Lecture 7: Offline RL
CS885 Reinforcement Learning

2022-10-03

Complementary readings:
Levine, Kumar, Tucker, Fu (2021) Offline reinforcement learning: Tutorial, review, and perspectives on open problems, *arxiv*.
Kumar, Zhou, Tucker, Levine (2020) Conservative Q-Learning for Offline Reinforcement Learning, *NeurIPS*.

Pascal Poupart
David R. Cheriton School of Computer Science

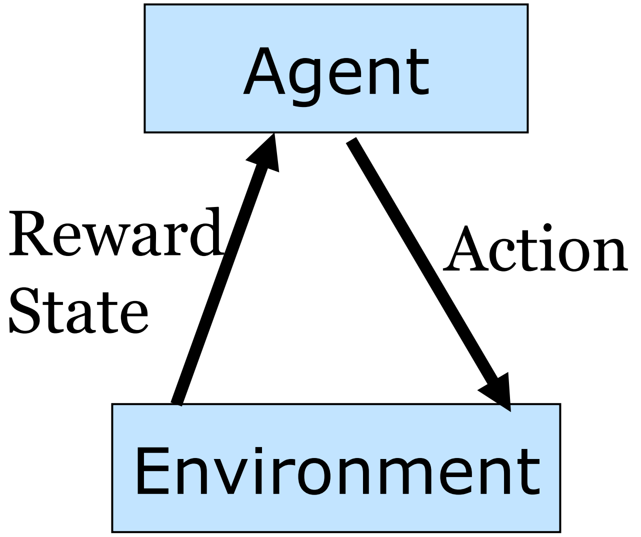
UNIVERSITY OF WATERLOO

Outline

- Can we optimize a policy without interacting with the environment (i.e., learn from previously saved data)?
- Offline RL (also known as batch RL)
  - Conservative Q-Learning
  - Conservative Soft Q-learning
  - Conservative Soft Actor Critic (SAC)

