Offline Reinforcement Learning

Weinan Zhang

Shanghai Jiao Tong University

http://wnzhang.net

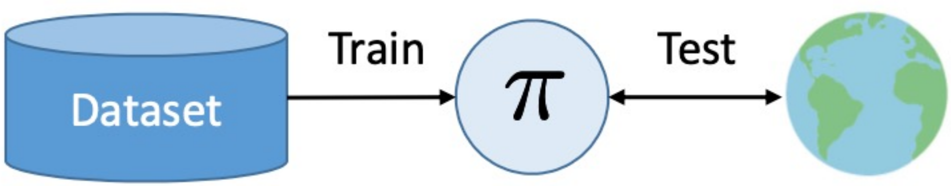
Apr. 2023

Content

- Overview of offline RL
- Offline RL training methods
  - Imitation learning
  - Model-free methods
  - Model-based methods
- Offline policy evaluation
- Offline RL benchmarks

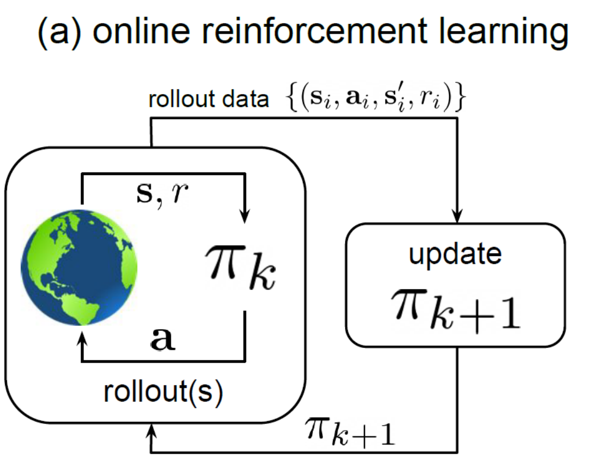
Overview of Offline RL (or Batch RL)

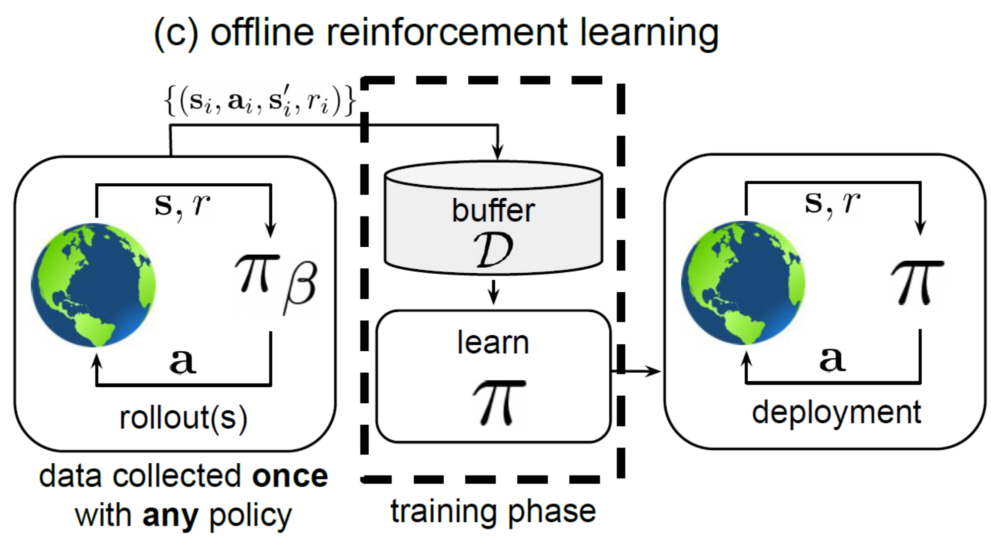
- Training an RL agent from zero in real environment is sometimes risky (auto-driving, health care) or costly (robot control, recommender system).

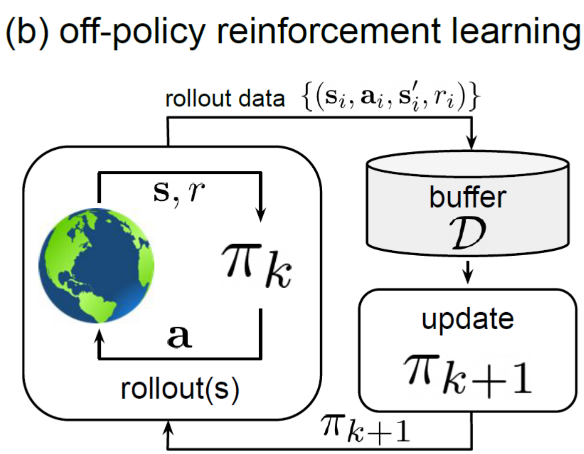


- Since there is often not a good simulator, the way of training the agent only from a pre-collected offline dataset is promoted, called offline RL, or batch RL.
- In offline setting, namely pure batch setting, the agent only has access to the offline data set, instead of interacting with the environment.

Overview of Offline RL





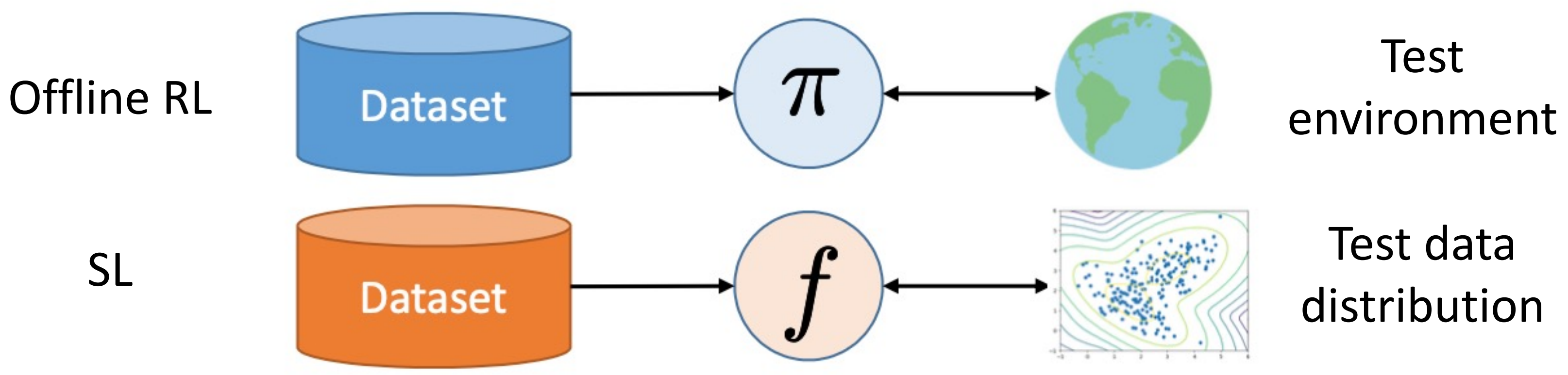


- Though both off-policy RL and offline RL evaluate the policy using the data sampled from a replay buffer, they are different.
- The key difference is whether the agent can interact with the environment while learning
- Offline RL techniques help deploy RL to real-world applications

Levine, Sergey, et al. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems." *arXiv 2005.01643* (2020).

Advantages of Offline RL

- Offline RL can help
    1. Pretrain an RL agent using existing datasets
    2. Empirically evaluate RL algorithms based on their ability to exploit a fixed dataset of interactions
    3. Bridge the gap between academic interest in RL and real-world applications
- Offline RL makes RL more like supervised learning



Gulcehre, Caglar, et al. "Rl unplugged: Benchmarks for offline reinforcement learning." *arXiv preprint arXiv:2006.13888* (2020).

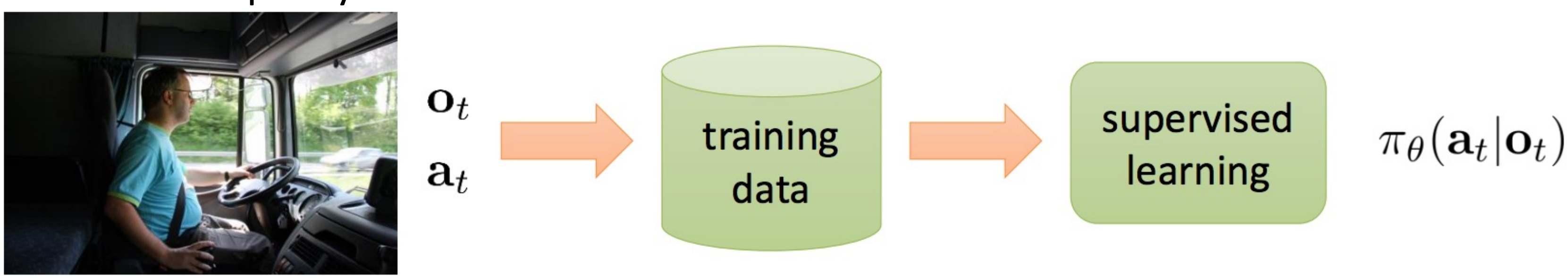
Content

- Overview of offline RL
- Offline RL training methods
  - Imitation learning
  - Model-free methods
  - Model-based methods
- Offline policy evaluation
- Offline RL benchmarks

Behavior Cloning as Offline RL

- Behavior cloning, the simplest imitation learning method, requires no environment interaction

Behavioral policy



- Learning objective of BC

Behavioral policy

$$\hat{\pi}^* = \arg\min_{\pi} \mathbb{E}_{s\sim\rho^s_{\pi_E}} \left[\ell\left(\pi_E(\cdot|s), \pi(\cdot|s)\right)\right]$$

- Obvious shortcomings of BC
  1. The policy upper bound is the behavioral policy
  2. Distribution shift

Overview of Offline RL Methods

Model-free Methods

- Explicit constraint
  - BCQ
  - BEAR
  - BRAC
  - CQL
- Implicit constraint
  - AWR
  - REM
  - BAIL
  - …

Model-based Methods

- Uncertainty estimation with the learned model
  - MOReL
  - MOPO
  - COMBO
  - …

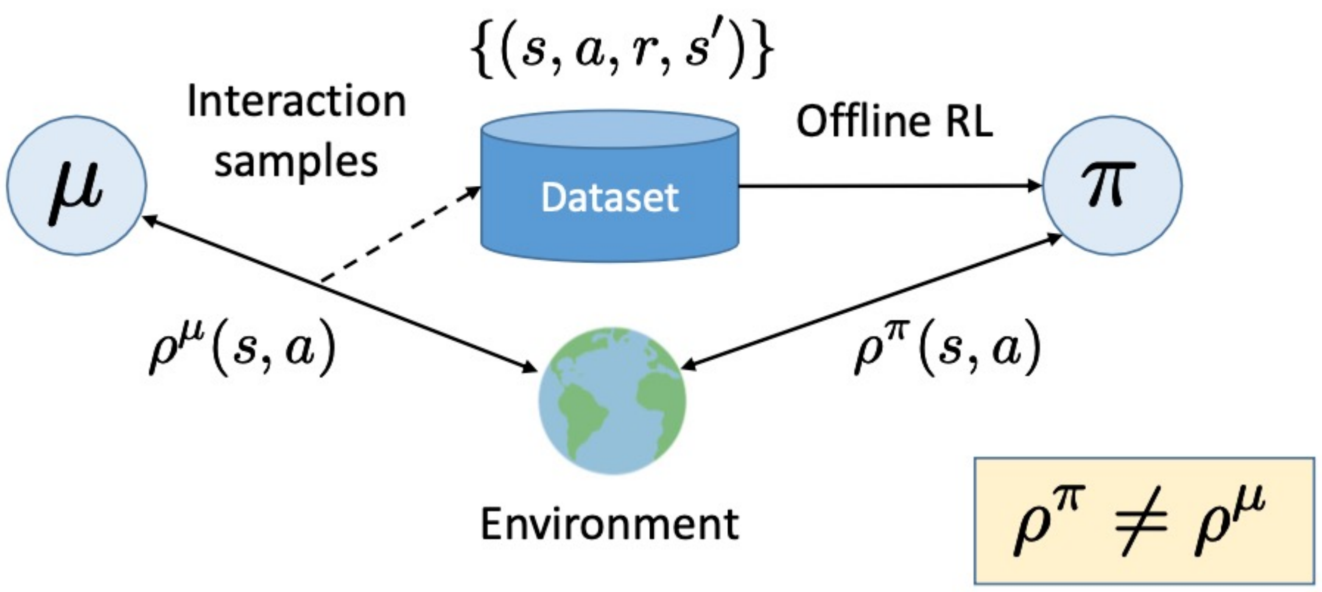
- The most severe problem that offline RL faces is the extrapolation error, i.e., the out-of-distribution problem.
  - What if the agent performs unseen state-action?

