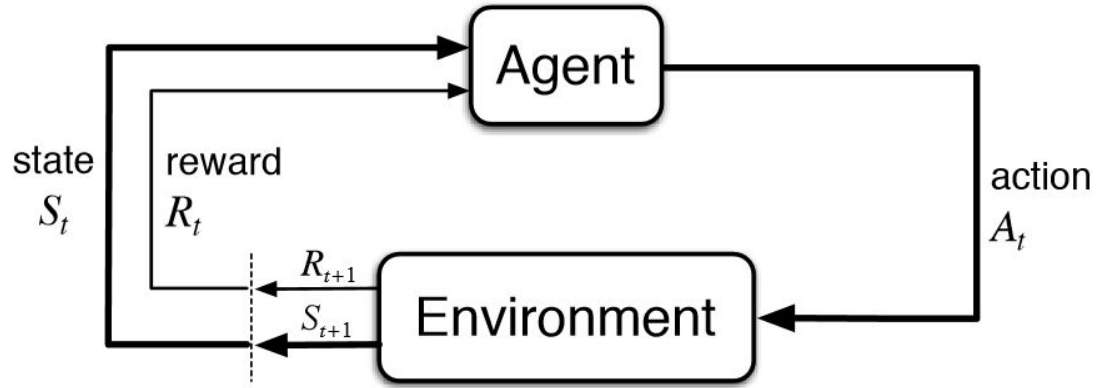


Off-Policy Deep Reinforcement Learning without Exploration

Presenter: Stephane Hatgis-Kessell

10/5/23

RL is a very powerful tool...



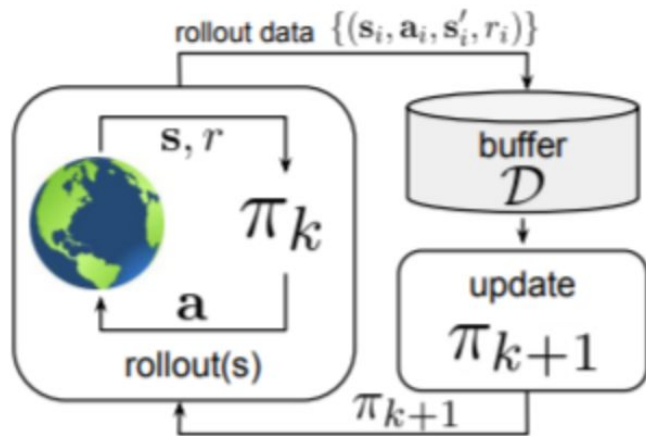
...But allowing an agent to explore the environment can be dangerous.

...But allowing an agent to explore the environment can be dangerous.

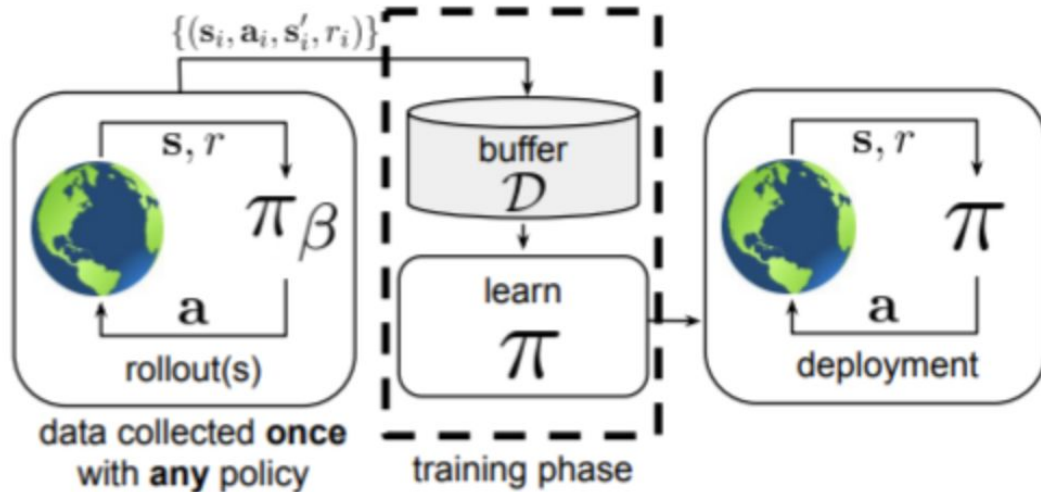


Can we learn a good policy directly from an offline dataset without having to explore the environment further?