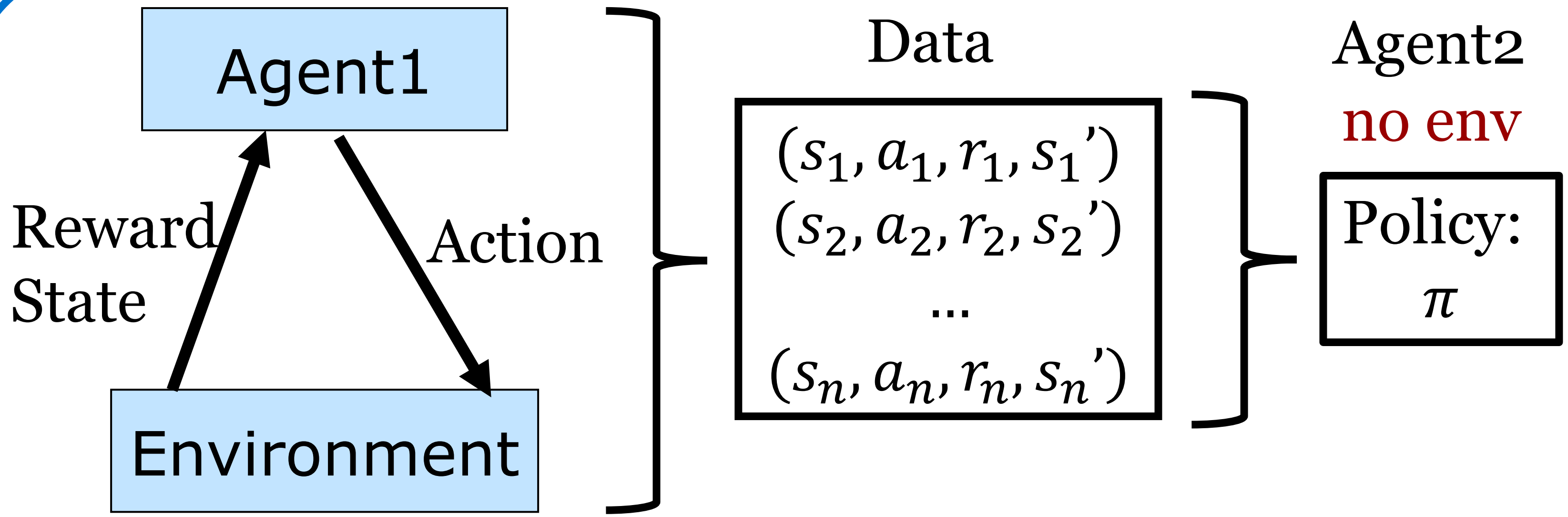
Reinforcement Learning

Online RL



Challenge: exploration

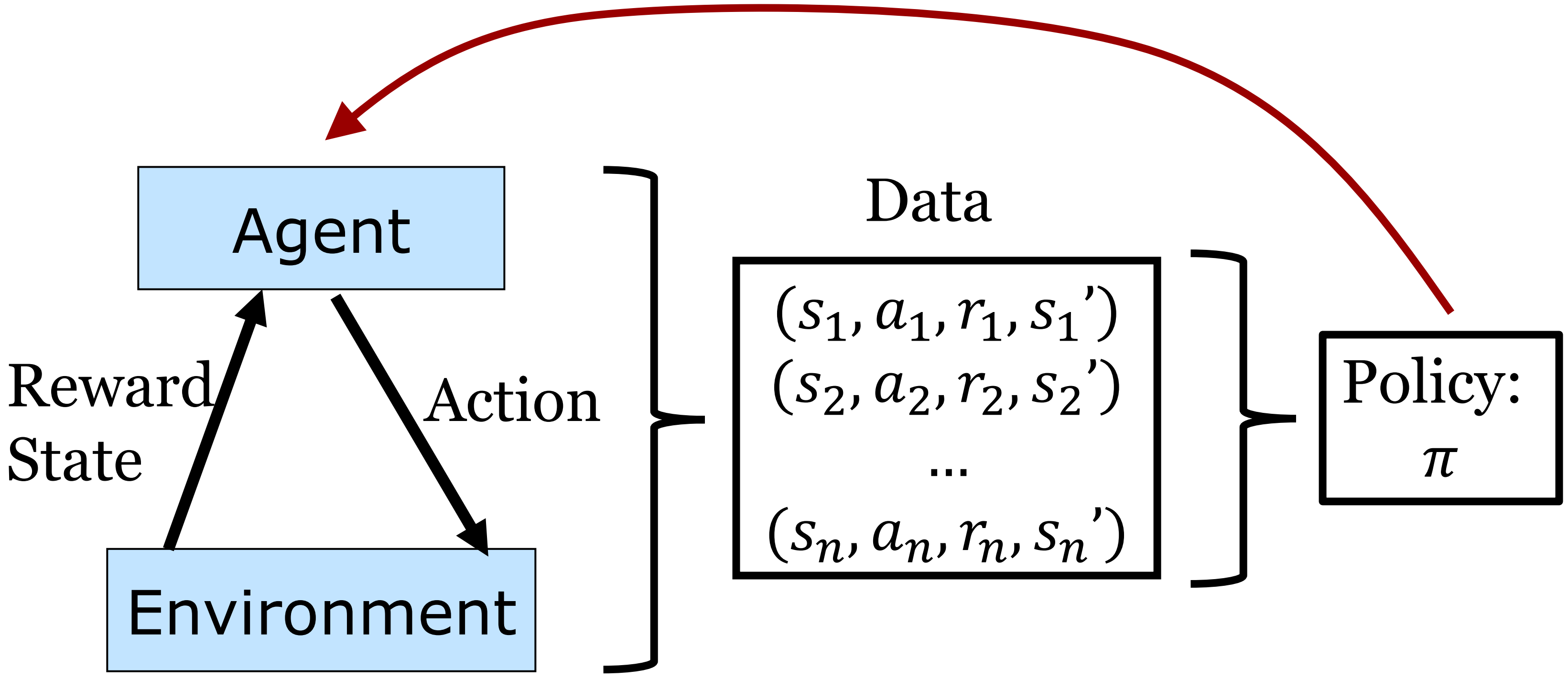
Offline RL



Challenge: distribution shift

Off-Policy RL

