Offline Reinforcement Learning with Munchausen Regularization

Hsin-Yu Liu
University of California San Diego
La Jolla, CA
hyl001@eng.ucsd.edu

Balaji, Bharathan
Amazon*
410 Terry Ave N, Seattle
bhabalaj@amazon.com

Rajesh Gupta
University of California San Diego
La Jolla, CA
gupta@eng.ucsd.edu

Dezhi Hong
University of California San Diego
La Jolla, CA
dehong@eng.ucsd.edu

Abstract

Most temporal differences based (TD-based) Reinforcement Learning (RL) methods focus on replacing the true value of a transiting state by their current estimate of this value. Munchausen-RL (M-RL) proposes the idea of incorporating the current policy to be leveraged to bootstrap RL. It regularize policies that are: 1. Too far from the previous one. 2. Too far from uniform policy. The concept of 1., penalizing two consecutive policies that are far from each other is also applicable to offline settings. In our work, we add the Munchausen term in the Q-update step to penalize policies that deviate from previous policy too far. Our results indicate that this method could be implemented in various offline Q-learning methods to help improve the performance. In addition, we evaluate another TD-based method, prioritized experience replay (PER), it prioritizes the mini-batch by its TD-error and weighted importance sampling. Our results show that Munchausen Offline RL outperforms the original methods that are without the regularization term. The results indicate that adding the extra regularization term indeed helps improve the performance of Q-learning methods even in offline settings.

1 Motivation

Current batch/offline RL methods are mainly focused on utilizing statistical methods or using regularization methods to mitigate the effect of distribution drift. Pessimistic Q-Learning (PQL [9]) adds a state Variational Auto Encoder (VAE [5]) and uses a filtration function to avoid Q-update when state-action visitation is not frequent enough in the batch. Bootstrapping Error Accumulation Reduction (BEAR [7]) uses the sampled version of Maximum Mean Discrepancy (MMD) between the unknown behaviour policy and the actor as constraint to avoid actions that lie outside of the training data distribution. Batch Constrained Q-learning (BCQ [3]) also minimizes the distance of selected action to the data in the batch and leads to states where familiar data could be observed. While effective, however, none of them considers another source of learning—the current policy.

Model-free batch RL is challenging because it is in the deadly triad of off-policy learning, function approximation, and bootstrapping [11]. The key insight of our work is that we improve offline methods in the off-policy Q-update itself. While other works focus on the extrapolation errors, bootstrapping errors, and function approximations.

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2021 (NeurIPS 2021)

2 Methodology

We add a scaled log-policy term in the Q-update step in the Batch RL Q-network architecture inspired by Munchausen-RL [13]. State-of-the-art batch RL algorithms, such as PQL and BEAR, are based on BCQ's architecture, and BCQ uses double-clipped Q-learning architecture. We follow a similar methodology and modify the Q-update step from:

$$r + \gamma \max_{a_i} \left[\lambda \min_{j=1,2} Q_{\theta'_j}(s', a_i) + (1-\lambda) \max_{j=1,2} Q_{\theta'_j}(s', a_i) \right] \text{ to}$$
$$r + \alpha_m \tau_m \ln(softmax(\tfrac{Q_{\theta'}}{\tau_m})) + \gamma \max_{a_i} \left[\lambda \min_{j=1,2} Q_{\theta'_j}(s', a_i) + (1-\lambda) \max_{j=1,2} Q_{\theta'_j}(s', a_i) \right]$$

with the Munchausen term highlighted in red, where α_m and τ are hyperparameters[2]. Algorithm 1 gives the full description. Additionally, we also adapt Prioritized Experience Replay (PER [10]) with BCQ. We compute the rank-based probability $P(j)$ based on priority p_j^α, importance-sampling weight ω_j, and TD-error δ_j. For each mini batch k for $j = 1$ to k:

$$P(j) = p_j^\alpha / \textstyle\sum_i p_i^\alpha$$
$$\omega_j = (N \cdot P(j))^{-\beta} / \max_i \omega_i$$
$$\delta_j = R_j + \gamma_j Q_{target}(S_j, argmax_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$$

Where N is the size of the replay period. We update the transition priority $p_j \leftarrow |\delta_j|$. α and β are the exponent hyperparameters. Finally, we update the critic network with:

$$\theta_i \leftarrow argmin_{\theta_i} N^{-1} \sum \omega (y - Q_{\theta_i})^2$$

We evaluate against the state-of-the-art BRL methods: BCQ, PQL, and BEAR, and compare their modified versions with Munchausen and PER variants. [3]

Table 1: Evaluated Algorithm Variants

Name	Description
BCM	BCQ with Munchausen-RL
PML	PQL with Munchausen-RL
BCQ_PER	BCQ with PER

3 Experimental Setup and Result

3.1 Experimental Setup

We evaluate our methods on MuJoCo [12] environments similar to prior works but use the latest version: *Hopper-v3*, *HalfCheetah-v3*, and *Walker2d-v3*. We use Deep Deterministic Policy Gradient (DDPG [8]) to generate buffers after training for one million time steps with a $\mathcal{N}(0, 0.1)$ Gaussian noise to select random actions. Then the agent is used to generate buffers across five random seeds also with a $\mathcal{N}(0, 0.1)$ Gaussian noise to emulate stochastic processes.

