Fast Proximal Policy Optimization

Weiqi Zhao, Haobo Jiang, and Jin Xie(✉)

PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, Nanjing University of Science and Technology, Nanjing, China
{zwq626,jiang.hao.bo,csjxie}@njust.edu.cn

Abstract. Proximal policy optimization (PPO) is one of the most promising deep reinforcement learning methods and has achieved remarkable success in a variety of challenging control tasks. However, its overall updating gradient of a batch of samples may mislead the optimization of some sub-samples. It potentially reduces the sample efficiency and degrades the final decision performance. Although the minimum operation of PPO can relieve it, its slow *escape speed* makes it difficult to escape the wrong optimization range within the limited epochs of the minibatch update. In this paper, we propose a novel fast version of PPO named fast-PPO that replaces the original minimum operation with two accelerating operations called *linear-pulling* and *quadratic-pulling*, respectively. Both of them can increase the updating weight of the gradient for the misled samples so that the gradient of the overall object follows their expected optimization direction. Extensive experiments on classic discrete control tasks and MuJoCo based continuous control tasks verify the effectiveness of our proposed fast PPO.

Keywords: Proximal Policy Optimization · Escape speed · Accelerating operations

1 Introduction

In recent years, deep reinforcement learning (DRL) has achieved great development and obtained impressive successes in different fields, such as competitive games (Doom [13], Atari 2600 [17,18], game Go [21], etc.), robot navigation [6,14,15] and control tasks [2,4,5,8,10,16]. Given a control problem formulated as a Markov Decision Problem (MDP), it is dedicated to learning an optimal policy that can obtain the highest cumulative rewards through extensive trial and error. In general, model-free DRL mainly contains the value function based method [11] and the policy gradient method [9], where the former indirectly learns the policy via learning an optimal action-value function while the latter directly optimizes the expected return by searching in the parameterized policy space. Compared to the value function based method, the policy gradient method presents a significant advantage on the complex continuous control problems and thus obtains more and more attention.

The trust region policy optimization (TRPO) method [19] is one of the widely studied policy gradient methods. It aims to safely perform policy updating with guaranteed

This work was supported by Shanghai Automotive Industry Science and Technology Development Foundation (No. 1917).

C. Wallraven et al. (Eds.): ACPR 2021, LNCS 13189, pp. 73–86, 2022.
https://doi.org/10.1007/978-3-031-02444-3_6

monotonic performance improvement by optimizing a surrogate object function (low bound of the original object) constrained by the divergence between the old policy distribution and the updated one. Although TRPO has a complete theoretical guarantee about monotonicity, its complicated second-order approximation for the constraint largely reduces its computation efficiency and hinders its applications in large-scale tasks. To relieve it, the Proximal Policy Optimization (PPO) [20] proposes a likelihood ratio based constraint for parameter updating, which is able to retain the stable optimization and sample efficiency of TRPO while only requires computationally efficient first-order optimization. In detail, the probability ratio between the old and new policy distributions is clipped within manually defined constant bounds (clipping range) so that it can remove the updating incentive for moving the likelihood ratio outside of the defined clipping bounds.

However, as demonstrated in [25], with the improper initialization, PPO may not perform sufficient exploration in its environment and thus suffers from the local optima. To handle it, Wang *et al.* proposes a trust-region guided PPO algorithm (TRGPPO), which can improve the exploration ability and achieve higher performance compared to the original PPO method through adaptively tuning its clipping range within the trust region. Furthermore, Wang *et al.* [24] finds that PPO could neither restrict the likelihood ratio within the clipping range strictly nor enforce the KL divergence lower than the specified bound. Therefore, PPO-RB [24] is proposed to replace the original flat clipping function with a rollback function (straight-reversed slope) which can weaken the incentive, driven by the overall function, of exceeding the clipping bounds. In addition, Zhu *et al.* [27] proposes a smoothed PPO variant that combines original PPO with PPO-RB to further improve the stability and sample efficiency of PPO-RB. Moreover, trust region-based PPO (TR-PPO) [24] substitutes the probability ratio based clipping function with the trust region based one, which is justified that such variant can sufficiently bound the KL divergence.

