J. Appl. Numer. Optim. 5 (2023), No. 2, pp. 189-200
Available online at http://jano.biemdas.com
https://doi.org/10.23952/jano.5.2023.2.01

MULTI-STEP ACTOR-CRITIC FRAMEWORK FOR REINFORCEMENT LEARNING IN CONTINUOUS CONTROL

TIANYI HUANG[1,*], GUANGFENG CHEN[2]

[1]*Institute of Advanced Technology, Westlake Institute for Advanced Study, Hangzhou, China*
[2]*School of Engineering, Westlake University, Hangzhou, China*

Abstract. Continuous control is an important issue in control theory. It controls an agent to take action in continuous spaces for transiting from one state to another until achieving the desired goal. A useful tool for this issue is the reinforcement learning where an optimal policy is learned for the agent by maximizing the cumulative reward of the state transitions. However, most existing reinforcement learning methods consider only the one-step transition and one-step reward in each state. In this case, it is hard to recognize the information hidden in the sequence of the previous states and accurately estimate the cumulative reward. Therefore, these methods cannot learn the optimal policy both fast and effectively for continuous control. To solve this problem, in this paper, we propose a new framework, called Multi-step Actor-critic Framework (MAF) for reinforcement learning. In MAF, the convolutional deterministic policy is used to learn the information hidden in the sequence of the previous states by convolutional neural networks, and then n-step temporal difference learning is used to accurately estimate the cumulative reward by considering the rewards from n-step states. Based on an effective reinforcement learning method, TD3, the implementation of our MAF is in nTD3. The theoretical analysis and experiment illustrate that our nTD3 can learn the policy not only better but also faster than the existing RL methods for continuous control.

Keywords. Continuous control; Convolutional deterministic policy; Multi-step actor-critic; Reinforcement learning; Temporal difference learning.

2020 Mathematics Subject Classification. 68T20, 68T35.

1. Introduction

Continuous control is an important issue in control theory and is widely used in real-world applications, such as robotics [1, 2, 3, 4]. It controls an agent to take action in continuous space for transiting from one state to another until achieving the desired goal [5, 6, 7]. Reinforcement Learning (RL) is an effective tool to optimize continuous control problems [8, 9]. It learns a policy for an agent to take action by maximizing the cumulative reward of the state transitions in the control environment [17].

Many effective RL methods have been proposed to optimize continuous control problems, such as Deep Deterministic Policy Gradient (DDPG), Twin Delayed Deep Deterministic policy

[*]Corresponding author.
E-mail address: huangtianyi@westlake.edu.cn (T. Huang).

Received January 19, 2023; Accepted June 17, 2023.

gradient (TD3), and Soft Actor-Critic (SAC) [11, 12, 13, 14]. They are based on the actor-critic framework with two components, critic and actor [15, 16]. Based on the sampled state transitions and rewards, the critic estimates the action-value function (Q-function) to compute the expected cumulative reward after taking action at each state, while the actor tries to update the policy for maximizing Q-function. However, most existing RL methods just consider the reward and state transition by one-step learning, so they are hard to efficiently learn the policy and accurately estimate Q-function in continuous control.

n-step temporal difference (TD) estimation for RL considers multi-step rewards when calculating Q-function, so Q-function can be estimated more accurately [17, 18]. However, just based on n-step TD estimation, RL methods still cannot directly consider multi-step state transitions when updating the policy. In this case, the policy cannot be updated efficiently enough. To efficiently update the policy for RL in continuous control, recently, Convolutional Deterministic Policy (CDP) is proposed to consider multi-step state transitions by Convolutional Neural Networks (CNN) [19, 20]. Although we can obtain high efficiency in policy learning by CDP, we cannot accurately estimate Q-function without n-step TD estimation.

Based on the above analysis, we, in this paper, propose a new actor-critic framework called Multi-step Actor-critic Framework (MAF). MAF collects multi-step state sequences and rewards in a replay buffer. Then CDP is used to consider the recognizable information in each state sequence when updating the policy, so Q-function can be maximized efficiently. At the same time, n-step TD estimation is used to estimate Q-function. In this case, Q-function can be estimated more accurately to increase the upper limit of the cumulative reward in RL. In this way, the RL method based on MAF can well optimize the continuous control problem. MAF can be easily added to any RL method for continuous control. As an implementation of the MAF, we add it to TD3 and then construct the nTD3 method. The convergence of MAF is analyzed theoretically to prove its rationality. We also compare our nTD3 with other state-of-art RL methods for continuous control in our experiment. The experimental result shows that nTD3 can optimize the continuous action control better and faster than compared RL methods. The ablation study in our experiment shows the effectiveness of n-step TD estimation and CDP, respectively.

The remainder of this paper is organized as follows. In Section 2, n-step TD estimation, CDP, and the existing RL methods for continuous control are reviewed. In Section 3, the MAF combining CDP and n-step TD estimation is proposed and then the nTD3 method is obtained by adding the MAF to TD3. In Section 4, the convergence of the MAF is analyzed. The experiment in Section 5 illustrates the effectiveness of nTD3 in continuous control problems. The conclusion is given in Section 6.

2. Related Work

In this section, we review n-step TD estimation, CDP, and the existing RL methods for continuous action control. They are highly related to our work.

2.1. The n-step TD estimation of multi-step return.

The n-step TD estimation method is an intermediate algorithm between the Monte Carlo method, which requires a lot of resources or a long wait to simulate, and the single-step TD method, which may lead to a high error in estimation. Because n is adjustable, the n-step TD can adapt to different RL tasks. In addition, as can be seen, the single-step TD estimation method is a special case of the n-step TD estimation method. Here $n=1$.

