}(Y-m)\big)\big\}$$

which is the median when $\tau=1/2$,

$$E_Y(\tau)=\underset{m}{\operatorname{argmin}}\big\{\mathbb{E}\big(\mathcal{R}^{\mathrm{e}}_{\tau}(X-m)\big)\big\}$$

which is the expected value when $\tau=1/2$.



Elicitable Measures

"elicitable" means "being a minimizer of a suitable expected score"

T is an elicatable function if there exits a scoring function $S:\mathbb{R}\times\mathbb{R}\to[0,\infty)$ (see functions $\ell(\cdot,\cdot)$ introduced above) such that

$$T(Y)=\underset{x\in\mathbb{R}}{\operatorname{argmin}}\left\{\int_{\mathbb{R}}S(x,y)dF(y)\right\}=\underset{x\in\mathbb{R}}{\operatorname{argmin}}\left\{\mathbb{E}\big[S(x,Y)\big]\text{ where }Y\sim F.\right\}$$

see Gneiting (2011) Making and evaluating point forecasts.

The mean, $T(Y)=\mathbb{E}[Y]$ is elicited by $S(x,y)=\|x-y\|^2_{\ell_2}$

The median, $T(Y)=\text{median}[Y]$ is elicited by $S(x,y)=\|x-y\|_{\ell_1}$

The quantile, $T(Y)=Q_Y(\tau)$ is elicited by $S(x,y)=\tau(y-x)_+ + (1-\tau)(y-x)_-$

The expectile, $T(Y)=E_Y(\tau)$ is elicited by $S(x,y)=\tau(y-x)^2_+ + (1-\tau)(y-x)^2_-$

在論文中的定義如下（和圖中的基本沒差別）：

\underset{m_{\tau}}{\arg \min } \mathbb{E}_{x \sim X}\left[L_{2}^{\tau}\left(x-m_{\tau}\ right)\right] \\ L_{2}^{\tau}(u)=|\tau-\mathbb{1}(u<0)| u^{2} \\

作者的目標是去預測條件期望回歸，即找到最小函數下的m_{\tau}(x)

\underset{m_{\tau}(x)}{\arg \min } \mathbb{E}_{(x, y) \sim \mathcal{D}}\left[L_{2}^{\tau} \left(y-m_{\tau}(x)\right)\right] \\

透過實驗，作者得出了不同的\tau對應的函數，並在二維分佈上實驗得到圖(right)

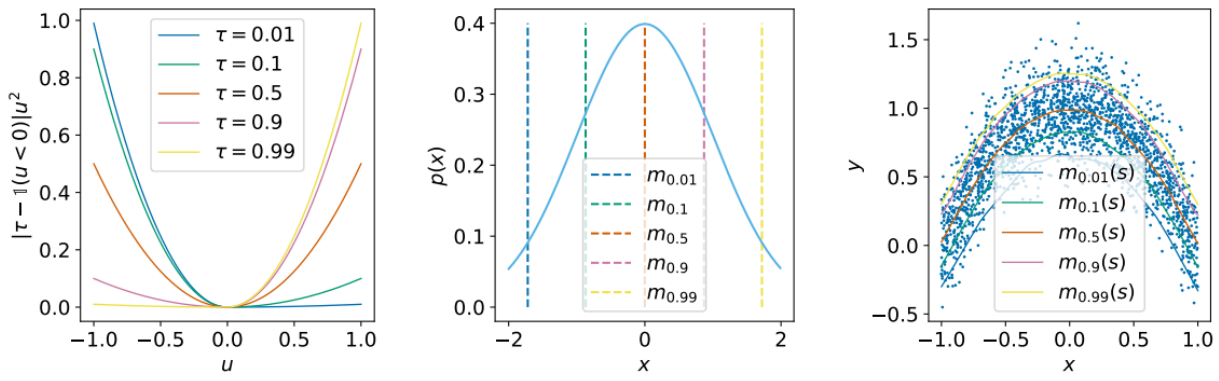
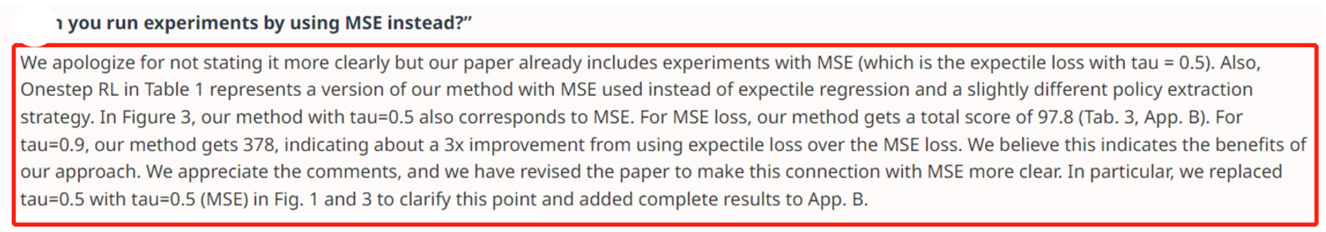


Figure 1: **Left:** The asymmetric squared loss used for expectile regression. $\tau=0.5$ corresponds to the standard mean squared error loss, while $\tau=0.9$ gives more weight to positives differences. **Center:** Expectiles of a normal distribution. **Right:** an example of estimating state conditional expectiles of a two-dimensional random variable. Each x corresponds to a distribution over y. We can approximate a maximum of this random variable with expectile regression: $\tau=0.5$ correspond to the conditional mean statistics of the distribution, while $\tau\approx 1$ approximates the maximum operator over in-support values of y.

這裡我在看文章時納悶，為什麼這個所謂的**Expectile Regression**要比直接的MSE以及quantiles好用，後來看了審稿意見，審稿人也提出了對應疑問

- There is lots of unknown about why expectile regression was selected for value function updates and it is not clear what is the motivation to go with expectile regression rather than mean regression. Particularly, what would happen if use mean regression, i.e. $L_V(\phi)=E_{(s,a)}[(Q_{\hat{\theta}}(s,a)-V_{\phi}(s))^2]$, where it requires no τ? How will performance change? Can you run experiments by using MSE loss instead?