- Form of online RL since agent can experiment with its policy in environment



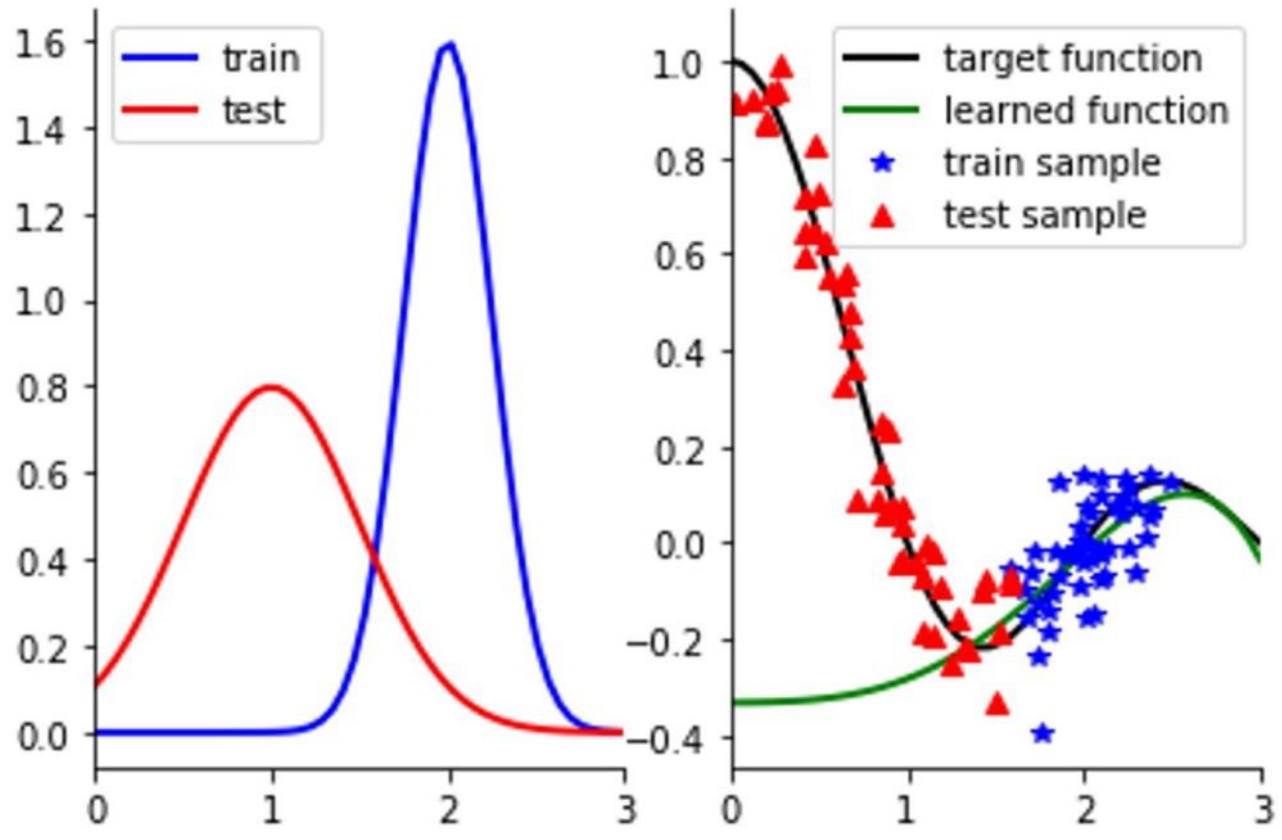
Distribution Shift

- Train distribution different from test distribution

- In RL: data generated by π, but goal is to learn improved π'

