Time Delayed Forward Looking for Real-time Reinforcement Learning Control

창의IT설계 III

한수희 교수님
20130394 주동욱

#Deep_Reinforcement_Learning #Control #Realtime

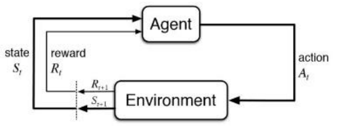
Problem

Most of successful Deep RL algorithms are mainly trained and evaluated on simulated/emulated environments. In the simulated system, they assume that there is no computational time constraint from observation to action, which is not the case in the real-time system. This project proposes Time Delayed Forward Looking RL for real-time control.

Hypothesis

By making the state of reinforcement learning algorithm be (state, previous action) pair, we can train an agent which is capable of outputting consecutive time-delayed actions for real-time control without losing performance.

Background



Reinforcement Learning

environment E, timestep t, observation x_t, action policy $\pi: \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$
transition $p(s_{t+1}|s_t, a_t)$, reward $r(s_t, a_t)$, return $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$
Bellman Equation

$$Q^\mu(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))\right]$$

Deep Deterministic Policy Gradient(DDPG)

Actor Critic, Deterministic Policy Gradient

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E}\left[\left(Q(s_t, a_t|\theta^Q) - y_t\right)^2\right]$$
$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q)$$

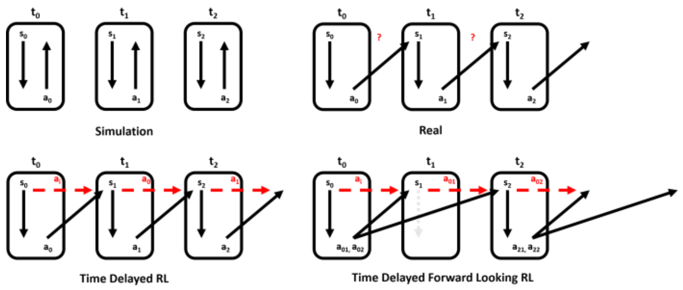
$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)}\right]$$
$$= \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s=s_t}\right]$$

Off Policy

Replay Buffer, Soft Target Optimization

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

Algorithm



$$Q([S_t, a_{(t-1)n}, \dots, a_{(t-1)N}], [a_{t0}, a_{t1}, \dots, a_{tN}])$$
$$= r_t + \gamma Q([S_{t+1}, a_{tn}, \dots, a_{tN}], [a_{(t+1)0}, \dots, a_{(t+1)N}])$$
$$= r_t + \gamma Q([S_{t+1}, a_{tn}, \dots, a_{tN}], \mu([S_{t+1}, a_{tn}, \dots, a_{tN}]))$$
$$\textit{for } n \textit{ s.t. } a_{(t-1)n} \textit{ is excuted when } S_t \textit{ is observed}$$

$$\nabla_{\theta^\mu} J_t$$
$$= \nabla_{[a_{t0}, a_{t1}, \dots, a_{tN}]} Q([S_t, a_{(t-1)n}, \dots, a_{(t-1)N}], [a_{t0}, a_{t1}, \dots, a_{tN}])$$
$$\nabla_{\theta^\mu} \mu([S_t, a_{(t-1)n}, \dots, a_{(t-1)N}])$$

$$L(\theta^Q)$$
$$= \{Q([S_t, a_{(t-1)n}, \dots, a_{(t-1)N}], [a_{t0}, a_{t1}, \dots, a_{tN}])$$
$$- r_t - \gamma Q([S_{t+1}, a_{tn}, \dots, a_{tN}], \mu([S_{t+1}, a_{tn}, \dots, a_{tN}]))\}^2$$

Component to be tested:

Time delay component
-> How many time can it be delayed?
Forward Looking component
-> How forward can it output proper actions?

More ideas developed during experiments

Steps to be Tested:

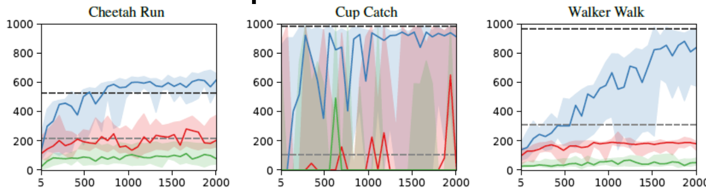
1. Naïve TD
2. Naïve FL
3. Naïve TDFL
4. Recurrent TDFL
5. Vision based TDFL for real environment

Experiments & Results

Limit Control Capable Period

	Cartpole Balance	Cartpole Swing Up	Finger Spin	Cheetah Run	Cup Catch	Walker Walk
DDPG	120ms	100ms	40ms	20ms	40ms	20ms
TDDDPG	**240ms**	**200ms**	**80ms**	**80ms**	**120ms**	**60ms**

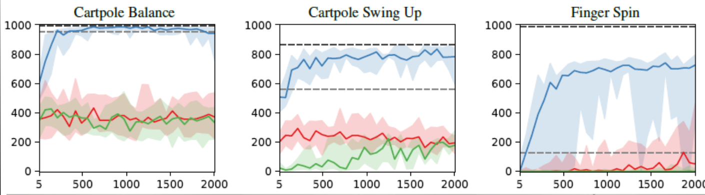
Number of multi steps



Reward curve comparison

10% more computation while saving 300% time for computation.

Real-time, Real-dynamics control



Conclusion

This project proposes Time Delayed Forward Looking Reinforcement Learning whose actor network outputs time-delayed multiple consecutive actions. Proposed model is capable of working on longer control period for enough computation time to output actions without losing performance of previous approach. This network can be applied to real-time control systems with lower performance processor and less frequent sensing by saving time for enough computation, especially for vision based control.

The project of this paper is submitted NeurIPS!