Off policy reinforcement learning

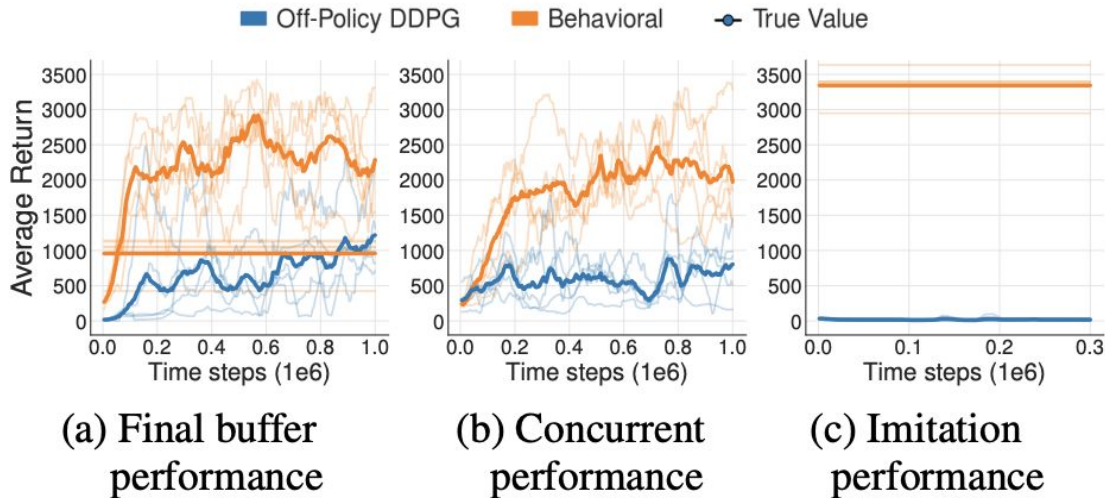


Offline reinforcement learning



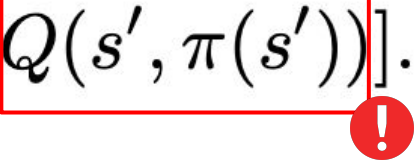
Insights:

1. The behavior policy used by off-policy algorithms are still very close to the target policy.
2. Learning with a truly off-policy behavior policy is hard. Even in the concurrent experiment, where both agents are trained with the same dataset, there is a large gap in performance in every single trial.



Extrapolation Error

- Unseen state-action pairs are erroneously estimated to have unrealistic values.
- Our dataset contains (s, a, s') triplets. In order to update the value of $Q(s, a)$, we backup the value for $Q(s', *)$ but this requires knowing the action at s' which may be different between the target and behaviour policies.

$$\mathcal{T}^\pi Q(s, a) = \mathbb{E}_{s'} [r + \gamma Q(s', \pi(s'))].$$


Why does extrapolation error occur?

- Absent data
 - a. If any state-action pair (s, a) is unavailable, then error is introduced. The estimated $Q(s', a')$ may be arbitrarily bad.
- Model Bias
 - b. For stochastic MDPs, the expectation is with respect to transitions in the batch not the entire MDP.

$$\mathcal{T}^\pi Q(s, a) \approx \mathbb{E}_{s' \sim \mathcal{B}}[r + \gamma Q(s', \pi(s'))] \neq \mathbb{E}_{s'}[r + \gamma Q(s', \pi(s'))].$$

- Training Mismatch
 - c. If the distribution of data in the batch does not correspond with the distribution under the current policy, the value function may be a poor estimate of actions selected by the current policy, due to the mismatch in training.