- Challenge: may choose actions with overestimated Q-values

From Christyn Zehnder (in-Q-Tel)



Offline RL Techniques

- Importance Sampling
- Policy constraints
- Penalty methods
  - Conservative Q-Learning, conservative Soft Actor Critic
- Model-based RL

Off-Policy Evaluation by Q-learning

- Let $\pi_\beta(a|s)$ be a behaviour policy to collect $D = \{(s, a, r, s')\}$
- We can evaluate a different policy π by off-policy Q-learning:

$$Q^\pi = argmin_Q E_{(s,a,r,s')\sim D}\left[\left(r + \gamma E_{a'\sim\pi(a'|s')}[Q(s', a')] - Q(s, a)\right)^2\right]$$

- Some Q-values underestimated and others overestimated
- Greedy policy improvement: $\pi_{k+1}(s) \leftarrow argmax_a Q^{\pi_k}(s, a)\ \forall s$
  - Problem: select actions with overestimated Q-values

Conservative Off-Policy Evaluation

- Introduce a penalty term

$$\hat{Q}^{\pi} = argmin_Q \ \eta E_{s\sim D, a\sim\pi(a|s)}[Q(s,a)] + E_{(s,a,r,s')\sim D}\left[\left(r + \gamma E_{a'\sim\pi(a'|s')} Q(s',a') - Q(s,a)\right)^2\right]$$

where η: weight that determines importance of penalty

- Let support$(\pi) = \{(s,a) |\ \pi$ reaches (s,a) with non-zero probability$\}$

Theorem: If support$(\pi) \subseteq$ support(π_β), then for sufficiently large η,

$$\hat{Q}^{\pi}(s,a) \leq Q^{\pi}(s,a) \ \ \forall s \in D, a$$

(from Kumar et al. 2020)

Improved Bound

- Remove $E_{s,a\sim D}[Q(s,a)]$ from penalty term

$$\tilde{Q}^{\pi} = argmin_Q \, \eta\big(E_{s\sim D, a\sim\pi(a|s)}[Q(s,a)] - E_{s,a\sim D}[Q(s,a)]\big)$$
$$+E_{(s,a,r,s')\sim D}\left[\left(r + \gamma E_{a'\sim\pi(a'|s')} Q(s',a') - Q(s,a)\right)^2\right]$$

- We cannot guarantee that $\tilde{Q}^{\pi}(s,a) \leq Q^{\pi}(s,a)\ \forall s \in D, a$ for sufficiently large η

- Let $V^{\pi}(s) = E_{a\sim\pi(a|s)} Q^{\pi}(s,a)$

Theorem: If support(π) $\subseteq$ support(π_β), then for sufficiently large η,

$$\tilde{V}^{\pi}(s) \leq V^{\pi}(s)\ \ \forall s \in D$$

(from Kumar et al. 2020)

Conservative Q-learning

- Idea: let π be the greedy policy: $\pi(s) = argmax_a Q(s,a)$

$$\tilde{Q}^* = argmin_Q\, \eta \left(E_{s\sim D}\left[\max_a Q(s,a)\right] - E_{s,a\sim D}[Q(s,a)] \right)$$
$$+ E_{(s,a,r,s')\sim D}\left[\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)^2 \right]$$

Conservative Q-Learning

Load fixed buffer of experiences

Initialize weights $\boldsymbol{w}$ and $\overline{\boldsymbol{w}}$ at random in $[-1,1]$

Loop

Sample minibatch of n experiences from buffer

Bellman error: $Err(\boldsymbol{w}) = \frac{1}{n}\sum_{(s,a,r,s')\in minibatch}\left[\left(Q_{\boldsymbol{w}}(s,a) - r - \gamma \max_{a'} Q_{\overline{\boldsymbol{w}}}(s',a')\right)^2\right]$