Based on policy π, one-step return value $G^{\pi}_{t,t+1}$ can be defined as

$$G^{\pi}_{t,t+1} = r_t(\boldsymbol{s_t},\boldsymbol{a_t}) + \gamma\hat{Q}(\boldsymbol{s}_{t+1},\boldsymbol{a}_{t+1})|_{\boldsymbol{a}_{t+1}\sim\pi(\boldsymbol{s}_{t+1})},$$

and two-step return value $G^{\pi}_{t,t+2}$ can be defined as

$$G^{\pi}_{t,t+2} = r_t(\boldsymbol{s_t},\boldsymbol{a_t}) + \gamma r_t(\boldsymbol{s}_{t+1},\boldsymbol{a}_{t+1}) + \gamma^2\hat{Q}(\boldsymbol{s}_{t+2},\boldsymbol{a}_{t+2})|_{\boldsymbol{a}_{t+2}\sim\pi(\boldsymbol{s}_{t+2})}.$$

So in n-step TD estimation, the return value of $G^{\pi}_{t,t+n}$ can be defined as

$$G^{\pi}_{t,t+n} = \sum_{i=t}^{t+n-1} \gamma^{i-t} r_i(\boldsymbol{s_i},\boldsymbol{a_i}) + \gamma^n\hat{Q}(\boldsymbol{s}_{t+n},\boldsymbol{a}_{t+n})|_{\boldsymbol{a}_{t+n}\sim\pi(\boldsymbol{s}_{t+n})}. \tag{2.1}$$

Based on $G^{\pi}_{t,t+n}$, Q-function can be estimated by

$$\hat{Q}(\boldsymbol{s_t},\boldsymbol{a_t}) = \hat{Q}(\boldsymbol{s_t},\boldsymbol{a_t}) + \alpha[G_{t,t+n} - \hat{Q}(\boldsymbol{s_t},\boldsymbol{a_t})]. \tag{2.2}$$

The iterative update of the n-step estimation can be seen in Algorithm 1.

Algorithm 1 n-step estimation

```
1:  Initialize $\hat{Q}(\boldsymbol{s_t},\boldsymbol{a_t})$;
2:  Initialize $\pi$;
3:  for $episode = 0$ to $episode_{max}$ do
4:      for $t = 0$ to $T$ do
5:          Select $\boldsymbol{a}_t = \pi(\boldsymbol{s_t})$;
6:          Execute $\boldsymbol{a}_t$, then obtain $r_t$ and $\boldsymbol{s}_{t+1}$
7:          Update $G_{t,t+n}$ by (2.1);
8:          Update $\hat{Q}(\boldsymbol{s_t},\boldsymbol{a_t})$ by (2.2);
9:          Update $\pi$ by greedy strategy w.r.t $\hat{Q}$;
10:     end for
11: end for
```

From the above analysis, it can be seen that the n-step estimation method can adjust the steps for the estimation of Q-function according to the actual task. Therefore, by n-step estimation, Q-function can be calculated more accurately and efficiently.

2.2. **Convolutional deterministic policy.**

CDP is proposed to efficiently optimize continuous control for RL [20]. Let $\boldsymbol{s_i} \in \mathbb{R}^d$ be the i-th state in a collected state sequence of length n at time step t. This sequence are definded as

$$\boldsymbol{s}_{t-n+1:t} = \boldsymbol{s}_{t-n+1} \bigoplus \boldsymbol{s}_{t-n+2} \bigoplus \ldots \bigoplus \boldsymbol{s}_t,$$

where $\bigoplus$ is the concatenation operator. For each state, CDP learns the recognizable information not only in the one-step transition of this state but also in the sequence of its previous states by CNN. CDP is defined as

$$\boldsymbol{a_t} = \mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t}).$$

Then, the performance objective of CDP is

$$J(\mu_{\theta^c}) = \sum_{t=0}^{T} \int_{\mathscr{S}} \rho^{\mu}(\boldsymbol{s_t}) r(\boldsymbol{s_t}, \mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t})) d\boldsymbol{s_t}.$$

If $t-i<0$ and $i \in \{1,2,...,n\}$, $\boldsymbol{s}_{(t-i)}$ will be set as $\mathbf{0} \in \mathbb{R}^d$, where $\mathbf{0}$ is a zero vector. Based on the above performance objective, the gradient of CDP can be computed by

$$\begin{aligned}\nabla_{\theta^c} J(\mu_{\theta^c}) &= \sum_{t=0}^{T} \int_{\mathscr{S}} \rho^{\mu}(\boldsymbol{s_t}) \nabla_{\theta^c} \mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t}) \nabla_{\mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t})} Q^{\mu}(\boldsymbol{s_t}, \mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t})) d\boldsymbol{s_t} \\ &= \mathbb{E}_{\boldsymbol{s_t} \sim \rho^{\mu}}[\sum_{t=0}^{T} \nabla_{\theta^c} \mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t}) \nabla_{\mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t})} Q^{\mu}(\boldsymbol{s_t}, \mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t}))].\end{aligned}$$

For CDP in actor-critic framework, the critic estimates Q-function by the parameter set ϕ while the actor ascends the performance gradient by the parameter set θ^c as follows.

