Implementation Talk: Offline RL and Conservative Q-Learning

Aviral Kumar

Berkeley
UNIVERSITY OF CALIFORNIA

Abstract: Conservative Q-Learning

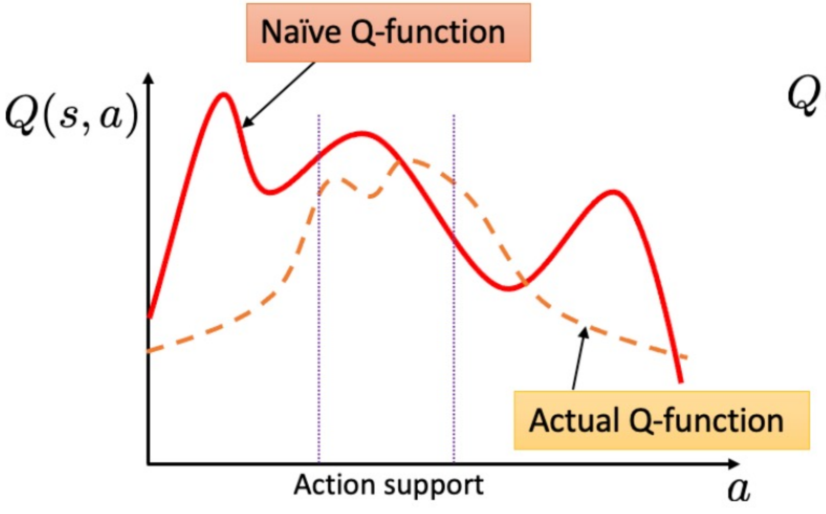
"Bake the pessimism" into the Q-function

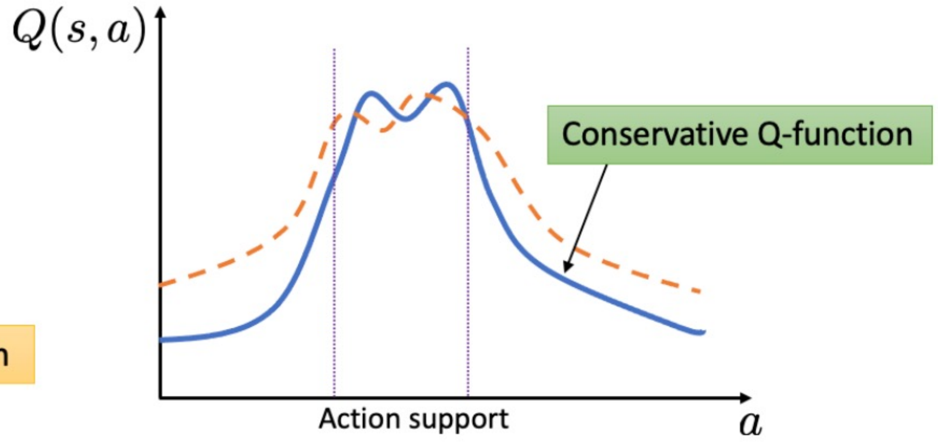
Maximize the data Q-values

Standard TD error

$$Q_{\mathrm{CQL}}(\mathbf{s},\mathbf{a}) := \arg\min_{Q}\max_{\mu}\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[Q(\mathbf{s},\mathbf{a})]\right) + \frac{1}{2\alpha}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[(Q(\mathbf{s},\mathbf{a}) - y(\mathbf{s},\mathbf{a}))^2\right]$$

Minimize OOD Q-values





K., Zhou, Tucker, Levine. **Conservative Q-Learning for Offline Reinforcement Learning**. NeurIPS 2020

Implementation in Discrete Action Settings

Reduces to a combination of TD-error + BC loss

Simple!

$$\min_{\theta} \alpha \underbrace{\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D}} \left[\log \left(\sum_{\mathbf{a}'} \exp(Q_\theta(\mathbf{s}, \mathbf{a}')) \right) \right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}} \left[Q_\theta(\mathbf{s}, \mathbf{a}) \right] \right)}_{\text{"standard NLL BC loss"}} + \mathsf{TDError}(\theta; \mathcal{D})$$

- Compute the log-sum-exp exactly!
- Often discretized representation of Q-values (C51) results in better training
- Use DR3 regularization to effectively leverage capacity

See the scaled Q-learning paper for how to use it with large networks!