Extrapolation Error in Offline RL

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$$

What if a' is an out-of-distribution action?

- Extrapolation error is introduced by the mismatch between the dataset and true state-action visitation of the current policy.



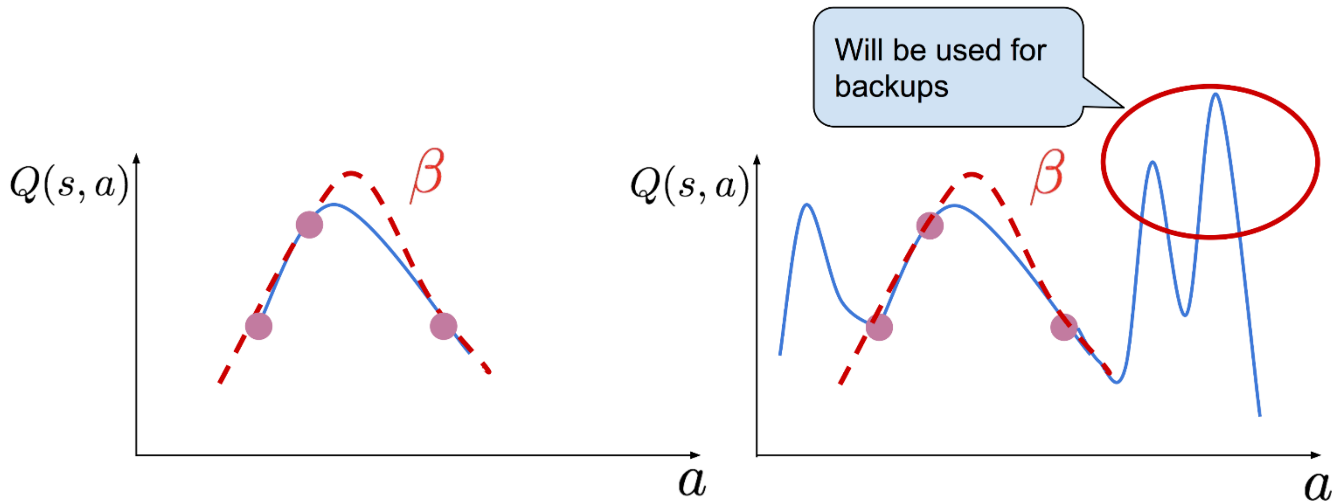
Gulcehre, Caglar, et al. "Rl unplugged: Benchmarks for offline reinforcement learning." *arXiv preprint arXiv:2006.13888* (2020).

Extrapolation Error in Offline RL

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s', a'))$$

What if a’ is an out-of-distribution action?

- Extrapolation error will be propagated via value function backups.



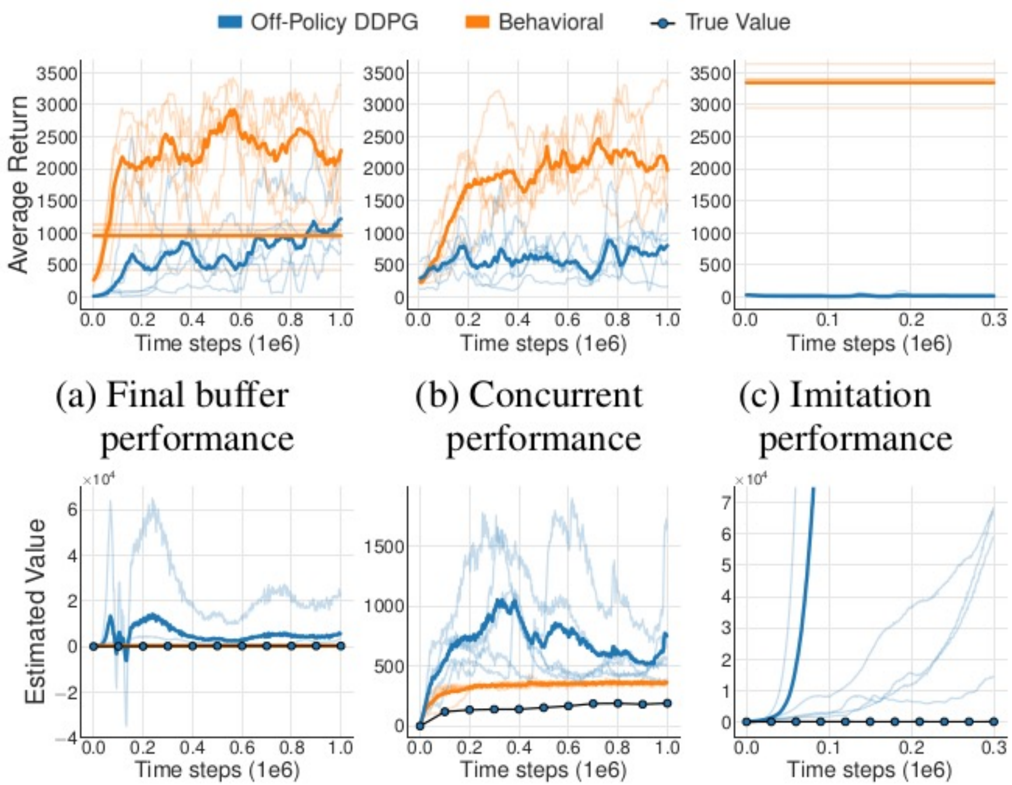
$$\boxed{\rho^{\pi} \neq \rho^{\mu}}$$

https://bair.berkeley.edu/blog/2020/12/07/offline/

BCQ: Batch-Constrained Q-learning

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$$

What if a’ is an out-of-distribution action?



(a) Final buffer performance (b) Concurrent performance (c) Imitation performance

(d) Final buffer value estimate (e) Concurrent value estimate (f) Imitation value estimate

- Extrapolation error

$$\rho^{\pi} \neq \rho^{\mu}$$

- Off-policy methods fail to work in offline setting due to extrapolation error

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. ICML 2019.

BCQ: Batch-Constrained Q-learning

- Tabular setting: only update while the transition is in the batch.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a' \text{ s.t. } (s',a') \in \mathcal{B}} Q(s',a'))$$

Batch-constrained policy

- General setting: use a VAE to give the action that may occur in real environment with high probability.

$$\pi(s) = \underset{a_i+\xi_\phi(s,a_i,\Phi)}{\arg\max} \; Q_\theta(s, a_i + \xi_\phi(s, a_i, \Phi))$$

Perturbation in $[-\Phi, +\Phi]$

$$\text{where } \{a_i \sim G_\omega(s)\}_{i=1}^n$$

Generative model (VAE)

  - The choice of n and Φ creates a trade-off between an imitation learning and reinforcement learning algorithm

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. ICML 2019.

BCQ: Batch-Constrained Q-learning

Input: Batch $\mathcal{B}$, horizon T, target network update rate τ, mini-batch size N, max perturbation Φ, number of sampled actions n, minimum weighting λ.
Initialize Q-networks $Q_{\theta_1}, Q_{\theta_2}$, perturbation network ξ_ϕ, and VAE $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, with random parameters θ_1, θ_2, ϕ, ω, and target networks $Q_{\theta'_1}, Q_{\theta'_2}$, $\xi_{\phi'}$ with $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$.
for $t = 1$ **to** T **do**
  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$
  $\omega \leftarrow \mathrm{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\mathrm{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$
  Sample n actions: $\{a_i \sim G_\omega(s')\}_{i=1}^n$
  Perturb each action: $\{a_i = a_i + \xi_\phi(s', a_i, \Phi)\}_{i=1}^n$
  Set value target y (Eqn. 13)

$$r + \gamma \max_{a_i} \left[\lambda \min_{j=1,2} Q_{\theta'_j}(s', a_i) + (1 - \lambda) \max_{j=1,2} Q_{\theta'_j}(s', a_i) \right]$$

  $\theta \leftarrow \mathrm{argmin}_\theta \sum (y - Q_\theta(s, a))^2$
  $\phi \leftarrow \mathrm{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$
  Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$
  $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
end for

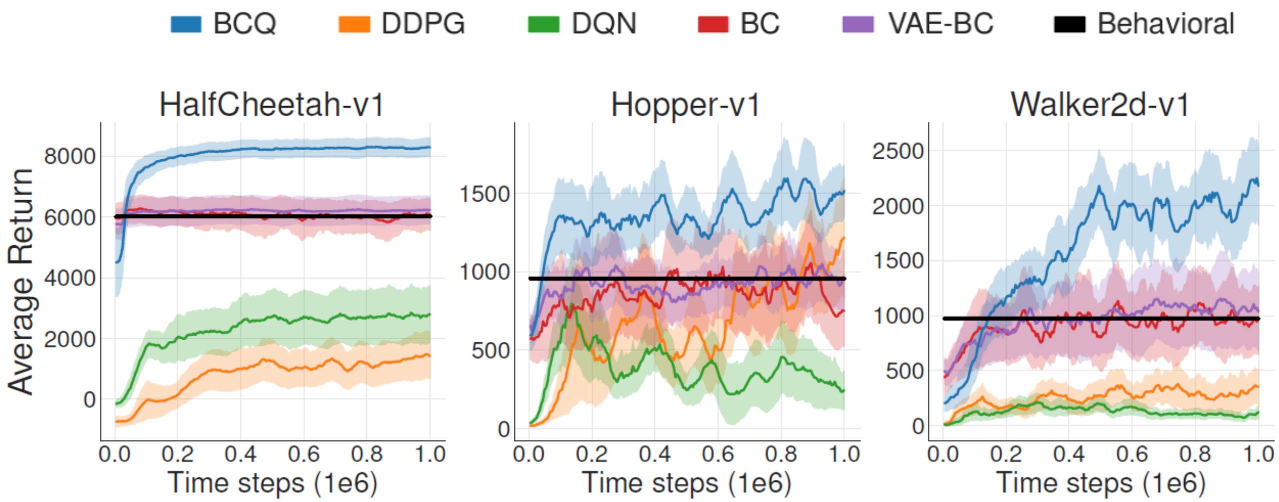
VAE performs imitation learning

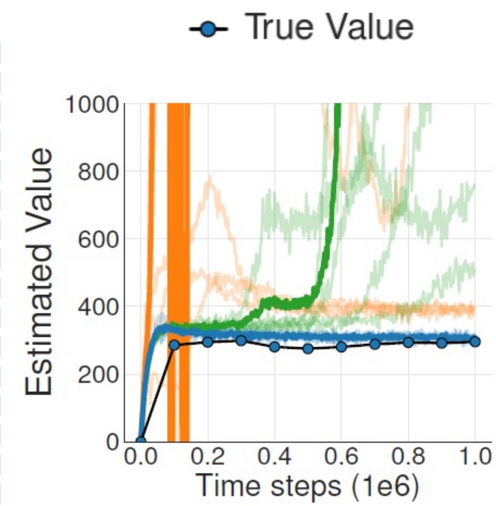
A tradeoff estimation of target

Perturbation function is like an actor

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. ICML 2019.

