2023 Spring RL Final Presentation

Batch Constraint Q-Learning

Members:
111550006 林庭寫
111550093 朱驛庭
111550106 何義翔
111550160 黃佑得

Table of contents

01

Introduction

Brief intro. to today’s main method, BCQ

Four Networks in BCQ

Perturbation | **VAE** | **Q value 1** | **Q value 2**

Exploration

adjust the action derived from VAE into a range.

Generate action

generate an action with given state, the action is near to behavioral policy

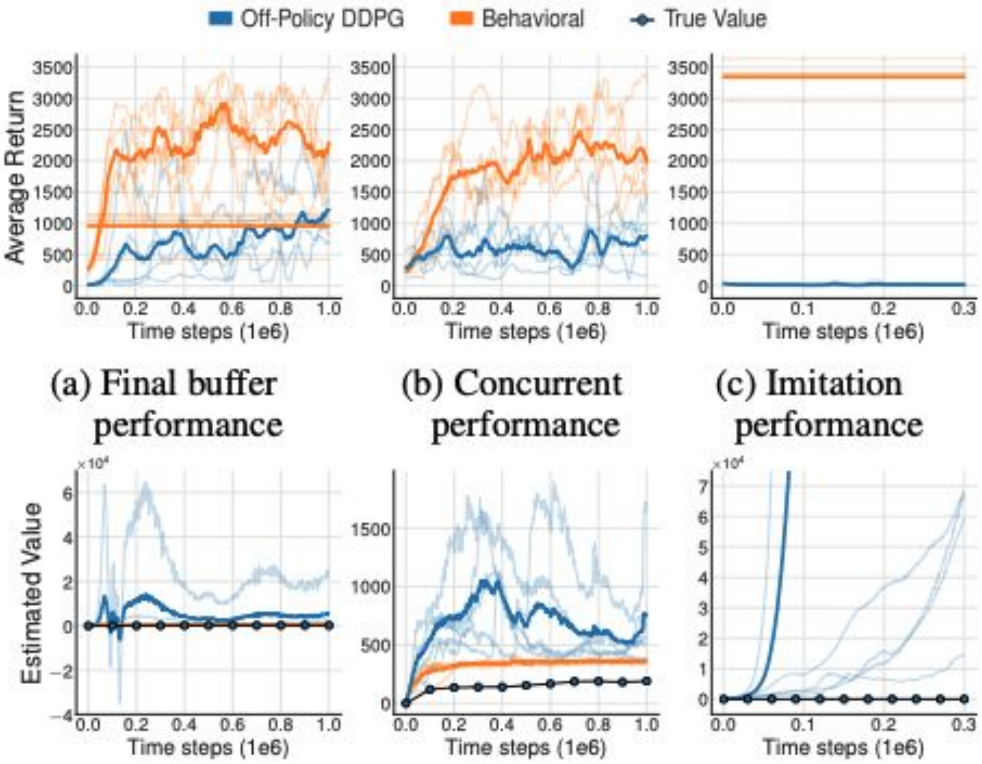
Double Q Learning

calculate the TD target with another two target Q value network

Why BCQ?

Try to solve **Extrapolation error**, caused by:

1. Absent data
2. Model Bias
3. Training mismatch



(a) Final buffer performance
(b) Concurrent performance
(c) Imitation performance
(d) Final buffer value estimate
(e) Concurrent value estimate
(f) Imitation value estimate

02

Reproduction

Reproduce the result from original paper

Maze2D & Hopper Reproduce

		seed=123	seed=456	seed=789	avg	paper
Maze2D	ant-maze-medium-diverse-v0	0.00	0.00	0.00	0.00	0.00
	ant-maze-medium-play-v0	0.00	0.30	0.00	0.10	x
	maze2d-large-v1	109.10	119.10	90.20	106.13	23.20
	maze2d-medium_v1	85.60	71.10	72.60	76.43	35.00
	maze2d-umaze-v1	91.50	76.90	81.30	83.23	41.50
hopper	hopper-random-v0	330.58	324.36	322.87	325.94	323.90
	hopper-medium-v0	929.84	2664.16	988.39	1527.46	1752.40
	hopper-expert-v0	3679.69	2793.27	3664.08	3379.02	x
	hopper-medium-replay-v0	1185.01	726.79	1165.64	1025.81	1057.80
	hopper-medium-expert-v0	3667.69	3518.68	3668.15	3618.17	3588.50