$$\begin{aligned}\delta &= r_t + \gamma Q_{\phi}(\boldsymbol{s}_{t+1}, \mu_{\theta^c}(\boldsymbol{s}_{t-n+2:t+1})) - Q_{\phi}(\boldsymbol{s_t}, \boldsymbol{a_t}), \\ \phi &\leftarrow \phi + \alpha_{\phi} \delta \nabla_{\phi} Q_{\phi}(\boldsymbol{s_t}, \boldsymbol{a_t}), \\ \theta^c &\leftarrow \theta^c + \alpha_{\theta^c} \nabla_{\theta^c} \mu_{\theta^c} \nabla_{\theta^c} Q_{\phi}(\boldsymbol{s_t}, \mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t})).\end{aligned}$$

2.3. **RL Methods for continuous action control.** RL aims to maximize a numerical reward signal by learning what to do and how to map the states to the actions in a Markov Decision Process (MDP) [21, 22, 23]. Recently, some effective RL methods have been proposed for continuous control optimization. These methods have been used in many real-world applications, such as robot control and denoising [24, 25]. The most representative RL method for continuous control is the DDPG [13]. Based on the actor-critic framework, the DDPG concurrently estimates a Q-function and learns a deterministic policy. The critic network $Q_{\phi}(\boldsymbol{s_t}, \boldsymbol{a_t})$ and its target $Q_{\phi'}(\boldsymbol{s_t}, \boldsymbol{a_t})$ are used to estimate Q-function while the network $\mu_{\theta}(\boldsymbol{s_t})$ and its target $\mu_{\theta'}(\boldsymbol{s_t})$ are used to update the deterministic policy.

Specifically, $Q_{\phi}(\boldsymbol{s_t}, \boldsymbol{a_t})$ is updated by minimizing the following mean-squared Bellman error $L(\phi, \mathscr{B})$ to estimate Q-function.

$$L(\phi, \mathscr{B}) = \mathbb{E}_{(\boldsymbol{s_t}, \boldsymbol{a_t}, r_t, \boldsymbol{s}^{t+1}) \sim \mathscr{B}} \left[(Q_{\phi}(\boldsymbol{s_t}, \boldsymbol{a_t}) - z_t)^2 \right],$$

where $\mathscr{B}$ is a collected set of transitions $(\boldsymbol{s_i}, \boldsymbol{a_i}, r_i, \boldsymbol{s}_{i+1})$ and

$$z_t = r_t + \gamma Q_{\phi'}(\boldsymbol{s}_{t+1}, \mu_{\theta'}(\boldsymbol{s}_{t+1})).$$

Accordingly, $Q_{\phi'}$ is updated by

$$\phi' \leftarrow \tau \phi + (1-\tau)\phi'.$$

In policy updating, the goal is to find a deterministic policy μ_{θ} for maximizing $Q_{\phi}(\boldsymbol{s_t}, \boldsymbol{a_t})$ as

$$\max_{\theta} \mathbb{E}_{\boldsymbol{s_t} \sim \mathscr{B}} \left[Q_{\phi}(\boldsymbol{s_t}, \mu_{\theta}(\boldsymbol{s_t})) \right].$$

Accordingly, $\mu_{\theta'}$ is updated by

$$\theta' \leftarrow \tau \theta + (1-\tau)\theta'.$$

TD3 and SAC are two important improvements of the DDPG: [11, 12]. TD3 aims to better estimate Q-function by training two critic networks, $Q_{\phi_1}(\boldsymbol{s_t}, \boldsymbol{a_t})$ and $Q_{\phi_2}(\boldsymbol{s_t}, \boldsymbol{a_t})$. Then it computes z_t by whichever of the two Q-functions has a smaller Q-value [26, 27]. SAC aims to solve the brittleness of DDPG by the approximate inference in prior off-policy maximum entropy algorithms based on soft Q-learning [12].

3. Reinforcement Learning based on Multi-step Actor-Critic Framework

In this section, MAF is proposed. Both multi-step rewards and multi-step state transitions are considered in this framework to improve the effectiveness of RL for continuous control optimization. Then, as an implementation of MAF, the nTD3 method is constructed by the combination of MAF and TD3.

3.1. Multistep actor-critic framework.

In this subsection, a new actor-critic framework called MAF is proposed by combining the n-step TD estimation with CDP in the previous actor-critic framework. With this new framework, multi-step information will be considered. In MAF, the number of state steps considered in the policy learning is defined as n_p and the number of the steps for the rewards considered in estimating Q-function is defined as n_Q. When estimating Q-function, the following objective function needs to be minimized

$$L(\phi^c, \mathfrak{B}^n) = \mathbb{E}_{(\boldsymbol{s}_{t-n_p+1:t}, \boldsymbol{a_t}, \boldsymbol{r}_{t-n_Q+1:t}, \boldsymbol{s}_{t-n_p+2:t+1}) \sim \mathfrak{B}^{(n)}} [(Q_{\phi^c}(\boldsymbol{s_t}, \mu_{\theta^c}(\boldsymbol{s}_{t-n_p+1:t})) - z_t^c)^2], \tag{3.1}$$

where $\mathfrak{B}^n$ is the set of collected state transition sequences $(\boldsymbol{s}_{i-n_p+1:t}, \boldsymbol{a_i}, \boldsymbol{r}_{t-n_Q+1}, \boldsymbol{s}_{i-n_p+2:i+1})$. The update for any $z^c_{t-n_Q+1}$ is as follows

$$z^c_{t-n_Q+1} = \sum_{i=t-n_Q+1}^{t} \gamma^{t-i} \boldsymbol{r_i}(\boldsymbol{s_i}, \boldsymbol{a_i}) + \gamma^{n_Q} \boldsymbol{Q}_{\phi'^c}(\boldsymbol{s}_{t+1}, \mu_{\theta'^c}(\boldsymbol{s}_{t-n_p+2:t+1})).$$

After the update of Q-function, the policy μ_{θ^c} is updated as

$$\max_{\theta^c} \mathbb{E}_{\boldsymbol{s}_{t-n+1:t} \sim \mathfrak{B}^{\boldsymbol{n}}} [Q_{\phi_1^c}(\boldsymbol{s_t}, \mu_{\theta^c}(\boldsymbol{s}_{t-n_p+1:t}))].$$

The algorithm of MAF is shown in Algorithm 2. It can be seen that the traditional actor-critic framework is a special case of MAF where $n_Q = n_p = 1$.

