Neurocomputing 393 (2020) 83–90

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Principled reward shaping for reinforcement learning via lyapunov stability theory☆

Yunlong Dong, Xiuchuan Tang, Ye Yuan*

The Key Laboratory of Imaging Processing and Intelligent Control, School of Artificial Intelligence and Automation, State Key Laboratory of Digital Manufacturing Equipments and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

ARTICLE INFO

Article history:
Received 21 October 2019
Revised 20 January 2020
Accepted 4 February 2020
Available online 6 February 2020

Communicated by Dr. Tie-Yan Liu

Keywords:
Reinforcement learning
Principled reward shaping
Lyapunov stability theory
Stochastic approximation

ABSTRACT

Reinforcement learning (RL) suffers from the designation in reward function and the large computational iterating steps until convergence. How to accelerate the training process in RL plays a vital role. In this paper, we proposed a Lyapunov function based approach to shape the reward function which can effectively accelerate the training. Furthermore, the shaped reward function leads to convergence guarantee via stochastic approximation, an invariant optimality condition using Bellman Equation and an asymptotical unbiased policy. Moreover, sufficient RL benchmarks have been experimented to demonstrate the effectiveness of our proposed method. It has been verified that our proposed method substantially accelerates the convergence process as well as improves the performance in terms of a higher accumulated reward.

1. Introduction

Reinforcement learning (RL), especially when coupled with deep learning [20], has gained great success in beyond-human level in Atari games [25], Go game [32], cooperative agents [9], dexterous robotic manipulation [2] and multi-agent RL [6], among others. However, despite its advanced capabilities, RL suffers severe drawbacks, related to the requirement of enormous training data size, long convergence iterations and the problem of reproducibility [15]. A large number of works [13,24,31,33] have dedicated to make RL efficient and available in various scenarios. Theoretical analysis about how to indeed accelerate the training process in RL remains necessary discussions.

In deep learning, predominant optimization methodologies include Momentum [34], Adam [18], Powerball [41] and many investigations [21,28,29,42] have been devoted to the stability analysis of the optimization in real applications. In terms of the RL, reward shaping [26] has been proved can accelerate the convergence process by adding a potential function term which shall be initialized to the value function. However this kind of acceleration is implicative, cause the value function is very hard to estimate at first, especially in high dimensional and complex RL tasks. Further reward shaping and Q-value initialization are shown to be equivalent in [39]. Later based on reward shaping, [7,12,39] have extended the results, which are still limited by the weakness of reward shaping in searching the potential function. It should also be noted that the acceleration in the RL process of reward shaping has not been guaranteed theoretically.

In this paper, we apply Lyapunov stability theory [22] to tempt the RL process into maximal reward region by driving the reward to make the Lyapunov function descend in time domain. There is certainly a concern whether the shaped reward will cause the variance in optimality or the biased greedy policy in [39]. It has been proved that our proposed shaped reward function leads to convergence guarantee via stochastic approximation, an invariant optimality condition using Bellman Equation and an asymptotical unbiased policy.

Reward shaping has produced a great long-term impact on the development of RL. And reward shaping has been extended to partially observed cases [10] and model-based RL [3] to enlarge the application scope. Furthermore, [23] has proved that reward shaping has an invariant optimal policy in multi-objective RL when the vectorized reward function exists, demonstrating that reward shaping is not limited in scalar reward function further promoting the generalization ability of reward shaping. Reward shaping also plays a vital role in multi-agent RL [8]. derives a potential-based difference reward to improve the joint policy learnt by different agents. [16] combines reward shaping and hierarchies scheme to largely scale the multi-agent reinforcement systems. There are recent works focusing on the application of reward shaping in [7] and [12]. In order to achieve desirable performance, poten-

☆ This work is supported by the National Natural Science Foundation of China under Grant 91748112.

* Corresponding author.
E-mail address: yye@hust.edu.cn (Y. Yuan).

https://doi.org/10.1016/j.neucom.2020.02.008
0925-2312/

tial function needs to be carefully chosen by hand-craft. Recently [11] uses natural language to guide the reward shaping for better performance. Therefore, how to choose or design the potential function in a more principled way poses a key problem in accelerating RL convergence speed.

At the beginning, the original choice of potential function is limited in the function of the state in RL leaving along the action. To overcome the limitation above, [39] incorporates action into the potential function and proves its equivalent invariant optimality in terms of Bellman equation, resulting in a biased greedy policy through involving another action term. Yet, these two methods, [26] and [39] require a good estimation of state value function or state-action value function respectively. The 'curse of dimensionality' [4] refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces. Therefore in high dimensional and complex environments, accurate estimation of the value functions can not be guaranteed.

Lyapunov stability analysis [22] has been widely used in both linear and nonlinear dynamical systems analysis, as well as controller design for dynamical systems. [27] employs Lyapunov stability theory to explore safe RL. Xu et al. [40] applies Lyapunov theory to design an adaptive controller as an actor model in the actor-critic RL architecture for nonlinear feedback tracking control.

We make our contributions as follows:

- We leverage Lyapunov stability theory rather than handcraft engineering to guide the reward shaping in RL.
- We prove the convergence guarantee through stochastic approximation theory.
- We conduct extensive experimental results to demonstrate the effectiveness of our proposed method.

In what follows, we introduce the background and formulate the problem in Section 2. Next, we propose the Lyapunov function based method in Section 3 and demonstrate its effectiveness on benchmark examples in Section 4, and finally conclude this paper in Section 5. And the code is attached in the supplementary codes.

2. Problem formulation

2.1. Preliminaries

According to [35], RL can be modeled as a Markov Decision Process (MDP), which is denoted as a tuple $M \triangleq (\mathcal{S}, \mathcal{A}, T, R, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $T(s'|s, a)$ represents the transition probability to state $s' \in \mathcal{S}$ starting from state $s \in \mathcal{S}$ with action $a \in \mathcal{A}$, $R(s, a, s')$ is the reward function received and γ is the discount rate that gives more weights to short term reward. The goal of RL is to find an optimal policy: $\pi^* : \mathcal{S} \to \mathcal{A}$ which maximizes the accumulated reward: $\sum_{t=0}^{\infty} \gamma^t R_t(s, a, s')$, where $R_t(s, a, s')$ is the reward received on the t_{th} time-step in MDP.