下面是作者給的回應：總之就是我使用不同的\tau效果就是好，有點Incremental的感覺。



ı you run experiments by using MSE instead?”

We apologize for not stating it more clearly but our paper already includes experiments with MSE (which is the expectile loss with tau = 0.5). Also, Onestep RL in Table 1 represents a version of our method with MSE used instead of expectile regression and a slightly different policy extraction strategy. In Figure 3, our method with tau=0.5 also corresponds to MSE. For MSE loss, our method gets a total score of 97.8 (Tab. 3, App. B). For tau=0.9, our method gets 378, indicating about a 3x improvement from using expectile loss over the MSE loss. We believe this indicates the benefits of our approach. We appreciate the comments, and we have revised the paper to make this connection with MSE more clear. In particular, we replaced tau=0.5 with tau=0.5 (MSE) in Fig. 1 and 3 to clarify this point and added complete results to App. B.

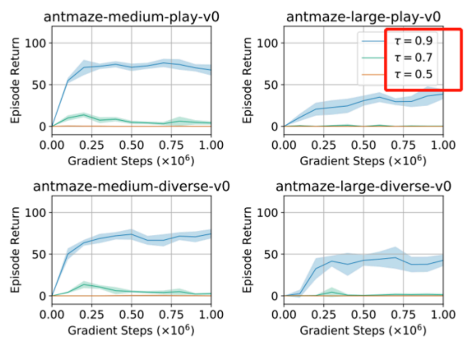


Figure 3: Estimating a larger expectile τ is crucial for antmaze tasks that require dynamical programming (’stitching’).

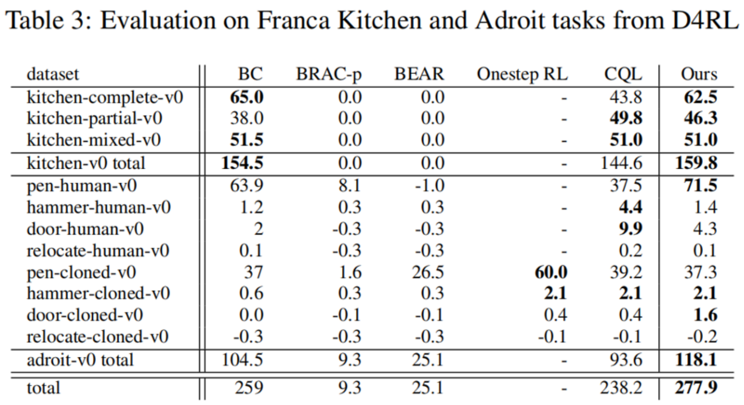
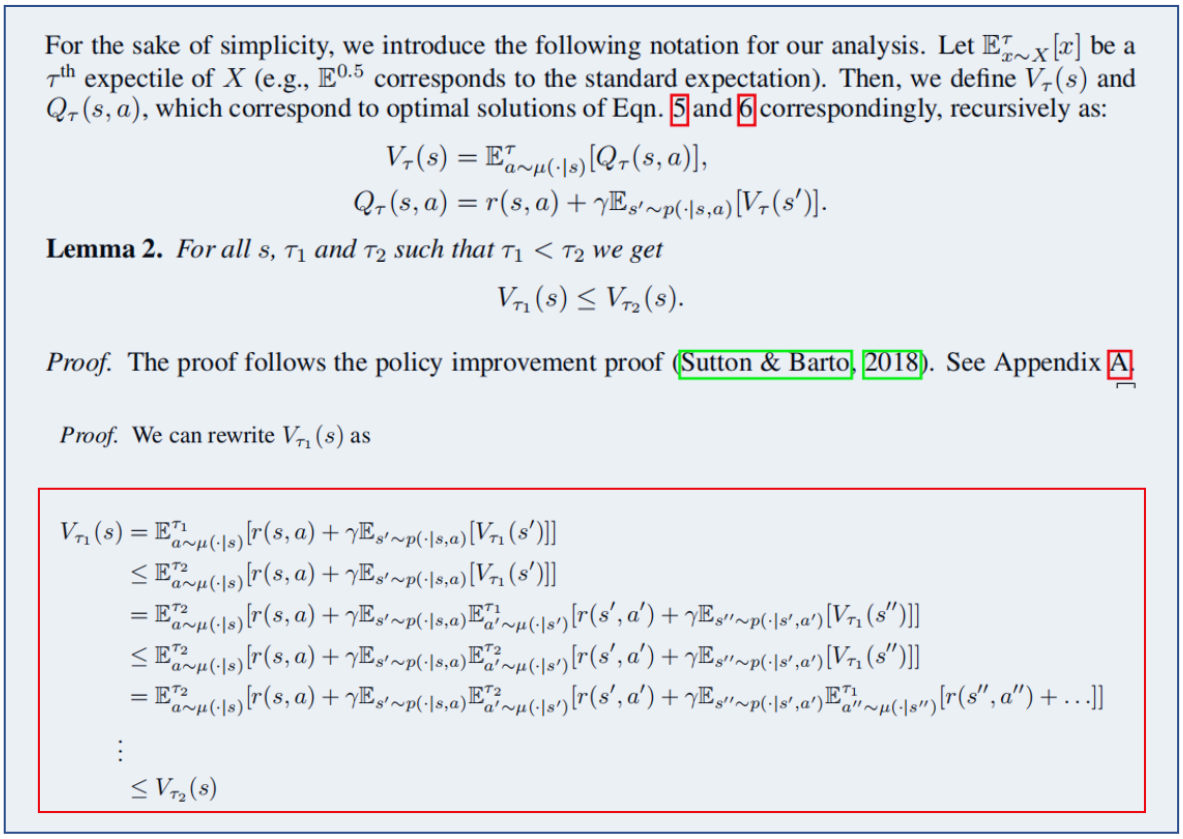