Theoretical Claims

- A policy is **batch-constrained** if for all (s, a) where $\mu_{\pi}(s) > 0$ and $\pi(a|s) > 0$ then $(s, a) \in B$
- A batch B is **coherent** if for all $(s, a, s') \in B$ then $s' \in B$ unless s' is a terminal state

Key results:

- In a deterministic MDP, the extrapolation error is 0 if and only if we have a batch constrained policy and a coherent batch.
- In a deterministic MDP with some mild convergence assumptions on the learning rate and sampling, and with a coherent batch, we can learn a Q function that induces an optimal batch-constrained policy,

In practice, how can we reduce extrapolation error?

In practice, how can we reduce extrapolation error?

Batch-Constrained deep Q-learning (BCQ): For a given state, BCQ generates plausible candidate actions with high similarity to the batch, and then selects the highest valued action through a learned Q-network.

The key idea here is that want to

- 1) Minimizing the distance of selected actions to the data in the batch.
- 2) Lead to states where familiar data can be observed.
- 3) Maximize the value function

by constraining the actions that we look at when maximizing the value function to be close to the actions in our dataset.

In practice, how can we reduce extrapolation error?

Intuition for action selection

1. Learn a generative model that takes in the current state and, when sampled from, outputs an action that is most likely to appear in the batch given the current state.
2. Sample n actions from this generative model. These will be our candidate actions to take.
3. Perturb these actions, which enables access to actions in a constrained region, without having to sample from the generative model a prohibitive number of times.
4. Choose the action with the highest estimated Q-value.

In practice, how can we reduce extrapolation error?

$$\pi(s) = \operatorname{argmax}_{a_i + \xi_\phi(s, a_i, \Phi)} Q_\theta(s, a_i + \underbrace{\xi_\phi(s, a_i, \Phi)}_{\text{perturbation model outputs action in range } [-\Phi, \Phi]}),$$

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

perturbation model outputs action in range $[-\Phi, \Phi]$.

generative model

In practice, how can we reduce extrapolation error?

$$\pi(s) = \underset{a_i + \xi_\phi(s, a_i, \Phi)}{\operatorname{argmax}} Q_\theta(s, a_i + \underbrace{\xi_\phi(s, a_i, \Phi)}_{\text{perturbation model outputs action in range } [-\Phi, \Phi]}),$$

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

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

- If $\Phi = 0$, and the number of sampled actions $n = 1$, then the policy resembles behavioral cloning
- If $\Phi \rightarrow a_{\max} - a_{\min}$ and $n \rightarrow \infty$, then the algorithm approaches Q-learning, as the policy begins to greedily maximize the value function over the entire action space.

In practice, how can we reduce extrapolation error?

Intuition: we plug the action selection method into a Q-learning algorithm and we are done!

Algorithm 1 BCQ

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 $\theta_1, \theta_2, \phi, \omega$, 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}

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow \operatorname{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

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)

$\theta \leftarrow \operatorname{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \operatorname{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

} Generative model is initialized as a conditional VAE

Algorithm 1 BCQ

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 $\theta_1, \theta_2, \phi, \omega$, 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}

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow \operatorname{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

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)

$\theta \leftarrow \operatorname{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \operatorname{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

} We are going to initialize 4 Q networks as well; 2 for learning and 2 for targets

Algorithm 1 BCQ

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 $\theta_1, \theta_2, \phi, \omega$, 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}

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow \operatorname{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

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)

$\theta \leftarrow \operatorname{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \operatorname{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

} We are going to initialize 4 Q networks as well; 2 for learning and 2 for targets

Why?

We could estimate the value by taking the minimum between two Q-networks. Instead, this work takes a convex combination of the two values (with a higher weight on the minimum).

Algorithm 1 BCQ

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 $\theta_1, \theta_2, \phi, \omega$, 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}

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow \operatorname{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

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)

$\theta \leftarrow \operatorname{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \operatorname{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

} We are going to initialize 4 Q networks as well; 2 for learning and 2 for targets

Why?