It is noted that $\mathbb{E}(\cdot; \bullet)$ stands for the conditional expectation given $\bullet$. Then given a policy $\pi : \mathcal{S} \to \mathcal{A}$, the value function $V^{\pi}(s)$ is defined by:

$$V^{\pi}(s) \triangleq \mathbb{E}[R_1 + \gamma R_2 + \gamma^2 R_3 + \ldots; \pi, s], \tag{1}$$

which indicates the expected accumulated reward from a given initial state s with executing policy π. And the optimal value function can be defined as:

$$V^*(s) \triangleq \sup_{\pi} V^{\pi}(s). \tag{2}$$

In order to obtain optimal policy, an important conception Q-value under policy π [38]:

$$Q^{\pi}(s, a) \triangleq \mathbb{E}_{s' \sim T(s'|s,a)}[R(s, a, s') + \gamma V^{\pi}(s')]. \tag{3}$$

Also the optimal Q-value is defined by:

$$Q^*(s, a) \triangleq \sup_{\pi} Q^{\pi}(s, a). \tag{4}$$

It is known that optimal Q-value satisfies the Bellman equation details in [35]:

$$Q^*(s, a) = \mathbb{E}_{s' \sim T(s'|s,a)}\Big[R(s, a, s') + \gamma \max_{b \in \mathcal{A}} Q^*(s', b)\Big]. \tag{5}$$

Generally the policy can be obtained by greedy strategy which takes the corresponding action with maximum Q-value given a certain state. In some cases [1,14] the greedy strategy may not be optimal. The greedy policy can be defined by:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}}(Q^*(s, a)). \tag{6}$$

In RL the optimal Q-value $Q^*(s, a)$ is the key to solve the optimal policy which maximizes accumulated reward. Q-learning [38] is a milestone which formalizes finding the optimal Q-value in an iterative scheme by:

$$\begin{aligned}\tilde{Q}^{i+1}(s, a) \leftarrow\ & (1 - \alpha_i)\tilde{Q}^i(s, a) \\ & + \alpha_i\Big(R + \gamma \max_{b \in \mathcal{A}} \tilde{Q}^i(s', b)\Big),\end{aligned} \tag{7}$$

where $\tilde{Q}^i(s, a)$ is the estimated optimal Q-value at ith update time-step proved to converge on $Q^*(s, a)$ with probability 1 as $i \to \infty$ and α_i presents the updating rate in the learning procedure. The learning procedure of Q-learning is listed in Algorithm 1.

Algorithm 1 Q-learning.

$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \tilde{Q}^0(s, a) \leftarrow 0$. For any transition tuple $(s, a, R(s, a), s') : i \leftarrow 0$
while not convergence **do**
  $\tilde{Q}^{i+1}(s, a) \leftarrow (1 - \alpha_i)\tilde{Q}^i(s, a) + \alpha_i\{R(s, a) + \gamma \max_{b \in \mathcal{A}} \tilde{Q}^i(s', b)\}$, $i \leftarrow i + 1$
end while

As a matter of fact in complex and high dimensional cases Q-learning is indeed difficult to converge due to the 'curse of dimensionality' [4], which is the main reason of slow convergence speed in RL. While reward shaping is a well developed theory which can speed up the convergence.

2.2. Reward shaping

The slow convergence speed commonly seen in RL has been a bottleneck especially when coupled with deep learning models. Efforts have been made to find a general principled method to accelerate the convergence in RL. Ng et al. [26] proves that a potential-based reward shaping can lead to substantial reductions in learning time and do not change the optimality of the original optimal policy, which can be stated as follows:

$$R'(s, a, s') \triangleq R(s, a, s') + \gamma \Phi(s') - \Phi(s), \tag{8}$$

where $\Phi : \mathcal{S} \to \mathbb{R}$ is a to-be-defined potential function. By conducting the Bellman equation one can show that reward shaping preserves the optimality as well as invariant optimal policy. Furthermore, if $\Phi(s)$ is initialized to the optimal value function $V^*(s)$, it then can accelerate the training procedure. Yet note that $V^*(s)$ is difficult to be estimated especially in complex and high dimensional cases.

Another drawback of reward shaping in [26] is that the potential function only involves with state s without action a, which largely limits the scope of reward shaping. Wiewiora et al. [39] extends the potential function to $\Psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which involves the action a as well to have:

$$R'(s, a, s', a') \triangleq R(s, a, s') + \gamma \Psi(s', a') - \Psi(s, a), \tag{9}$$

where a' is the action taken at next time. The optimality is preserved through Bellman equation. But issue raised is the biased

greedy policy (i.e., $\pi^b(s) \neq \pi^*(s)$):

$$\pi^b(s) \triangleq \arg\max_{a\in\mathcal{A}} Q^*(s,a) + \Psi(s,a), \tag{10}$$

in which the greedy policy is biased by $\Psi(s, a)$ compared with Eq. (6).

3. Lyapunov function based reward shaping

Motivated by these pioneering works, In this paper, we shall propose a principled design of $\Psi(s, a)$, which has the following properties: (a) guaranteed convergence; (b) preserved optimality; and (c) unbiased optimal greedy policy.

We propose the Lyapunov Function Based Reward Shaping method as follows:

$$R^{lyap} = R(s,a) + \lambda\big(\gamma R(s',a') - R(s,a)\big), \tag{11}$$

where λ is a tuning parameter that weights the shaped term $\gamma R(s', a') - R(s, a)$.

From the perspective of stability analysis, we consider the MDP RL problem as an optimal control problem. The control scheme is to minimize a cost (negative reward) function $\mathcal{L}(s, a) \triangleq -R(s, a)$. We make the following assumptions:

Assumption 1. The MDP RL problem is assumed to have a maximal point (s^*, a^*) which makes the reward maximal.

Remark 1. This assumption is reasonable because when designing the cost function, the goal of any agent is defined as its achievable maximal reward.

Assumption 2. γ in MDP is assumed to be equal to 1 as $\gamma \approx 1$ in real applications.

Remark 2. Although the experimental results do not rely on this assumption, this is a standard assumption for theoretical developments [26].

Theorem 1. *Let $\mathcal{L}(s, a)$ be the Lyapunov function and assume that Assumption 1 holds. If $\forall(s', a')$, the following inequality holds*

$$\mathcal{L}(s',a') - \mathcal{L}(s,a) \leq 0, \tag{12}$$

then the state and action tuple (s, a) will converge to maximal point (s^*, a^*) asymptotically.

Proof. Due to the descent property of the Lyapunov function $\mathcal{L}(s, a)$, and the existence of a maximal point (s^*, a^*) (therefore $\mathcal{L}(s, a)$ is lower bounded), then the state and action tuple (s, a) will converge to maximal point (s^*, a^*) finally due to Lyapunov stability. □

Based on Theorem 1, we encourage the agent in MDP to make Eq. (12) hold by rewarding the shaped term defined in Eq. (11) cause maximizing $R(s', a') - R(s, a)$ equals to minimizing $\mathcal{L}(s', a') - \mathcal{L}(s, a)$ which can make Eq. (12) hold as much as possible. Indeed we can prove the convergence of the Q-learning procedure with Lyapunov Function Based Reward Shaping.