Algorithm 2 MAF

```
 1: Initialize $\theta^c$ and $\phi$;
 2: Initialize replay buffer $\mathfrak{B}^n$;
 3: for $episode = 0$ to $episode_{max}$ do
 4:     for $t = 0$ to $T$ do
 5:         Select $\boldsymbol{a}_t = \mu_{\theta^c}(\boldsymbol{s}_{t-n_p+1:t})$;
 6:         Execute $\boldsymbol{a}_t$, then obtain $r_t$ and $\boldsymbol{s}_{t+1}$;
 7:         Store transition $(\boldsymbol{s}_{t-n_p+1:t}, \boldsymbol{a_t}, \boldsymbol{r}_{t-n_Q+1:t}, \boldsymbol{s}_{t-n_p+2:t+1})$ in $\mathfrak{B}^n$;
 8:         Update $\phi$ by minimizing (3.1);
 9:         Update $\theta^c$ by (3.2);
10:     end for
11: end for
```

3.2. ***n*TD3 method.** In this subsection, a new reinforcement learning method, nTD3, for continuous control is proposed based on MAF. It uses CDP to consider multi-step states to update the policy and uses multi-step rewards to estimate Q-function. In nTD3, there are four neural networks $Q_{\phi_1}, Q_{\phi_1'}, Q_{\phi_2}, Q_{\phi_2'}$ for estimating Q-function. In this process, $Q_{\phi_1}(\boldsymbol{s_t}, \mu_{\theta^c(\boldsymbol{s}_{t-np+1:t})})$ and $Q_{\phi_2}(\boldsymbol{s_t}, \mu_{\theta^c(\boldsymbol{s}_{t-np+1:t})})$ are updated by minimizing the following objective function:

$$\begin{aligned} L(\phi_1, \mathfrak{B}^n) &= \mathbb{E}_{(\boldsymbol{s}_{t-np+1:t}, \boldsymbol{a_t}, \boldsymbol{r}_{t-n_Q+1:t}, \boldsymbol{s}_{t-np+2:t+1}) \sim \mathfrak{B}^{(n)}}[(Q_{\phi_1}(\boldsymbol{s_t}, \mu_{\theta^c}(\boldsymbol{s}_{t-n_p+1:t})) - z_t^c)^2], \\ L(\phi_2, \mathfrak{B}^n) &= \mathbb{E}_{(\boldsymbol{s}_{t-np+1:t}, \boldsymbol{a_t}, \boldsymbol{r}_{t-n_Q+1:t}, \boldsymbol{s}_{t-np+2:t+1}) \sim \mathfrak{B}^{(n)}}[(Q_{\phi_2}(\boldsymbol{s_t}, \mu_{\theta^c}(\boldsymbol{s}_{t-n_p+1:t})) - z_t^c)^2]. \end{aligned} \tag{3.2}$$

The update of any $z^c_{t-n_Q+1}$ is as follows:

$$z^c_{t-n_Q+1} = \sum_{i=t-n_Q+1}^{t} \gamma^{t-i} r_i(s_i, a_i) + \gamma^{n_Q} \min_{i=1,2} Q_{\phi_i'}(\boldsymbol{s}_{t+1}, \mu_{\theta'^c}(\boldsymbol{s}_{t-n_p+2:t+1})). \tag{3.3}$$

Q_{ϕ_i} is updated by

$$\begin{aligned} \phi_1' &\leftarrow \tau\phi_1 + (1-\tau)\phi_1', \\ \phi_2' &\leftarrow \tau\phi_2 + (1-\tau)\phi_2' \end{aligned} \tag{3.4}$$

In nTD3, there are also two action networks μ_{θ^c} and $\mu_{\theta'^c}$ to update the policy. The update for μ_{θ^c} is as

$$\max_{\theta^c} \mathbb{E}_{\boldsymbol{s}_{t-np+1:t} \sim \mathfrak{B}^c}[Q_{\phi_1}(\boldsymbol{s_t}, \mu_{\theta^c}(\boldsymbol{s}_{t-n_p+1:t}))]. \tag{3.5}$$

Accordinglyy $\mu_{\theta'^c}$ is updated by

$$\theta'^c \leftarrow \tau\theta^c + (1-\tau)\theta'^c. \tag{3.6}$$

As in TD3, exploration noise ε can be added to the CDP to explore new actions as

$$\boldsymbol{a_t} \leftarrow \mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t}) + \varepsilon, \varepsilon \sim \mathcal{N}.$$

The delay policy update in TD3 can also be used to improve the performance of nTD3. In this update, parameter e is used to control the delay step. When the Q-function estimation error is high, the update of the policy will lead to the instability of the obtained action. Therefore, the update frequency of the policy network should be lower than that of the network used to estimate the Q-function to minimize estimation errors before the policy update [11]. nTD3 algorithm is in Algorithm 3.