...And having the additional 2 target networks, which are updated periodically, improves training stability.

Algorithm 1 BCQ

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 $\theta_1, \theta_2, \phi, \omega$, 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}

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow \operatorname{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

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)

$\theta \leftarrow \operatorname{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \operatorname{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

} Update generative model

Algorithm 1 BCQ

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 $\theta_1, \theta_2, \phi, \omega$, 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}

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow \operatorname{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

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)

$\theta \leftarrow \operatorname{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \operatorname{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

} Sample and perturb candidate actions

Algorithm 1 BCQ

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 $\theta_1, \theta_2, \phi, \omega$, 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}

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow \operatorname{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

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)

$\theta \leftarrow \operatorname{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \operatorname{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

} Update value function

Algorithm 1 BCQ

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 $\theta_1, \theta_2, \phi, \omega$, 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}

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow \operatorname{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

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)

$\theta \leftarrow \operatorname{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \operatorname{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end for

} Update perturbation model to maximize Q-function

Algorithm 1 BCQ

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 $\theta_1, \theta_2, \phi, \omega$, 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}

$\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow \operatorname{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

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)

$\theta \leftarrow \operatorname{argmin}_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow \operatorname{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

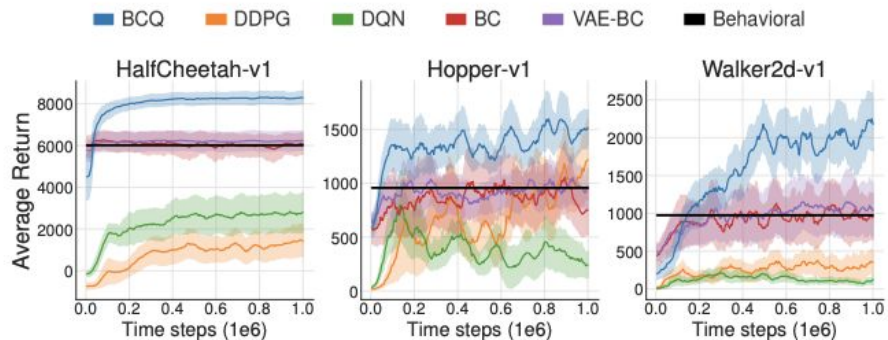
Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$