All of our experiments are conducted with Intel Xeon Gold 6230 CPUs (2.10GHz) and NVidia Quadro RTX 8000 GPUs with Ubuntu 18.04 OS. All results shown are trained and evaluated with five buffers with different random seeds.

[2] M-RL regularization consists of two parts: the first part is the one we add on BRL architectures by using Kullback-Leiber divergence to penalize policies that are far from the previous policy, and the other is using an entropy term to penalize policies that deviate far from uniform distribution [14]. Our evaluation shows that penalizing only the first term yields the best outcome.

[3] We also implement PQL_PER, however due to the heuristic in PQL that avoids Q-update when visiting low-data region, the results are not improving, so we omit it in the comparison.

Algorithm 1: BCM algorithm

Input : Batch $\mathcal{B}$, horizon T, target network update rate τ, mini-batch size N, max perturbation Φ, number of sampled actions n, minimum weighting λ, M-RL hyperparameters α_m and temperature parameter scaling the entropy τ_m

Initialize Q-networks $Q_{\theta_1}, Q_{\theta_2}$, perturbation network ξ_ϕ, and VAE $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, with random parameters $\theta_1, \theta_2, \phi, \omega$, and target networks $Q_{\theta_1'}, Q_{\theta_2'}, \xi_{\phi'}$ with

$\theta_1^{'} \leftarrow \theta_1, \theta_2^{'} \leftarrow \theta_2, \phi^{'} \leftarrow \phi$

for $t \leftarrow 1$ **to** T **do**

Sample mini-batch of N transitions $(s, a, r, s^{'})$ from $\mathcal{B}$

$\mu, \sigma = E_{\omega_1}(s, a), \widetilde{a} = D_{\omega_2}(s, z), z \sim \mathcal{N}(\mu, \sigma)$

$\omega \leftarrow argmin_\omega \sum (a - \widetilde{a})^2 + D_{KL}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$

Sample n actions: $\{a_i \sim G_\omega(s^{'})\}_{i=1}^n$

Perturb each action: $\{a_i = a_i + \xi_\phi(s^{'}, a_i, \Phi)\}_{i=1}^n$

Set value target:

$$y = r + \alpha_m \tau_m \ln\left(softmax(\tfrac{Q_{\theta'}}{\tau_m})\right) + \gamma \max_{a_i} \left[\lambda \min_{j=1,2} Q_{\theta_j'}(s^{'}, a_i) + (1-\lambda) \max_{j=1,2} Q_{\theta_j'}(s^{'}, a_i)\right]$$

$\theta \leftarrow argmin_\theta \sum (y - Q_\theta(s, a))^2$

$\phi \leftarrow argmin_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$

Update target networks: $\theta_i^{'} \leftarrow \tau\theta + (1-\tau)\theta_i^{'}$

$\phi^{'} \leftarrow \tau\phi + (1-\tau)\phi^{'}$

end

3.2 Metrics and Results

We report the mean and median scrores across our experiments. Following Agarwal et al. [1], we also report inter-quantile mean (IQM), optimality gap, performance profile and probability of improvement to account for inherent uncertainty in deep RL training.

Aggregate Metrics In Fig.1, aggregate metrics are with 95% of confidence interval (CI) and stratified sampling using percentile bootstrapping $50K$ times. IQM discards the bottom and the top 25% of the scores, then calculates the mean. Optimality gap is the amount by which the algorithm fails to meet a minimum score of $\gamma = 1.0$, typically we set the aim as the normalized human/expert score. We can see that PML has a smaller optimality gap and higher median, IQM, and mean compared with the second-best algorithm, PQL.

Performance Profile Performance profile is commonly used in benchmarking optimization software. However, it does not consider uncertainty estimation. A revised version of performance profile is called run-score distribution. It shows the fraction of runs above a certain normalized score. It is an unbiased estimator of the underlying distribution and more robust than average-score distribution. In Fig. 2 we observe that PML outperforms other methods almost under any condition. On the other hand, the addition of Munchausen regularization and PER are helpful for improving BCQ. The results shown here are bootstrapped with $2K$ times.

Probability of Improvement Probability of improvement is a metric which indicates how likely one method outperforms the other on a randomly selected task. This metric does not account for the size of improvement. As we can see in Fig. 3, PML is most likely to dominate among the methods we have evaluated.The results shown here are bootstrapped with 200 times.