4. Convergence Analysis

In this section, the convergence of MAF is analyzed. The optimal Bellman operator $H^\pi_{n_Q}$ based on the multi-step actor-critic framework can be defined as [28, 29]

$$H^\pi_{n_Q} Q(\boldsymbol{s_t}, \boldsymbol{a_t}) = \max_\pi \mathbb{E}_{\boldsymbol{a}_{t:t+n_Q} \sim \pi, s_{t:t+n_Q} \sim \rho^\pi}[\sum_{i=1}^{t+n_Q-1} \gamma^{i-t} r_i(s_i, a_i) + \gamma^{n_Q} Q^\pi(\boldsymbol{s}_{t+n_Q, a_{t+n_Q}})].$$

Algorithm 3 *n*TD3 algorithm

```
 1: Initialize $\theta^c$, $\theta'^c$, $\phi_1$, $\phi_2$, $\phi_1'^c$ and $\phi_2'^c$
 2: Initialize replay buffer $\mathscr{B}^n$
 3: Initialize a random process $\mathscr{N}$ for action exploration;
 4: for $episode = 0$ to $episode_{max}$ do
 5:     for $t = 0$ to $T$ do
 6:         Select $\boldsymbol{a}_t = \mu_{\theta^c}(\boldsymbol{s}_{t-n+1:t}) + \varepsilon$, $\varepsilon \sim \mathscr{N}$;
 7:         Execute $\boldsymbol{a}_t$, then obtain $r_t$ and $\boldsymbol{s}_{t+1}$;
 8:         Store transition $(\boldsymbol{s}_{i-n_p+1:t}, \boldsymbol{a}_i, \boldsymbol{r}_{t-n_Q:1}, \boldsymbol{s}_{i-n_p+2:i+1})$ in $\mathscr{B}^n$;
 9:         Sample mini-batch of the transitions from $\mathscr{B}^n$;
10:         Update $\phi_1$ and $\phi_2$ by minimizing (3.2)
11:         if $t \mod e$ then
12:             Update $\theta^c$ by (3.5)
13:             Update $\phi_1'$ and $\phi_2'$ by (3.4)
14:             Update $\theta'^c$ by (3.6)
15:         end if
16:     end for
17: end for
```

Now we prove that $H^{\pi}_{n_Q}$ is a contractive mapping of infinite norm. The proof is as follows:

$$\begin{aligned}
&||H^{\pi}_{n_Q}Q_1(\boldsymbol{s_t},\boldsymbol{a_t}) - H^{\pi}_{n_Q}Q_2(\boldsymbol{s_t},\boldsymbol{a_t})||_\infty \\
&= \max_{\boldsymbol{s_t},\boldsymbol{a_t}} \Big| \int_S p(\boldsymbol{s_t} \to \boldsymbol{s}_{t+n_Q}|n_Q,\boldsymbol{a_t})[\gamma^{n_Q} \max_\mu Q_1(\boldsymbol{s}_{t+n_Q},\mu(\boldsymbol{s}_{t+n_Q})) \\
&\quad - \gamma^{n_Q} \max_\mu Q_2(\boldsymbol{s}_{t+n_Q},\mu(\boldsymbol{s}_{t+n_Q}))]d\boldsymbol{s_t}\Big| \\
&\le \gamma^{n_Q} \max_{\boldsymbol{s}_{t+n_Q},\boldsymbol{a}_{t+n_Q}} |[Q_1(\boldsymbol{s}_{t+n_Q},\boldsymbol{a}_{t+n_Q}) - Q_2(\boldsymbol{s}_{t+n_Q},\boldsymbol{a}_{t+n_Q})]| \\
&= ||Q_1(\boldsymbol{s}_{t+n_Q},\boldsymbol{a}_{t+n_Q}) - Q_2(\boldsymbol{s}_{t+n_Q},\boldsymbol{a}_{t+n_Q})||_\infty.
\end{aligned}$$

Therefore, $H^{\pi}_{n_Q}$ is an infinite norm contraction mapping.

Theorem 4.1. *Given a finite MDP, based on the Bellman operator $H^{\pi}_{n_Q}$, Q converges to Q^* in probability.*

Proof. Firstly, we have that the optimal Q-function is a fixed point of $\boldsymbol{H}^{\pi}_{n_Q}$ as $Q^* = \boldsymbol{H}^{\pi}_{n_Q}Q^*$ and $\boldsymbol{H}^{\pi}_{n_Q}$ is a contraction in the sup-norm, i.e., for any Q_1 and Q_2,

$$||\boldsymbol{H}^{\pi}_{n_Q}Q_1 - \boldsymbol{H}^{\pi}_{n_Q}Q_2||_\infty \le \gamma^{n_Q}||Q_1 - Q_2||_\infty.$$

Our goal is to update policy $\pi_\theta(\boldsymbol{a_t}|\boldsymbol{s_t})$ to maximize $Q(\boldsymbol{s_t},\boldsymbol{a_t})$ in each step. For this goal, we can define the function $F_{t:t+n_Q}(\boldsymbol{s_t},\boldsymbol{a_t}|\pi)$ as

$$\begin{aligned}
&F_{t:t+n_Q}(\boldsymbol{s_t},\boldsymbol{a_t}|\pi) \\
&= \sum_{i=t}^{t+n_Q-1} \gamma^{i-t} r_i(\boldsymbol{s_t},\boldsymbol{a_t}) + \gamma^{n_Q} \max_\pi Q(\boldsymbol{s}_{t+n_Q},\boldsymbol{a}_{t+n_Q}) - Q^*(\boldsymbol{s}_{t+n_Q},\boldsymbol{a}_{t+n_Q})|_{\boldsymbol{a}_{t:t+n_Q} \sim \pi(\boldsymbol{s}_{t:t+n_Q})}.
\end{aligned}$$