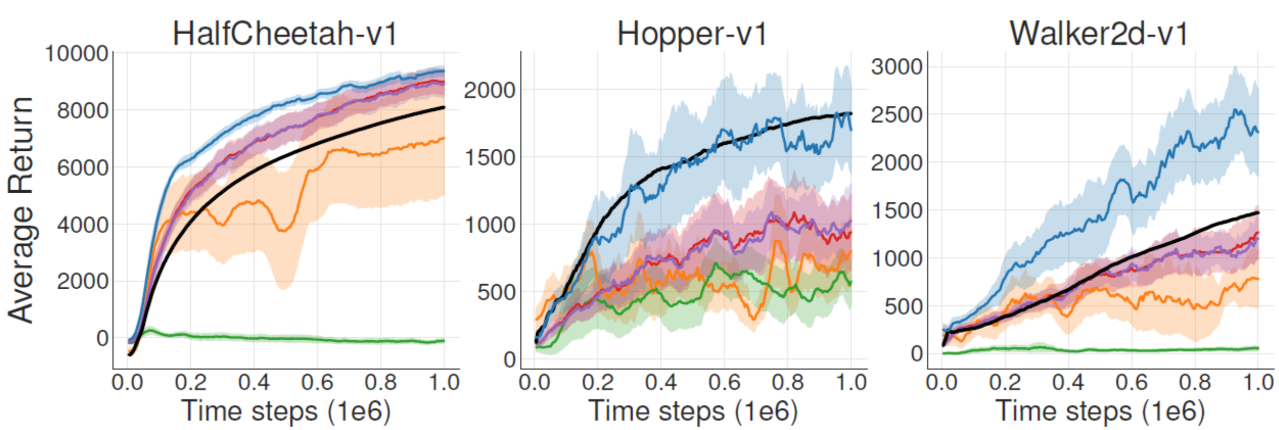
BCQ: Batch-Constrained Q-learning

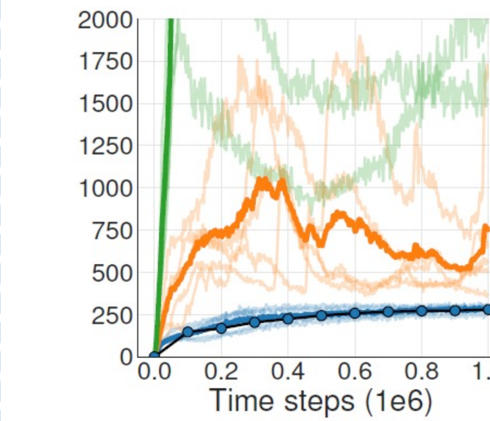




(a) Final buffer performance

Final buffer value estimates





(b) Concurrent performance

Concurrent value estimates

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. ICML 2019.

CQL: Conservative Q-Learning

- Learn a conservative, lower-bound Q function to avoid overestimation.
- Add penalty terms on a standard Bellman error objective.
  - Penalize the overestimated Q on unseen actions (from new policy μ)

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \ \alpha \boxed{\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]} + \frac{1}{2}\,\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}(\mathbf{s},\mathbf{a})\right)^2\right]$$

  - Maximize Q under a behavioral data distribution $\hat{\pi}_\beta$

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \ \alpha \cdot \left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] - \boxed{\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]}\right)$$
$$+\frac{1}{2}\,\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}(\mathbf{s},\mathbf{a})\right)^2\right]$$

  - use max μ to approximate current policy π, and add regularizer

$$\min_{Q}\max_{\mu} \ \alpha\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right)$$
$$+\frac{1}{2}\,\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^{k}(\mathbf{s},\mathbf{a})\right)^2\right] + \mathcal{R}(\mu) \quad (\mathrm{CQL}(\mathcal{R}))$$

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. NeuIPS 2020.
https://sites.google.com/view/cql-offline-rl

CQL: Conservative Q-Learning

$$\min_{Q} \max_{\mu} \alpha \left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right)$$
$$+\frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right] + \mathcal{R}(\mu) \quad (\mathrm{CQL}(\mathcal{R}))$$

- Empirically, a good regularizer is

$$\mathcal{R}(\mu) = -D_{\mathrm{KL}}(\mu, \mathrm{Unif}(\mathbf{a}))$$

- With SAC, π is updated in the same way as SAC.

Algorithm 1 Conservative Q-Learning (both variants)

1: Initialize Q-function, Q_θ, and optionally a policy, π_ϕ.
2: **for** step t in $\{1, \ldots, \mathrm{N}\}$ **do**
3: Train the Q-function using G_Q gradient steps on objective from Equation 4
$\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta \mathrm{CQL}(\mathcal{R})(\theta)$
(Use $\mathcal{B}^*$ for Q-learning, $\mathcal{B}^{\pi_{\phi_t}}$ for actor-critic)
4: (only with actor-critic) Improve policy π_ϕ via G_π gradient steps on ϕ with SAC-style entropy regularization:
$\phi_t := \phi_{t-1} + \eta_\pi \nabla_\phi \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\phi(\cdot|\mathbf{s})}[Q_\theta(\mathbf{s},\mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})]$
5: **end for**

https://sites.google.com/view/cql-offline-rl

CQL: Conservative Q-Learning

Task Name	SAC	BC	BEAR	BRAC-p	BRAC-v	CQL($\mathcal{H}$)
halfcheetah-random	30.5	2.1	25.5	23.5	28.1	**35.4**
hopper-random	**11.3**	9.8	9.5	**11.1**	**12.0**	**10.8**
walker2d-random	4.1	1.6	**6.7**	0.8	0.5	**7.0**
halfcheetah-medium	-4.3	36.1	38.6	**44.0**	**45.5**	**44.4**
walker2d-medium	0.9	6.6	33.2	72.7	**81.3**	79.2
hopper-medium	0.8	29.0	47.6	31.2	32.3	**58.0**
halfcheetah-expert	-1.9	**107.0**	**108.2**	3.8	-1.1	104.8
hopper-expert	0.7	**109.0**	**110.3**	6.6	3.7	**109.9**
walker2d-expert	-0.3	125.7	106.1	-0.2	-0.0	**153.9**
halfcheetah-medium-expert	1.8	35.8	51.7	43.8	45.3	**62.4**
walker2d-medium-expert	1.9	11.3	10.8	-0.3	0.9	**98.7**
hopper-medium-expert	1.6	**111.9**	4.0	1.1	0.8	**111.0**
halfcheetah-random-expert	53.0	1.3	24.6	30.2	2.2	**92.5**
walker2d-random-expert	0.8	0.7	1.9	0.2	2.7	**91.1**
hopper-random-expert	5.6	10.1	10.1	5.8	11.1	**110.5**
halfcheetah-mixed	-2.4	38.4	36.2	**45.6**	**45.9**	**46.2**
hopper-mixed	3.5	11.8	25.3	0.7	0.8	**48.6**
walker2d-mixed	1.9	11.3	10.8	-0.3	0.9	**26.7**

- Offline settings with different experts in Gym environments, CQL performs (almost) the best

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. NeulPS 2020.

Overview of Offline RL Methods

Model-free Methods

- Explicit constraint
  - BCQ
  - BEAR
  - BRAC
  - CQL
- Implicit constraint
  - AWR
  - REM
  - BAIL
  - …

Model-based Methods

- Uncertainty estimation with the learned model
  - MOReL
  - MOPO
  - COMBO
  - …

- The most severe problem that offline RL faces is the extrapolation error, i.e., the out-of-distribution problem.
  - What if the agent performs unseen state-action?

AWR: Advantage-Weighted Regression

- Policy optimization objective

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] = \mathbb{E}_{\mathbf{s} \sim d_\pi(\mathbf{s})} \mathbb{E}_{a \sim \pi(\mathbf{a}|\mathbf{s})} \left[r(\mathbf{s}, \mathbf{a}) \right]$$

Expected improvement

$$\eta(\pi) = J(\pi) - J(\mu)$$ [as derived in TRPO]

$$= \mathbb{E}_{\mathbf{s} \sim d_\pi(\mathbf{s})} \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \left[A^\mu(\mathbf{s}, \mathbf{a}) \right] = \mathbb{E}_{\mathbf{s} \sim d_\pi(\mathbf{s})} \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \left[\mathcal{R}^\mu_{\mathbf{s},\mathbf{a}} - V^\mu(\mathbf{s}) \right]$$

- Based on reward-weighted regression (RWR)

$$\pi_{k+1} = \arg\max_\pi \quad \mathbb{E}_{\mathbf{s} \sim d_{\pi_k}(\mathbf{s})} \mathbb{E}_{\mathbf{a} \sim \pi_k(\mathbf{a}|\mathbf{s})} \left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta} \underset{\uparrow\,\text{return}}{\mathcal{R}_{\mathbf{s},\mathbf{a}}} \right) \right]$$

- Regarded as solving a maximum likelihood problem that fits a new policy to samples collected under the current policy, where the likelihood is weighted by the exponentiated return.

Xue Bin Peng et al . Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177, 2019.

AWR: Advantage-Weighted Regression

- Replace the weight with the advantage function deriving the AWR

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 π_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**

- For offline setting, regard behavior policy as a mixture of policies.