Penalty: $Penalty(\boldsymbol{w}) = \frac{1}{n}\sum_{(s,a)\in minibatch}[\max_{\hat{a}} Q_{\boldsymbol{w}}(s,\hat{a}) - Q_{\boldsymbol{w}}(s,a)]$

Update weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha\left(\frac{\partial Err}{\partial \boldsymbol{w}} + \eta\,\frac{\partial Penalty}{\partial \boldsymbol{w}}\right)$

Every c steps, update target: $\overline{\boldsymbol{w}} \leftarrow \boldsymbol{w}$

Conservative Soft Q-Learning

Load fixed buffer of experiences

Initialize weights $\boldsymbol{w}$ and $\overline{\boldsymbol{w}}$ at random in $[-1,1]$

Loop

Sample minibatch of n experiences from buffer

Bellman error: $Err(\boldsymbol{w}) = \frac{1}{n}\sum_{(s,a,r,s')\in minibatch}\left[\left(Q_{\boldsymbol{w}}(s,a) - r - \gamma \,\widetilde{\max}_{\lambda\, a'}\, Q_{\overline{\boldsymbol{w}}}(s',a')\right)^2\right]$

Penalty: $Penalty(\boldsymbol{w}) = \frac{1}{n}\sum_{(s,a)\in minibatch}[\widetilde{\max}_{\lambda\, \hat{a}}\, Q_{\boldsymbol{w}}(s,\hat{a}) - Q_{\boldsymbol{w}}(s,a)]$

Update weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha\left(\frac{\partial Err}{\partial \boldsymbol{w}} + \eta\,\frac{\partial Penalty}{\partial \boldsymbol{w}}\right)$

Every c steps, update target: $\overline{\boldsymbol{w}} \leftarrow \boldsymbol{w}$

Conservative Soft Actor Critic (SAC)

Load fixed buffer of experiences

Initialize weights $\boldsymbol{w}, \bar{\boldsymbol{w}}$ and $\boldsymbol{\theta}$ at random in $[-1,1]$

Loop

Sample minibatch of n experiences from buffer

For each experience (s, a, r, s') in minibatch, sample $a' \sim \pi_\theta(a'|s')$

Bellman error: $Err(\boldsymbol{w}) = \frac{1}{n}\sum_{(s,a,r,s',a')\in minibatch}\left[\left(Q_{\boldsymbol{w}}(s,a) - r - \gamma[Q_{\bar{\boldsymbol{w}}}(s',a') + \lambda H(\pi_{\boldsymbol{\theta}}(\cdot\,|s'))]\right)^2\right]$

Penalty: $Penalty(\boldsymbol{w}) = \frac{1}{n}\sum_{(s,a)\in minibatch}[\widetilde{\max}_{\hat{a}}{}_{\lambda}\, Q_{\boldsymbol{w}}(s,\hat{a}) - Q_{\boldsymbol{w}}(s,a)]$

Q-function update: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha\left(\frac{\partial Err}{\partial \boldsymbol{w}} + \eta\frac{\partial Penalty}{\partial \boldsymbol{w}}\right)$

Policy update: Update policy: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\frac{\partial KL\left(\pi_{\boldsymbol{\theta}} \middle| softmax\left(Q_{\bar{\boldsymbol{w}}}/\lambda\right)\right)}{\partial \boldsymbol{\theta}}$

Every c steps, update target: $\bar{\boldsymbol{w}} \leftarrow \boldsymbol{w}$

Empirical Evaluation

- Kumar et al. (NeurIPS-2020)

Conservative SAC

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**

Table 1: Performance of CQL($\mathcal{H}$) and prior methods on gym domains from D4RL, on the normalized return metric, averaged over 4 seeds. Note that CQL performs similarly or better than the best prior method with simple datasets, and greatly outperforms prior methods with complex distributions ("–mixed", "–random-expert", "–medium-expert").