Although the variants of PPO discussed above can obtain great performance gain, all of them just focus on safe constraints on the likelihood ratio between the old and new policies, while neglecting the issue that the overall optimization gradient of a batch of samples may mislead the optimization for some sub-sample, named *negative optimization*. Intuitively, *negative optimization* means that the updated policy network may reduce the decision probability of the action that has the positive advantage value. Although the minimum operation of PPO can relieve it, it's still difficult to escape the wrong optimization within the finite epochs. To improve the *escape speed*, the weight of such samples in the overall object's gradient should be increased. To this end, we propose an improved PPO called Fast Proximal Policy Optimization algorithm (FPPO) to decrease the number of samples suffering from *negative optimization* after the finite updating epochs. Specifically, we first replace the minimum operation with an accelerating operation called *linear-pulling* which multiplies the objective function by an accelerating factor so that the weight of the misled examples can be increased in the linear sense. To adjust the accelerating factor adaptively, we propose another accelerating operation named *quadratic-pulling*, where the acceleration grows when the ratio goes away from the boundary. We theoretically prove that both of the proposed

accelerating variants can improve the *escape speed* of PPO. Also, extensive experiment results further demonstrate their effectiveness.

2 Related Work

Policy gradient algorithm [12,22] is one of the popular studying directions in reinforcement learning. By parameterizing the policy network, it aims to update its network parameters following the gradient-direction of performance improvement. However, during the optimization process, selecting a proper updating step size is an important but challenging problem. Kakade *et al.* firstly proposed that it is better to update the policy within a region in the policy space. Inspired by the theory about the restricted region, the trust region policy optimization (TRPO) [19] proposed to enforce a hard constraint of KL divergence onto the surrogate objective function. Also, Wu *et al.* [26] proposed to use Kronecker-Factored trust regions to optimize the policy. To optimize the computational complexity of TRPO, the Proximal Policy Optimization algorithm (PPO) [20], a first-order algorithm, was proposed. It adopts a clipping mechanism to restrict the likelihood ratio between the old policy and the new one.

Despite the huge success that PPO had achieved in a range of challenging tasks, the original method still has some flaws that affect its performance. Wang *et al.* [25] proved that the clipping mechanism with a constant clipping range might fail in the case that the policy is initialized from a bad one. Thus, they proposed to adaptively adjust the clipping range of PPO to get sufficient exploration. Furthermore, Wang *et al.* [24] found that PPO can neither restrict the ratio within the clipping range strictly nor restrict strictly the policy within the trust region. To address it, they proposed two improvements: replace the flat constraint with a rollback operation, use the KL divergence bound as the clipping range. Moreover, [3,7] also empirically analyzed the implementation details and the code-level optimization of PPO. They argued that the practical success of PPO might be owed to the tricks adopted in the code.

All of the improved methods above neglect the *negative optimization* caused by improper gradient-direction of the overall object. Instead, our method is devoted to handling such issues and the proposed variant is proved to be able to effectively escape the *negative optimization*, which can potentially improve its sample efficiency as well as the final performance. We proposed two *accelerating operations* to improve the *escape speed* and justify them theoretically.

3 Preliminaries

We consider a finite Markov Decision Process (MDP) described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces; $\mathcal{T}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition probability distribution; $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is the reward function; $\gamma \in (0, 1)$ is the discount factor. In a finite MDP, the agent takes an action $\mathbf{a}_t \in \mathcal{A}$ in a state $\mathbf{s}_t \in \mathcal{S}$ and then gets a reward $r_t = \mathcal{R}(\mathbf{s}_{t+1})$ and the next state $\mathbf{s}_{t+1} \in \mathcal{S}$. The policy π maps each state $\mathbf{s} \in \mathcal{S}$ to a distribution over $\mathcal{A}$ and our goal is to find an optimal policy that can achieve the maximum accumulated rewards.

In policy gradient methods [19,22], the policy π is usually represented by a policy network π_θ where θ denotes the network parameters. Then, the policy gradient algorithm aims to find the optimal parameter θ^* with the gradient ascent so that the object function $J(\theta)$ defined as below can be maximized:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[Q^{\pi_\theta}(\mathbf{s}, \mathbf{a})\right], \tag{1}$$

where the state-action value function $Q^{\pi}(\mathbf{s}, \mathbf{a})$ denotes the expected value of accumulated rewards that an agent can obtain after performing an action $\mathbf{a}$ in the state $\mathbf{s}$ following a policy π_θ:

$$Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right]. \tag{2}$$

The parameter θ is updated along the gradient of the objective function as below:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta \left(Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) - b(\mathbf{s})\right)\right], \tag{3}$$

where $b(\mathbf{s}) : \mathcal{S} \rightarrow \mathbb{R}$ is a baseline which can reduce variance without changing the expected value of the gradient. The baseline is usually set to a state function $V^{\pi_\theta}(\mathbf{s}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \mathbf{s}_0 = \mathbf{s}\right]$. Next, we denote a advantage function $A^{\pi_\theta}(\mathbf{s}, \mathbf{a}) = Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) - V^{\pi_\theta}(\mathbf{s})$ and the policy gradient can be rewritten as:

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta \left(Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) - V^{\pi_\theta}(\mathbf{s})\right)\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta \left(A^{\pi_\theta}(\mathbf{s}, \mathbf{a})\right)\right]. \end{aligned} \tag{4}$$

3.1 Trust Region Policy Optimization

Trust region policy optimization (TRPO) exploits the lower bound of the original optimization target as its objective function (i.e., surrogate objective), which is subjected to a constraint on the KL-divergence between the current policy distribution $\pi_\theta(\cdot \mid \mathbf{s}_t)$ and the old one $\pi_{\theta_{old}}(\cdot \mid \mathbf{s}_t)$. For simplicity, we denote A_t as the advantage value at sample $(\mathbf{s}_t, \mathbf{a}_t)$, i.e., $A_t \triangleq A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$ and denote $r_t(\theta)$ as the likelihood ratio of sample $(\mathbf{s}_t, \mathbf{a}_t)$ between the current and the old policies, i.e., $r_t(\theta) \triangleq \frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_{\text{old}}}(\mathbf{a}_t|\mathbf{s}_t)}$. Then, the optimization object of TRPO can be written as below:

$$\max_\theta \ \mathbb{E}_{\pi_\theta}\left[r_t(\theta) A_t\right] \tag{5}$$

$$\text{subject to} \quad \mathbb{E}_{\pi_\theta}\left[\mathrm{KL}\left[\pi_{\theta_{\text{old}}}(\cdot \mid \mathbf{s}_t), \pi_\theta(\cdot \mid \mathbf{s}_t)\right]\right] \leqslant \sigma \tag{6}$$

where the hyper-parameter σ constrains the KL-divergence between the current and old policies. Although the constraint on updating step above can effectively improve the optimization stability of TRPO, its second-order optimization usually has high computational complexity which is inefficient for large-scale applications, such as the tasks with the high-dimensional sensory input.

3.2 Proximal Policy Optimization

In order to handle the inefficient second-order optimization issue in TRPO, the Proximal Policy Optimization (PPO) algorithm [20] is proposed to directly clip the likelihood ratio $r_t(\theta)$ between the current and the old policies for robust policy optimization. Notably, such clipping operation retains the monotonicity of TRPO as well as just requires the first-order computation. Specifically, the objective function of PPO is defined as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_{\pi_\theta}\left[\min(r_t(\theta)A_t, \mathcal{F}^{\text{CLIP}}(r_t(\theta), \epsilon)A_t\right], \tag{7}$$

where the clipping function $\mathcal{F}^{\text{CLIP}}(r_t(\theta), \epsilon) = \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$, which remove the incentive that the ratio moves out of the clipping range $[1-\epsilon, 1+\epsilon]$, and the hyper-parameter $\epsilon \in (0, 1)$. The minimum operation is to guarantee that the final objective is a lower bound on the unclipped objective [20].

As demonstrated in Sect. 4.1, although the minimum operation in PPO can relieve the *negative optimization* problem to some extent, there are still some samples that suffer from the *negative optimization* after finite epochs of minibatch updates. Please refer to Sect. 4.1 for more details.