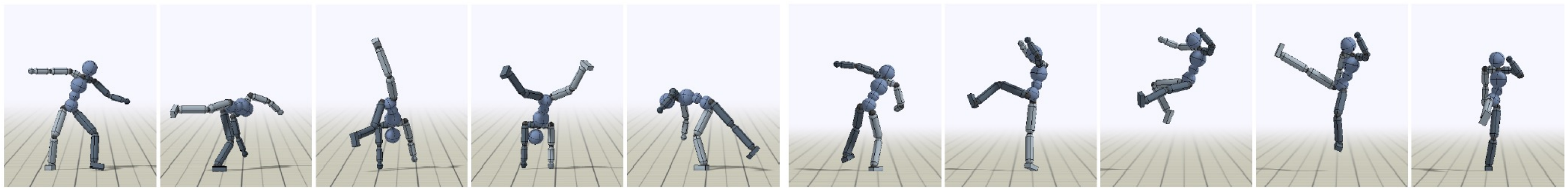
$$\eta(\pi) = J(\pi) - \sum_i w_i J(\pi_i) = \mathbb{E}_{\mathbf{s}\sim d_\pi(\mathbf{s})}\mathbb{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[\sum_i w_i A^{\pi_i}(\mathbf{s},\mathbf{a})\right]$$

Expected improvement

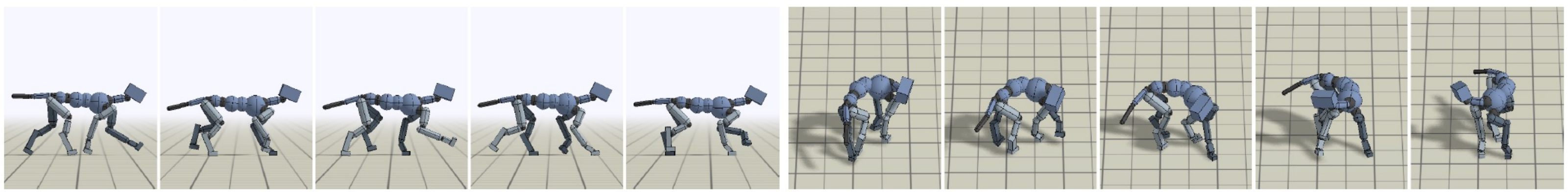
$$\boxed{A^{\pi_i}(\mathbf{s},\mathbf{a}) = \mathcal{R}_{\mathbf{s},\mathbf{a}}^{\pi_i} - V^{\pi_i}(\mathbf{s})}$$

Xue Bin Peng et al . Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177, 2019.

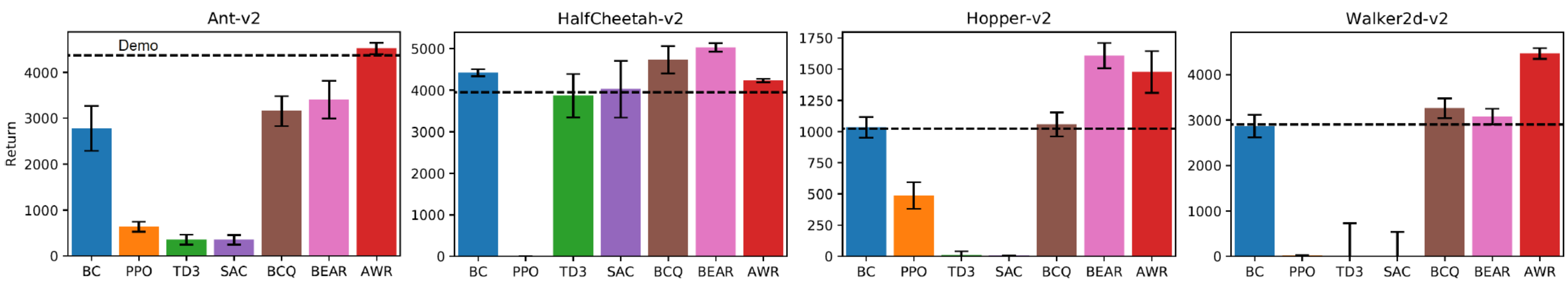
AWR: Advantage-Weighted Regression



(a) Humanoid: Cartwheel (b) Humanoid: Spinkick



(c) Dog: Trot (d) Dog: Turn



- Performance of various algorithms on off-policy learning tasks with static datasets. AWR is able to learn policies that are comparable or better than the original demo policies.

Xue Bin Peng et al . Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177, 2019.

BAIL: Best-Action Imitation Learning

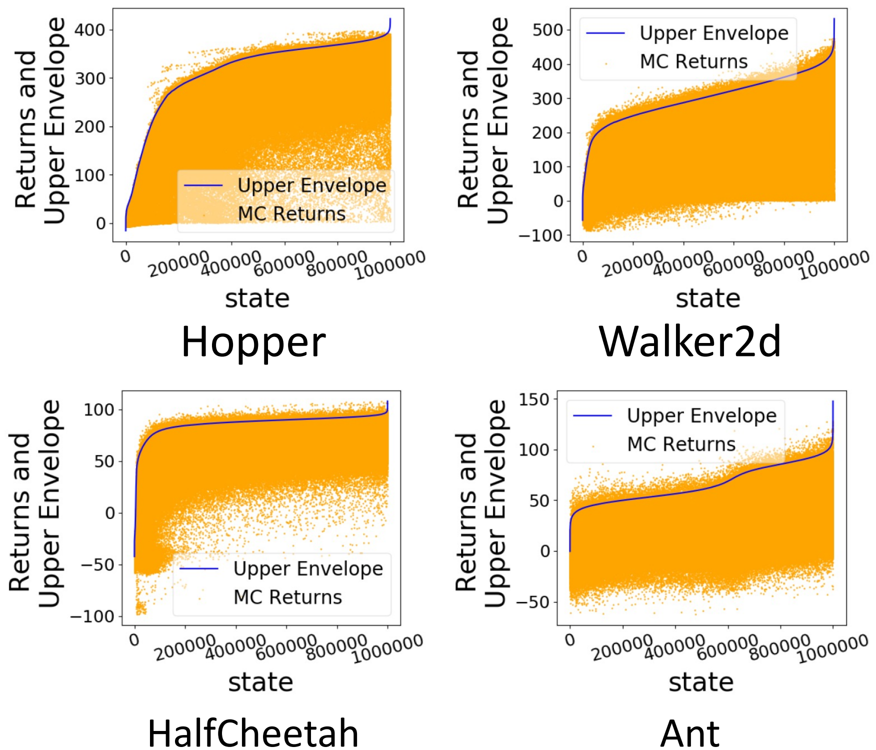
- BAIL does not suffer from the extrapolation error, since it does not maximize over the actions space.
- Step 1: learn an upper envelope of state by solving a constrained optimization problem:

$$\min_{\phi} \sum_{i=1}^{m} [V_\phi(s_i) - G_i]^2 + \lambda \|w\|^2$$

$$s.t. \quad V_\phi(s_i) \geq G_i, \quad i = 1, 2, \ldots, m$$

or its unconstrained penalty-loss version

$$L^K(\phi) = \sum_{i=1}^{m} (V_\phi(s_i) - G_i)^2 \{\mathbb{1}_{(V_\phi(s_i) \geq G_i)} + K \cdot \mathbb{1}_{(V_\phi(s_i) < G_i)}\} + \lambda \|w\|^2$$



X. Chen et al. BAIL: Best-action imitation learning for batch deep reinforcement learning. NeuIPS 2020.

BAIL: Best-Action Imitation Learning

- Step 2: Select actions satisfying $G_i > x\,V(s_i)$ to perform simple imitation learning (BC). x is set such that 25% samples are selected.

Algorithm 1 Static BAIL

Initialize upper envelope parameters ϕ, ϕ', policy parameters θ. Obtain batch data $\mathcal{B}$. Randomly split data into training set $\mathcal{B}_t$ and validation set $\mathcal{B}_v$ for the upper envelope.
Compute return G_i for each data point i in $\mathcal{B}$.
Obtain upper envelope by minimizing the loss $L^K(\phi)$:
for $j = 1, \ldots, J$ **do**
  Sample a mini-batch B from $\mathcal{B}$.
  Update ϕ using the gradient: $\nabla_\phi \sum_{i \in B} (V_\phi(s_i) - G_i)^2 \{\mathbb{1}_{(V_\phi(s_i) > G_i)} + K \mathbb{1}_{(V_\phi(s_i) < G_i)}\} + \lambda \|\phi\|^2$
  if time to do validation for the upper envelope **then**
    Compute validation loss on B_v
    Update ϕ and ϕ' according to the validation loss
  end if
end for
Select data point i if $G_i > xV_\phi(s_i)$, where x is such that $p\%$ of data in $\mathcal{B}$ are selected. Let $\mathcal{U}$ be the set of selected data points.
for $l = 1, \ldots, L$ **do**
  Sample a mini-batch U of data from $\mathcal{U}$.
  Update θ using the gradient: $\nabla_\theta \sum_{i \in U} (\pi_\theta(s_i) - a_i)^2$
end for

X. Chen et al. BAIL: Best-action imitation learning for batch deep reinforcement learning. NeuIPS 2020.

BAIL: Best-Action Imitation Learning

Environment	BAIL	BCQ	BEAR	BC	MARWIL
$\sigma = 0.1$ Hopper B1	**2173 ± 291**	1219 ± 114	505 ± 285	626 ± 112	827 ± 220
$\sigma = 0.1$ Hopper B2	**2078 ± 180**	1178 ± 87	985 ± 3	579 ± 141	620 ± 336
$\sigma = 0.1$ Walker B1	**1125 ± 113**	576 ± 309	610 ± 212	514 ± 17	436 ± 24
$\sigma = 0.1$ Walker B2	**3141 ± 300**	2338 ± 388	2707 ± 425	1741 ± 239	1810 ± 200
$\sigma = 0.1$ HC B1	**5746 ± 29**	**5883 ± 43**	0 ± 0	**5546 ± 29**	**5573 ± 35**
$\sigma = 0.1$ HC B2	**7212 ± 43**	**7562 ± 31**	0 ± 0	6765 ± 108	**6828 ± 111**
$\sigma = 0.5$ Hopper B1	**2054 ± 158**	1145 ± 300	203 ± 42	919 ± 52	946 ± 103
$\sigma = 0.5$ Hopper B2	**2623 ± 282**	1823 ± 555	241 ± 239	694 ± 64	818 ± 112
$\sigma = 0.5$ Walker B1	**2522 ± 51**	1552 ± 455	1248 ± 181	2178 ± 178	2111 ± 52
$\sigma = 0.5$ Walker B2	**3115 ± 133**	2785 ± 123	2302 ± 630	2483 ± 94	2364 ± 228
$\sigma = 0.5$ HC B1	1055 ± 9	**1222 ± 38**	924 ± 579	570 ± 35	512 ± 43
$\sigma = 0.5$ HC B2	**7173 ± 120**	5807 ± 249	-114 ± 140	**6545 ± 171**	**6668 ± 93**
SAC Hopper B1	**3296 ± 105**	2681 ± 438	1000 ± 110	2853 ± 318	2897 ± 227
SAC Hopper B2	1831 ± 915	**2134 ± 917**	1139 ± 317	**2240 ± 367**	**2063 ± 168**
SAC Walker B1	**2455 ± 211**	**2408 ± 84**	-3 ± 5	1674 ± 277	1484 ± 140
SAC Walker B2	**4767 ± 130**	3794 ± 398	325 ± 75	2599 ± 145	2651 ± 268
SAC HC B1	**10143 ± 77**	8607 ± 473	7392 ± 257	8874 ± 221	9105 ± 90
SAC HC B2	**10772 ± 59**	**10106 ± 134**	7217 ± 273	9523 ± 164	9488 ± 136
SAC Ant B1	**4284 ± 64**	**4042 ± 113**	3452 ± 128	**3986 ± 112**	**4033 ± 130**
SAC Ant B2	**4946 ± 148**	**4640 ± 76**	3712 ± 236	**4618 ± 111**	**4589 ± 130**
SAC Humanoid B1	**3852 ± 430**	1411 ± 250	0 ± 0	543 ± 378	589 ± 121
SAC Humanoid B2	**3565 ± 153**	1221 ± 207	0 ± 0	1216 ± 826	1033 ± 257

X. Chen et al. BAIL: Best-action imitation learning for batch deep reinforcement learning. NeuIPS 2020.

Overview of Offline RL Methods

Model-free Methods

- Explicit constraint
  - BCQ
  - BEAR
  - BRAC
  - CQL
- Implicit constraint
  - AWR
  - REM
  - BAIL
  - …