Proposition 1 (Convergence guarantee)**.** *Considering a MDP based RL problem, given the Lyapunov function based reward shaping term defined in Eq. (11), and the updating procedure given in Procedure 2, and assume that Assumptions 1 and 2 hold, then for any state action pair (s, a), the estimated state action value $\tilde{Q}^i(s, a)$ will converge to $\tilde{Q}^*(s, a)$ with probability 1 as $i \to \infty$ following the assumptions in stochastic approximation [17].*

Algorithm 2 Updating procedure.

$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \tilde{Q}^0(s,a) \leftarrow 0$. For any transition tuple $(s, a, R(s, a), s', a', R(s', a'))$, $i \leftarrow 0$
while not convergence **do**
$\tilde{Q}^{i+1}(s,a) \leftarrow (1-\alpha_i)\tilde{Q}^i(s,a) + \alpha_i\{R(s,a) + \lambda(R(s',a') - R(s,a)) + \gamma \max_{b\in\mathcal{A}} \tilde{Q}^i(s',b)\}$, $i \leftarrow i+1$
end while

Proof. Lyapunov function based shaping reward R^{lyap} will lead to a new MDP tuple denoted as $M' \triangleq (\mathcal{S}, \mathcal{A}, T, R^{lyap}, \gamma)$. In order to complete a more convenient proof, we define an operator $\mathbf{H}$ on state action value function $Q(s, a)$ over any transition tuple $(s, a, R(s, a), s', a', R(s', a'))$ under M' by:

$$(\mathbf{H}Q)(s,a) = \mathbb{E}_{s'\sim T(s,a)}\mathbb{E}_{a'\in\mathcal{A}}\left\{R^{lyap} + \gamma \max_{b\in\mathcal{A}} Q(s',b)\right\}. \tag{13}$$

It should be noted that $(\mathbf{H}Q^*)(s, a) = Q^*(s, a)$, which can be conducted from Eq. (5). Here we proof that operator $\mathbf{H}$ is a contraction operator by:

$$\begin{aligned}
\|\mathbf{H}Q_1 - \mathbf{H}Q_2\|_\infty &= \max_{s,s'\in\mathcal{S}} \mid \mathbb{E}_{s'\sim T(s,a)}\mathbb{E}_{a'\in\mathcal{A}}(\gamma \max_{b_1\in\mathcal{A}} Q_1(s',b_1) \\
&\qquad - \gamma \max_{b_2\in\mathcal{A}} Q_2(s',b_2) \mid \\
&\leq \gamma\mathbb{E}_{s'\sim T(s,a)} \mid \max_{b_1\in\mathcal{A}} Q_1(s',b_1) - \max_{b_2\in\mathcal{A}} Q_2(s',b_2) \mid \\
&\leq \gamma\mathbb{E}_{s'\sim T(s,a)} \mid \max_{b\in\mathcal{A}}(Q_1(s',b) - Q_2(s',b)) \mid \\
&\leq \gamma \max_{s'\in\mathcal{S},b\in\mathcal{A}} \mid Q_1(s',b) - Q_2(s',b) \mid \\
&= \gamma\|Q_1 - Q_2\|_\infty,
\end{aligned} \tag{14}$$

where $Q_1(s, a)$ and $Q_2(s, a)$ is two different estimated state action value function under M'.

Combine the proof in [17] and the contraction operator $\mathbf{H}$, $\tilde{Q}^i(s, a)$ will converge to $\tilde{Q}^*(s, a)$ with probability 1 as $i \to \infty$ if

1. The state space $\mathcal{S}$ and action space $\mathcal{A}$ are finite;
2. $\sum_i \alpha_i = \infty$ and $\sum_i \alpha_i^2 < \infty$, where α_i is the updating rate in Procedure 2;
3. the variance of $\{R_t\}$ is bounded.

The details about proof is attached in the Appendix Theorem 2. □

Moreover, we disccuss the optimality and asymptotical unbiased property of our proposed method.

Proposition 2 (Optimality preserving)**.** *Given R^{lyap} in Eq. (11) and assume Assumption 2 holds, then the reward shaping preserves optimality.*

Proof. For the original MDP M (with $\gamma = 1$), the Bellman equation can be written by:

$$Q_M^*(s,a) = \mathbb{E}_{s'\sim T(s'|s,a)}\big[R(s,a) + V_M^*(s')\big], \tag{15}$$

where $Q_M^*(s, a), V_M^*(s')$ denote the optimal Q-value function and value function respectively under MDP M.

With similar idea to the proof of Section 4.1 in [39], we can derive

$$Q_{M'}^*(s,a) = Q_M^*(s,a) - \lambda R(s,a), \tag{16}$$

where $Q_{M'}^*(s, a)$ satisfies the Bellman equation as well. □

Proposition 3 (Unbiased greedy policy)**.** *Given R^{lyap} in Eq. (11) and assume Assumption 2 holds, then it will lead to an asymptotically unbiased optimal greedy policy as in Eq. (6).*

Proof. The greedy policy in M' is stated as follows:

$$\begin{aligned}
\pi_{M'}^*(s) &= \arg\max_{a\in\mathcal{A}}(Q_{M'}^*(s,a)) \\
&= \arg\max_{a\in\mathcal{A}}(Q_M^*(s,a) - \lambda R(s,a)).
\end{aligned} \tag{17}$$

Table 1
Details of the reward functions and experimented environments in discrete cases. The reward representation in continuous cases is complicated and not intuitive, here we illustrate our modification compared with official reward detail in OpenAI gym [5]. The specific meaning of parameters in reward functions can be found in the official reward function. And `ReLU(•)` stands for the rectified linear unit which is `max(0, •)`.