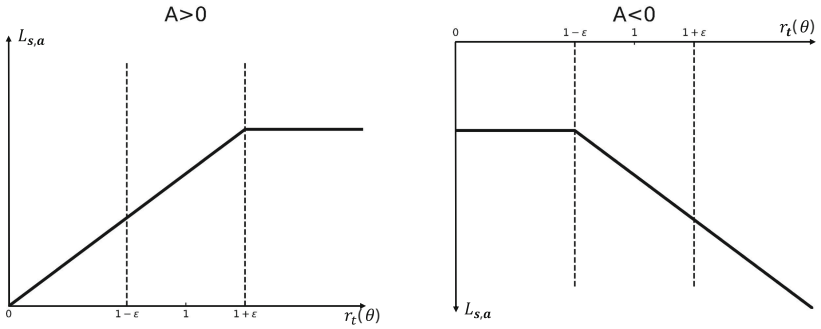

Fig. 1. Plots show how the slope of the surrogate function L^{CLIP} varies with different $r_t(\theta)$, for positive advantages (left) and negative advantages (right). We can see that it loses gradient when $A_t > 0, r_t(\theta) > 1+\epsilon$ or $A_t < 0, r_t(\theta) < 1-\epsilon$. However it does not suffer from this issue when $A_t > 0, r_t(\theta) < 1-\epsilon$ or $A_t < 0, r_t(\theta) > 1+\epsilon$ because of the minimum operation.

4 Method

4.1 Sample Efficiency of PPO

As we can see in Fig. 1, $r_t(\theta)$ may be lower than $1-\epsilon$ when $A_t > 0$. That is to say, the probability of an action $\mathbf{a}$ whose advantage value is positive decreases, called *negative optimization*. The diversity between the overall gradient of a batch of samples and that of a single sample lead to this case in practice. Without the minimum operation, the gradient of such examples will be zero so that their gradient can never be updated. The minimum operation in PPO can help those examples to regain their original gradients.

However, such examples can hardly escape from *negative optimization* in finite epochs by minimum operation. This will significantly hinder the sample efficiency of PPO. To address this issue, we propose to increase the weight of such examples to correct the fused gradient so that the examples could escape from *negative optimization* in the remaining epochs. For example, the angle between $\mathbf{g}_1$ and $\mathbf{g}$ (the left one of Fig. 2) is greater than 90° in the beginning while the corresponding example enjoys positive optimization again after increasing its gradient.

In original PPO, the ratio could be driven to go farther away from the bound in the case that *negative optimization* has already occurred. Formally, we give a theorem as follows.

Theorem 1. *Suppose there is a parameter θ_0 that $r_t(\theta_0)$ satisfies the condition of negative optimization. Let $\theta_1^{PPO} = \theta_0 + \delta \nabla \hat{L}^{PPO}(\theta_0)$, where δ is the step size and $\nabla \hat{L}^{PPO}(\theta_0)$ is the gradient of $\hat{L}^{PPO}(\theta)$ at θ_0. On the condition that*

$$\langle \nabla \hat{L}^{PPO}(\theta_0), \nabla r_t(\theta_0) \rangle A_t < 0, \tag{8}$$

there exists such $\delta^* > 0$ that we have following property for any $\delta \in (0, \delta^*)$

$$\left| r_t(\theta_1) - 1 \right| < \left| r_t(\theta_0) - 1 \right| < \epsilon. \tag{9}$$

Following [24], we give an formal proof as follows:

Proof. Let $\psi(\delta) = r_t(\theta_0 + \delta \nabla \hat{L}^{\text{PPO}}(\theta_0))$, then we can get the gradient:

$$\psi'(0) = \langle \nabla \hat{L}^{\text{PPO}}(\theta_0), \nabla r_t(\theta_0) \rangle \tag{10}$$

We have $\psi'(0) < 0$ when $r_t(\theta_0) < 1 - \epsilon$ and $A_t > 0$. Thus there is $\delta^* > 0$ such that for any $\delta \in (0, \delta^*)$, we have:

$$\psi(\delta) < \psi(0) \tag{11}$$

Then

$$r_t(\theta_1) < r_t(\theta_0) \leq 1 - \epsilon \tag{12}$$

That is

$$\left| r_t(\theta_1) - 1 \right| < \left| r_t(\theta_0) - 1 \right| \tag{13}$$

□

The condition (8) will be triggered when the gradient-direction of the overall objective $\hat{L}^{\text{PPO}}(\theta_0)$ and that of $r_t(\theta_0) A_t$ is significantly different. This condition is possibly caused by the highly differentiated gradient-directions of a minibatch of samples.