$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

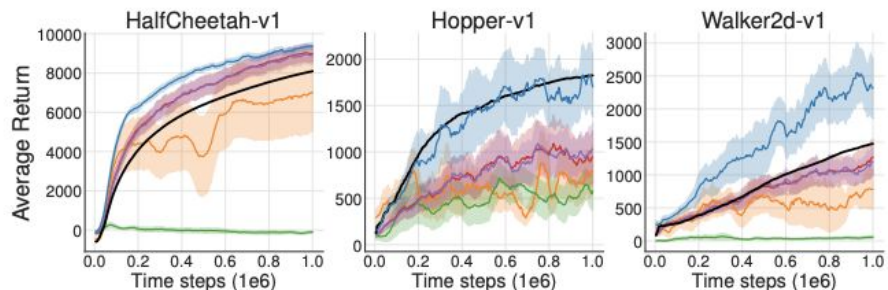
end for

} Update target networks

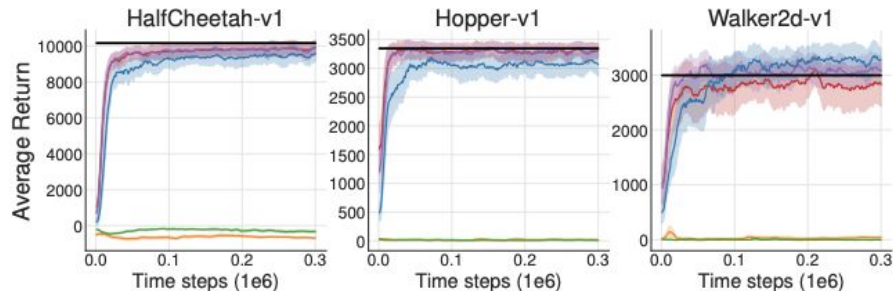
Results



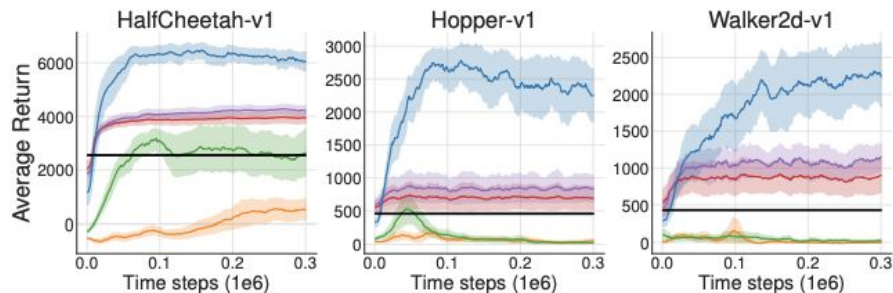
(a) Final buffer performance



(b) Concurrent performance

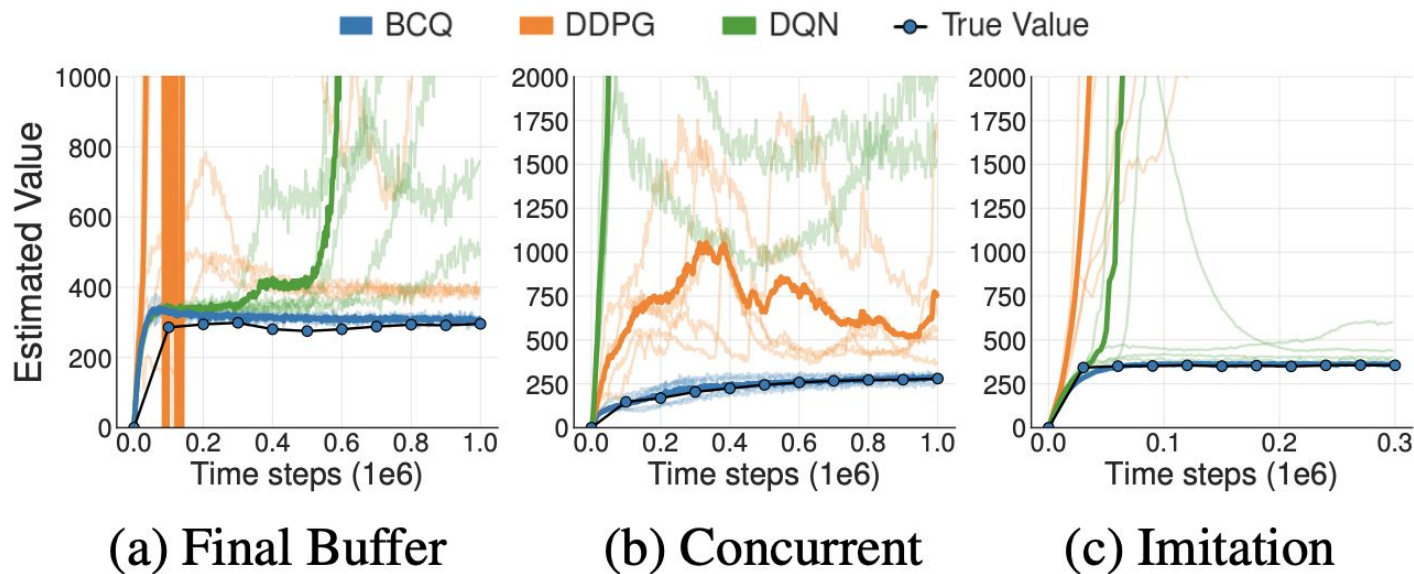


(c) Imitation performance



(d) Imperfect demonstrations performance

Results



Discussion of Results

- Theoretically:
 - In deterministic MDPs with a coherent batch and some mild assumptions, BCQL is guaranteed to induce an optimal batch constrained policy. If the batch is coherent this policy will have an extrapolation error of 0.
- Empirically:
 - In deterministic MDPs with continuous states/actions, BCQ matches or outperforms the behavior policy in every environment and dataset.
 - BCQ exhibits a highly stable value function in each task.