Table 3: Evaluation on Franca Kitchen and Adroit tasks from D4RL

dataset	BC	BRAC-p	BEAR	Onestep RL	CQL	Ours
kitchen-complete-v0	**65.0**	0.0	0.0	-	43.8	**62.5**
kitchen-partial-v0	38.0	0.0	0.0	-	**49.8**	**46.3**
kitchen-mixed-v0	**51.5**	0.0	0.0	-	**51.0**	**51.0**
kitchen-v0 total	**154.5**	0.0	0.0	-	144.6	**159.8**
pen-human-v0	63.9	8.1	-1.0	-	37.5	**71.5**
hammer-human-v0	1.2	0.3	0.3	-	**4.4**	1.4
door-human-v0	2	-0.3	-0.3	-	**9.9**	4.3
relocate-human-v0	0.1	-0.3	-0.3	-	0.2	0.1
pen-cloned-v0	37	1.6	26.5	**60.0**	39.2	37.3
hammer-cloned-v0	0.6	0.3	0.3	**2.1**	**2.1**	**2.1**
door-cloned-v0	0.0	-0.1	-0.1	0.4	0.4	**1.6**
relocate-cloned-v0	-0.3	-0.3	-0.3	-0.1	-0.1	-0.2
adroit-v0 total	104.5	9.3	25.1	-	93.6	**118.1**
total	259	9.3	25.1	-	238.2	**277.9**

具體的關於相關的分析如下：



For the sake of simplicity, we introduce the following notation for our analysis. Let $\mathbb{E}^{\tau}_{x\sim X}[x]$ be a τ^{th} expectile of X (e.g., $\mathbb{E}^{0.5}$ corresponds to the standard expectation). Then, we define $V_\tau(s)$ and $Q_\tau(s,a)$, which correspond to optimal solutions of Eqn. 5 and 6 correspondingly, recursively as:

$$V_\tau(s) = \mathbb{E}^{\tau}_{a\sim\mu(\cdot|s)}[Q_\tau(s,a)],$$
$$Q_\tau(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}[V_\tau(s')].$$

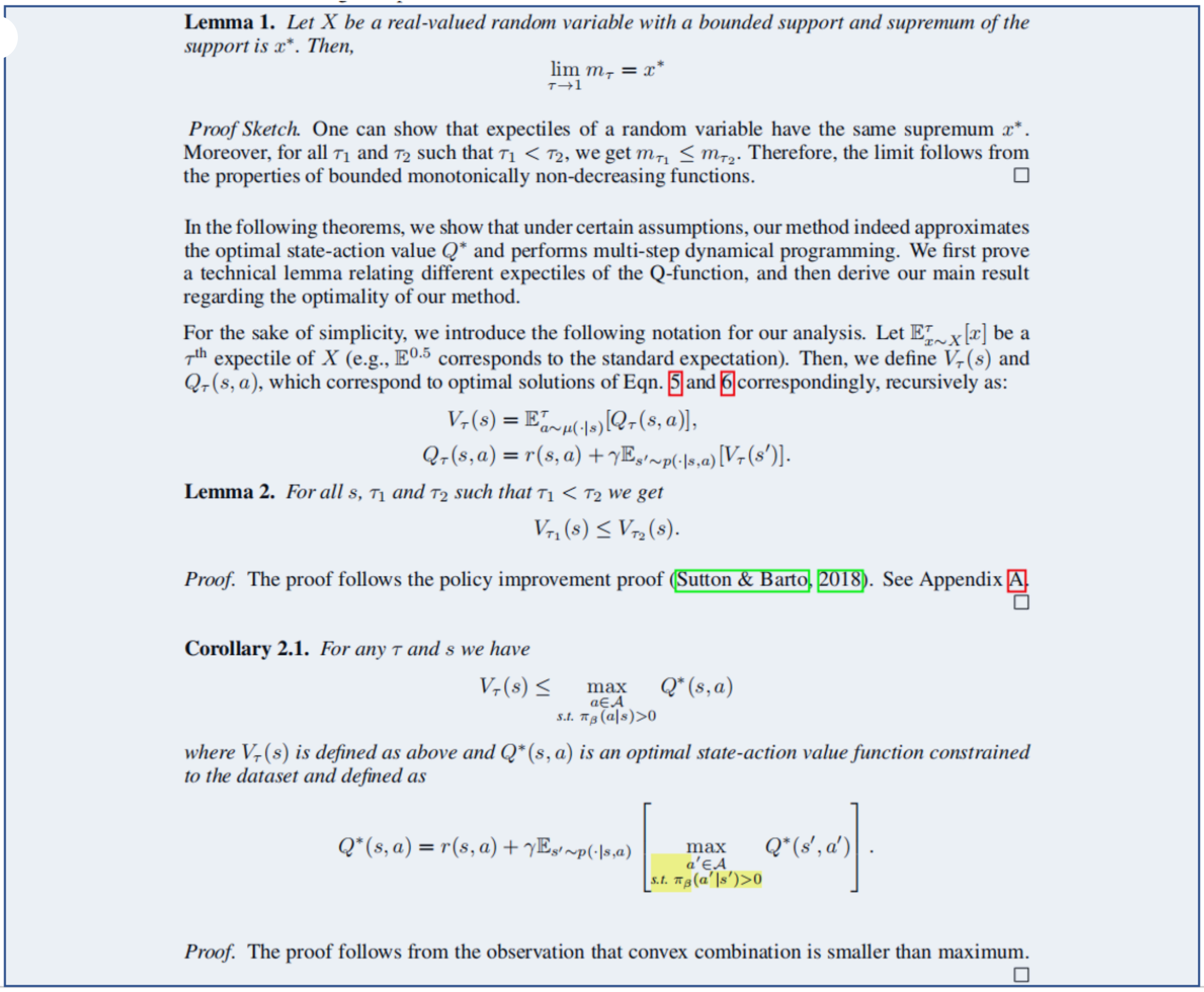
Lemma 2. *For all s, τ_1 and τ_2 such that $\tau_1 < \tau_2$ we get*

$$V_{\tau_1}(s) \le V_{\tau_2}(s).$$

Proof. The proof follows the policy improvement proof (Sutton & Barto, 2018). See Appendix A.