Halfcheetah & Walker2D Reproduce

		seed=123	seed=456	seed=789	avg	paper
halfcheetah	halfcheetah-random-v0	-1.12	-1.21	-1.46	-1.26	-1.30
	halfcheetah-medium-v0	4688.00	4462.00	4721.00	4623.67	4767.90
	halfcheetah-expert-v0	11211.00	10806.00	12307.00	11441.33	x
	halfch.-medium-replay-v0	4371.00	4315.00	4125.00	4270.33	4463.90
	halfch.-medium-expert-v0	12073.00	10727.00	12615.00	11805.00	7750.80

walker2d	walker2d-random-v0	220.94	221.52	222.11	221.52	228.00
	walker2d-medium-v0	2815.42	2847.64	2164.88	2609.31	2441.00
	walker2d-expert-v0	4549.01	3366.86	4521.77	4145.88	x
	walker2d-medium-replay-v0	716.67	779.04	822.28	772.66	688.70
	walker2d-medium-expert-v0	3413.47	3823.71	3335.65	3524.28	2640.30

03

Ablation Study

Find better hyperparameters for gym task.

hopper-random-v0

The reward is relatively stable

Iterations = 1e5

Quick convergence of hopper-random-v0

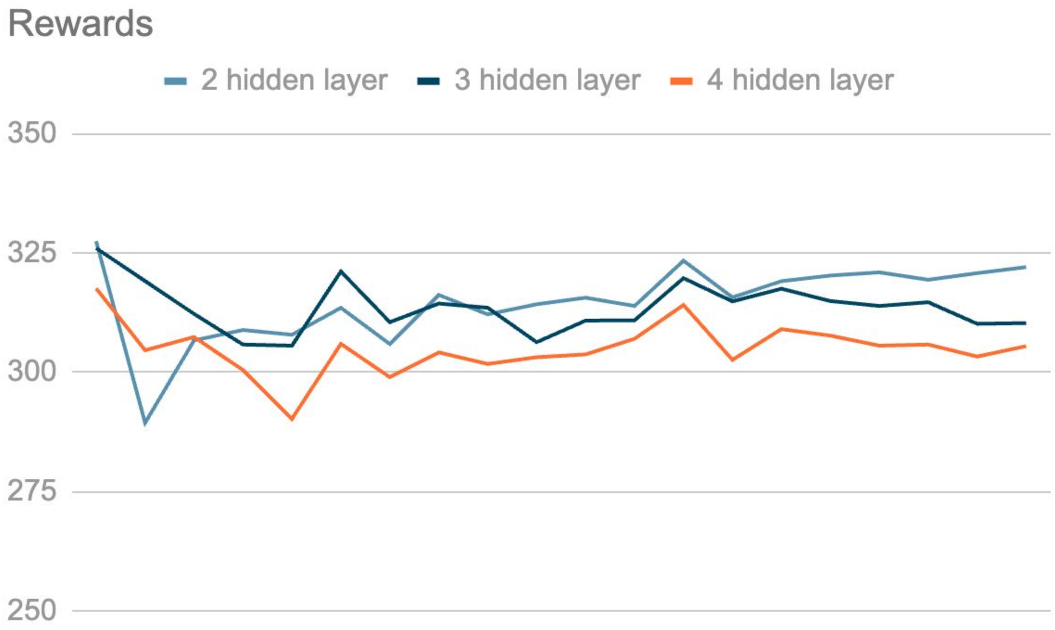
Seed = 123, 456, 789

Get the average reward of three seeds

Deeper Q Network

Multiple hidden layers

The increasing complexity doesn't enhance the overall performance.



Follow the link in the graph to modify its data and then paste the new one here. **[For more info, click.]**

Deeper VAE & Q Network

Multiple VAE hidden layers

The performance maintains .