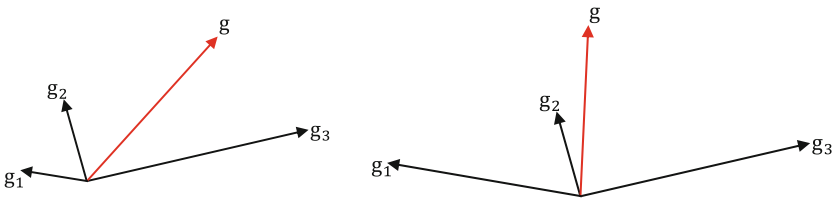

Fig. 2. $\mathbf{g}$ is the fused gradient, and $\mathbf{g}_1$ is the gradient of a sample that owns a positive advantage but suffers from *negative optimization* in the left figure. We can correct the fused gradient by increasing the gradient of the negatively optimized samples. If the correction mechanism is effective enough, all examples will enjoy proper optimization gradients just as depicted in the right figure.

4.2 Fast Proximal Policy Optimization with Linear-Pulling

To better explain our method, we rewrite the Eq. 7 as the following:

$$L(\theta) = \mathbb{E}_{\pi_\theta}\left[\min\left(r_t(\theta)A_t, \mathcal{F}(r_t(\theta), \epsilon)A_t\right)\right] \tag{14}$$

where $\mathcal{F}$ is a clipping function to restrict the update step size of the policy. In PPO, $\mathcal{F} = \text{clip}\left(r_t\left(\theta\right), 1-\epsilon, 1+\epsilon\right)$. To enforce the gradient of the overall object across a batch of samples to correct, we can increase the weight of the examples suffering from *negative optimization*. Specifically, $\mathcal{F}$ multiplies by a factor greater than one in the case that $A_t > 0, r_t\left(\theta\right) < 1-\epsilon$ or $A_t < 0, r_t\left(\theta\right) > 1+\epsilon$. Our mechanism is to pull the ratio into the clipping range with a linear function, we called this operation *linear-pulling*. Then we get a new clipping function which is defined as

$$\mathcal{F}^{\text{FPPO-LP}} = \begin{cases} \alpha r_t(\theta) + (1-\alpha)(1-\epsilon), & A_t > 0 \text{ and } r_t(\theta) < 1-\epsilon \\ \alpha r_t(\theta) + (1-\alpha)(1+\epsilon), & A_t < 0 \text{ and } r_t(\theta) > 1+\epsilon \\ \text{CLIP}(r_t(\theta)), & \text{otherwise} \end{cases} \tag{15}$$

where the hyper-parameter $\alpha \in [1, +\infty)$ is to decide the force of acceleration. Figure 3 plots $L^{\text{FPPO-LP}}(\theta)$ as function of the ratio $r_t(\theta)$. As the figure depicted, when $r_t(\theta)$ is out of the clipping range, the slope of the side that suffers from *negative optimization* becomes larger. By correcting the overall object's gradient, *linear-pulling* could more forcefully keep the ratio $r_t(\theta)$ within the clipping range compared to the minimum operation in PPO. Formally, we have the following theorem.

Theorem 2. *Suppose there is a parameter* θ_0, $\theta_1^{PPO} = \theta_0 + \delta\nabla\hat{L}^{PPO}(\theta_0)$, *and* $\theta_1^{FPPO\text{-}LP} = \theta_0 + \delta\nabla\hat{L}^{FPPO\text{-}LP}(\theta_0)$. *The set of indexes of the samples suffering from negative optimization is denoted as* $\Omega = \{t|1 \le t \le T, |r_t(\theta_0) - 1| \ge \epsilon$ *and* $r_t(\theta_0)A_t \le r_t(\theta_{old})A_t\}$. *If* $t \in \Omega$ *and* $r_t(\theta_0)$ *satisfies such condition that* $\sum_{t' \in \Omega}\langle\nabla r_t(\theta_0), \nabla r_{t'}(\theta_0)\rangle A_t A_{t'} > 0$, *then there exists some* $\delta^* > 0$ *such that for any* $\delta \in (0, \delta^*)$, *we have*

$$\left|r_t(\theta_1^{FPPO\text{-}LP}) - 1\right| < \left|r_t(\theta_1^{PPO}) - 1\right|. \tag{16}$$

Proof. Let $\psi(\delta) = r_t(\theta_0 + \delta\nabla\hat{L}^{\text{FPPO-LP}}(\theta_0)) - r_t(\theta_0 + \delta\nabla\hat{L}^{\text{PPO}}(\theta_0))$. We can get the gradient:

$$\begin{aligned}\psi'(0) &= \nabla r_t^{\top}(\theta_0)\left(\nabla\hat{L}^{\text{FPPO-LP}}(\theta_0) - \nabla\hat{L}^{\text{PPO}}(\theta_0)\right) \\ &= (\alpha-1)\sum_{t'\in\Omega}\langle\nabla r_t(\theta_0), \nabla r_{t'}(\theta_0)\rangle A_{t'}\end{aligned} \tag{17}$$