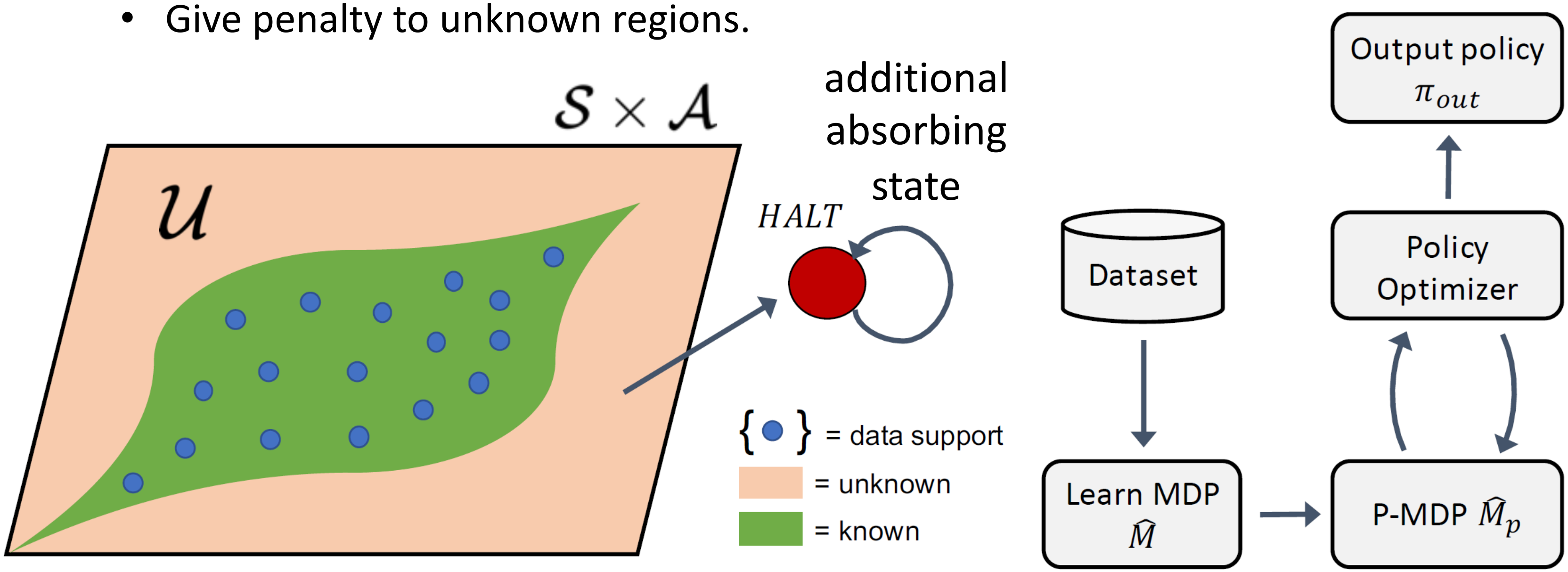
Model-based Methods

- Uncertainty estimation with the learned model
  - MOReL
  - MOPO
  - COMBO
  - …

- The most severe problem that offline RL faces is the extrapolation error, i.e., the out-of-distribution problem.
  - What if the agent performs unseen state-action?

MOReL: Model-Based Offline RL

- Model-based methods allow (pseudo) exploration.
- Construct a Pessimistic MDP (P-MDP) using the offline dataset.
    - Require a careful use of the model in regions outside of support, to ensure that policies do not visit states where the model is inaccurate.
    - Partition the (s, a) space into known and unknown regions.
    - Give penalty to unknown regions.



Kidambi R, Rajeswaran A. MOReL: Model-Based Offline Reinforcement Learning. NeurIPS 2020.

MOReL: Model-Based Offline RL

- Unknown state-action detector (USAD)

  (α-USAD) Given a state-action pair (s, a), define an unknown state action detector as

Total variation distance

$$U^{\alpha}(s,a) = \begin{cases} \textit{FALSE} \;\; (\textit{i.e. Known}) & \text{if } \; D_{TV}\left(\hat{P}(\cdot|s,a), P(\cdot|s,a)\right) \leq \alpha \;\; \textit{can be guaranteed} \\ \textit{TRUE} \;\; (\textit{i.e. Unknown}) & \textit{otherwise} \end{cases}$$

HALT is an additional absorbing state

- Pessimistic MDP $\hat{\mathcal{M}}_p := \{S \cup \textit{HALT}, A, r_p, \hat{P}_p, \hat{\rho}_0, \gamma\}$

$$\hat{P}_p(s'|s,a) = \begin{cases} \delta(s' = \text{HALT}) & \text{if } U^{\alpha}(s,a) = \text{TRUE} \;\text{ or }\; s = \text{HALT} \\ \hat{P}(s'|s,a) & \textit{otherwise} \end{cases}$$

$$r_p(s,a) = \begin{cases} -\kappa & \text{if } s = \text{HALT} \\ r(s,a) & \textit{otherwise} \end{cases}$$

Kidambi R, Rajeswaran A. MOReL: Model-Based Offline Reinforcement Learning. NeurIPS 2020.

MOReL: Model-Based Offline RL

- Planning: the final step in MOReL is to perform planning in the P-MDP with various MBRL methods like MPC, MBPO etc.
- Give penalty to unknown regions by learning optimal policy in P-MDP.

Algorithm 1 `MOReL`: Model Based Offline Reinforcement Learning

1: **Require** Dataset $\mathcal{D}$
2: Learn approximate dynamics model $\hat{P} : S \times A \rightarrow S$ using $\mathcal{D}$.
3: Construct α-USAD, $U^{\alpha} : S \times A \rightarrow \{\text{TRUE}, \text{FALSE}\}$ using $\mathcal{D}$ (see Definition 1).
4: Construct the *pessimistic* MDP $\hat{\mathcal{M}}_p = \{S \cup \text{HALT}, A, r_p, \hat{P}_p, \hat{\rho}_0, \gamma\}$ (see Definition 2).
5: (OPTIONAL) Use a behavior cloning approach to estimate the behavior policy $\hat{\pi}_b$.
6: $\pi_{\text{out}} \leftarrow \text{PLANNER}(\hat{\mathcal{M}}_p, \pi_{\text{init}} = \hat{\pi}_b)$
7: **Return** π_{out}.

Kidambi R, Rajeswaran A. MOReL: Model-Based Offline Reinforcement Learning. NeurIPS 2020.

MOReL: Model-Based Offline RL

Environment: Ant-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	1921	2100	<u>2839</u>	2839	**3663±247**
Eps-1	1864	1897	<u>2672</u>	2672	**3305±413**
Eps-3	1504	2008	<u>2602</u>	2602	**3008±231**
Gauss-1	1731	2054	<u>2667</u>	2667	**3329±270**
Gauss-3	1887	2018	2640	2661	**3693±33**

Environment: Hopper-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	1543	0	2291	2774	**3642±54**
Eps-1	1652	1620	2282	2360	**3724±46**
Eps-3	1632	2213	1892	2892	**3535±91**
Gauss-1	1599	1825	<u>2255</u>	2255	**3653±52**
Gauss-3	1590	1720	1458	2097	**3648±148**

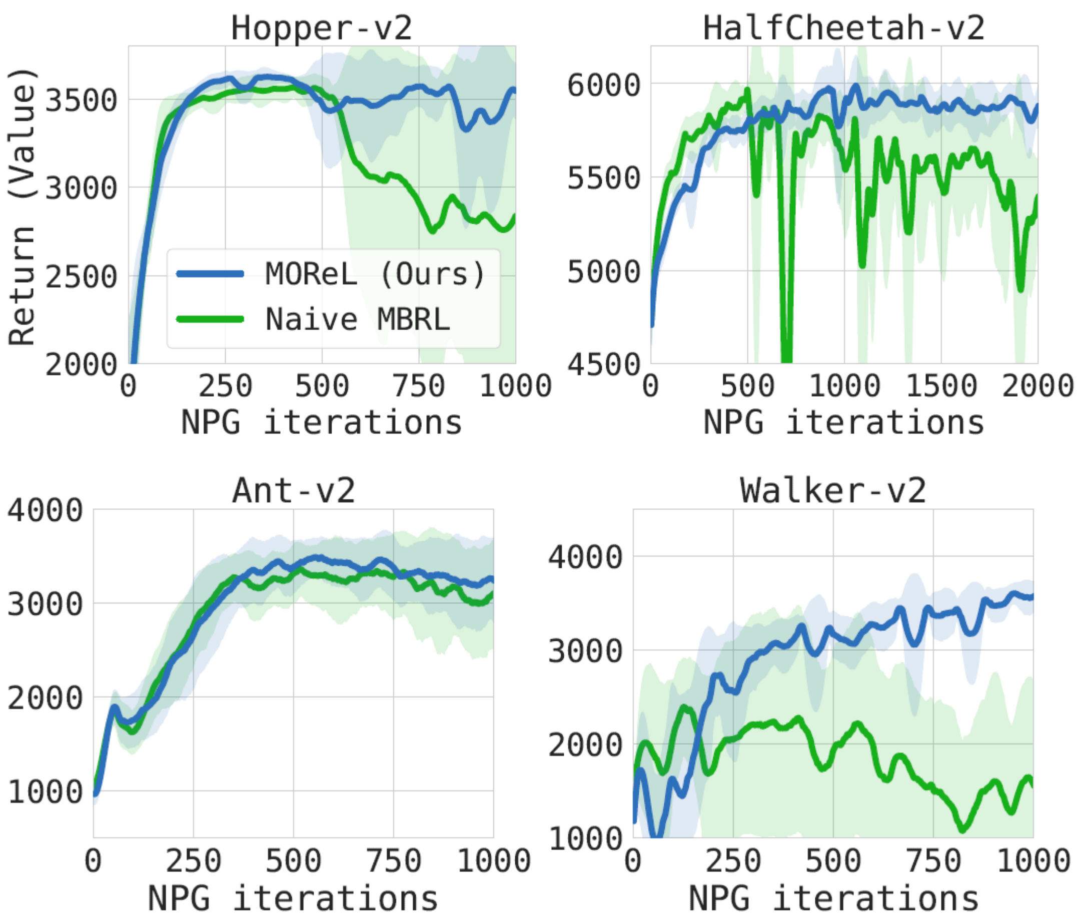
Environment: HalfCheetah-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	5064	5325	6207	**6209**	**6028±192**
Eps-1	5693	5435	<u>**6307**</u>	**6307**	5861±192
Eps-3	5588	5149	6263	**6359**	5869±139
Gauss-1	5614	5394	<u>**6323**</u>	**6323**	6026±74
Gauss-3	5837	5329	<u>**6400**</u>	**6400**	5892±128

Environment: Walker-v2					
Algorithm	BCQ [15]	BEAR [16]	BRAC [18]	Best Baseline	MOReL (Ours)
Pure	2095	2646	2694	2907	**3709±159**
Eps-1	1921	2695	3241	**3490**	2899±588
Eps-3	1953	2608	<u>**3255**</u>	**3255**	**3186±92**
Gauss-1	2094	2539	2893	3193	**4027±314**
Gauss-3	1734	2194	<u>**3368**</u>	**3368**	**2828±589**

- MORel achieves SOTA performance in 12 out of 20 tasks
- The error bar is the standard deviation of five random seeds

Kidambi R, Rajeswaran A. MOReL: Model-Based Offline Reinforcement Learning. NeurIPS 2020.