Env / Reward	function $\mathcal{A}$	function $\mathcal{B}$	function $\mathcal{C}$
MountainCar	`-0.6 + position`	$20 \times e^{(\frac{\texttt{position}+2.1}{1.8})^2} \times (\texttt{velocity})^2$	$(10 \times \texttt{velocity})^2$
CartPole	$\cos(\theta)^2$	$\frac{\pi}{2} - \lvert\theta\rvert$	$\cos(\theta)$
Acrobot	`ReLU(1-position) × -0.25`	`ReLU(1-position)`^{2} `× -0.1`	`ReLU(1-position)`^{3} `× -0.02`

Table 2
Performance of the proposed method for both continuous and discrete cases. All the metrics in terms of accumulated reward here are sampled from ten repetitive experiments. In total, five different environments experimented among three different type of reward functions within discrete cases (denoted as $\mathcal{A}, \mathcal{B}, \mathcal{C}$ respectively) are listed below. The $\frac{1}{5}$ before min, avg, max stand for the metrics in the last $\frac{1}{5}$ episode. The large margin over averaged reward and last $\frac{1}{5}$ episode averaged reward metrics have demonstrated that Lyapunov function based reward shaping can consistently improve the convergence and performance. In terms of max and $\frac{1}{5}$ max metrics, our proposed method can substantially improve the maximum margin among different environments and reward functions.

Env / Reward	min	avg	max	$\frac{1}{5}$ min	$\frac{1}{5}$ avg	$\frac{1}{5}$ max
CartPole/$\mathcal{A}$ (w/ Lyapunov)	7.88	**168.88**	**199.97**	86.12	**189.70**	**199.97**
CartPole/$\mathcal{A}$ (w/o Lyapunov)	**8.84**	136.49	199.96	**122.49**	187.76	199.95
CartPole/$\mathcal{B}$ (w/ Lyapunov)	**11.68**	**257.23**	**312.97**	167.11	297.21	**312.97**
CartPole/$\mathcal{B}$ (w/o Lyapunov)	11.66	221.03	312.40	**209.87**	**300.78**	312.40
CartPole/$\mathcal{C}$ (w/ Lyapunov)	**8.93**	**170.00**	**199.99**	**146.49**	**198.13**	**199.99**
CartPole/$\mathcal{C}$ (w/o Lyapunov)	7.94	134.85	199.98	116.67	190.39	199.97
MountainCar/$\mathcal{A}$ (w/ Lyapunov)	**-240.64**	**-167.47**	**-82.98**	**-232.51**	**-139.40**	**-83.21**
MountainCar/$\mathcal{A}$ (w/o Lyapunov)	-248.00	-212.45	-88.35	-237.38	-201.42	-88.35
MountainCar/$\mathcal{B}$ (w/ Lyapunov)	0.02	**6.76**	**14.55**	**0.03**	**8.61**	**14.55**
MountainCar/$\mathcal{B}$ (w/o Lyapunov)	0.02	0.41	2.47	0.02	0.40	2.05
MountainCar/$\mathcal{C}$ (w/ Lyapunov)	0.00	**19.47**	**28.66**	**5.19**	**21.83**	**28.25**
MountainCar/$\mathcal{C}$ (w/o Lyapunov)	**0.05**	1.01	7.01	0.07	0.99	6.84
Acrobot/$\mathcal{A}$ (w/ Lyapunov)	**-147.68**	**-75.56**	**-41.15**	**-112.27**	**-62.89**	**-41.15**
Acrobot/$\mathcal{A}$ (w/o Lyapunov)	-149.14	-113.15	-65.56	-141.54	-109.58	-65.56
Acrobot/$\mathcal{B}$ (w/ Lyapunov)	-177.18	**-67.60**	**-41.10**	**-95.62**	**-55.50**	**-41.10**
Acrobot/$\mathcal{B}$ (w/o Lyapunov)	**-176.50**	-104.22	-56.81	-143.18	-98.14	-64.63
Acrobot/$\mathcal{C}$ (w/ Lyapunov)	**-103.96**	**-41.40**	**-24.01**	**-55.42**	**-34.11**	**-24.01**
Acrobot/$\mathcal{C}$ (w/o Lyapunov)	-104.76	-70.25	-39.52	-96.48	-67.88	-39.52
Ant (w/ Lyapunov)	0.091	**0.646**	0.851	**0.788**	**0.825**	**0.851**
Ant (w/o Lyapunov)	**0.094**	0.573	**0.799**	0.671	0.758	0.799
Hopper (w/ Lyapunov)	**0.009**	**0.274**	**0.584**	**0.520**	**0.548**	**0.584**
Hopper (w/o Lyapunov)	0.008	0.169	0.389	0.313	0.339	0.389
HalfCheetah (w/ Lyapunov)	0.000	**0.285**	**0.751**	**0.489**	**0.639**	**0.751**
HalfCheetah (w/o Lyapunov)	0.000	0.206	0.444	0.352	0.415	0.444

Substitute Eq. (15) into Eq. (17) and for simplification noting $\mathbb{E}_{s' \sim T(s'|s,a)}$ as $\mathbb{E}$ we will have

$$\pi^*_{M'}(s) = \arg\max_{a \in \mathcal{A}} \mathbb{E}\Big[(1-\lambda)R(s,a) + V^*_M(s')\Big]. \tag{18}$$

To be convenient, s, a, s' are all implicit with time-step t. Combine Eqs. (18) and (6), successively we obtain (based on Theorem 1 and the convergence fact that $\lim_{t\to\infty} s = s'$)

$$\lim_{t\to\infty} \pi^*_{M'}(s) = \arg\max_{a \in \mathcal{A}} \mathbb{E}\Big[(1-\lambda)R(s,a) + V^*_M(s)\Big]$$

$$\lim_{t\to\infty} \pi^*_{M}(s) = \arg\max_{a \in \mathcal{A}} \mathbb{E}\Big[R(s,a) + V^*_M(s)\Big].$$

As $V^*_M(s)$ is independent of action a, we then prove the asymptotical unbiased policy:

$$\lim_{t\to\infty} (\pi^*_{M'}(s) - \pi^*_M(s)) = 0. \qquad \square$$

4. Experiments

The proposed method can be applied to RL by only substituting the reward function. Here we evaluate our method on two state-of-the-art methods, DQN [25] for discrete cases and PPO2 [31] for continuous cases respectively. In discrete cases the action space is discrete in a certain countable set such as $a \in \{1, 0, -1\}$, while continuous cases make the action space continuous and uncountable in a range such as $a \in (-1, 1)$. Both DQN[1] and PPO2[2] codes are imported from public online repository to insure the proposed method's independence in implemental specification. The environments experimented are from OpenAI gym [5]. The continuous cases are used from Mujoco [37]. In order to remove the randomness introduced during the initialization of network parameters in DQN, PPO2 and other random factors, each experiment is repeated 10 times. We visualize average curves in figures and use the discounted accumulated reward as metric. The reward functions and environments during the experiments are listed in details in Table 1.

[1] https://github.com/keras-rl/keras-rl.
[2] https://github.com/openai/baselines.