Then

$$
\begin{aligned}
*\mathbb{E}[F_t(\boldsymbol{s_t},\boldsymbol{a_t})|\pi] &= \int_{\mathscr{S}} p(\boldsymbol{s_t} \to \boldsymbol{s}_{t+n_Q}|n_Q,\pi)[\sum_{i=t}^{t+n_Q-1} \gamma^{i-t} r_i(\boldsymbol{s_t},\boldsymbol{a_t}) + \gamma^{n_Q} \max_{\pi} Q(\boldsymbol{s}_{t+n_Q},\boldsymbol{a}_{t+n_Q}) \\
&\quad - Q^*(\boldsymbol{s}_{t+n_Q},\boldsymbol{a}_{t+n_Q})|_{\boldsymbol{a}_{t:t+n_Q}\sim\pi(\boldsymbol{s}_{t:t+n_Q})}]d\boldsymbol{s}_{t+n_Q} \\
&= \boldsymbol{H}^{\pi}_{n_Q} Q(\boldsymbol{s_t},\boldsymbol{a_t}) - Q^*(\boldsymbol{s_t},\boldsymbol{a_t}) \\
&= \boldsymbol{H}^{\pi}_{n_Q} Q(\boldsymbol{s_t},\boldsymbol{a_t}) - \boldsymbol{H}^{\pi}_{n_Q} Q^*(\boldsymbol{s_t},\boldsymbol{a_t}).
\end{aligned}
$$

Finally, we have

$$||\mathbb{E}[F_t(\boldsymbol{s_t},\boldsymbol{a_t})|\pi]||_\infty \leq \gamma^{n_Q}||Q(\boldsymbol{s_t},\boldsymbol{a_t}) - Q^*(\boldsymbol{s_t},\boldsymbol{a_t})||_\infty$$

Therefore,

$$||\boldsymbol{H}^{\pi}_{n_Q} Q(\boldsymbol{s_t},\boldsymbol{a_t}) - \boldsymbol{H}^{\pi}_{n_Q} Q^*(\boldsymbol{s_t},\boldsymbol{a_t})||_\infty \leq \gamma^{n_Q}||Q^{\pi}(\boldsymbol{s_t},\boldsymbol{a_t}) - Q^*(\boldsymbol{s_t},\boldsymbol{a_t})||_\infty$$

Thus Q converges to the optimal Q-function Q^* in probability. □

5. Experiment

In this section, our *n*TD3 is compared with four reinforcement learning methods to illustrate the effectiveness of *n*TD3. Then the ablation study is performed to prove the advantages and rationality of combining *n*-step learning with CDP in MAF.

5.1. **Experimental Setting.** We list down the hyper-parameters of *n*TD3 in Table 1. Then we list down our network architectures.

*n*TD3 actor architecture

(n_p, state-dim)
Conv2d(in-channel=1, out-channels=512, kernel-size=(8, state-dim), stride=1)
ReLU
Max-overtime-Pooling((1, 1))
cat{(512, 128), (state-dim, 128)}
ReLU
(256, 256)
ReLU
Dropout
(256, action-dim)

*n*TD3 critic architecture

(state-dim + action-dim, 256)
ReLU
(256, 256)
ReLU
(256, 1)

5.2. **Experimental results.** In this subsection, our experiment demonstrates the effectiveness of nTD3 on the MuJoCo continuous control tasks which are interfaced through OpenAIGym [30]. Our nTD3 is compared with DDPG [13], SAC [12], and TD3 [11]. Furthermore, we combine TD3 with prioritized experience replay (TD3-EPR) as another compared method in our experiment [31]. The numerical results of our experiment are presented in Table 2. The learning curves are demonstrated in Fig. 1.

TABLE 1. Parameter setting

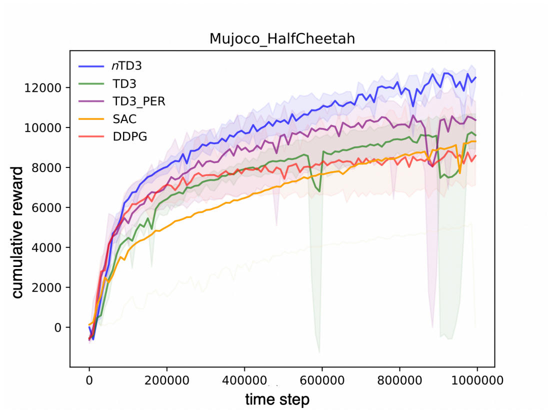
parameter	value	profile
start-timesteps	10000	Steps used by random strategies
Batch-size	256	Size of batch
e	2	Steps of delayed update
σ	0.1	Explore the variance of noise
n_p	21	Continuous steps considering state
n_Q	1 to 4	Continuous steps considering reward
γ	0.99	decay factor
τ	0.02	Update rate of label network
l_p	0.0006	Update rate of action network
l_Q	0.0003	Update rate of Q network
p_r	0.75	Probability of selecting the nearest sample

TABLE 2. Average of maximum returns over 5 trails of 10^6 time steps.