Proof. We can rewrite $V_{\tau_1}(s)$ as

$$\begin{aligned}
V_{\tau_1}(s) &= \mathbb{E}^{\tau_1}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}[V_{\tau_1}(s')]] \\
&\le \mathbb{E}^{\tau_2}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}[V_{\tau_1}(s')]] \\
&= \mathbb{E}^{\tau_2}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}\mathbb{E}^{\tau_1}_{a'\sim\mu(\cdot|s')}[r(s',a') + \gamma\mathbb{E}_{s''\sim p(\cdot|s',a')}[V_{\tau_1}(s'')]] \\
&\le \mathbb{E}^{\tau_2}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}\mathbb{E}^{\tau_2}_{a'\sim\mu(\cdot|s')}[r(s',a') + \gamma\mathbb{E}_{s''\sim p(\cdot|s',a')}[V_{\tau_1}(s'')]] \\
&= \mathbb{E}^{\tau_2}_{a\sim\mu(\cdot|s)}[r(s,a) + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}\mathbb{E}^{\tau_2}_{a'\sim\mu(\cdot|s')}[r(s',a') + \gamma\mathbb{E}_{s''\sim p(\cdot|s',a')}\mathbb{E}^{\tau_1}_{a''\sim\mu(\cdot|s'')}[r(s'',a'') + \ldots]] \\
&\vdots \\
&\le V_{\tau_2}(s)
\end{aligned}$$



Lemma 1. *Let X be a real-valued random variable with a bounded support and supremum of the support is x^*. Then,*

$$\lim_{\tau \to 1} m_\tau = x^*$$

Proof Sketch. One can show that expectiles of a random variable have the same supremum x^*. Moreover, for all τ_1 and τ_2 such that $\tau_1 < \tau_2$, we get $m_{\tau_1} \leq m_{\tau_2}$. Therefore, the limit follows from the properties of bounded monotonically non-decreasing functions. □

In the following theorems, we show that under certain assumptions, our method indeed approximates the optimal state-action value Q^* and performs multi-step dynamical programming. We first prove a technical lemma relating different expectiles of the Q-function, and then derive our main result regarding the optimality of our method.

For the sake of simplicity, we introduce the following notation for our analysis. Let $\mathbb{E}^\tau_{x\sim X}[x]$ be a τ^{th} expectile of X (e.g., $\mathbb{E}^{0.5}$ corresponds to the standard expectation). Then, we define $V_\tau(s)$ and $Q_\tau(s,a)$, which correspond to optimal solutions of Eqn. 5 and 6 correspondingly, recursively as:

$$V_\tau(s) = \mathbb{E}^\tau_{a\sim\mu(\cdot|s)}[Q_\tau(s,a)],$$
$$Q_\tau(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}[V_\tau(s')].$$

Lemma 2. *For all s, τ_1 and τ_2 such that $\tau_1 < \tau_2$ we get*

$$V_{\tau_1}(s) \leq V_{\tau_2}(s).$$

Proof. The proof follows the policy improvement proof (Sutton & Barto, 2018). See Appendix A. □

Corollary 2.1. *For any τ and s we have*

$$V_\tau(s) \leq \max_{\substack{a\in\mathcal{A} \\ \text{s.t. } \pi_\beta(a|s)>0}} Q^*(s,a)$$

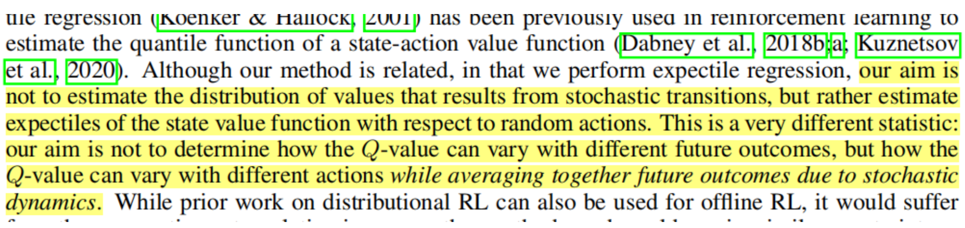
where $V_\tau(s)$ is defined as above and $Q^*(s,a)$ is an optimal state-action value function constrained to the dataset and defined as

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}\left[\max_{\substack{a'\in\mathcal{A} \\ \text{s.t. } \pi_\beta(a'|s')>0}} Q^*(s',a')\right].$$

Proof. The proof follows from the observation that convex combination is smaller than maximum. □

2. IQL原理部分

作者在文章中產生了IQL與普通演算法的最大的差異在於這裡：



the regression (Koenker & Hallock, 2001) has been previously used in reinforcement learning to estimate the quantile function of a state-action value function (Dabney et al., 2018b;a; Kuznetsov et al., 2020). Although our method is related, in that we perform expectile regression, our aim is not to estimate the distribution of values that results from stochastic transitions, but rather estimate expectiles of the state value function with respect to random actions. This is a very different statistic: our aim is not to determine how the Q-value can vary with different future outcomes, but how the Q-value can vary with different actions *while averaging together future outcomes due to stochastic dynamics*. While prior work on distributional RL can also be used for offline RL, it would suffer

> **譯**：我們的目標不是估計隨機轉換（stochastic transitions）產生的值的分佈，而是**估計狀態值函數相對於隨機動作的期望值**。這是一個非常不同的統計數據：我們的目標不是確定Q值如何隨著不同的未來結果而變化，而是**確定Q值如何隨著不同的動作而變化**，同時隨機動態（stochastic dynamics.）的對未來結果進行平均。

2.1 值函數構造(Expectile Regression構造)