We have $\psi'(0) > 0$ in the case where $r_t(\theta_0) \leq 1-\epsilon$ and $A_t > 0$. Thus there exists $\delta^* > 0$ such that for any $\delta \in (0, \delta^*)$

$$\phi(\delta) > \phi(0) \tag{18}$$

Then we have

$$r_t(\theta_1^{\text{FPPO-LP}}) > r_t(\theta_1^{\text{PPO}}) \tag{19}$$

That is

$$\left|r_t(\theta_1^{\text{FPPO-LP}}) - 1\right| < \left|r_t(\theta_1^{\text{PPO}}) - 1\right| \tag{20}$$

Similarly, we can also get $\left|r_t(\theta_1^{\text{FPPO-LP}}) - 1\right| < \left|r_t(\theta_1^{\text{PPO}}) - 1\right|$ in the case where $r_t(\theta_0) \geqslant 1+\epsilon$ and $A_t < 0$. □

This theorem proves that *linear-pulling* can prevent the out-of-the-range ratios from going farther beyond the clipping range more forcefully. In other words, the *escape speed* is increased. Ideally, we can tune α to guarantee the new policy within the clipping range more effectively.

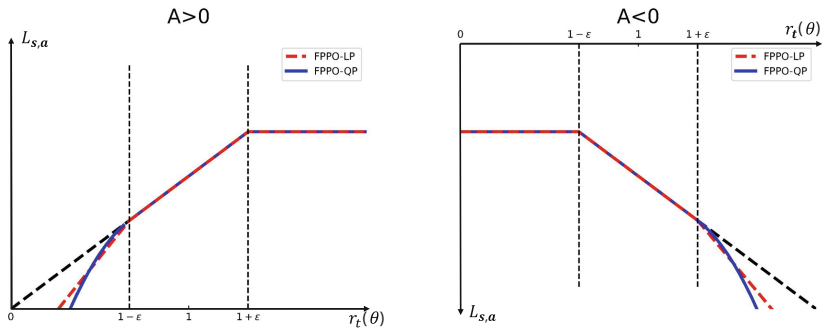

Fig. 3. Plots show the surrogate function $L_{\mathbf{s},\mathbf{a}}$ as a function of $r_t(\theta)$ (the probability ratio), for positive advantages (left) and negative advantages (right). We can see that the slope of FPPO (both FPPO-LP and FPPO-QP) becomes larger than PPO when $A_t > 0$, $r_t(\theta) < 1+\epsilon$ or $A_t < 0$, $r_t(\theta) > 1+\epsilon$. The red dotted line corresponds to *linear-pulling* and the blue line corresponds to *quadratic-pulling*. (Color figure online)

4.3 Fast Proximal Policy Optimization with Quadratic-Pulling

In FPPO-LP, the accelerating factor α is a fixed parameter, however, the updating epochs of batch samples are finite, the examples suffering from *negative optimization* cannot escape within a few epochs if the ratio is too far away from the bound. Thus it is more reasonable that the farther the ratio goes away from the clipping range, the larger the parameter α in FPPO-LP should be set. Inspired by the motivation above, we have the following differential equation:

$$\mathcal{F}'(r_t(\theta) = \begin{cases} k(1-\epsilon-r_t(\theta))+1, & A_t > 0 \text{ and } r_t(\theta) < 1-\epsilon \\ k(r_t(\theta)-1-\epsilon)+1, & A_t < 0 \text{ and } r_t(\theta) > 1+\epsilon \\ \nabla \operatorname{CLIP}(r_t(\theta)), & \text{otherwise} \end{cases} \tag{21}$$

where k is a hyper-parameter to control the change of slope. By solving differential equation Eq. 21, we obtain the *quadratic-pulling* operation:

$$\mathcal{F}^{\text{FPPO-QP}} = \begin{cases} k(1-\epsilon-r_t(\theta))^2 + r_t(\theta), & A_t > 0 \text{ and } r_t(\theta) < 1-\epsilon \\ k(r_t(\theta)-1-\epsilon)^2 + r_t(\theta), & A_t < 0 \text{ and } r_t(\theta) > 1+\epsilon \\ \operatorname{CLIP}(r_t(\theta)), & \text{otherwise} \end{cases} \tag{22}$$

where the value of k is half of that in Eq. 21. Similarly, FPPO-QP has the same property as Theorem 2. Formally, we give the proof as following:

Proof. Let $\psi(\delta) = r_t(\theta_0+\delta\nabla\hat{L}^{\text{FPPO-QP}}(\theta_0)) - r_t(\theta_0+\delta\nabla\hat{L}^{\text{PPO}}\theta_0)$. When $r_t(\theta_0) < 1-\epsilon$ and $A_t > 0$, We can get the gradient:

$$\begin{aligned} \psi'(0) &= \nabla r_t^\top(\theta_0)\left(\nabla\hat{L}^{\text{FPPO-QP}}(\theta_0) - \nabla\hat{L}^{\text{PPO}}(\theta_0)\right) \\ &= 2k\left(1-\epsilon-r_t(\theta_0)\right)\sum_{t'\in\Omega}\langle\nabla r_t(\theta_0), \nabla r_{t'}(\theta_0)\rangle A_{t'} \end{aligned} \tag{23}$$

We have $\psi'(0) > 0$, thus there exists $\delta^* > 0$ such that for any $\delta \in (0, \delta^*)$

$$\psi(\delta) > \psi(0) \tag{24}$$

Then we have

$$\left|r_t(\theta_1^{\text{FPPO-QP}}) - 1\right| < \left|r_t(\theta_1^{\text{PPO}}) - 1\right| \tag{25}$$

Similarly, we can also get $\left|r_t(\theta_1^{\text{FPPO-QP}})-1\right| < \left|r_t(\theta_1^{\text{PPO}})-1\right|$ in the case where $r_t(\theta_0) > 1+\epsilon$ and $A_t < 0$. □

Theoretically, *quadratic-pulling* can guarantee that the slope of the surrogate objective function be tuned feasibly.

5 Experiments

In this section, we firstly present the performance of our algorithm in contrast with other policy gradient methods and then show the results of our ablation study.

The following algorithms are evaluated. (a) PPO: the version with clipping mechanism, we use the author recommended hyper-parameter $\epsilon = 0.2$ [20]. (b) TR-PPO: the ratio is clipped when the updated policy is out of the trust region [24]. The hyper-parameter $\delta = 0.035$. (c) PPO-RB: PPO with the rollback operation [24]. The rollback coefficient $\alpha = 0.3$ (d) FPPO-LP: fast PPO with linear-pulling. We choose α as recommended in Table 2. (e) FPPO-QP: fast PPO with quadratic-pulling. We use k as recommended in Table 2.

Table 1. Max average return over 5 trials of 100 thousand timesteps (classic discrete tasks) or 1 million steps (MuJoCo tasks).

	Task	FPPO-LP	FPPO-QP	PPO	TR-PPO	PPO-RB
Discrete tasks	CartPole	$\mathbf{200.0 \pm 0.03}$	$\mathbf{200.0 \pm 0.01}$	**200.0**	192.7	**200.0**
	Acrobot	-84.2 ± 1.2	$\mathbf{-82.0 \pm 0.7}$	-88.7	-86.8	-89.2
	MountainCar	$\mathbf{-130.6 \pm 0.9}$	-131.0 ± 1.4	-143.3	-140.2	-146.2
Continuous tasks	Walker2d	$\mathbf{4030.4 \pm 122.8}$	3893.4 ± 145.8	3368.5	3279.8	2858.4
	Ant	2778.8 ± 183.7	$\mathbf{3323.6 \pm 200.5}$	2024.3	2013.0	2640.1
	Reacher	$\mathbf{-6.1 \pm 0.28}$	-6.5 ± 0.29	-7.6	$\mathbf{-6.0}$	-7.8
	Hopper	2726.1 ± 118.4	$\mathbf{2908.8 \pm 116.0}$	2364.2	2341.8	1997.7
	HalfCheetah	4107.0 ± 105.8	$\mathbf{4478.3 \pm 175.0}$	4141.7	3455.8	3420.1
	Swimmer	113.6 ± 1.6	$\mathbf{115.5 \pm 1.9}$	100.2	108.9	99.8