Learning Curves Fig. 4, 5, and 6 illustrate the learning curves of all the algorithms evaluated with training time steps as the x-axis and the average episode rewards on the y-axis. Each solid line shows the average between runs, and half-transparent regions indicate the range. The results again verify the robustness of the add-on of Munchausen regularization.[4]

[4] All results are based on five runs, except for BEAR, some runs were aborted due to MuJoCo simulator feedbacks system state for large numbers or inf./NaN.

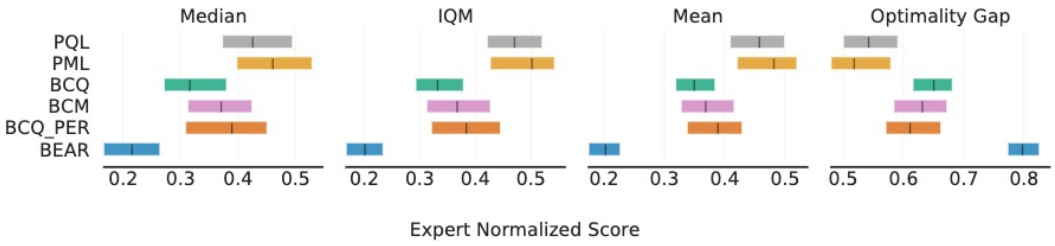

Figure 1: Aggregate Metrics

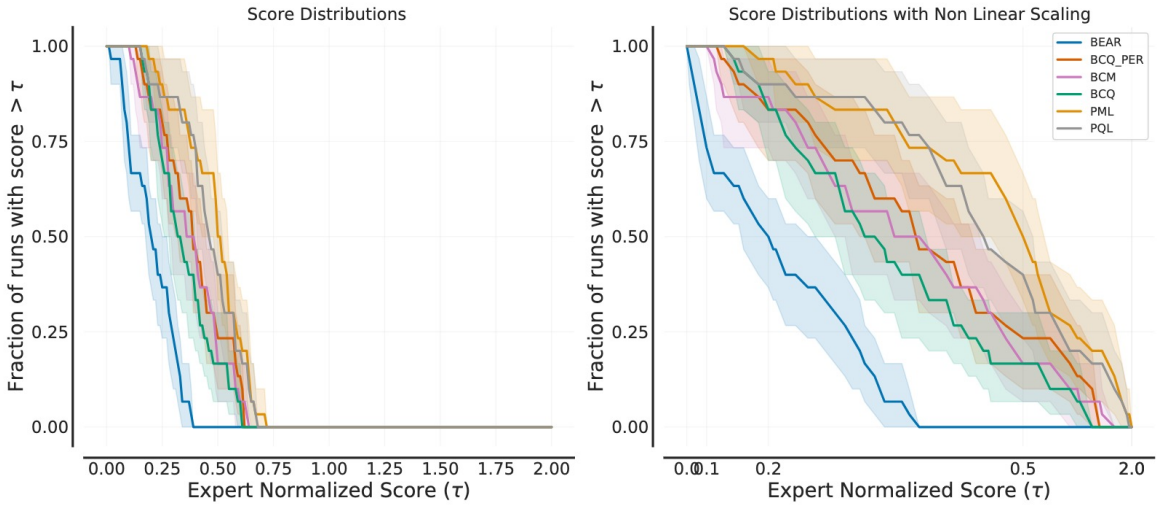

Figure 2: Score distribution with linear/non-linear scaling

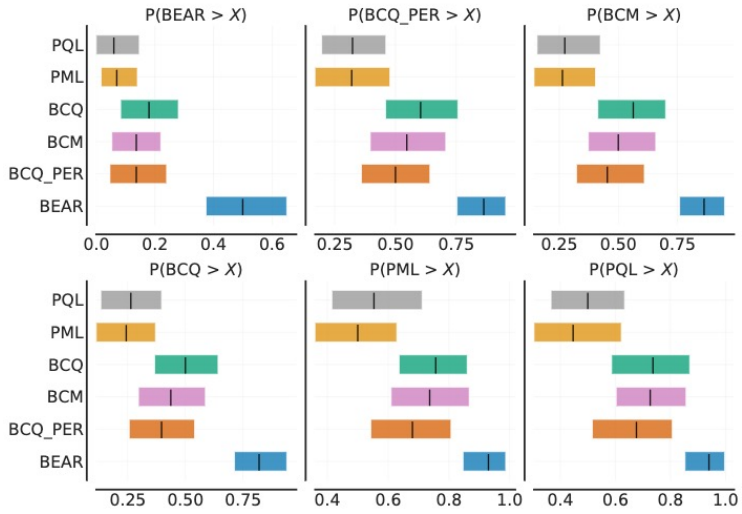

Figure 3: Probabilities of improvement

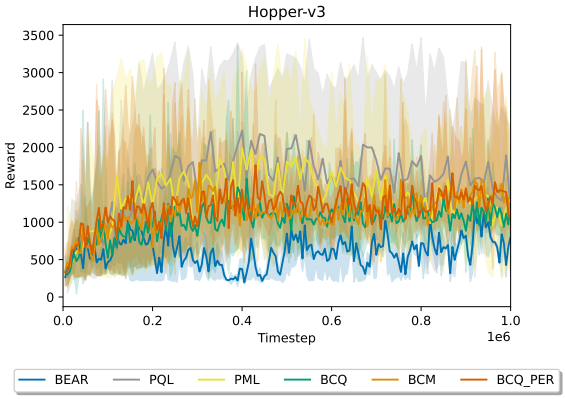

Figure 4: Learning curves of *Hopper-v3*

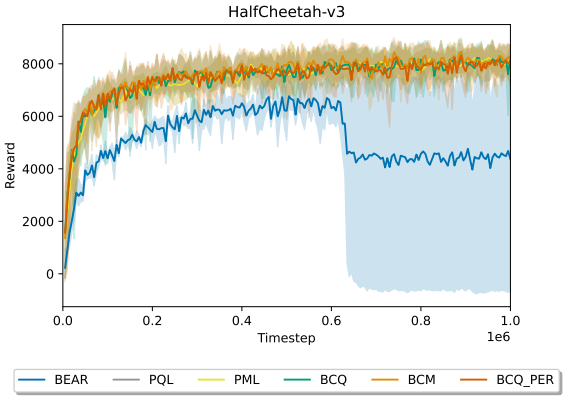

Figure 5: Learning curves of *HalfCheetah-v3*

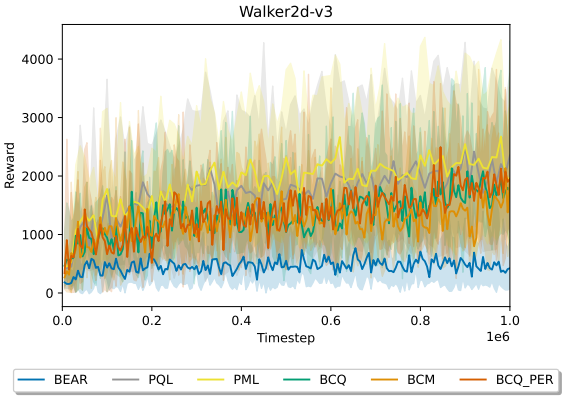

Figure 6: Learning curves of *Walker2d-v3*

4 Conclusion and Discussion

In this work, we show that Munchausen regularization is effective in improving BRL methods. It penalizes policies that are far from the previous ones. It can serve as a strong learning signal to enhance the performance of models. Moreover, prioritized replay with weighted importance sampling could also improve BRL methods with consistent Q-update. Due to the massive amount of resources required in continuous spaces DRL algorithm evaluation, usually DRL studies conduct a handful of runs ($3 \sim 10$). We use aggregate statistical metrics that consider the uncertainty to provide a more robust comparison. These results are encouraging to us to discover more opportunities to boost BRL performances with regularization approaches. We expect to implement more benchmarks and further improvements as our future work.