以下是作者根據1.4將的Expectile Regression建構Loss函數

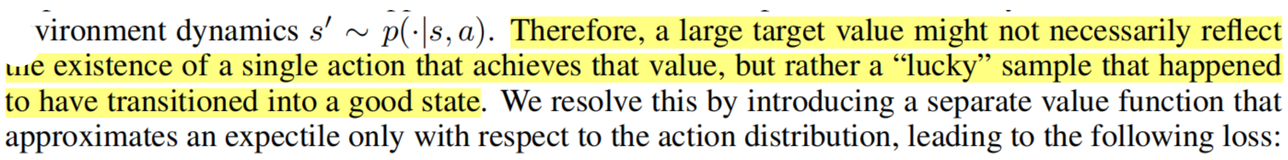
L(\theta)=\mathbb{E}_{\left(s, a, s^{\prime}, a^{\prime}\right) \sim \mathcal{D}}\left[L_{2 }^{\tau}\left(r(s, a)+\gamma Q_{\hat{\theta}}\left(s^{\prime}, a^{\prime}\right)-Q_{\ theta}(s, a)\right)\right] \\

其中的L_{2}^{\tau}表示為Expectile Regression函數，作者在不同的\tau下進行比較實驗，最後得出當\tau=0.9時效果最佳，相較於MSE效能提升3倍以上。

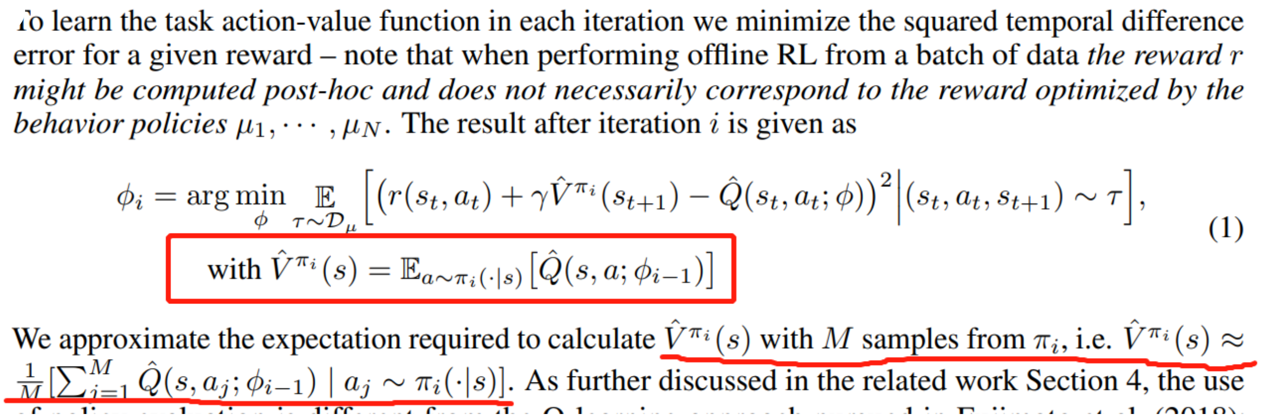
另外作者對值函數和狀態值函數分別分開的進行了構造，其中separate value function和Q-function 的計算過程如下：

L_{V}(\psi)=\mathbb{E}_{(s, a) \sim \mathcal{D}}\left[L_{2}^{\tau}\left(Q_{\hat{\ theta}}(s, a)-V_{\psi}(s)\right)\right] \\ L_{Q}(\theta)=\mathbb{E}_{\left(s, a, s^ {\prime}\right) \sim \mathcal{D}}\left[\left(r(s, a)+\gamma V_{\psi}\left(s^{\prime}\right)-Q_{ \theta}(s, a)\right)^{2}\right] \\

> **備註**：這裡作者說一個大的target value不能代表一個好的action


vironment dynamics $s' \sim p(\cdot|s,a)$. Therefore, a large target value might not necessarily reflect the existence of a single action that achieves that value, but rather a “lucky” sample that happened to have transitioned into a good state. We resolve this by introducing a separate value function that approximates an expectile only with respect to the action distribution, leading to the following loss:

於是提出單獨的網路來表示值函數，而該網路只近似（函數逼近）對動作分佈的期望（這裡作者參考了Behavioral Modelling Priors for Offline Reinforcement Learning）中的V的定義\text { with } \hat {V}^{\pi_{i}}(s)=\mathbb{E}_{a \sim \pi_{i}(\cdot \mid s)}\left[\hat{Q}\left(s , a ; \phi_{i-1}\right)\right]


To learn the task action-value function in each iteration we minimize the squared temporal difference error for a given reward – note that when performing offline RL from a batch of data *the reward r might be computed post-hoc and does not necessarily correspond to the reward optimized by the behavior policies $\mu_1, \cdots, \mu_N$*. The result after iteration i is given as

$$\phi_i = \arg\min_{\phi} \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}_\mu}\left[\left(r(s_t,a_t)+\gamma\hat{V}^{\pi_i}(s_{t+1})-\hat{Q}(s_t,a_t;\phi)\right)^2 \middle| (s_t,a_t,s_{t+1})\sim\tau\right], \tag{1}$$
$$\boxed{\text{with } \hat{V}^{\pi_i}(s)=\mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[\hat{Q}(s,a;\phi_{i-1})\right]}$$

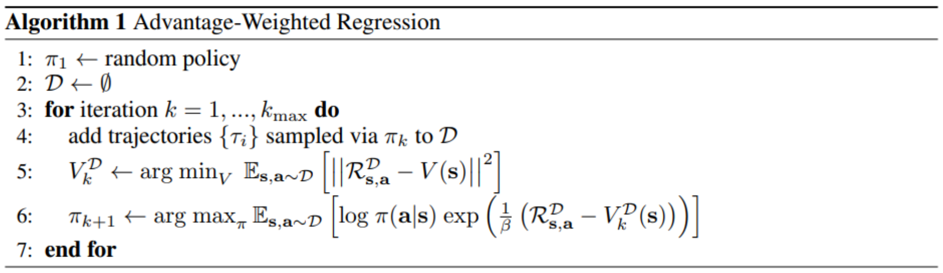
We approximate the expectation required to calculate $\hat{V}^{\pi_i}(s)$ with M samples from π_i, i.e. $\hat{V}^{\pi_i}(s) \approx \frac{1}{M}\left[\sum_{j=1}^{M}\hat{Q}(s,a_j;\phi_{i-1}) \mid a_j \sim \pi_i(\cdot|s)\right]$. As further discussed in the related work Section 4, the use