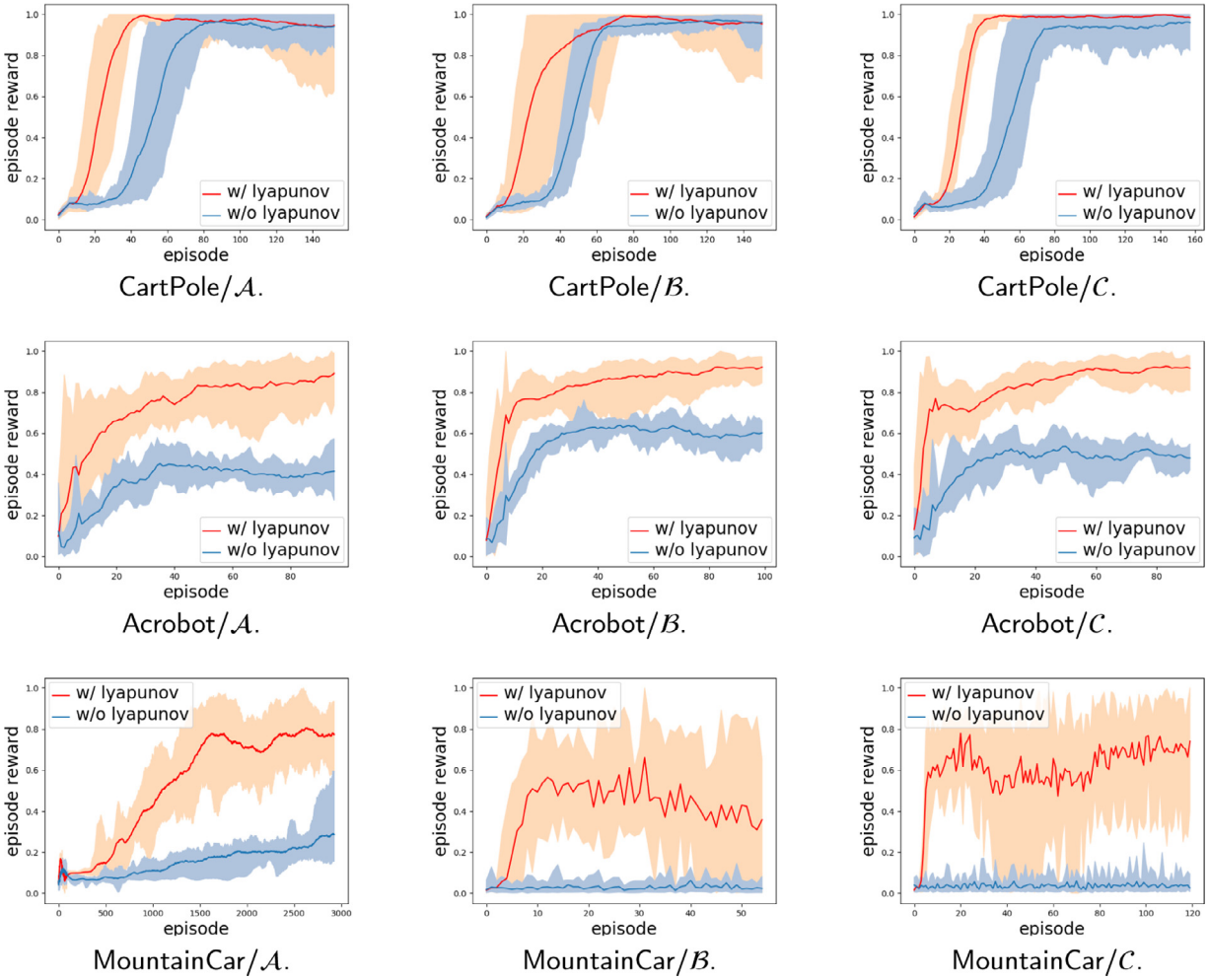

Fig. 1. Experimental results on different discrete environments with different reward functions. All the figures here are sampled from 10 repetitive experiments to eliminate the impact imposed by random factors. The name under each figure stands for environment name / reward function, details in supplementary codes. We use DQN [25] as the general solver for discrete cases. Red line and blue line indicate the performance w/ (with) Lyapunov function based reward shaping and w/o (without) respectively. The episode reward is normalized.

4.1. Discrete case

We apply DQN as a general solver in discrete cases. DQN uses a neural network function iteratively to approximate Q-value for each state and action pair which is designed for discrete case. The extensive experimental results are shown in Fig. 1 and listed statistically in Table 2. The figures and numerical results demonstrate that Lyapunov function based reward shaping can accelerate the convergence speed of RL and improve the performance under different environments and reward functions.

4.2. Continuous case

PPO2 is adopted as the general solver for continuous cases. Because in continuous cases the action space $\mathcal{A}$ is uncountable, the brute search of greedy policy is infeasible and time-consuming to loop over all the actions inside $\mathcal{A}$. PPO2 is a policy gradient [36] and actor-critic [19] based method which uses gradient method to optimize the parametric policy. And also PPO2 has benefited a lot from previous method TRPO [30] to punish exploring new updated policy far away from current policy via $\mathcal{KL}$ divergence. In practice PPO2 has been demonstrated state-of-the-art method in solving continuous RL cases.

Here we apply PPO2 along with our proposed Lyapunov function based reward shaping to verify the effectiveness. The extensive experimental results are shown in Fig. 2 and listed statistically in Table 2. The results strongly demonstrate the effectiveness of our proposed Lyapunov function based reward shaping in different continuous RL cases which has more satisfying learning speed and performance.

4.3. Effectiveness of parameter λ

Here we further explore the effectiveness of tuning parameter λ in Eq. (11) to investigate the sensitivity of λ which can be of great essential in conducting Lyapunov function based reward shaping. We vary the parameter λ in a certain range to conduct the experiments.

The results are summarized in Fig. 3. λ reflects the extent of impact of Lyapunov function based reward shaping term. When setting λ to 0, it does not change anything of the original MDP. Experiments are conducted in CartPole, Acrobot and MountainCar environments by only varying λ to different values with setting other configurations unchanged. The results strongly reveal that the performance can be guaranteed within a wide range of λ, which means the efforts in tuning parameters can be liberated.