Acknowledgments and Disclosure of Funding

This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] Rishabh Agarwal et al. “Deep Reinforcement Learning at the Edge of the Statistical Precipice”. In: *arXiv preprint arXiv:2108.13264* (2021).

[2] Scott Fujimoto and Shixiang Shane Gu. "A Minimalist Approach to Offline Reinforcement Learning". In: *arXiv preprint arXiv:2106.06860* (2021).

[3] Scott Fujimoto, David Meger, and Doina Precup. "Off-policy deep reinforcement learning without exploration". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2052–2062.

[4] Peter Henderson et al. "Deep reinforcement learning that matters". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[5] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[6] Aviral Kumar et al. "Conservative q-learning for offline reinforcement learning". In: *arXiv preprint arXiv:2006.04779* (2020).

[7] Aviral Kumar et al. "Stabilizing off-policy q-learning via bootstrapping error reduction". In: *arXiv preprint arXiv:1906.00949* (2019).

[8] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[9] Yao Liu et al. "Provably good batch reinforcement learning without great exploration". In: *arXiv preprint arXiv:2007.08202* (2020).

[10] Tom Schaul et al. "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952* (2015).

[11] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[12] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033.

[13] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. "Munchausen reinforcement learning". In: *arXiv preprint arXiv:2007.14430* (2020).

[14] Nino Vieillard et al. "Leverage the average: an analysis of KL regularization in reinforcement learning". In: *NeurIPS-34th Conference on Neural Information Processing Systems*. 2020.