Batch Reinforcement Learning Through Continuation Method

Yijie Guo[1] **Shengyu Feng**[1] **Nicolas Le Roux**[2] **Ed Chi**[2] **Honglak Lee**[1,2] **Minmin Chen**[2]

[1] University of Michigan [2] Google AI

Neural Information Processing Systems (NeurIPS), Offline Reinforcement Learning Workshop, 2020

Introduction

- Under **batch RL** setting, the agent is asked to learn its policy from a **fixed set** of interactions collected by a different (and possibly unknown) policy commonly referred to as the behavior policy, without the flexibility to gather new interactions.
- Batch setup makes the **policy optimization** even harder as it adds large variance to the gradient estimate, especially when the learned policy differs from the behavior policy used to generate the fixed trajectories.
- **Continuation methods** attempt to solve the global optimization problem by progressively solving a sequence of new objectives that can be optimized more efficiently and then trace back the solutions to the original one.
- We change the objective function of policy optimization by including an additional term penalizing the **KL divergence** between the target policy and the behavior policy. We then **gradually decrease** the weight of that penalty, eventually converging to optimizing the expected return.

Optimizing Expected Return with KL regularization

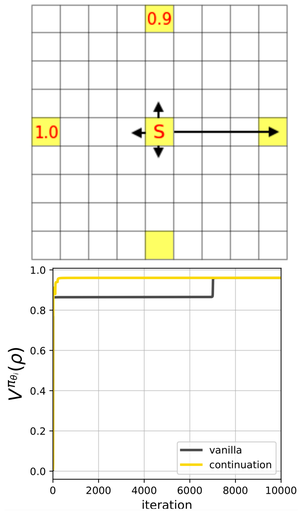
- Objective function of expected return:

$$V^{\pi}(s) = \mathbb{E}_{s_0=s, a_t\sim\pi(\cdot|s_t), s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right]$$

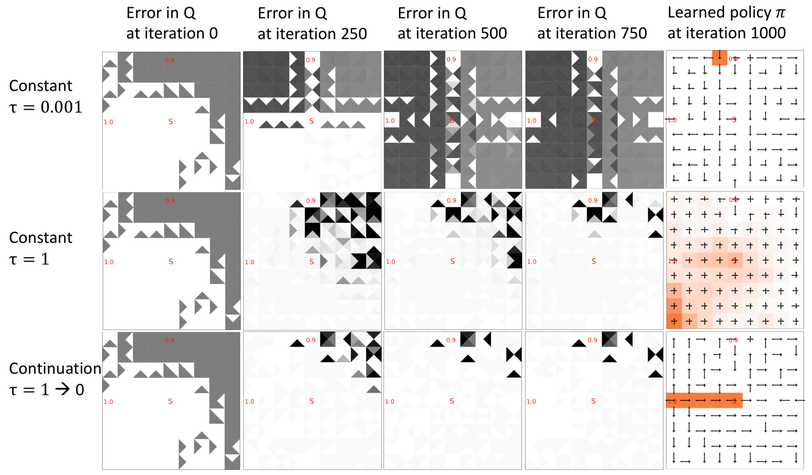
- Objective function of KL regularized expected return

$$\tilde{V}^{\pi,\tau}(s) = \mathbb{E}_{s_0=s, a_t\sim\pi(\cdot|s_t), s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}\left[\sum_{t=0}^{\infty}\gamma^t\left(r(s_t,a_t) - \tau\log\frac{\pi(a_t|s_t)}{\beta(a_t|s_t)}\right)\right]$$

- In tabular setting, with exact policy gradient, expected return can be optimized with convergence rate $\mathcal{O}(1/t)$; KL regularized expected return can be optimized with convergence rate $\mathcal{O}(e^{-t})$.
- The **faster convergence rate** inspires us to optimize KL regularized expected return $\tilde{V}^{\pi,\tau}(\rho)$ to reach π^*_τ, then use π^*_τ as initialization, gradually decrease the temperature τ to 0, and eventually move from π^*_τ to $\pi^* = \arg\max_\pi V^{\pi}(\rho)$.
- On the toy example, optimizing a sequence of objective functions $\tilde{V}^{\pi,\tau}(\rho)$ can reach the optimal solution for $V^{\pi}(\rho)$ significantly faster.
- Without exact policy gradient, we propose **soft policy iteration,** alternating between **soft policy evaluation** and **soft policy improvement**, and it will provably converge to the optimal policy maximizing KL regularized expected return.
- As the temperature goes to 0, the continuation method is guaranteed to **asymptotically converge** to the optimal policy maximizing expected return.