2.2 策略函數建構(Policy Extraction by AWR)

建構了值函數候，下一步就是建構策略函數，作者在這裡提出了使用AWR中的策略抽取（policy extraction）的方法,數學表達如下：

L_{\pi}(\phi)=\mathbb{E}_{(s, a) \sim \mathcal{D}}\left[\exp \left(\beta\left(Q_{\hat{\theta }}(s, a)-V_{\psi}(s)\right)\right) \log \pi_{\phi}(a \mid s)\right] \\

這裡我找到了AWR的抽取方法，具體的偽代碼如下：

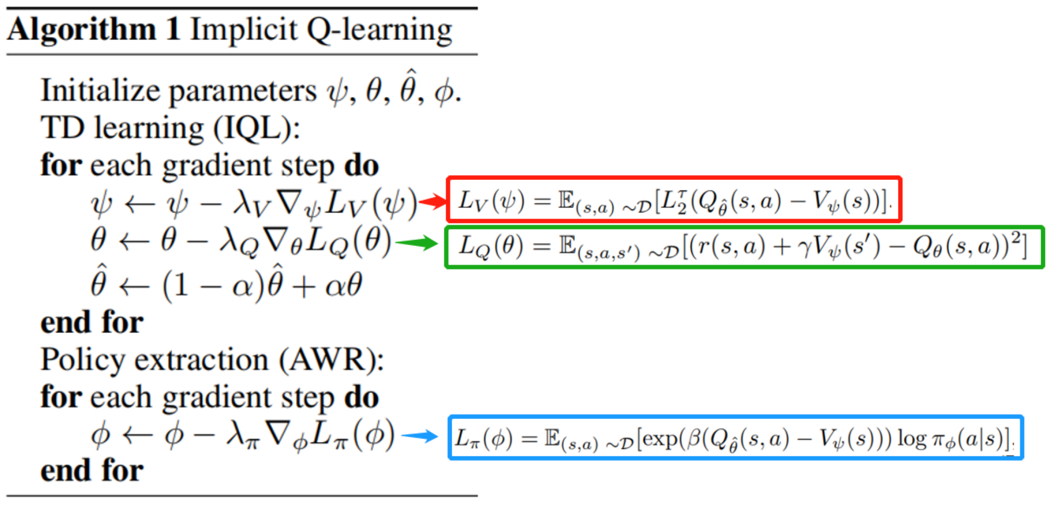

Algorithm 1 Advantage-Weighted Regression

```
1: $\pi_1 \leftarrow$ random policy
2: $\mathcal{D} \leftarrow \emptyset$
3: for iteration $k = 1, ..., k_{\max}$ do
4:     add trajectories $\{\tau_i\}$ sampled via $\pi_k$ to $\mathcal{D}$
5:     $V_k^{\mathcal{D}} \leftarrow \arg\min_V \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left|\left|\mathcal{R}_{\mathbf{s},\mathbf{a}}^{\mathcal{D}} - V(\mathbf{s})\right|\right|^2\right]$
6:     $\pi_{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\log\pi(\mathbf{a}|\mathbf{s})\exp\left(\frac{1}{\beta}\left(\mathcal{R}_{\mathbf{s},\mathbf{a}}^{\mathcal{D}} - V_k^{\mathcal{D}}(\mathbf{s})\right)\right)\right]$
7: end for
```

從偽代碼可以看出，作者構造的L_{\pi}(\phi)和AWR基本上相似，接下來就是演算法的執行過程。

2.3 偽代碼

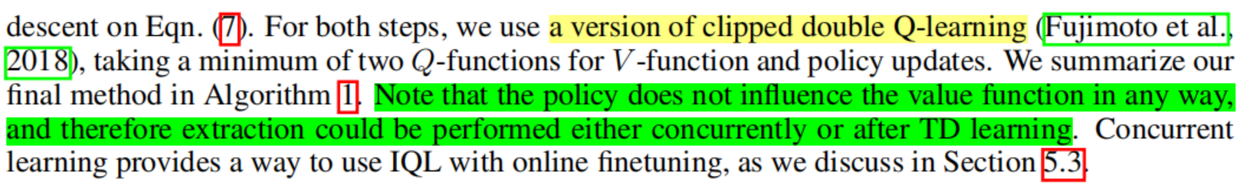
下圖是本文的核心程式碼，



Algorithm 1 Implicit Q-learning

```
Initialize parameters $\psi$, $\theta$, $\hat{\theta}$, $\phi$.
TD learning (IQL):
for each gradient step do
    $\psi \leftarrow \psi - \lambda_V \nabla_\psi L_V(\psi)$ → $L_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\tau(Q_{\hat{\theta}}(s,a) - V_\psi(s))]$.
    $\theta \leftarrow \theta - \lambda_Q \nabla_\theta L_Q(\theta)$ → $L_Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a))^2]$
    $\hat{\theta} \leftarrow (1-\alpha)\hat{\theta} + \alpha\theta$
end for
Policy extraction (AWR):
for each gradient step do
    $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi L_\pi(\phi)$ → $L_\pi(\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[\exp(\beta(Q_{\hat{\theta}}(s,a) - V_\psi(s))) \log \pi_\phi(a|s)]$.
end for
```

作者表明：可以簡單地透過修改SARSA style的TD backup 損失函數來做到這一點，而無需在目標值Q中使用樣本外(OOD)操作。**一旦這個Q函數收斂，就可以使用優勢加權行為克隆來提取對應的策略。** ,另外作者在文中強調：



descent on Eqn. (7). For both steps, we use a version of clipped double Q-learning (Fujimoto et al., 2018), taking a minimum of two Q-functions for V-function and policy updates. We summarize our final method in Algorithm 1. Note that the policy does not influence the value function in any way, and therefore extraction could be performed either concurrently or after TD learning. Concurrent learning provides a way to use IQL with online finetuning, as we discuss in Section 5.3.