MOReL: Model-Based Offline RL



- The naïve MBRL approach first learns a dynamics model using the offline data, then runs MBRL without any safeguards against model inaccuracy
- The naive MBRL algorithm is highly unstable while MOReL leads to stable and near-monotonic learning

Kidambi R, Rajeswaran A. MOReL: Model-Based Offline Reinforcement Learning. NeurIPS 2020.

MOPO: Model-based Offline Policy Optimization

- Idea: given the model will not be globally accurate
  - need an uncertainty-penalized MDP (for inaccurate state transitions)
  - discourage policy from visiting regions where model uncertainty is high.
- The gap of expected return under $\widehat{M}$ with $(\hat{T}, r)$ and M with (T, r)

Let $G^{\pi}_{\widehat{M}}(s,a) := \mathop{\mathbb{E}}_{s' \sim \widehat{T}(s,a)}[V^{\pi}_{M}(s')] - \mathop{\mathbb{E}}_{s' \sim T(s,a)}[V^{\pi}_{M}(s')]$

Improper expectation over OM

$$\eta_{\widehat{M}}(\pi) - \eta_M(\pi) = \gamma \mathop{\bar{\mathbb{E}}}_{(s,a)\sim\rho^{\pi}_{\widehat{T}}} \left[G^{\pi}_{\widehat{M}}(s,a) \right]$$

$$\eta_M(\pi) = \mathop{\bar{\mathbb{E}}}_{(s,a)\sim\rho^{\pi}_{\widehat{T}}} \left[r(s,a) - \gamma G^{\pi}_{\widehat{M}}(s,a) \right] \geq \mathop{\bar{\mathbb{E}}}_{(s,a)\sim\rho^{\pi}_{\widehat{T}}} \left[r(s,a) - \gamma |G^{\pi}_{\widehat{M}}(s,a)| \right]$$

- When the reward function is bounded $\forall(s,a), |r(s,a)| \leq r_{\max}$:

$$|G^{\pi}_{\widehat{M}}(s,a)| \leq \frac{r_{\max}}{1-\gamma} D_{\mathrm{TV}}(\widehat{T}(s,a), T(s,a))$$

build a lower bound of the true value

Yu T, Thomas G, Yu L, et al. MOPO: Model-based offline policy optimization. NeuIPS 2020.

MOPO: Model-based Offline Policy Optimization

- Optimize π in an uncertainty-penalized MDP $\widetilde{M} = (S, A, \widehat{T}, \tilde{r}, \gamma)$, where

$$\tilde{r}(s,a) := r(s,a) - \lambda u(s,a)$$

- If $u(s,a)$ is an admissible error estimator, i.e., upper bounds the error of $\widehat{T}(s'|s,a)$, then for a particular choice of λ:

$$\eta_M(\pi) \geq \bar{\mathbb{E}}_{(s,a)\sim\rho^\pi_{\widehat{T}}}\left[r(s,a) - \gamma\left|G^\pi_{\widehat{M}}(s,a)\right|\right] \geq \bar{\mathbb{E}}_{(s,a)\sim\rho^\pi_{\widehat{T}}}\left[r(s,a) - \lambda u(s,a)\right]$$
$$= \bar{\mathbb{E}}_{(s,a)\sim\rho^\pi_{\widehat{T}}}\left[\tilde{r}(s,a)\right] = \eta_{\widetilde{M}}(\pi)$$

- Practical algorithm uses ensemble of models:

$$\{\widehat{T}^i_{\theta,\phi} = \mathcal{N}(\mu^i_\theta, \Sigma^i_\phi)\}_{i=1}^N \qquad \widehat{T}_{\theta,\phi}(s_{t+1}, r|s_t, a_t) = \mathcal{N}(\mu_\theta(s_t, a_t), \Sigma_\phi(s_t, a_t))$$

- Penalize reward function using heuristic uncertainty estimate:

$$u(s,a) = \max_{i=1}^N \|\Sigma^i_\phi(s,a)\|_{\mathrm{F}}$$

the maximum standard deviation of the learned models in the ensemble

Yu T, Thomas G, Yu L, et al. MOPO: Model-based offline policy optimization. NeuIPS 2020.

MOPO: Model-based Offline Policy Optimization

Algorithm 1 Framework for Model-based Offline Policy Optimization (MOPO) with Reward Penalty

Require: Dynamics model $\widehat{T}$ with admissible error estimator $u(s,a)$; constant λ.

1: Define $\tilde{r}(s,a) = r(s,a) - \lambda u(s,a)$. Let $\widetilde{M}$ be the MDP with dynamics $\widehat{T}$ and reward $\tilde{r}$.

2: Run any RL algorithm on $\widetilde{M}$ until convergence to obtain $\hat{\pi} = \text{argmax}_\pi \eta_{\widetilde{M}}(\pi)$

Algorithm 2 MOPO instantiation with regularized probabilistic dynamics and ensemble uncertainty

Require: reward penalty coefficient λ rollout horizon h, rollout batch size b.

1: Train on batch data $\mathcal{D}_{\text{env}}$ an ensemble of N probabilistic dynamics $\{\widehat{T}^i(s', r \mid s, a) = \mathcal{N}(\mu^i(s,a), \Sigma^i(s,a))\}_{i=1}^N$ with Lipschitz-regularized $\mu^i(s,a)$.

2: Initialize policy π and empty replay buffer $\mathcal{D}_{\text{model}} \leftarrow \varnothing$.

3: **for** epoch $1, 2, \ldots$ **do** ▷ This for-loop is essentially one outer iteration of MBPO

4:     **for** $1, 2, \ldots, b$ (in parallel) **do**

5:         Sample state s_1 from $\mathcal{D}_{\text{env}}$ for the initialization of the rollout.

6:         **for** $j = 1, 2, \ldots, h$ **do**

7:             Sample an action $a_j \sim \pi(s_j)$.

8:             Randomly pick dynamics $\widehat{T}$ from $\{\widehat{T}^i\}_{i=1}^N$ and sample $s_{j+1}, r_j \sim \widehat{T}(s_j, a_j)$.

9:             Compute $\boxed{\tilde{r}_j = r_j - \lambda \max_{i=1}^N \|\Sigma^i(s_j, a_j)\|_{\text{F}}.}$

10:            Add sample $(s_j, a_j, \tilde{r}_j, s_{j+1})$ to $\mathcal{D}_{\text{model}}$.

11:     Drawing samples from $\mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}$, use SAC to update π.

Yu T, Thomas G, Yu L, et al. MOPO: Model-based offline policy optimization. NeuIPS 2020.

MOPO: Model-based Offline Policy Optimization

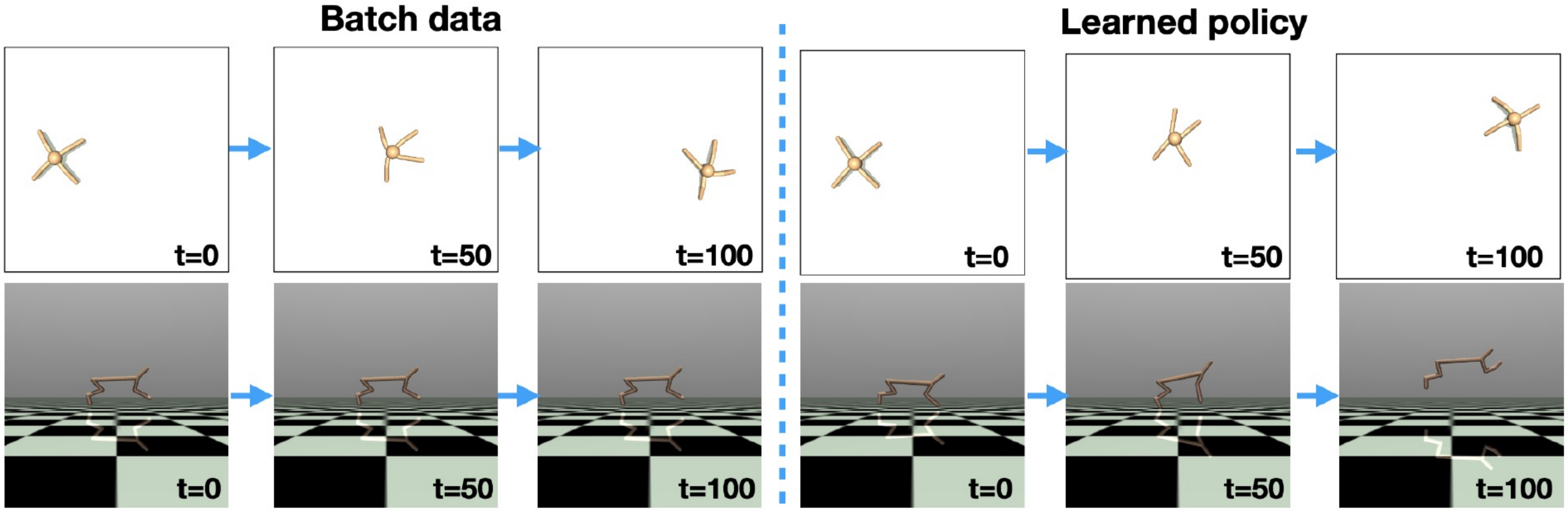
- Overall performance on D4RL

Dataset type	Environment	BC	MOPO (ours)	MBPO	SAC	BEAR	BRAC-v
random	halfcheetah	2.1	**35.4** $\pm$ 2.5	30.7 $\pm$ 3.9	30.5	25.5	28.1
random	hopper	1.6	11.7 $\pm$ 0.4	4.5 $\pm$ 6.0	11.3	9.5	**12.0**
random	walker2d	9.8	**13.6** $\pm$ 2.6	8.6 $\pm$ 8.1	4.1	6.7	0.5
medium	halfcheetah	36.1	42.3 $\pm$ 1.6	28.3 $\pm$ 22.7	-4.3	38.6	**45.5**
medium	hopper	29.0	28.0 $\pm$ 12.4	4.9 $\pm$ 3.3	0.8	**47.6**	32.3
medium	walker2d	6.6	17.8 $\pm$ 19.3	12.7 $\pm$ 7.6	0.9	33.2	**81.3**
mixed	halfcheetah	38.4	**53.1** $\pm$ 2.0	47.3 $\pm$ 12.6	-2.4	36.2	45.9
mixed	hopper	11.8	**67.5** $\pm$ 24.7	49.8 $\pm$ 30.4	1.9	10.8	0.9
mixed	walker2d	11.3	**39.0** $\pm$ 9.6	22.2 $\pm$ 12.7	3.5	25.3	0.8
med-expert	halfcheetah	35.8	**63.3** $\pm$38.0	9.7 $\pm$ 9.5	1.8	51.7	45.3
med-expert	hopper	111.9	23.7 $\pm$ 6.0	**56.0** $\pm$ 34.5	1.6	4.0	0.8
med-expert	walker2d	6.4	44.6 $\pm$ 12.9	7.6 $\pm$ 3.7	-0.1	26.0	**66.6**

Yu T, Thomas G, Yu L, et al. MOPO: Model-based offline policy optimization. NeuIPS 2020.