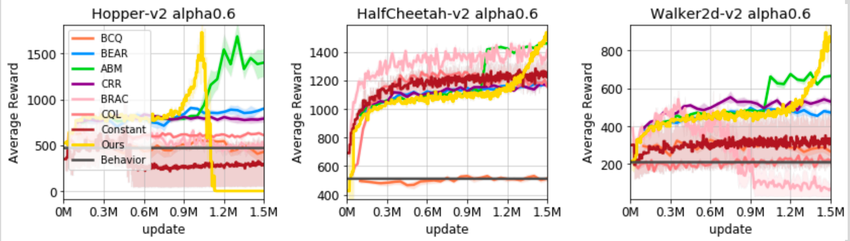
Error in Value Estimation



- In the toy example, gradually relaxing KL regularization towards zero alleviates the propagation of errors in the soft Q estimate and helps the agent converge to the optimal policy.

Experimental Results

- On the Mujoco dataset with relatively reasonable quality (i.e. α = 0.2, 0.4, 0.6), ours performs comparable or better than the baselines.
- With α = 0.2, i.e., close to optimal behavior policy, all the methods perform similarly and one can achieve a good return by simply cloning the behavior policy.
- With α = 0.8, i.e., low-quality behavior policy, there are few good trajectories in the dataset for any methods to learn.
- The advantage of our method is most obvious when α = 0.6, as the dataset contains trajectories of both high and low cumulative rewards. Our method can learn from the relatively large number of good trajectories and at the same time deviate from the behavior policy to avoid those bad trajectories and achieve higher rewards.

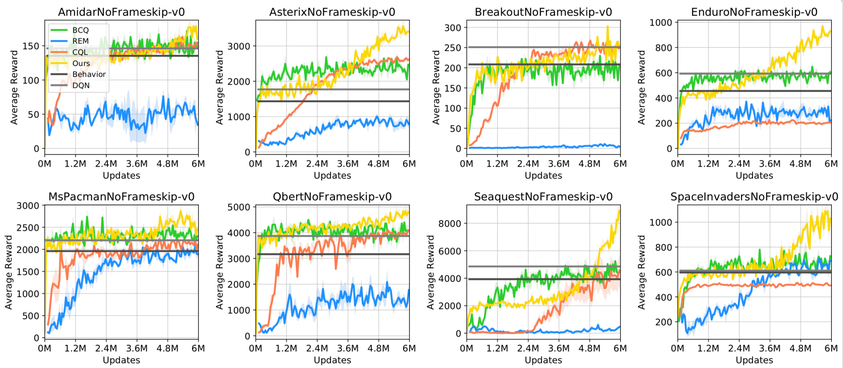


Experimental Results

- On Atari datasets, our approach achieves higher scores than the baselines on 7 out of 8 games, and perform comparably on the other one.

	Amidar	Asterix	Breakout	Enduro	MsPacman	Qbert	Seaquest	SpaceInvaders
BCQ	154.4 ±11.0	2466.7 ±273.4	203.8 ±19.6	604.7 ±39.7	2299.7 ±150.2	4088.3 ±332.8	4420.0 ±548.7	726.5 ±58.7
REM	32.0 ±3.5	741.4 ±236.7	3.1 ±0.0	244.2 ±19.8	1997.4 ±6.4	2062.9 ±511.5	474.0 ±61.9	678.4 ±41.3
CQL	145.0 ±1.6	2618.7 ±102.1	**253.7 ±14.4**	206.6 ±5.8	2234.6 ±203.2	4094.7 ±74.1	4652.6 ±2017.1	493.5 ±11.4
Ours	**174.5 ±7.1**	**3476.7 ±229.0**	199.0 ±32.0	**922.9 ±31.7**	**2494.0 ±301.3**	**4732.5 ±172.5**	**9935.0 ±1175.9**	**1070.3 ±137.1**

Table 2: Results on Atari, the mean and standard deviation of scores achieved in 3 indenpendent runs.



- The problem of recommending movies for each user can be converted to a contextual bandit problem.
- We compare our method with two baselines, as they are commonly used in current industrial recommender system.
- Our method clearly outperforms it across datasets collected using different behavior policies.

	data with avg. reward 0.53		data with avg. reward 0.65		data avg. reward 0.80	
sample size	30,000	60,000	30,000	60,000	30,000	60,000
Cross-Entropy	73.2±0.02	72.6±0.02	80.0±0.01	79.8±0.01	85.7±0.01	84.8±0.01
IPS	78.0±0.03	79.9±0.02	87.2±0.02	88.3±0.01	87.8 ± 0.01	89.4 ± 0.01
Ours	**82.6±0.01**	**83.9±0.01**	**89.8±0.01**	**90.5±0.00**	**90.1±0.01**	**91.9±0.00**

Take-home Message and Conclusion

- We propose a simple yet effective approach, soft policy iteration algorithm through continuation method to alleviate the two challenges in policy optimization under batch reinforcement learning: (1) highly non-smooth objective function which is difficult to optimize (2) high variance in value estimates.
- We provide theoretical ground and visualization tools to help understand this technique. We demonstrate its efficacy on multiple complex tasks.