DR3: Add an *explicit* regularizer to minimize feature dot products!

DR3 normalization: Normalize features to have norm = 1

K., Agarwal, Ma, Courville, Tucker, Levine. **DR3: Value-Based Deep RL Requires Explicit Regularization.** ICLR 2022

K., Agarwal, Geng, Tucker*, Levine*. **Offline Q-Learning on Diverse, Multi-Task Data Both Scales and Generalizes.** arXiv 2022.

Implementation in Continuous Control

More tricky than discrete settings due to computation of log-sum-exp

$$\log \sum_a \exp(Q_\theta(s,a)) - Q(s, a_{\text{data}}) \Rightarrow \log \int_a \exp(Q_\theta(s,a))da - Q(s, a_{\text{data}})$$

$$\log \int_a p(a|s) \exp(Q_\theta(s,a) - \log p(a|s))da = \log \mathbb{E}_{a\sim p(a|s)}[\exp(Q_\theta(s,a) - \log p(a|s))]$$

Can be computed with samples!

Typically, CQL chooses p(a|s) to be: $p(a|s) = \frac{1}{2}\pi(a|s) + \frac{1}{2}\text{Unif}(a)$

- 4-10 samples of actions suffice
- Can also omit log p(a|s) if the action space is too large

Singh, Yu, Yang, Zhang, **K.**, Levine. **COG: Connecting New Skills to Past Experience via Offline RL.** CoRL 2020.

Offline Hyperparameter Tuning

Network

Generally, use bigger networks
(e.g., on D4RL tasks (256, 256) -> (512, 512, 512))

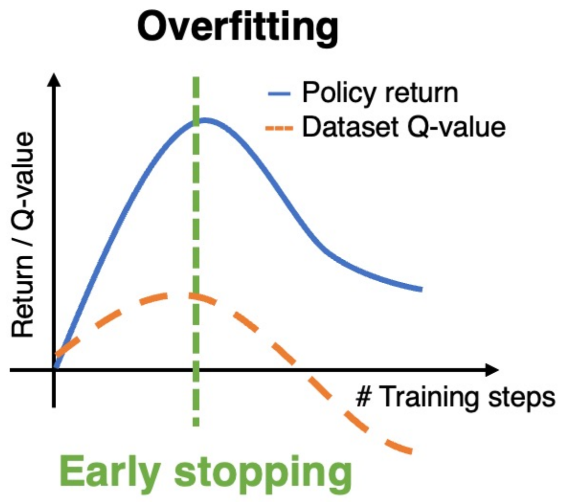
Tuning the hyperparameter α

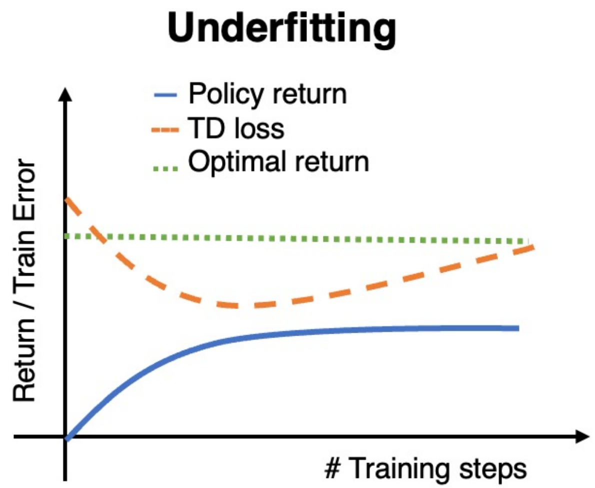
Run a sweep over a certain range of values, pick a sweet spot where TD error is small, and CQL regularizer is small.

If hard to minimize both CQL loss and TD-error, pick a larger model size!

If the CQL regularizer can be minimized to very small (or if Q-values are too small), pick a smaller model size or apply regularization!