Follow the link in the graph to modify its data and then paste the new one here. **For more info, click.**

CQL + BCQ

$$\alpha \mathbb{E}_{s\sim\mathcal{D}}\left[\log \sum_a \exp(Q(s,a)) - \mathbb{E}_{a\sim\hat{\pi}_\beta(a|s)}\left[Q(s,a)\right]\right] + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k\right)^2\right]$$

$$\log \sum_a \exp(Q(s,a))$$

Logsumexp

Try to approximate the original bound

$$\mathbb{E}_{a\sim\hat{\pi}_\beta(a|s)}\left[Q(s,a)\right]$$

Current Q

Directly use current Q-value

$$\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k\right)^2\right]$$

Critic Loss

The Bellman error

CQL + BCQ

```
current_Q1, current_Q2 = self.critic(state, action)
vae_Q1, vae_Q2         = self.critic(state, self.vae.decode(next_state))
pred_Q1, pred_Q2       = self.critic(state, self.actor.forward(state, action))
```

For reference, click.

CQL + BCQ

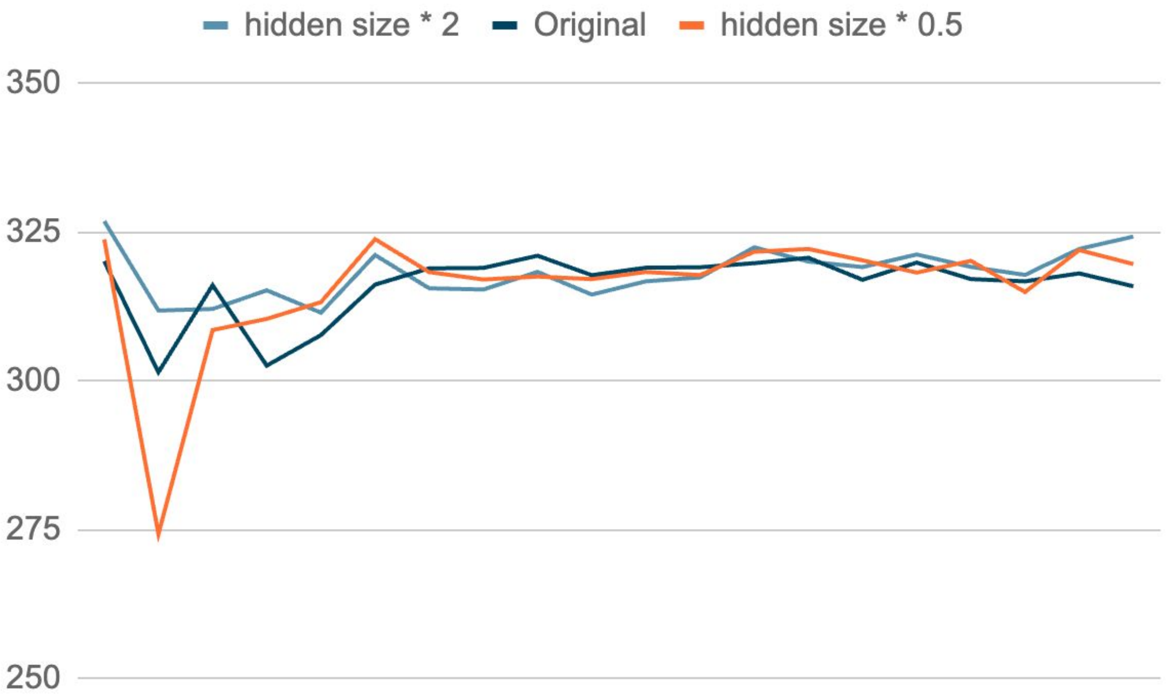
```
alpha = torch.tensor(0.4, dtype=torch.float32)
q1_cat = torch.cat([current_Q1, vae_Q1, pred_Q1], 1)
q2_cat = torch.cat([current_Q2, vae_Q2, pred_Q2], 1)

logsumexp_Q1 = torch.logsumexp(q1_cat, dim=1).mean()
logsumexp_Q2 = torch.logsumexp(q2_cat, dim=1).mean()

penalty1 = alpha*(logsumexp_Q1 - current_Q1)
penalty2 = alpha*(logsumexp_Q2 - current_Q2)

bellman_er1 = F.mse_loss(current_Q1, target_Q)
bellman_er2 = F.mse_loss(current_Q2, target_Q)
critic_loss = (0.5 * bellman_er1 + penalty1) + (0.5 * bellman_er2 +
penalty2)
```