MOPO: Model-based Offline Policy Optimization

- Performance on tasks requiring out-of-distribution generalization



- Ant-angle: the ant is rewarded for running forward in a 30 degree angle and the training offline dataset contains data of the ant running forward directly
- Halfcheetah-jump: the agent is asked to run while jumping as high as possible given a training offline dataset of halfcheetah running

Environment	Batch Mean	Batch Max	MOPO (ours)	MBPO	SAC	BEAR	BRAC-p	BRAC-v
halfcheetah-jump	-1022.6	1808.6	**4016.6±144**	2971.4±1262	-3588.2±1436	16.8±60	1069.9±232	871±41
ant-angle	866.7	2311.9	**2530.9±137**	13.6±66	-966.4±778	1658.2±16	1806.7±265	2333±139

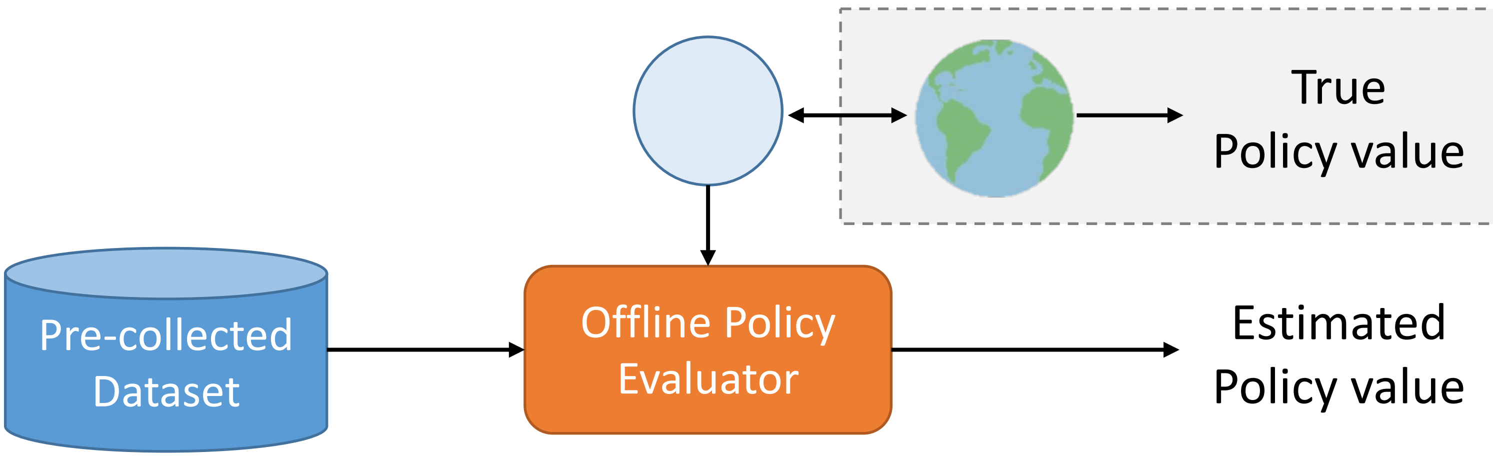
Yu T, Thomas G, Yu L, et al. MOPO: Model-based offline policy optimization. NeuIPS 2020.

Content

- Overview of offline RL
- Offline RL training methods
  - Imitation learning
  - Model-free methods
  - Model-based methods
- Offline policy evaluation
- Offline RL benchmarks

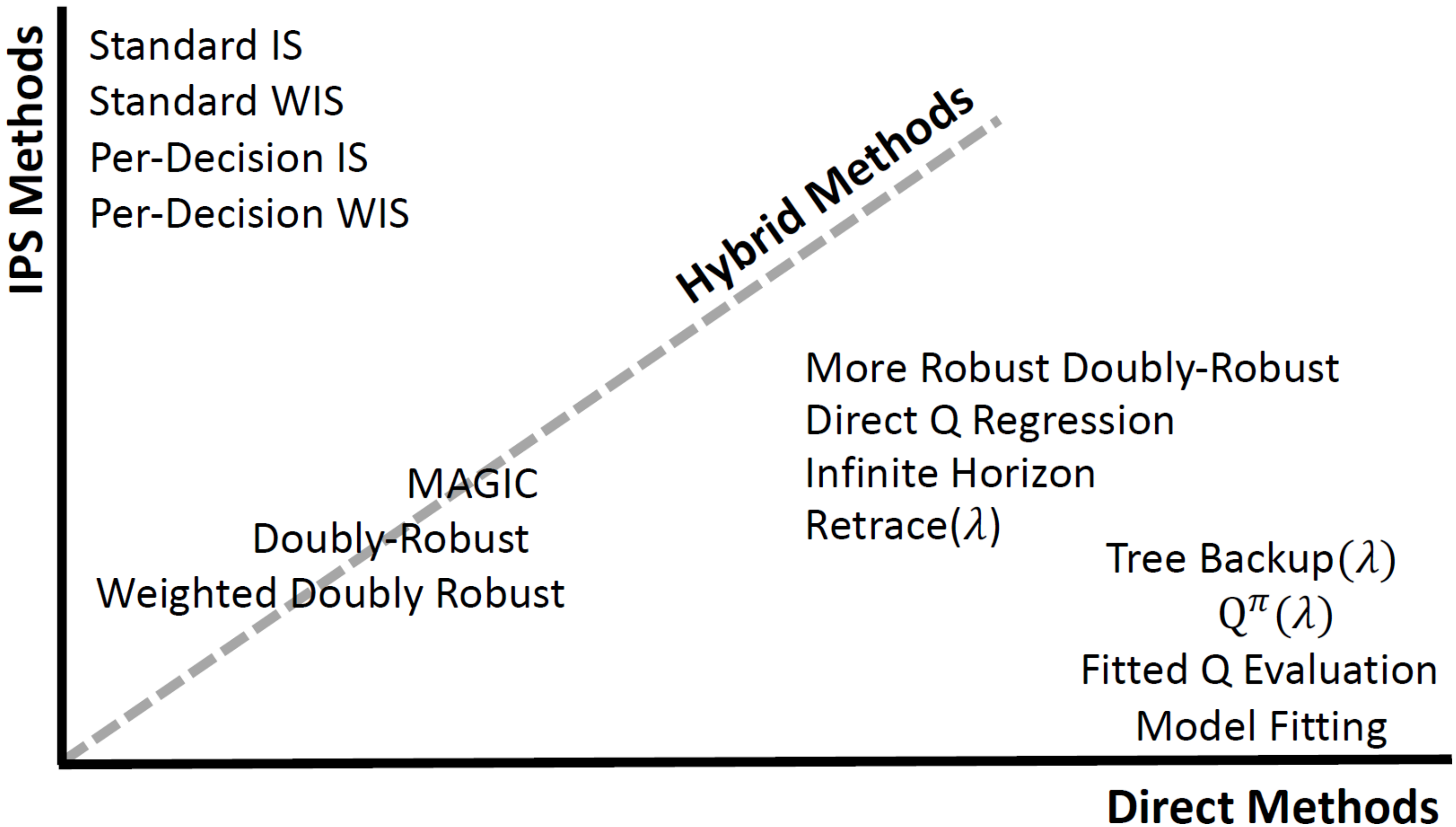
OPE: Offline Policy Evaluation

- Offline (or off-policy) policy evaluation (OPE) concerns estimating the value of a target policy using only pre-collected data from other policies



- OPE serves for policy evaluation, policy ranking or selection for the final online deployment

Overview of OPE Methods



Voloshin, Cameron, et al. "Empirical study of off-policy policy evaluation for reinforcement learning." arXiv:1911.06854. 2019.

FQE: Fitted Q Evaluation

Algorithm 3 Fitted Off-Policy Evaluation with Function Approximation: $\texttt{FQE}(\pi, c)$

Input: Dataset $\mathrm{D} = \{x_i, a_i, x'_i, c_i\}_{i=1}^{n} \sim \pi_{\mathrm{D}}$. Function class F. Policy π to be evaluated

1: Initialize $Q_0 \in \mathrm{F}$ randomly
2: **for** $k = 1, 2, \ldots, K$ **do**
3: Compute target $y_i = c_i + \gamma Q_{k-1}(x'_i, \pi(x'_i)) \quad \forall i$
4: Build training set $\widetilde{\mathrm{D}}_k = \{(x_i, a_i), y_i\}_{i=1}^{n}$
5: Solve a supervised learning problem:
$$Q_k = \underset{f \in \mathrm{F}}{\arg\min}\ \frac{1}{n} \sum_{i=1}^{n} (f(x_i, a_i) - y_i)^2$$

Output: $\widehat{C}^{\pi}(x) = Q_K(x, \pi(x)) \quad \forall x$

- FQE takes a policy as input and performs policy evaluation on the fixed dataset by Bellman backup
- After learning the Q function of the policy, the performance is measured by the mean Q values on the states sampled from the dataset and actions by the policy.

Le, Hoang, Cameron Voloshin, and Yisong Yue. "Batch policy learning under constraints." ICML 2019.

Content

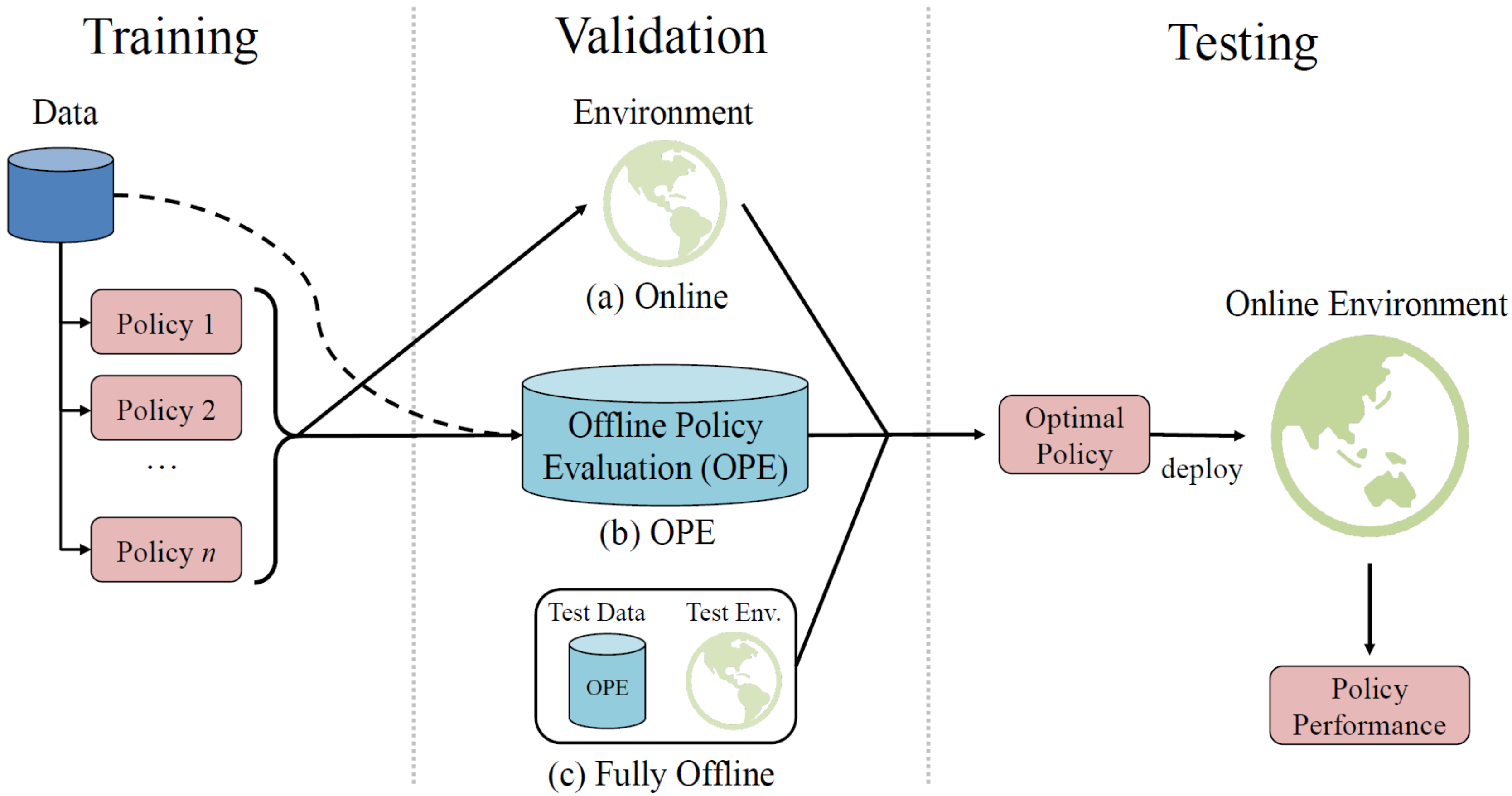
- Overview of offline RL
- Offline RL training methods
  - Imitation learning
  - Model-free methods
  - Model-based methods
- Offline policy evaluation
- Offline RL benchmarks

