Recurrent Open-loop Control in Offline Reinforcement Learning

Alex Lewandowski[†], Vincent Zhang[†], Dale Schuurmans[†‡]

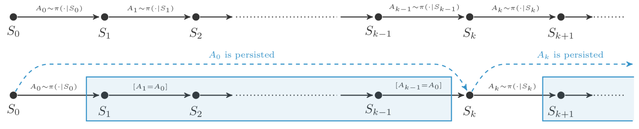
[†]Department of Computing Science, University of Alberta, [‡] Google Brain

Motivation

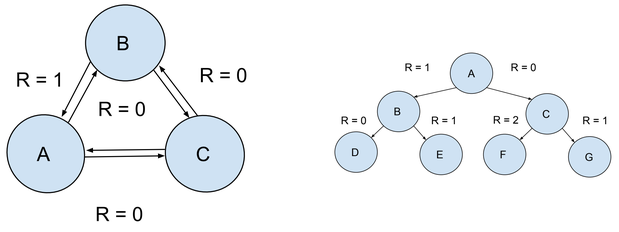
- Environments come preconfigured with temporal hyperparameters such as frame skips in Atari and discretization rates in continuous control environments.
- Temporal hyperparameters determines policy speed, the rate at which the policy interacts with the environment. Faster policies can simulate slower policies, but may be difficult to learn.
- In Offline RL, there is another benefit to learning slower policies. Action gaps may be smaller than approximation error for a fixed dataset.
- Slower policies increase action gaps, overcoming approximation error and enabling better control.

Open-loop Control in Reinforcement Learning

- Slower policies make decisions in the environment without getting feedback, in the form of states, from the environment.
- Persistent FQI (Metelli et al., 2020) learns a policy that optimally repeats actions a fixed number of times.
- Can we learn more general open-loop action selection strategies?



Denote $Q(S, A, w)$ as an open-loop action-value for a state-action pair and time $w > 0$. If $w = 0$, the action-value is closed-loop.



Left: Closed-loop action-value $Q(B, \text{left}, w = 0)$ has access to the state B. The open-loop action-value approximates the next time step $Q(B, \text{left}, w = 1) \approx Q(A, \text{left}, w = 0)$.

Right: A deterministic MDP, where persistent action selection would result in a pessimal return but a general open-loop controller can learn the optimal policy.

Recurrent Open-loop Control in Reinforcement Learning

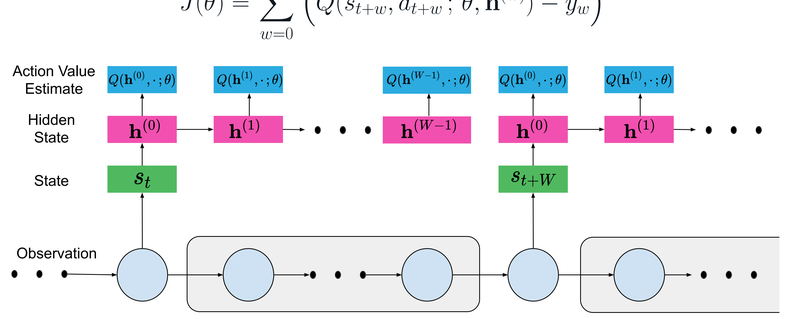
- Recurrent network is initialized by the first state in the window.
- Action-values are calculated from the hidden memory cell.
- The greedy action is selected based on the action value

For 1-step, let $y_w = r_{t+1} + \gamma \max_a Q(s_{t+w+1}, a\,;\, \theta', \mathbf{h}^{(0)})$. This is equivalent to persistent FQI if $\mathbf{h}^{(w)} = \mathbf{h}^{(0)}$ for all w.

An RNN enables us to use n-step returns, let $y_w = \gamma^{W-w} \max_a Q(s_{t+W}, a\,;\, \theta, \mathbf{h}^{(0)}) + \sum_{s=w}^{W-1} r_{t+s+1}$.

Then the FQI objective can be written with these targets as,

$$J(\theta) = \sum_{w=0}^{W-1} \left(Q(s_{t+w}, a_{t+w}\,;\, \theta, \mathbf{h}^{(w)}) - y_w \right)^2$$



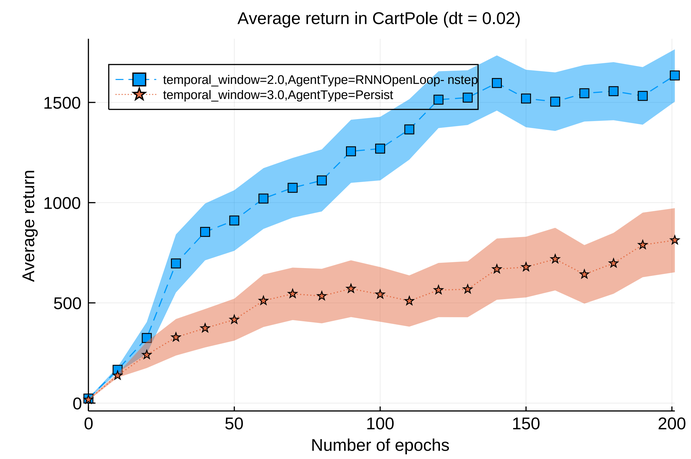
Preliminary Results - Cartpole

- Two Cartpole envs where $dt = 0.02$ with a maximum of 2000 time-steps and $dt = 0.02/4$ with a maximum of 8000 time-steps.
- Compare recurrent (1-step and n-step) against persistent FQI with window sizes $W = \{1, 2, 3, 5\}$, where $W = 1$ reduces to FQI.
- Train agents offline with ADAM on 100 episodes from a uniformly random behavior policy, for 200 epochs.
- All experiments are averaged over 100 seeds.

	Cartpole ($dt = 0.02$)			
	W=1	W=2	W=3	W=5
Persistent FQI	196.9 (26.3)	470.5 (126.8)	**541.8** (135.4)	106.9 (5.0)
RNN (1-step)	178.4 (9.8)	906.2 (173.1)	886.8 (168.0)	140.4 (38.7)
RNN (n-step)	244.4 (95.5)	**1596.8** (137.63)	733.83 (155.7)	177.2 (17.9)

	Fast Cartpole ($dt = 0.02/4$)			
	W=1	W=2	W=3	W=5
Persistent FQI	1207.6 (405.7)	1299.06 (404.68)	1301.9 (71.6)	**1579.1** (37.6)
RNN (1-step)	1021.2 (228.5)	2365.8 (620.5)	2575.6 (627.3)	1656.1 (513.3)
RNN (n-step)	1205.5 (419.6)	2722.9 (624.3)	**2998.1** (613.6)	2186.1 (425.5)

Preliminary Results - Cartpole Continued



References

Metelli, A. M., Mazzolini, F., Bisi, L., Sabbioni, L., and Restelli, M. (2020). Control frequency adaptation via action persistence in batch reinforcement learning. *arXiv:2002.06836*.