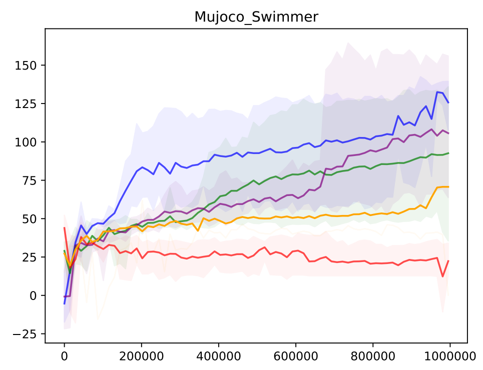
Environment	nTD3	TD3	TD3-PER	SAC	DDPG
HalfCheetah	13124.48	9940.46	10976.55	9272.49	8915.98
Swimmer	128.11	98.32	112.75	77.55	49.60
Walker2d	6056.13	4140.14	4696.98	4442.774	3702.31
Hopper	3758.30	3444.97	3019.76	3431.78	3662.24
InvDoublePendulum	9359.56	9359.76	9358.51	9359.93	0359.61

Overall, nTD3 can make an agent learn a better policy than that of the compared RL methods for continuous control while the learning speed of nTD3 is also faster than that of the compared RL methods. Concretely, in Fig. 1, the learning curves of nTD3 rise quicker than those of other methods before convergence. In most cases, nTD3 performs much better than other methods before 2×10^5 time steps. Furthermore, in most of the cases, nTD3 has a better learning result than that of other RL methods before 10^6 time steps as shown in Table 2.

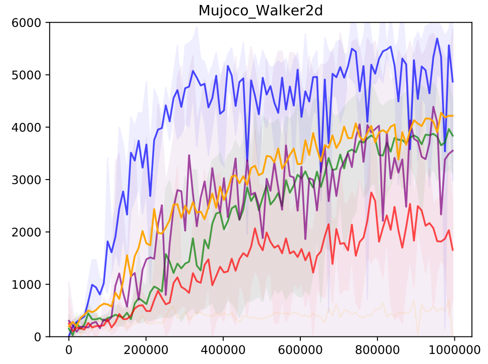
5.3. **Ablation study.** In Fig. 2, we perform the ablation study for nTD3. In this figure, CTD3 considers multi-step state transitions by CDP, but does not consider the rewards from the multi-step state transitions, e.g., $n_Q = 1$. The nTD3 and CTD3 are compared on HalfCheetah and Walker2d, respectively. For nTD3, in HalfCheetah $n_Q = 2$ and in Walker2d $n_Q = 4$. From Fig. 2, it can be seen that the learning speeds of the two methods are similar at the beginning of training. This means that the change of n_Q cannot improve the training speed. However, with



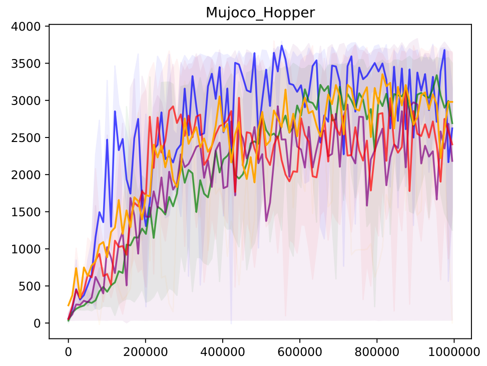
(a)



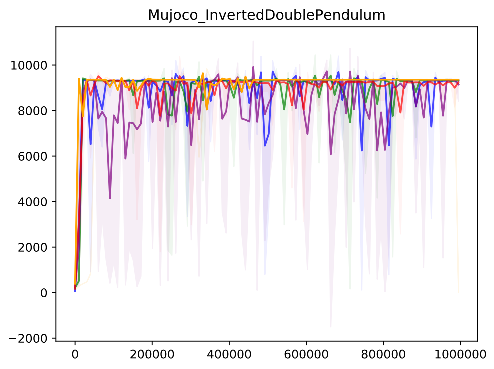
(b)



(c)



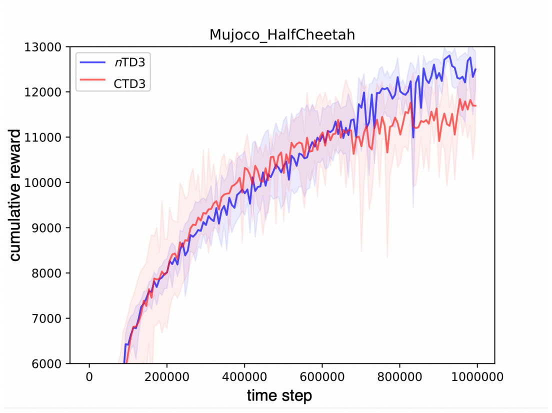
(d)



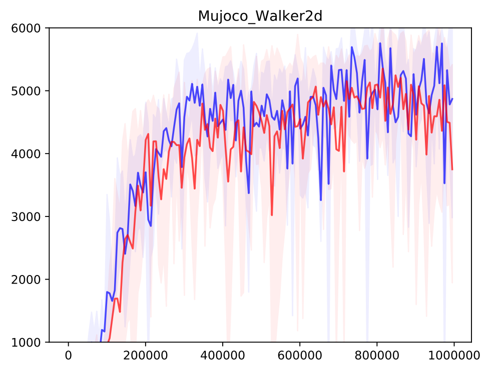
(e)

FIGURE 1. Learning curves of different RL methods on 10^6 time steps.

the increase of training times, the upper limit of the cumulative reward of nTD3 is gradually higher than that of CTD3. This illustrates that the high accuracy of Q-function estimation greatly affects whether CDP can find good actions. By adjusting n_Q, nTD3 makes Q-function be estimated more accurately. Thus the upper limit of the cumulative reward can be increased.