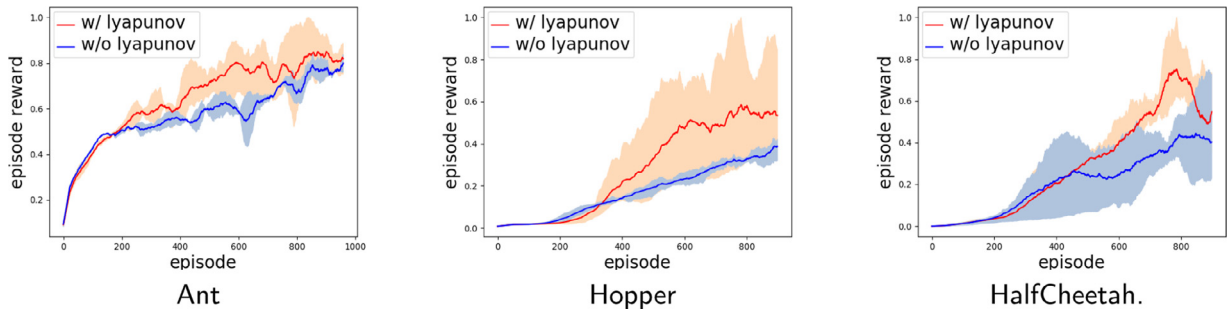

Fig. 2. Experimental results on three different continuous environments. All the figures here are sampled from 10 repetitive experiments to eliminate the impact imposed by random factors. We use PPO2 [31] as the general solver for continuous cases. Red line and blue line indicate the performance w/ (with) Lyapunov function based reward shaping and w/o (without) respectively. The episode reward is normalized.

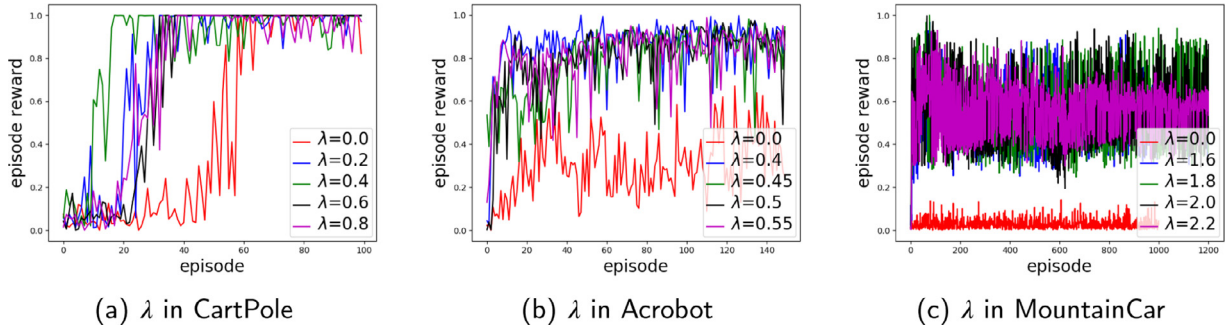

Fig. 3. The effectiveness of tuning parameter λ in the Lyapunov function based reward shaping is tested in CartPole, Acrobot and MountainCar environments with different value of λ, while keeping other configurations the same.

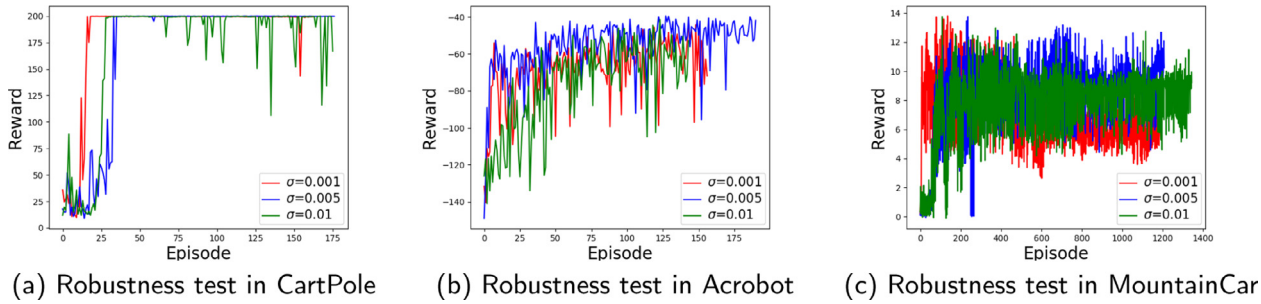

Fig. 4. The robustness test of the proposed method in three different environments. The noise is generated from a zero-mean Gaussian distribution with different standard variations σ.

4.4. Robustness to noise

Next we consider the robustness of the proposed method, as the reward received can be very noisy in real applications. We manually generate additive gaussian noise with different standard variations to the reward received in the environments as follows:

$$R_{noise} = R + \mathcal{N}(0, \sigma),$$

where R is the original reward, $\mathcal{N}$ indicates zero mean Gaussian noise with standard variance σ and R_{noise} stands for the noisy reward used in the training process. In real applications, noise is unavoidable which will mislead and confuse the communication of agent with environments. It is essential to analyze the proposed method's resistivity to noise of varying degrees. In experiments, we vary σ within a certain range while keeping other configurations unchanged. The results are shown in Fig. 4. The experimental results demonstrated that Lyapunov function based reward shaping is robust to noise in a certain satisfying range, which promotes its application in real noisy circumstances.

4.5. Limitations

The proposed method guarantees the convergence but lacks the theoretical analysis about the convergence speed. Without the theoretical analysis on the convergence speed, the choice of parameter λ in Eq. (11) would be tricky, which could introduce some other uncertainties. Our future work will focus on how to provide convergence speed of our proposed method.

5. Conclusion

In this paper, we proposed a novel reward shaping method based on Lyapunov stability analysis by encouraging the agent to reach the region of maximal reward as far as possible. The proposed method is guaranteed to be invariant in optimality and asymptotically unbiased greedy policy. And we give theoretical convergence proof of our proposed method via the early established stochastic approximation theorem. Our proposed Lyapunov function based reward shaping effectively liberates the efforts in searching well-defined potential function because potential func-

tion here is exactly chosen to be the reward function. Also we have conducted extensive experiments for both discrete and continuous RL public benchmarks to verify our proposed method. Both theoretical proof and experimental results strongly demonstrate the effectiveness of our proposed method, which can substantially fasten the convergence speed and promote the performance in RL. In the future work, theoretical analysis about the convergence speed shall be carefully discussed to further guide the designation of reward shaping.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Ye Yuan: Supervision.

Acknowledgment

We would like to thank Cheng for useful discussions and help with the paper revision.

