Offline RL at Multiple Frequencies

Anonymous Author(s)
Affiliation
email

Abstract: Building a large dataset for robot learning requires bridging together data from diverse environments, which gives rise to non-standard data formats that may not be considered in algorithm design. We focus on one important but often overlooked axis of variation: the frequency of data collected. These frequency differences are incompatible with the parameters of reinforcement learning algorithms: maximum trajectory lengths, discount factors, and value functions are all connected to the assumption of a fixed and given frequency. These incompatibilities are only exacerbated by the training process. We show that naively mixing data from different discretizations with reinforcement learning offline leads to unstable learning in two simulated control tasks. Guided by the intuition that data at different discretizations will propagate learned Q-values at different rates, we present a simple but effective solution to stabilize learning: N-step returns where N is scaled with the discretization. Our method leverages this finding to outperform naive mixing both in terms of absolute performance and training speed across two simulated control tasks.

Keywords: offline reinforcement learning

1 Introduction

The frequency of observations and actions is an important but often overlooked aspect of variation in robotics data. Different frequencies arise commonly when data is collected independently from different lab environments because the discretization of the controller, the sampling rate of a sensor, or the demands of a task of interest (with some tasks requiring faster feedback than others at the cost of training efficiency) may differ in each setting. Of special importance is the difference between the frequency used when collecting demonstrations and training a policy: data collection interfaces may be unintuitive for human users with large discretizations, but training a policy on this frequency of data is challenging because higher frequencies require longer training times and have been shown to be unstable during training [1]. Figure 1 illustrates how increasing the frequency of the same data results in larger training times: it requires more update steps to propagate a value function in the same physical distance as with lower frequency data.

Straightforward approaches, such as standardizing the data, fail because the same policy may be infeasible or less performant at a different discretization. While past work has looked at mixing irregularly sampled data together [2, 3] and presented solutions to the fragility of reinforcement learning with small discretizations [1], this is the first work to look at mixing data of different discretizations together in a reinforcement learning algorithm.

Intuitively, we want to mimic the "update stride" of coarser discretizations when training on finer-grained discretizations to improve the speed of learning and the consistency of q-value updates. In Figure 1, we visualize this intuition for value iteration in a simple grid environment. The agents in the environments in the bottom row can take actions of size 2, while agents in the top row can take actions of size 1. We adjust the learning rate in accordance with the action magnitude. On the right hand side, we let the value iteration algorithm compensate for the decreased action magnitude by looking ahead in provided data. Instead of the standard value iteration update, we would update

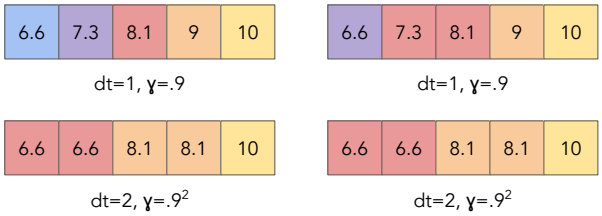

Figure 1: On the left hand side, we perform value iteration on a simple grid at two different discretizations. On the right hand side, we allow the smaller discretization to look ahead to the next two states. Modifying the update rule in this way leads to more consistent value function updates. Each squre of the grid is color-coded based on the iteration at which it's updated.

with an adapted N-step return—$V_k(s_t) = \sum_{t'=t}^{N-1} \gamma^{t'-t} r(s_{t'}, a_{t'}) + V_{k-1}(s_{t+N})$—where $N = 1$ for $dt = 2$ and $N = 2$ for $dt = 1$.

We integrate this discretization-dependent N-step update rule into the q-target calculation for an offline RL algorithm across 2 simulated control tasks. We show that simply scaling the value of N with the discretization leads to more stable training and improved final performance for Pendulum swing-up and Metaworld door-open tasks. By studying data mixing settings where there is incomplete state space coverage in the data from some discretizations, we find that a discretization-dependent N-step counteracts underestimation in naive offline mixing.

2 Related Work

Irregular Time Series. Data of varying frequencies arise whenever observations are pulled from a continuous time system. Instead of fixing a discretization, continuous time methods model a differential equation that describes the dynamics of the environment. Many approaches to consuming irregularly sampled data [4, 2, 3] model the latent state of the data-consuming process as a differential equation. Singh et al. [5] studies how this architectural change can be applied specifically to continuous control tasks. Our setting differs because we're interested in training stability when working with data where the discretization differs between sources but remains fixed within each source. More relevant to our setting is the analysis of timescale shift at test time in Gu et al. [3].

Continuous RL. We focus on control tasks learned with offline reinforcement learning, so in addition to learning a policy that can operate at multiple time scales, we also need to learn a value function at multiple time scales. Past work [6, 7, 8] uses the Hamilton-Jacobi Bellman equation to derive algorithms for estimating the value function. These formalisms have been used to create algorithms that can handle settings where the environment evolves concurrently with planning [9] and to learn to budget coarse and fine-grained time scales during training [10]. In addition to modeling challenges, continuous time introduces challenges during training as well. Tallec et al. [1] shows that deep q-learning can fail in near-continuous time settings and uses the formalisms from continuous RL to make q-learning more robust to discretization.