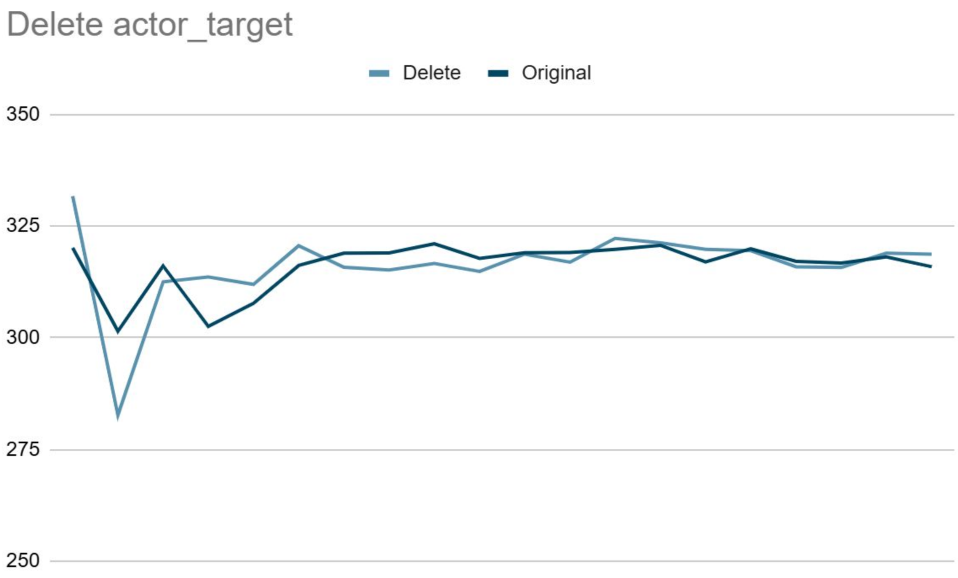
Scaling Hidden Size



Delete Target Perturbation Network

Use the same network to evaluate Q target

The performance maintains.



Follow the link in the graph to modify its data and then paste the new one here. **For more info, click.**

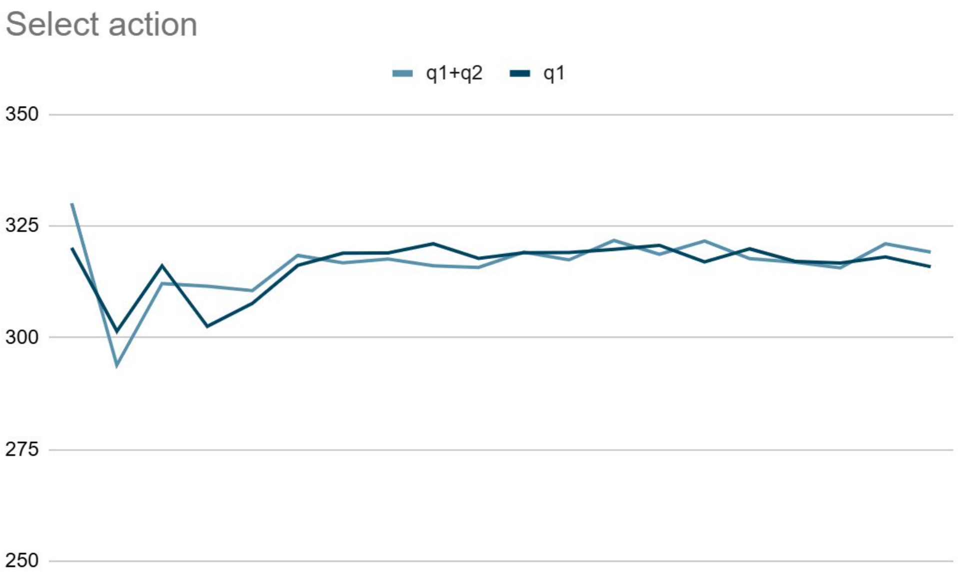
Delete Target Perturbation Network

```
def train(self, replay_buffer, iterations, batch_size=100):
# ...
# Critic Training
with torch.no_grad():
    # Duplicate next state 10 times
    next_state = torch.repeat_interleave(next_state, 10, 0)
    # Compute value of perturbed actions sampled from the VAE
    target_Q1, target_Q2 = self.critic_target(next_state, self.actor(next_state, self.vae.decode(next_state)))
    # ...
```

Use (q1 + q2) to Select Action

The original implement use only q1

The performance maintains.