Appendix A

Lemma 1. *Define a random process* $\{\Delta_i(x)\}$ *taking values in* $x \in \mathbb{R}^n$ *as:*

$$\Delta_{i+1}(x) \leftarrow (1-\alpha_i)\Delta_i(x) + \alpha_i F_i(x),$$

Where $\{F_i(x)\}$ *is also a random process over x.* $\Delta_i(x)$ *converges to zero with probability 1 as* $i \to \infty$ *under following assumptions:*

1. $\sum_i \alpha_i = \infty$ *and* $\sum_i \alpha_i^2 < \infty$;
2. $\|\mathbb{E}(F_i(x))\|_W \le \gamma \|\Delta_i(x)\|_W$, *with* $\gamma < 1$;
3. $\mathbf{var}\{F_i(x)\} \le C(1 + \|\Delta_i(x)\|_W^2)$, *for some constant value* $C > 0$.

It should be noted that $\|\cdot\|_W$ *defines the maximum norm weighted by W.*

Proof. The details about the proof can be seen in [17]. □

Theorem 2. *Given the MDP with Lyapunov Function Based Reward Shaping denoted as* $M' \triangleq (\mathcal{S}, \mathcal{A}, T, R^{lyap}, \gamma)$ *where* R^{layp} *is defined in Eq. (11). Given the following update rule*

$$\begin{aligned}\tilde{Q}^{i+1}(s,a) \leftarrow\ & (1-\alpha_i)\tilde{Q}^i(s,a) \\ & + \alpha_i\Big[\underbrace{R(s,a) + \lambda(R(s',a') - R(s,a))}_{R^{lyap}} \\ & + \gamma \max_{b\in\mathcal{A}} \tilde{Q}^i(s',b)\Big],\end{aligned} \tag{19}$$

where i indicates the updating ith *step along the learning procedure.* $\tilde{Q}^i(s,a)$ *will converge to the optimal state action value function* $Q^*(s,a)$ *with probability 1 as* $i \to \infty$ *under following assumptions:*

1. *The state space* $\mathcal{S}$ *and action space* $\mathcal{A}$ *are finite;*
2. $\sum_i \alpha_i = \infty$ *and* $\sum_i \alpha_i^2 < \infty$, *where* α_i *is the updating rate in Algorithm 2;*
3. *the variance of* $\{R^{lyap}\}$ *is bounded.*

Proof. Subtract optimal state action value function $Q^*(s,a)$ from both sides of Eq. (19) and define $\Delta_i(s,a) \triangleq \tilde{Q}^i(s,a) - Q^*(s,a)$ to yield

$$\begin{aligned}\Delta_{i+1}(s,a) \leftarrow\ & (1-\alpha_i)\Delta_i(s,a) \\ & + \alpha_i\Big[\underbrace{R(s,a) + \lambda(R(s',a') - R(s,a))}_{R^{lyap}} \\ & + \gamma \max_{b\in\mathcal{A}} \tilde{Q}^i(s',b) - Q^*(s,a)\Big].\end{aligned} \tag{20}$$

Then we define

$$\begin{aligned}F_i(s,a) \triangleq\ & \underbrace{R(s,a) + \lambda(R(s',a') - R(s,a))}_{R^{lyap}} \\ & + \gamma \max_{b\in\mathcal{A}} \tilde{Q}^i(s',b) - Q^*(s,a).\end{aligned}$$

Combining the operator **H** defined in Eq. (13) we can derive the expectation of $F_i(s,a)$ as follows:

$$\begin{aligned}\mathbb{E}(F_i(s,a)) &= \mathbb{E}_{s'\sim T(s,a)}\mathbb{E}_{a'\in\mathcal{A}}\Big[\underbrace{R(s,a) + \lambda(R(s',a') - R(s,a))}_{R^{lyap}} \\ &\quad + \gamma \max_{b\in\mathcal{A}} \tilde{Q}^i(s',b) - Q^*(s,a)\Big] \\ &= (\mathbf{H}\tilde{Q}^i)(s,a) - Q^*(s,a).\end{aligned}$$

Based on the fact that $(\mathbf{H}Q^*)(s,a) = Q^*(s,a)$ we will have

$$\mathbb{E}(F_i(s,a)) = (\mathbf{H}\tilde{Q}^i)(s,a) - (\mathbf{H}Q^*)(s,a).$$

Using the contraction property of **H** in Eq. (14), we can easily get

$$\|\mathbb{E}(F_i(s,a))\|_\infty \le \gamma \|\Delta_i(s,a)\|_\infty.$$

Moreover consider the variance of $F_i(s,a)$,

$$\begin{aligned}\mathbf{var}\Big[F_i(s,a)\Big] &= \mathbb{E}\Big[\Big(\underbrace{R(s,a) + \lambda(R(s',a') - R(s,a))}_{R^{lyap}} \\ &\quad + \gamma \max_{b\in\mathcal{A}} \tilde{Q}^i(s',b) - Q^*(s,a) - \mathbb{E}(F_i(s,a))\Big)^2\Big] \\ &= \mathbb{E}\Big[\Big(\underbrace{R(s,a) + \lambda(R(s',a') - R(s,a))}_{R^{lyap}} \\ &\quad + \gamma \max_{b\in\mathcal{A}} \tilde{Q}^i(s',b) - (\mathbf{H}\tilde{Q}^i)(s,a)\Big)^2\Big] \\ &= \mathbf{var}\Big[\underbrace{R(s,a) + \lambda(R(s',a') - R(s,a))}_{R^{lyap}} \\ &\quad + \gamma \max_{b\in\mathcal{A}} \tilde{Q}^i(s',b)\Big].\end{aligned}$$

As long as the assumption that $\mathbf{var}(R^{lyap})$ is bounded holds, it can be clearly verified that

$$\begin{aligned}\mathbf{var}\Big[(F_i(s,a)\Big] &= \mathbf{var}\Big[R^{lyap} + \gamma \max_{b\in\mathcal{A}} \tilde{Q}^i(s',b)\Big] \\ &\le C(1 + \|\Delta_i(s,a)\|_W^2)\end{aligned}$$

for some constant value $C > 0$ and weighted matrix W, which is in agreement with [17].

Finally, combining Lemma 1 and Theorem 2 on the condition that their assumptions hold, $\Delta_i(s,a)$ will converge to zero with probability 1 as $i \to \infty$, *i.e.*, $\tilde{Q}^i(s,a)$ will converge to the optimal state action value $Q^*(s,a)$ with probability 1 as $i \to \infty$. □

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.neucom.2020.02.008.

References

[1] B.H. Abed-alguni, M.A. Ottom, Double delayed q-learning, Int. J. Artif. Intell. 16 (2) (2018) 41–59.