3 Discretization-Dependent N-Step

We present a simple q-value update rule that can be integrated into any offline reinforcement learning algorithm. At every update step, we calculate the target q-value as:

$$Q^{dt}(s_t, a_t) = \sum_{t'=0}^{\frac{N}{dt}-1} \gamma^{t'} r(s_{t+t'}, a_{t+t'}) + \gamma^{\frac{N}{dt}} Q^{dt}(s_{\frac{N}{dt}}, a_{\frac{N}{dt}})$$

In all of our experiments, we select N to be the maximum discretization size.

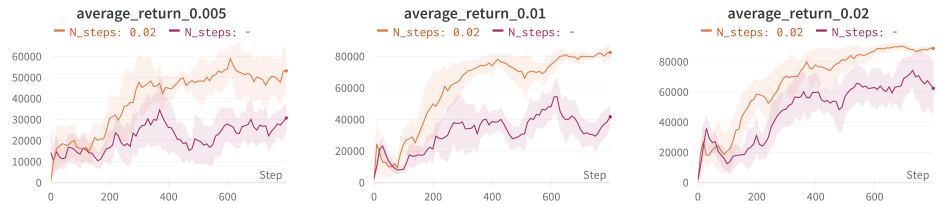

Figure 2: Within the sparse Pendulum swing-up task, discretization-dependent N-step returns improves training stability, especially for smaller discretizations.

Although this update rule could be incorporated in any offline RL algorithm, we focus on Conservative Q-Learning [11], which updates the q-value as:

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \alpha \left(\mathbb{E}_{a\sim\mu(a|s)}\left[Q(s,a)\right] - \mathbb{E}_{a\sim\hat{\pi}_{\beta}(a|s)}\left[Q(s,a)\right]\right) + \frac{1}{2}\mathbb{E}\left[\left(Q(s,a) - \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}(s,a)\right)^2\right]$$

Unless otherwise noted, states and actions are sampled from the dataset. Minimizing over $\mathbb{E}_{a\sim\mu(a|s)}\left[Q(s,a)\right]$ enforces pessimism on actions outside of the training distribution and maximizing over $\mathbb{E}_{a\sim\hat{\pi}_{\beta}(a|s)}\left[Q(s,a)\right]$ maintains optimism on the distribution of actions from the policy. We replace the Bellman operator, $\hat{\mathcal{B}}^{\pi}$, with discretization-dependent N-step returns.

4 Experiments

We demonstrate how simply scaling the value of N with dt can enable stable training when mixing discretizations offline. Our results on offline mixing span two simulated control tasks: pendulum swing-up implemented in Open AI Gym [12] and Metaworld [13] door open. We collect data for offline training by running DAU [1] and storing all of the data collected in the replay buffer. When mixing data of different discretizations offline, we build on top of the CQL implementation from Geng [14].

4.1 Pendulum

We modify the pendulum environment by changing the underlying discretization of the simulator and collect data with disctretizations of 0.02, 0.01, 0.005. We make the task sparse by creating a window of angles and velocities around 0 where the algorithm receives a constant reward. We focus on the setting where different discretizations cover different parts of the state space. To simulate this setting, we remove all transitions where the pendulum is to the right of the fixed point (or equivalently, where the value, θ, of the angle of the pendulum is negative) for data from discretizations 0.005 and 0.01. We condition the q-value and policy on the discretization. The training curves in Figure 2 demonstrate how scaling N with dt improves the training stability and final performance of the tasks for the discretizations where the state space is not fully covered. In Figure 3, we visualize the value function across the entire state action space. For discretizations without complete state-space coverage, naive mixing learns strongly negative values of θ, while scaling N with dt maintains optimism on the unexplored region.

4.2 Metaworld

We modify the Metaworld door-open environment to support different discretizations by adjusting the frame-skip. We experiment with frame skips of 1, 2, 5, and 10. In this environment, we found not conditioning on dt to give stronger performance. To evaluate performance, we look at the maximum success score across all states in the trajectory over 2 seeds. Figure 4 shows the results with mixing. In the lowest discretization, scaling N provides gains in training stability and final performance.