Follow the link in the graph to modify its data and then paste the new one here. **For more info, click.**

Use (q1 + q2) to Select Action

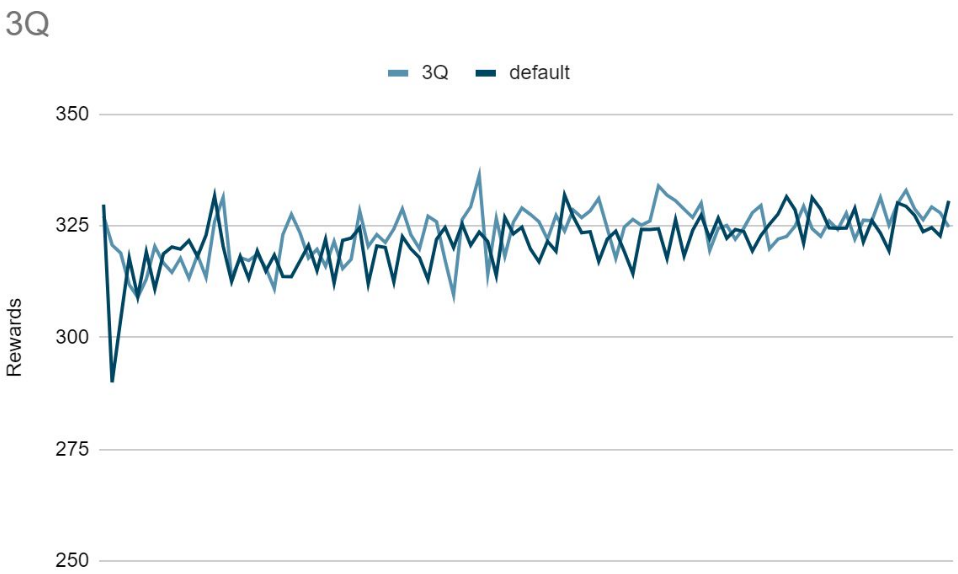
```
def select_action(self, state):
    with torch.no_grad():
        state = torch.FloatTensor(state.reshape(1, -1)).repeat(100, 1).to(self.device)
        action = self.actor(state, self.vae.decode(state))
        q1 = self.critic.q1(state, action)
        ind = q1.argmax(0)
    return action[ind].cpu().data.numpy().flatten()
```

Triple Q Networks

Use

3 Q-Networks

The performance maintains .



Follow the link in the graph to modify its data and then paste the new one here. **For more info, click.**

Triple Q Networks

```
# Critic Training
with torch.no_grad():
    # Duplicate next state 10 times
    next_state = torch.repeat_interleave(next_state, 10, 0)

    # Compute value of perturbed actions sampled from the VAE
    target_Q1, target_Q2 ,target_Q3= self.critic_target(next_state, self.actor_target(next_state, self.vae.decode(next_state)))

    # Soft Clipped Double Q-learning
    #mini = torch.min(target_Q1, target_Q2)
    #maxi = torch.max(target_Q1, target_Q2)
    mini = torch.min( torch.min(target_Q1, target_Q2), target_Q3)
    maxi = torch.max( torch.max(target_Q1, target_Q2), target_Q3)
    target_Q = self.lmbda * mini + (1. - self.lmbda) * maxi
    # Take max over each action sampled from the VAE
    target_Q = target_Q.reshape(batch_size, -1).max(1)[0].reshape(-1, 1)

    target_Q = reward + not_done * self.discount * target_Q

current_Q1, current_Q2, current_Q3 = self.critic(state, action)
critic_loss = F.mse_loss(current_Q1, target_Q) + F.mse_loss(current_Q2, target_Q) + F.mse_loss(current_Q3, target_Q)

self.critic_optimizer.zero_grad()
critic_loss.backward()
self.critic_optimizer.step()
```

04

Adroit Task

solve a more complex task with BCQ

door

door-[human/cloned/expert]

Iterations = 5e5

Seed = 123, 456, 789

Get the average reward of three seeds

Aroit Task Results

	3 VAE + 3 hidden	wide hidden	Triple Q	delete actor target	select action	CQL+BCQ	Origin
door -human	-57.12	-57.03	-59.36	-59.95	-59.81	-53.46	-56.6
door -cloned	-59.21	-56.26	-55.27	-57.52	-57.45		-56.3
door -expert			2948.32	2992.68	2972.49	2943.71	2850.7

Thanks!