[2] M. Andrychowicz, B. Baker, M. Chociej, et al., Learning Dexterous In-hand Manipulation, International Journal of Robotics Research 39 (1) (2020) 3–20.
[3] J. Asmuth, M.L. Littman, R. Zinkov, Potential-based shaping in model-based reinforcement learning., in: Proceedings of the AAAI, 2008, pp. 604–609.
[4] R. Bellman, Dynamic Programming, Courier Corporation, 2013.
[5] G. Brockman, V. Cheung, L. Pettersson, et al., Openai Gym, arXiv preprint arXiv:1606.01540 (2016).
[6] Z. Cao, C. Lin, Hierarchical Critics Assignment for Multi-agent Reinforcement Learning, arXiv preprint arXiv:1902.03079 (2019).
[7] S. Devlin, D. Kudenko, Dynamic potential-based reward shaping, in: Proceedings of the 11th International Conference on Autonomous Agents and MultiAgent Systems, 2012, pp. 433–440.
[8] S. Devlin, L. Yliniemi, D. Kudenko, K. Tumer, Potential-based difference rewards for multiagent reinforcement learning, in: Proceedings of the 13th International Conference on Autonomous Agents and Multi Agent Systems, 2014, pp. 165–172.
[9] E.A.O. Diallo, A. Sugiyama, T. Sugawara, Coordinated behavior of cooperative agents using deep reinforcement learning, Neurocomputing (2019).
[10] A. Eck, L.-K. Soh, S. Devlin, D. Kudenko, Potential-based reward shaping for pomdps, in: Proceedings of the 12th International Conference on Autonomous Agents and MultiAgent Systems, 2013, pp. 1123–1124.
[11] P. Goyal, S. Niekum, R.J. Mooney, Using natural language for reward shaping in reinforcement learning, arXiv preprint arXiv:1903.02020 (2019).
[12] M. Grześ, Reward shaping in episodic reinforcement learning, in: Proceedings of the 16th International Conference on Autonomous Agents and MultiAgent Systems, 2017, pp. 565–573.
[13] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor, arXiv:1801.01290 (2018).
[14] H.V. Hasselt, Double q-learning, in: Proceedings of the Advances in Neural Information Processing Systems, 2010, pp. 2613–2621.
[15] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[16] C. HolmesParker, A.K. Agogino, K. Tumer, Combining reward shaping and hierarchies for scaling to large multiagent systems, Knowl. Eng. Rev. 31 (1) (2016) 3–18.
[17] T. Jaakkola, M.I. Jordan, S.P. Singh, Convergence of stochastic iterative dynamic programming algorithms, in: Proceedings of the Advances in Neural Information Processing Systems, 1994, pp. 703–710.
[18] D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, arXiv preprint arXiv:1412.6980 (2014).
[19] V.R. Konda, J.N. Tsitsiklis, Actor-critic algorithms, in: Proceedings of the Advances in Neural Information Processing Systems, 2000, pp. 1008–1014.
[20] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.
[21] L. Luo, Y. Xiong, Y. Liu, X. Sun, Adaptive gradient methods with dynamic bound of learning rate, arXiv preprint arXiv:1902.09843 (2019).
[22] A.M. Lyapunov, The general problem of the stability of motion, Int. J. Control 55 (3) (1992) 531–534.
[23] P. Mannion, S. Devlin, K. Mason, J. Duggan, E. Howley, Policy invariance under reward transformations for multi-objective reinforcement learning, Neurocomputing 263 (2017) 60–73.
[24] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 1928–1937.
[25] V. Mnih, K. Kavukcuoglu, D. Silver, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[26] A.Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in: Proceedings of the ICML, 1999.
[27] T.J. Perkins, A.G. Barto, Lyapunov design for safe reinforcement learning, J. Mach. Learn. Res. 3 (Dec) (2002) 803–832.
[28] S.J. Reddi, S. Kale, S. Kumar, On the convergence of adam and beyond, in: Proceedings of the ICLR, 2018.
[29] S. Ruder, An Overview of Gradient Descent Optimization Algorithms, arXiv preprint arXiv:1609.04747 (2016).
[30] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: Proceedings of the International Conference on Machine Learning, 2015, pp. 1889–1897.
[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).
[32] D. Silver, A. Huang, C.J. Maddison, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484.
[33] D. Silver, G. Lever, N. Heess, et al., Deterministic policy gradient algorithms, in: Proceedings of the ICML, 2014.
[34] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: Proceedings of the ICML, 2013.
[35] R.S. Sutton, A.G. Barto, Introduction to reinforcement learning, 135, MIT press Cambridge, 1998.
[36] R.S. Sutton, D.A. McAllester, S.P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Proceedings of the Advances in neural information processing systems, 2000, pp. 1057–1063.
[37] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012, pp. 5026–5033.
[38] C.J. Watkins, P. Dayan, Q-Learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[39] E. Wiewiora, G.W. Cottrell, C. Elkan, Principled methods for advising reinforcement learning agents, in: Proceedings of the ICML, 2003.
[40] B. Xu, C. Yang, Z. Shi, Reinforcement learning output feedback nn control using deterministic learning technique, IEEE Trans. Neural Netw. Learn Syst. 25 (3) (2014) 635–641.
[41] Y. Yuan, M. Li, J. Liu, C. Tomlin, On the powerball method: variants of descent methods for accelerated optimization, IEEE Control Syst. Lett. 3 (3) (2019) 601–606, doi:10.1109/LCSYS.2019.2913770.
[42] Y. Yuan, X. Tang, W. Zhou, W. Pan, X. Li, H.-T. Zhang, H. Ding, J. Goncalves, Data driven discovery of cyber physical systems, Nat. Commun. 10 (1) (2019) 4894, doi:10.1038/s41467-019-12490-1.

Yunlong Dong received the B.E degree from the School of Automation, Huazhong University of Science and Technology, Wuhan, China, in 2017. He is currently working towards the Ph.D. degree at School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China. His research interests include RL, control theory and robotics.

Xiuchuan Tang received the B.E degree from the school of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, in 2016. He is currently working towards the Ph.D. degree at School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China. His research interests include Machine Learning and Smart Manufacturing.

Ye Yuan received the B.Eng. degree in automation (Valedictorian) from the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, in September 2008, and the M.Phil. and Ph.D. degrees in control science and engineering from the Department of Engineering, University of Cambridge, Cambridge, U.K., in October 2009 and February 2012, respectively. He is currently a Full Professor at the Huazhong University of Science and Technology, Wuhan, China. He was a Postdoctoral Researcher at UC Berkeley, a Junior Research Fellow at Darwin College, University of Cambridge. His research interests include system identification and control with applications to cyber-physical systems.