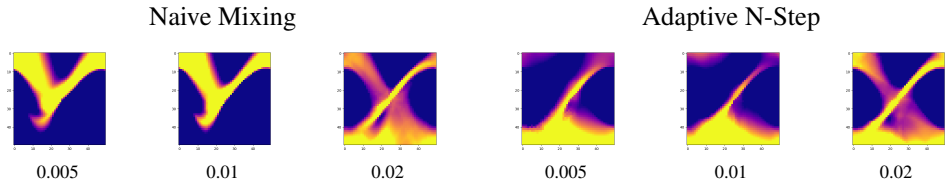

Figure 3: We visualize the q-values across the entire state-action space. The y-axis represents the angle of the pendulum and the x-axis represents the velocity. For the smaller discretizations, all data with a negative angle is removed. Discretization-dependent N-step returns maintains optimism over the unexplored region while naive mixing is pessimistic.

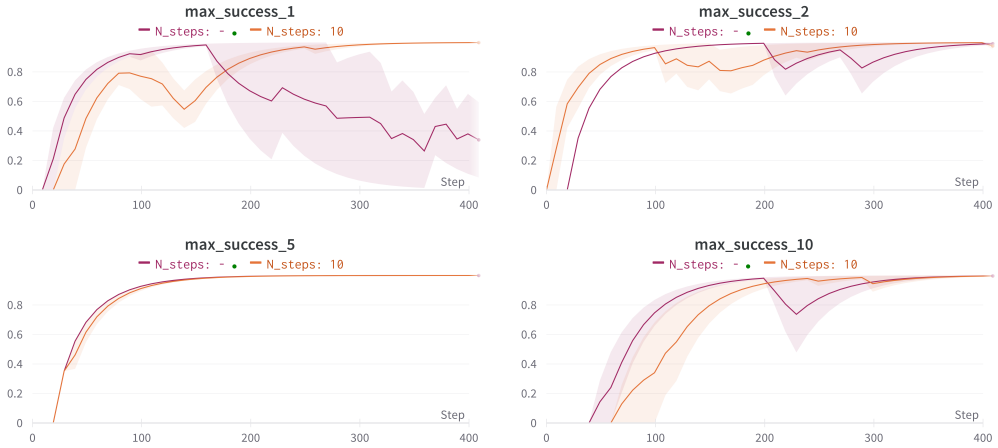

Figure 4: Within the Metaworld door open task, discretization-dependent N-step improves training stability and final performance for the smallest discretizations.

We do not directly test state space coverage, but we find that adding a discretization-dependent N-step has a similar affect on the pessimism of q-values. Figure 5 shows that, without N-step, the q-values of pairs of states sampled from the buffer and actions sampled from the policy diverge.

5 Conclusion

We study mixing data from different discretizations offline and present a simple and effective approach for improving training stability and final performance across two simulated control tasks. We find that our discretization-dependent N-step method counteracts underestimation of naive mixing. This finding contradicts the rationale behind the pessimism term in conservative Q-learning. Better understanding this relationship is an interesting question for future work.

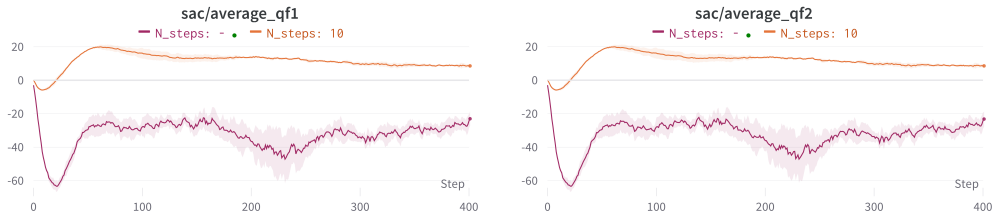

Figure 5: Discretization-dependent N-step returns prevents q-value divergence.

References

[1] C. Tallec, L. Blier, and Y. Ollivier. Making deep q-learning methods robust to time discretization. In *International Conference on Machine Learning*, pages 6096–6104. PMLR, 2019.

[2] P. Kidger, J. Morrill, J. Foster, and T. Lyons. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707, 2020.

[3] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems*, 33:1474–1487, 2020.

[4] Y. Rubanova, R. T. Chen, and D. K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

[5] S. Singh, F. M. Ramirez, J. Varley, A. Zeng, and V. Sindhwani. Multiscale sensor fusion and continuous control with neural cdes. *arXiv preprint arXiv:2203.08715*, 2022.

[6] R. Munos and P. Bourgine. Reinforcement learning for continuous stochastic control problems. *Advances in neural information processing systems*, 10, 1997.

[7] K. Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1): 219–245, 2000.

[8] J. Kim and I. Yang. Hamilton-jacobi-bellman equations for q-learning in continuous time. In *Learning for Dynamics and Control*, pages 739–748. PMLR, 2020.

[9] T. Xiao, E. Jang, D. Kalashnikov, S. Levine, J. Ibarz, K. Hausman, and A. Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. *International Conference on Learning Representations*, 2020.

[10] T. Ni and E. Jang. Continuous control on time. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022. URL https://openreview.net/forum?id=BtbG3NT4y-c.

[11] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[13] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[14] X. Geng. Cql. https://github.com/young-geng/cql, 2022.