Open Questions/Limitations

- If I have a dataset of optimal or near-optimal demonstrations and I want to use BCQ to learn a policy offline, is it beneficial to add sub-optimal data to this dataset?
- When is it better to use imitation learning algorithms versus offline batch RL algorithms? Can offline batch RL algorithms replace imitation learning algorithms, getting rid of assumptions about the optimality of the given dataset?
- How can we address “model bias” in off-policy RL without having prohibitively large replay buffers?

Related Work

- **Batch Reinforcement Learning**

- All prior work in the offline setting either makes no guarantees on the quality of the policy without infinite data, or come without convergence guarantees.
- Reinforcement learning with a replay buffer can be considered a form of batch reinforcement learning but as shown does not perform well in the truly off-policy setting.

- **Imitation Learning**

- Imitation learning algorithms have successfully been applied to deep RL tasks, but are inadequate for batch RL.
- They require either an explicit distinction between expert and non-expert data, further on-policy data collection, or access to an oracle

- **Uncertainty in Reinforcement Learning**

- This work connects to policy methods with conservative updates such as trust region methods (i.e. TRPO) which aim to keep the updated policy similar to the previous policy. BCQ aims to learn a policy that is close, in output space, to any combination of the previous policies which performed data collection.

Extended Readings

- Kumar, Aviral, et al. "**Conservative q-learning for offline reinforcement learning.**" *Advances in Neural Information Processing Systems* 33 (2020): 1179-1191.
- Kostrikov, Ilya, Ashvin Nair, and Sergey Levine. "**Offline reinforcement learning with implicit q-learning.**" *arXiv preprint arXiv:2110.06169* (2021).
- Janner, Michael, Qiyang Li, and Sergey Levine. "**Offline reinforcement learning as one big sequence modeling problem.**" *Advances in neural information processing systems* 34 (2021): 1273-1286.
- Yu, Tianhe, et al. "**Mopo: Model-based offline policy optimization.**" *Advances in Neural Information Processing Systems* 33 (2020): 14129-14142.
- Kidambi, Rahul, et al. "**Morel: Model-based offline reinforcement learning.**" *Advances in neural information processing systems* 33 (2020): 21810-21823.

Summary

- **Motivation:**

- RL algorithms require exploration to learn, but this can be dangerous or costly in the real world.

- **Problem**

- Can we learn a good policy directly from an offline dataset without having to explore the environment further?

- **Key contributions**

- Empirical evidence that SOTA off-policy RL algorithms fail to learn good policies when the behavior policy is truly off-policy.
- Formal definitions for extrapolation error and the conditions which give rise to it.
- BCQ: a modified q-learning algorithm for learning a policy that constrains its actions to be near those in the given dataset.
- Empirical evidence that BCQ matches or outperforms the behavioral policy in a wide range of continuous control tasks.

Some theoretical grounding...lets create a new MDP



$$p_{\mathcal{B}}(s' | s, a) = \frac{N(s, a, s')}{\sum_{\tilde{s}} N(s, a, \tilde{s})}$$

where $N(s, a, s')$ is the the number of times the tuple (s, a, s') is observed in the current batch

Formalizing extrapolation error (definitions)

Theorem 1. *Performing Q -learning by sampling from a batch \mathcal{B} converges to the optimal value function under the MDP $M_{\mathcal{B}}$.*

$$\epsilon_{\text{MDP}}(s, a) = Q^{\pi}(s, a) - Q_{\mathcal{B}}^{\pi}(s, a).$$

$$\epsilon_{\text{MDP}}^{\pi} = \sum_s \mu_{\pi}(s) \sum_a \pi(a|s) |\epsilon_{\text{MDP}}(s, a)|.$$