An Effective Negotiating Agent Framework based on Deep Offline Reinforcement Learning (Supplementary Material)

Siqi Chen[1] **Jianing Zhao**[1] **Gerhard Weiss**[2] **Ran Su**[1] **Kaiyou Lei**[*3]

[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Department of Advanced Computing Sciences, Maastricht University, Maastricht, the Netherlands
[3]College of Computer and Information Science, Southwest University, Chongqing, China

A EXPERIMENTAL SETUP DETAILS

Overall, all of our implementations are based on the rikit[1], a collection of reinforcement learning techniques. Using the same library helps us reduces the impact of implementation differences. We created several agents that are trained by a reinforcement learning algorithm. SAC is used to implement all agents, including baseline agents. In addition, we train a neural network called the density ratio estimator to estimate density ratios of samples. This network can calculate the priority of different trajectories in the reinforcement learning process, allowing the online fine-tuning process to safely utilize online samples by leveraging relevant, near-on-policy offline samples. First, we present our DOREA agent and baseline agents' detailed implementation and corresponding hyper-parameters. Second, we present the model structure and hyper-parameters for the neural network-based density ratio estimator.

A.1 IMPLEMENTAIN DETAILS FOR ALL AGENTS

Due to all the below agents based on SAC, Table 2 shows the general hyper-parameters used in our experiments and training process.

Offline RL. For training offline DOREA agents, we train ensemble DOREA w/o sft by training N=5 (ensemble size) CQL based Q-functions and policies for 5 random seeds separately then combine them. For every offline strategies, we use 2-layer MLPs for value and policy network. Other parameter settings are identical to the setup of Kumar et al. [2020]. See Table 1 for specific parameters.

Online fine-tuning. For fine-tuning processing, we initialize parameter of ensemble agent by the parameter of offline DOREA w/o sft. We also use ensemble size N=5, and trained for 1000 steps every 1000 additional samples were collected. The model has the same network structure, we employed the Adam optimizer [Kingma and Ba, 2014] with policy learning rate of 3e-4 and value learning rate of 3e-4. See Table 1 for specific parameters and architecture.

Baseline. For the SAC-sft, we initialize its strategy with an offline trained CQL agent (We don't use ensemble here). Additionally, both SAC and SAC-sft used the implementation from rlkit with default hyperparameters describe in Table 2.

Furthermore, it should be noted that in all the above agent, we use trainable temperature factor α in our SAC version. Besides this, two critic networks with the same structure are created, each with its own layers and weights. The second critic networks is commonly referred to as the target critic network. The weights from the critic network are copied with smoothing via $target\ update\ \tau$ to the target critic network after each $target\ update\ period$ train step.

[*]Corresponding author, Kaiyou Lei <kylei2022@163.com>
[1]See `https://github.com/vitchyr/rlkit`

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

A.2 IMPLEMENTAIN DETAILS FOR DENSITY RATIO ESTIMATOR

Density ratio estimaor. In balanced replay (BR) component, for training the density ratio estimation network $\omega_\phi(s, a)$, which was designed as a 2-layer MLP, we use batch size 256 (i.e., 256 offline samples and 256 online samples), and learning rate is 3e-4. We apply self-normalization to the estimated density ratio over $\mathcal{B}^{off}$, we calculate priority values:

$$\widetilde{w}_\psi(x) = \frac{w_\psi(x)^{1/T}}{E_{x\sim P}\left[w_\psi(x)^{1/T}\right]} \tag{1}$$

where x and P denote (s, a) and $\mathcal{B}^{off}$ respectively, and T is the temperature hyper-parameter and we set it 5. Before starting fine-tuning, we will add offline samples to the replay buffer at a priority of 1.0. Since it is necessary to ensure that new online samples can be updated at the initial stage, we set a high default priority for newly added samples to ensure this. Specifically, letting M denote the size of the offline buffer. We set the default priority to make the initial 1000 online samples collected have the probability ρ of been seen, where ρ is a hyper-parameter, i.e., priority value of $P_0 := \frac{M}{1000} \cdot \frac{\rho}{1-\rho}$. We used $\rho = 0.75$. After the used online data is updated in RL, the priority of the given sample will be updated appropriately, and then the default priority value will be updated to the maximum priority value seen during fine tuning. For detailed algorithm, please refer to Lee et al. [2022].

B HYPER-PARAMETERS

Table 1: Specific hyper-parameters for different Algo, CQL and BR stand for offline and online training, respectively. The meaning of hyper-parameter names can be found in the original paper.

Algo	Hyper-param name	Value
CQL	conservative weight	10
	# of actions sampled	10
BR	offline buffer size	2e6
	online buffer size	2e6
	density ratio estimation network arch.	[|$\mathcal{S}$|+|$\mathcal{A}$|, 256, 256, 1]
	density ratio estimation network temp T	5
	ρ	0.75

Table 2: General hyper-parameters, CQL and BR stand for offline and online training, respectively.

	CQL	BR SAC-ft SAC(scratch)
Phase	offline	online
General hyper-params		
π Arch.	[|$\mathcal{S}$|+|$\mathcal{A}$|,256,256,1]	
Q Arch.	[|$\mathcal{S}$|+|$\mathcal{A}$|,256,256,1]	
# Q nets	2	
# target update period	1	
soft target update τ	0.005	
Activation	ReLU	
Optimizer	Adam for all	
Adam params	betas = (0.9, 0.999), eps = 1e-8, weight decay = 0	
π lr	1e-4	3e-4
Q lr	3e-4	3e-4
α lr	1e-4	3e-4
# epochs	1000	
# step/epoch	1000	
# batch/step	1	
Batch size	256	
Hyper-params for the base SAC impl.		
Entropy target H	|$\mathcal{A}$|	
Uni-model Gaussian	Yes	
Squashed Gaussian	Yes	

References

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1702–1712. PMLR, 08–11 Nov 2022. URL `https://proceedings.mlr.press/v164/lee22d.html`.