3. 實驗結果分析

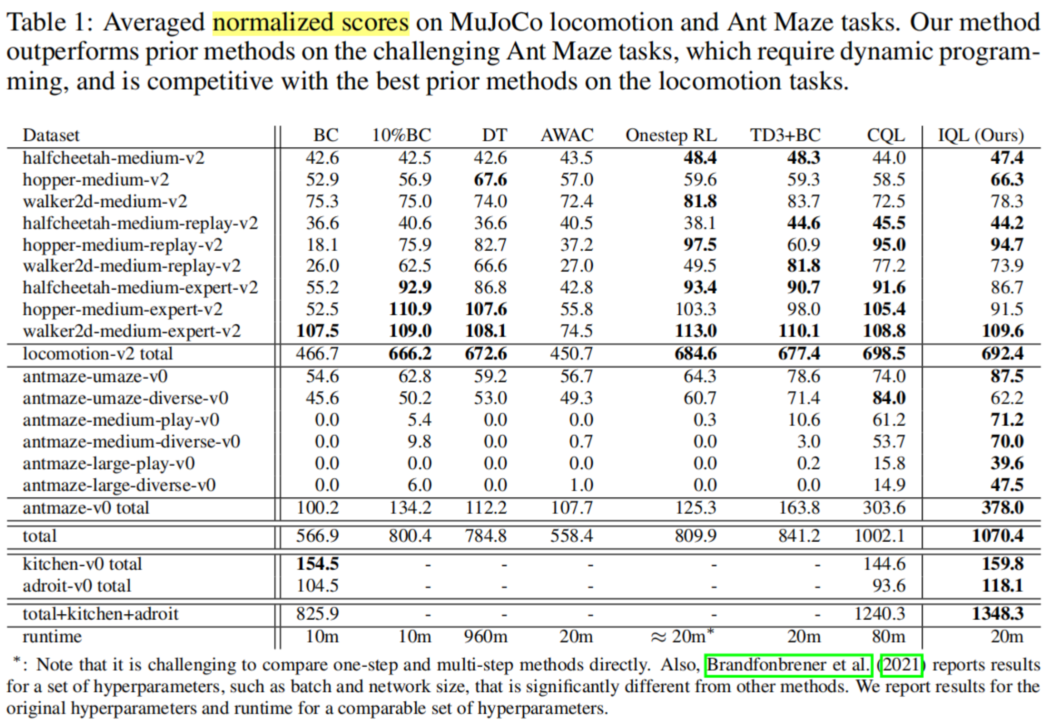


Table 1: Averaged normalized scores on MuJoCo locomotion and Ant Maze tasks. Our method outperforms prior methods on the challenging Ant Maze tasks, which require dynamic programming, and is competitive with the best prior methods on the locomotion tasks.

Dataset	BC	10%BC	DT	AWAC	Onestep RL	TD3+BC	CQL	IQL (Ours)
halfcheetah-medium-v2	42.6	42.5	42.6	43.5	**48.4**	**48.3**	44.0	**47.4**
hopper-medium-v2	52.9	56.9	**67.6**	57.0	59.6	59.3	58.5	**66.3**
walker2d-medium-v2	75.3	75.0	74.0	72.4	**81.8**	83.7	72.5	78.3
halfcheetah-medium-replay-v2	36.6	40.6	36.6	40.5	38.1	**44.6**	**45.5**	**44.2**
hopper-medium-replay-v2	18.1	75.9	82.7	37.2	**97.5**	60.9	**95.0**	**94.7**
walker2d-medium-replay-v2	26.0	62.5	66.6	27.0	49.5	**81.8**	77.2	73.9
halfcheetah-medium-expert-v2	55.2	**92.9**	86.8	42.8	**93.4**	**90.7**	**91.6**	86.7
hopper-medium-expert-v2	52.5	**110.9**	**107.6**	55.8	103.3	98.0	**105.4**	91.5
walker2d-medium-expert-v2	**107.5**	**109.0**	**108.1**	74.5	**113.0**	**110.1**	**108.8**	**109.6**
locomotion-v2 total	466.7	**666.2**	**672.6**	450.7	**684.6**	**677.4**	**698.5**	**692.4**
antmaze-umaze-v0	54.6	62.8	59.2	56.7	64.3	78.6	74.0	**87.5**
antmaze-umaze-diverse-v0	45.6	50.2	53.0	49.3	60.7	71.4	**84.0**	62.2
antmaze-medium-play-v0	0.0	5.4	0.0	0.0	0.3	10.6	61.2	**71.2**
antmaze-medium-diverse-v0	0.0	9.8	0.0	0.7	0.0	3.0	53.7	**70.0**
antmaze-large-play-v0	0.0	0.0	0.0	0.0	0.0	0.2	15.8	**39.6**
antmaze-large-diverse-v0	0.0	6.0	0.0	1.0	0.0	0.0	14.9	**47.5**
antmaze-v0 total	100.2	134.2	112.2	107.7	125.3	163.8	303.6	**378.0**
total	566.9	800.4	784.8	558.4	809.9	841.2	1002.1	**1070.4**
kitchen-v0 total	**154.5**	-	-	-	-	-	144.6	**159.8**
adroit-v0 total	104.5	-	-	-	-	-	93.6	**118.1**
total+kitchen+adroit	825.9	-	-	-	-	-	1240.3	**1348.3**
runtime	10m	10m	960m	20m	≈ 20m*	20m	80m	20m

*: Note that it is challenging to compare one-step and multi-step methods directly. Also, Brandfonbrener et al. (2021) reports results for a set of hyperparameters, such as batch and network size, that is significantly different from other methods. We report results for the original hyperparameters and runtime for a comparable set of hyperparameters.