Offline RL Benchmarks

- D4RL: Datasets for Deep Data-Driven RL
  - UC Berkeley & Google Brain
  - https://sites.google.com/view/d4rl/
- RL Unplugged: Benchmarks for Offline RL
  - Google DeepMind & Brain
  - https://github.com/deepmind/deepmind-research/tree/master/rl_unplugged
- NeoRL: A Near Real-World Benchmark for Offline RL
  - Nanjing University, Polixir and SJTU
  - http://polixir.ai/research/neorl

Discuss more on this benchmark here

Pipeline of Training & Deploying Offline RL



Rongjun Qin et al. NeoRL: A Near Real-World Benchmark for Offline Reinforcement Learning. NeurIPS 2022.

Realistic Challenges for Offline RL

- NeoRL considers the following reality gaps
  1. Conservative data
     - The behavior policy is not explorative (not diverse)
  2. Limited available data
     - Only a small batch of data is available
  3. Highly stochastic environments
     - Aleatoric uncertainty or non-stationary nature of the environment
  4. Offline evaluation before deployment
     - Due to the limited data, predicting the approximate accumulated rewards can be challenging
- In addition to Mujoco, three realistic environments are used
  - Industrial benchmark, FinRL, CityLearn

Rongjun Qin et al. NeoRL: A Near Real-World Benchmark for Offline Reinforcement Learning. NeurIPS 2022.

Experiments and Findings of NeoRL

Task	Expert Policy	Det. Policy	Behavior Policy	Random	BC	CQL	PLAS	BCQ	MOPO	MB-PPO w/ KL	MB-PPO w/o KL
IB-L-99	-180240	-344311	-344311	-317624	-333400	-298161	-341327*	-410860*	-379405*	**-230455**	-278524
IB-L-999	-180240	-344311	-344311	-317624	-339959	-341099*	-338732	-410531*	-372254*	**-220832**	-329973
IB-L-9999	-180240	-344311	-344311	-317624	-340716	-323374	-322796	-407141*	-409110*	-339899	**-310140**
IB-M-99	-180240	-283121	-283121	-317624	-281225	-277511	-410918*	-410196*	-345350*	-234696	**-217471**
IB-M-999	-180240	-283121	-283121	-317624	-382304	-279299	-283242	-862628*	-381155	-283838	**-223842**
IB-M-9999	-180240	-283121	-283121	-317624	-277010	-282285*	-410751*	-410820*	-406456*	**-218091**	-235499
IB-H-99	-180240	-220156	-220156	-317624	-217240	-223178*	-410869*	-406571*	-410431*	-272607*	**-213269**
IB-H-999	-180240	-220156	-220156	-317624	-220528	-213588	-411404*	-410517*	-314221*	**-207958**	-257114*
IB-H-9999	-180240	-220156	-220156	-317624	**-220370**	-280470*	-222682*	-410618*	-410726*	-261822*	-224130*
FinRL-L-99	631	150	152	206	136	**487**	447	330	369	136	328
FinRL-L-999	631	150	152	206	137	416	396	323	341	**752**	656
FinRL-M-99	631	300	357	206	355	700	388	376	357	593	**1213**
FinRL-M-999	631	300	357	206	504	621	470*	356*	373*	504	**698**
FinRL-H-99	631	441	419	206	252	**671**	464	426	531	640	484
FinRL-H-999	631	441	419	206	270	444	495	330	373	581	**787**
CL-L-99	50350	28500	29514	16280	29420	**30670**	29918	23238*	21135*	29902	23326*
CL-L-999	50350	28500	29514	16280	30317	**31611**	30141*	30451	21756*	23528*	23019*
CL-L-9999	50350	28500	29514	16280	30231	**32285**	30716	25107*	21396*	23293*	23431*
CL-M-99	50350	37800	36900	16280	27422	39551	**40957**	31398	21553*	18606*	23166*
CL-M-999	50350	37800	36900	16280	39132	42737	**54742**	25320*	22586*	38951*	23093*
CL-M-9999	50350	37800	36900	16280	38331	**42917**	40416	37222*	21499*	23365*	23467*
CL-H-99	50350	48600	48818	16280	53071	**55158**	53402	43254*	24867*	23289*	23733*
CL-H-999	50350	48600	48818	16280	54622	43437*	54397*	37040*	22151*	**54900**	23288*
CL-H-9999	50350	48600	48818	16280	52957	54696	55166	48427*	21849*	**56874**	23284*
Average Rank	-	5.06	6.04	9.04	4.63	2.45	4.61	5.63	7.08	3.90	6.29

Rongjun Qin et al. NeoRL: A Near Real-World Benchmark for Offline Reinforcement Learning. NeurIPS 2022.

Experiments and Findings of NeoRL

- OPE performance (FQE method)
  - RC score: ranking correlation
  - Top-k score: top selected policy value (normalized to [0,1])

Task Name	RC Score	Top-1 Mean Score	Top-3 Mean Score	Top-5 Mean Score	Top-1 Max Score	Top-3 Max Score	Top-5 Max Score	Policy Mean Score
Walker2d-v3-L-99	$-.282 \pm .062$	$.182 \pm .131$	$.172 \pm .019$	$.165 \pm .022$	$.182 \pm .131$	$.366 \pm .000$	$.384 \pm .012$	$.335 \pm .000$
Walker2d-v3-L-999	$.118 \pm .092$	$.039 \pm .034$	$.044 \pm .044$	$.054 \pm .030$	$.039 \pm .034$	$.083 \pm .096$	$.161 \pm .103$	$.390 \pm .000$
Walker2d-v3-L-9999	$.341 \pm .025$	$.009 \pm .001$	$.035 \pm .015$	$.047 \pm .009$	$.009 \pm .001$	$.054 \pm .019$	$.084 \pm .002$	$.427 \pm .000$
Walker2d-v3-M-99	$-.152 \pm .080$	$.264 \pm .353$	$.297 \pm .027$	$.357 \pm .080$	$.264 \pm .353$	$.832 \pm .097$	$.887 \pm .089$	$.473 \pm .000$
Walker2d-v3-M-999	$-.131 \pm .038$	$.006 \pm .001$	$.091 \pm .106$	$.143 \pm .095$	$.006 \pm .001$	$.240 \pm .306$	$.387 \pm .262$	$.466 \pm .000$
Walker2d-v3-M-9999	$.101 \pm .030$	$.031 \pm .015$	$.025 \pm .007$	$.022 \pm .004$	$.031 \pm .015$	$.043 \pm .001$	$.043 \pm .001$	$.459 \pm .000$
Walker2d-v3-H-99	$-.116 \pm .078$	$.274 \pm .379$	$.364 \pm .126$	$.397 \pm .137$	$.274 \pm .379$	$.721 \pm .125$	$.734 \pm .136$	$.481 \pm .000$
Walker2d-v3-H-999	$-.216 \pm .083$	$.006 \pm .001$	$.005 \pm .001$	$.005 \pm .001$	$.006 \pm .001$	$.006 \pm .001$	$.007 \pm .000$	$.388 \pm .000$
Walker2d-v3-H-9999	$.296 \pm .015$	$.005 \pm .000$	$.007 \pm .004$	$.068 \pm .086$	$.005 \pm .000$	$.013 \pm .011$	$.315 \pm .420$	$.448 \pm .000$
Average	$-.005 \pm .222$	$.091 \pm .209$	$.116 \pm .138$	$.140 \pm .153$	$.091 \pm .209$	$.262 \pm .321$	$.334 \pm .340$	$.430 \pm .046$

Ineffective policy ranking and selection via OPE (FQE method)

Rongjun Qin et al. NeoRL: A Near Real-World Benchmark for Offline Reinforcement Learning. NeurIPS 2022.

Experiments and Findings of NeoRL

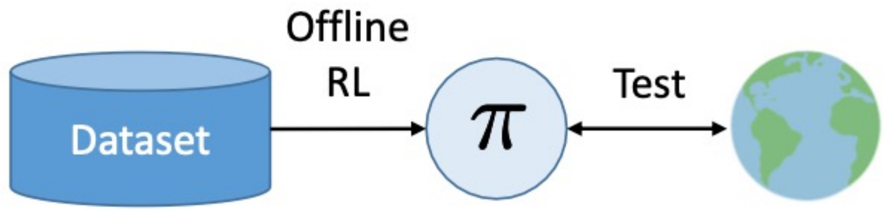
Ratio of winning the 3 baselines over the 51 tasks by online evaluation.

Baseline	CQL	PLAS	BCQ	MOPO	MB-PPO w/ KL	MB-PPO w/o KL
Behavior Policy	94.12%	72.55%	56.86%	21.57%	78.43%	29.41%
Deterministic Policy	88.24%	60.78%	43.14%	23.53%	64.71%	29.41%
BC	84.31%	54.90%	45.10%	25.49%	56.86%	29.41%

- Maybe-pessimistic findings with realistic challenges
  1. Offline RL algorithms fail to outperform neither the simplest behavior cloning method nor the deterministic behavior policy, only except CQL.
  2. Model-based methods are overall worse than model-free ones in the realistic environments, although may have better potential to achieve the out-of-data generalization ability.

Rongjun Qin et al. NeoRL: A Near Real-World Benchmark for Offline Reinforcement Learning. NeurIPS 2022.

Summary



- Offline RL trains a policy from a batch of pre-collected data, which makes RL closer to real applications
- The most important problem studied in offline RL is about extrapolation (or out-of-distribution) problem
- Model-free offline RL methods build constraints on value function or policies
- Model-based offline RL methods measure the uncertainty of state transition and use it to penalize the out-of-distribution actions
- Offline RL performance is still far from perfect

Thank You!
Questions?

Weinan Zhang

Associate Professor

APEX Data & Knowledge Management Lab

Department of Computer Science and Engineering

Shanghai Jiao Tong University

http://wnzhang.net