(a)



(b)

FIGURE 2. Learning Curve of *n*TD3 and CTD.(a)HalfCheetah; (b)Walker2d

6. Conclusion

For optimizing continuous control, RL methods need to consider multi-step state transitions to update the policy and multi-step rewards to estimate Q-function. To solve this problem, MAF is proposed for RL methods to better optimize continuous control. In MAF, CDP is used to learn the information hidden in the sequence of the previous states by CNN, and n-step TD learning is used to accurately estimate the cumulative reward by considering the rewards from n-step state transitions. Then, a new RL method, nTD3 is constructed by adding MAF to TD3. Theoretical analysis proves the rationality of MAF and our experiment shows that compared with the existing RL methods, nTD3 can optimize the continuous action control better and faster.

There are hyper-parameters that need to be tuned in nTD3. Sometimes it is difficult to find the best parameter setting. In future work, we will combine our MAF with meta RL learning [32, 33]. In this way, the hyper-parameters can be tuned automatically.

References

[1] M.J. Matarić, Reinforcement learning in the multi-robot domain, In: Robot Colonies, pp. 73-83, 1997.
[2] V. Mnih, K. Kavukcuoglu, D. Silver, et al., Human-level control through deep reinforcement learning, Nature, 518 (2015), 529–533.
[3] K. Ota, D.K. Jha, A. Kanezaki, Training larger networks for deep reinforcement learning, arXiv preprint arXiv:2102.07920, 2021.
[4] A.E. Sallab, M. Abdou, E. Perot, et al., Deep reinforcement learning framework for autonomous driving, Electronic Imaging 19 (2017), 70–76.
[5] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine learning. Neural and Statistical Classification, 1994.
[6] K.P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.
[7] A. Perrusquía, W. Yu, Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: An overview. Neurocomputing 438 (2021), 145-154.
[8] M.L. Littman, Markov games as a framework for multi-agent reinforcement learning, In: Machine Learning Proceedings 1994, pp. 157–163. Elsevier, New Brunswick, NJ, 1994.
[9] N.M. Seel, Goal-Directed Learning, Encyclopedia of the Sciences of Learning, Springer, Boston, MA, 2012.
[10] R.S Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT Press, 2018.

[11] S. Fujimoto, H.V. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, International Conference on Machine Learning, vol. 80, pp. 1587–1596, Stockholm, 2018.

[12] T. Haarnoja, A. Zhou, P, Abbeel, et al., Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, International Conference on Machine Learning, vol. 80, pp. 1861–1870, Stockholm, 2018.

[13] T. P Lillicrap, J.J. Hunt, A. Pritzel, et al., Continuous control with deep reinforcement learning, International Conference on Learning Representations, San Juan, Puerto Rico, 2016.

[14] J. Schulman, P. Moritz, S. Levine, High-dimensional continuous control using generalized advantage estimation, arXiv preprint arXiv:1506.02438, 2015.

[15] L.P. Kaelbling, M.L. Littman, A. W Moore, Reinforcement learning: A survey, J. Artificial Intelligence Res. 4 (1996), 237–285.

[16] V.R. Konda, J.N. Tsitsiklis, Actor-critic algorithms, Advances in Neural Information Processing Systems, pp. 1008–1014, Denver, 2000.

[17] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT Press, 2018.

[18] G. Tesauro, Temporal difference learning and td-gammon, Commun. ACM 38 (1995), 58–68.

[19] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM, 60 (2017), 84–90.

[20] T. Huang, M. Li, X. Qin, et al., A cnn-based policy for optimizing continuous action control by learning state sequences, Neurocomputing, 468 (2022), 286–295.

[21] L. Baird, Residual algorithms: Reinforcement learning with function approximation, Machine Learning Proceedings 1995, pp. 30–37, Elsevier, 1995.

[22] Y. Duan, X. Chen, R. Houthooft, et al., Benchmarking deep reinforcement learning for continuous control, International Conference on Machine Learning, vol. 28, pp. 1329–1338, New York, 2016.

[23] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning 8 (1992), 229–256.

[24] S. Dankwa, W. Zheng, Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent, Proceedings of the 3rd International Conference on Vision, Image and Signal Processing, pp. 1–5, 2019.

[25] T. Huang, Z. Cai, R. Li, et al., Consolidation of structure of high noise data by a new noise index and reinforcement learning, Info. Sci. 614 (2022), 206–222.

[26] H.V. Hasselt, Double q-learning, Advances in Neural Information Processing Systems, pp. 2613–2621, Vancouver, 2010.

[27] H.V. Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, New York, 2016.

[28] F.S. Melo, Convergence of q-learning: A simple proof, Institute Of Systems and Robotics, Tech. Rep, 2001.

[29] C. Ribeiro, C. Szepesvari, Q-learning combined with spreading: Convergence and results, Procs. of the ISRF-IEE International Conf. on Intelligent and Cognitive Systems (Neural Networks Symposium), 1996.

[30] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026–5033, Algarve, Portugal, 2012.

[31] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, International Conference on Learning Representations, Puerto Rico, 2016.

[32] Y. Duan, J. Schulman, X. Chen, et al., Rl^2: Fast reinforcement learning via slow reinforcement learning, arXiv preprint arXiv:1611.02779, 2016.

[33] A. Gupta, R. Mendonca, Y. Liu, et al. Meta-reinforcement learning of structured exploration strategies, Advances in Neural Information Processing Systems, 2018.