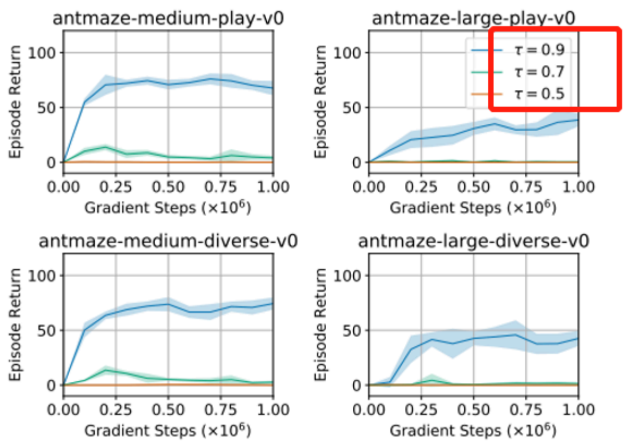
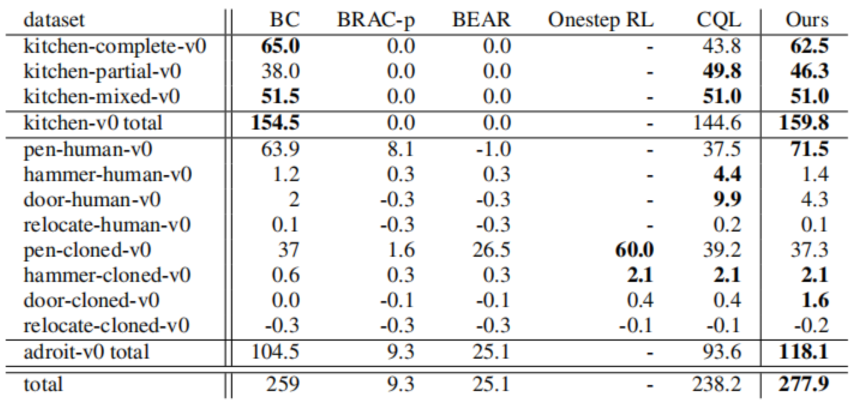


Figure 3: Estimating a larger expectile τ is crucial for antmaze tasks that require dynamical programming ('stitching').

Table 3: Evaluation on Franca Kitchen and Adroit tasks from D4RL



dataset	BC	BRAC-p	BEAR	Onestep RL	CQL	Ours
kitchen-complete-v0	**65.0**	0.0	0.0	-	43.8	**62.5**
kitchen-partial-v0	38.0	0.0	0.0	-	**49.8**	**46.3**
kitchen-mixed-v0	**51.5**	0.0	0.0	-	**51.0**	**51.0**
kitchen-v0 total	**154.5**	0.0	0.0	-	144.6	**159.8**
pen-human-v0	63.9	8.1	-1.0	-	37.5	**71.5**
hammer-human-v0	1.2	0.3	0.3	-	**4.4**	1.4
door-human-v0	2	-0.3	-0.3	-	**9.9**	4.3
relocate-human-v0	0.1	-0.3	-0.3	-	0.2	0.1
pen-cloned-v0	37	1.6	26.5	**60.0**	39.2	37.3
hammer-cloned-v0	0.6	0.3	0.3	**2.1**	**2.1**	**2.1**
door-cloned-v0	0.0	-0.1	-0.1	0.4	0.4	**1.6**
relocate-cloned-v0	-0.3	-0.3	-0.3	-0.1	-0.1	-0.2
adroit-v0 total	104.5	9.3	25.1	-	93.6	**118.1**
total	259	9.3	25.1	-	238.2	**277.9**

4. 程式碼實現

```
"""
        QF Loss
        """
        q1_pred = self.qf1(obs, actions)
        q2_pred = self.qf2(obs, actions)
        target_vf_pred = self.vf(next_obs).detach()

        q_target = self.reward_scale * rewards + (1. - terminals) * self.discount * target_vf_pred
        q_target = q_target.detach()
        qf1_loss = self.qf_criterion(q1_pred, q_target)
        qf2_loss = self.qf_criterion(q2_pred, q_target)

        """
        VF Loss
        """
        q_pred = torch.min(
            self.target_qf1(obs, actions),
            self.target_qf2(obs, actions),
        ).detach()
        vf_pred = self.vf(obs)
        vf_err = vf_pred - q_pred
        vf_sign = (vf_err > 0).float()
        vf_weight = (1 - vf_sign) * self.quantile + vf_sign * (1 - self.quantile)
        vf_loss = (vf_weight * (vf_err ** 2)).mean()

        """
        Policy Loss
        """
        policy_logpp = dist.log_prob(actions)

        adv = q_pred - vf_pred
        exp_adv = torch.exp(adv / self.beta)
        if self.clip_score is not None:
            exp_adv = torch.clamp(exp_adv, max=self.clip_score)

        weights = exp_adv[:, 0].detach()
        policy_loss = (-policy_logpp * weights).mean()

        """
        Update networks
        """
        if self._n_train_steps_total % self.q_update_period == 0:
            self.qf1_optimizer.zero_grad()
            qf1_loss.backward()
            self.qf1_optimizer.step()

            self.qf2_optimizer.zero_grad()
            qf2_loss.backward()
            self.qf2_optimizer.step()

            self.vf_optimizer.zero_grad()
            vf_loss.backward()
            self.vf_optimizer.step()

        if self._n_train_steps_total % self.policy_update_period == 0:
            self.policy_optimizer.zero_grad()
            policy_loss.backward()
            self.policy_optimizer.step()
```

參考文獻

[1]. Ilya Kostrikov, Ashvin Nair, Sergey Levine: “Offline Reinforcement Learning with Implicit Q-Learning”, 2021; **arXiv:2110.06169** .

[2]. Arthur Charpentier, Quantile and Expectile Regresions, Erasmus School of Economics, May 2017, **Access**

[3]. Hajo Holzmann, Bernhard Klar: “Expectile Asymptotics”, 2015; **arXiv:1509.06866** .

[4]. David Brandfonbrener, William F. Whitney, Rajesh Ranganath, Joan Bruna: “Offline RL Without Off-Policy Evaluation”, 2021; **arXiv:2106.08909** .

[5]. Noah Y. Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, Martin Riedmiller: “Keep Doing What Worked: Behavioral Modelling Priors for Offline Reinforcement Ling” , ICLR 2020; **arXiv:2002.08396** .