Tuning overfitting, underfitting and checkpoint selection in this paper!





K.*, Singh*, Tian, Finn, Levine. **A Workflow for Offline Model-Free Robotic Reinforcement Learning.** CoRL 2021.

General Offline RL Recommendations

Run SARSA first to check if basic details are fine

- SARSA would help identify what's going wrong irrespective of OOD actions!
- **For example**: target network update rate, size of the Q-function, discount factor

Once SARSA passes, try to apply algorithm-specific tuning guidelines

Conservatism α, network capacity

Overfitting, underfitting

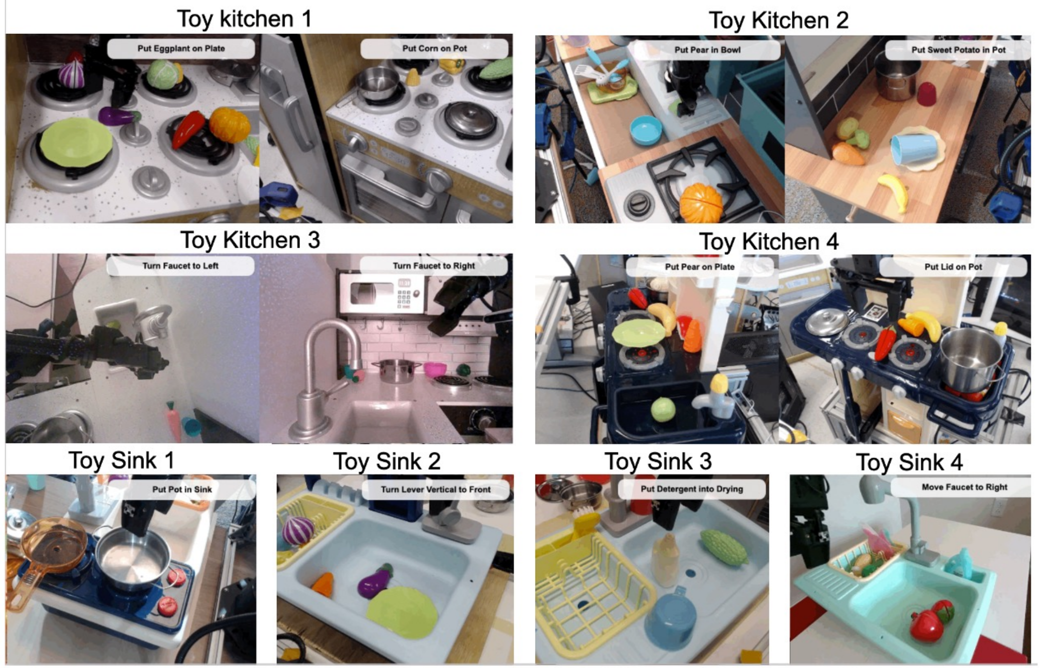
What's the Outcome?

Train a single policy on 40 Atari games

Metric	DDQN vs. Baseline	DDQN + CQL vs. Baseline
Sessions	not stat sig	+ 0.24%
WAU	-0.69%	+ 0.18%
Volume	+7.72%	-1.73%
CTR	-7.79%	+2.26%

fications. These two metrics are usually harder to move without increasing Volume than CTR metric, and hence +0.24% sessions and +0.18% WAU are considered significant business impact. We have ramped the DDQN + CQL model to all users based on this result.

Real-Time Mobile Notification Systems @ LinkedIn [Deployed]



Pre-training on broad data → **Fine-tuning on limited, task-specific data**

Real-robot pre-training and fine-tuning

Code References

- CQL implementation for continuous actions: https://github.com/young-geng/JaxCQL
- CQL implementation in Jax + parallelizable on TPUs + end-to-end from vision on robots: https://github.com/Asap7772/PTR
- CQL implementation in discrete action settings: https://github.com/aviralkumar2907/CQL/tree/master/atari/batch_rl
- Parallel implementation (runs on TPUs) of scaled CQL: https://tinyurl.com/scaled-ql-code

Thank You!