5.1 Classic Discrete Control Tasks

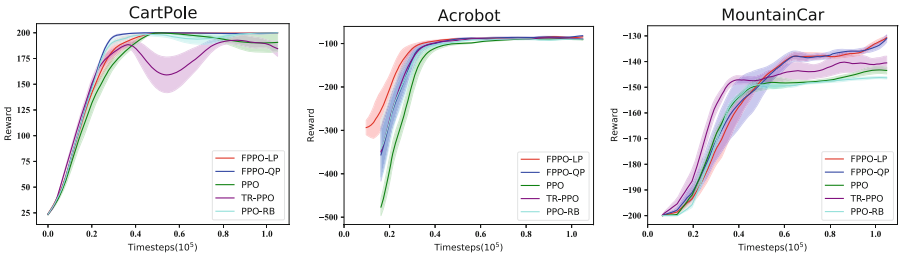

Fig. 4. Learning curves for the classic discrete control tasks.

We first conducted experiments on 3 classic discrete control tasks implemented in OpenAI Gym [1]. Figure 4 plots the performance during 10^5 training timesteps and Table 1

shows the maximum average return over 5 trials of 100 thousand timesteps. For CartPole, all the tested algorithms obtain the highest score in finite timesteps except for TR-PPO, and notably, FPPO-QP hits the highest score faster than other methods. For Acrobot and MountainCar, FPPO performs better than its competitors.

However, the improvements of FFPO in comparison with PPO are not so prominent, especially in the first two tasks. The main reason is that the exploration space is very small compared with continuous tasks so that different policy gradient methods have similar performance. Actually, value-based methods tend to perform better in such tasks. In addition, we notice that MountainCar is a sparse reward task, and thus the methods may be trapped in local optima if the *escape speed* is too low. Because of the *accelerating operations*, Both FPPO-LP and FPPO-QP obtain higher *escape speed* so that more prominent performance could be achieved.

5.2 Benchmark Tasks

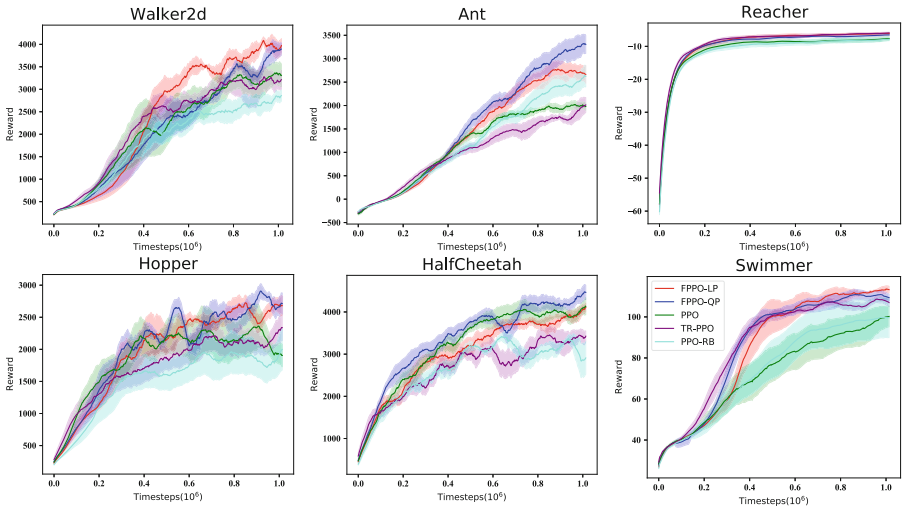

Fig. 5. Learning curves for the Mujoco continuous control tasks. The lines represent the average rewards over 5 random seeds and the shaded region represents the mean $\pm$ half of std. Curves are smoothed to get visual clarity.

In order to verify the effectiveness of our method, we evaluate PPO and its variants on continuous control benchmark tasks implemented in OpenAI Gym [1], simulated by MuJoCo [23]. 6 benchmark tasks are chosen to be tested on: Walker2d-v2, Ant-v2, Reacher-v2, Hopper-v2, HalfCheetah-v2, and Swimmer-v2. All tasks will run with 10^6 timesteps over 5 random seeds.

The mean of the reward and the standard deviation are plotted in the figure. As our results suggest, FPPO performs better than other algorithms in all 6 tasks. It is worth

noting that FPPO is particularly prominent on Hopper, Ant, and Walker2d. Walker2d has the largest sizes of action and observation spaces, which needs higher sample efficiency. The prominent performance on such a task proves the significance of high *escape speed*.

5.3 Ablation Study

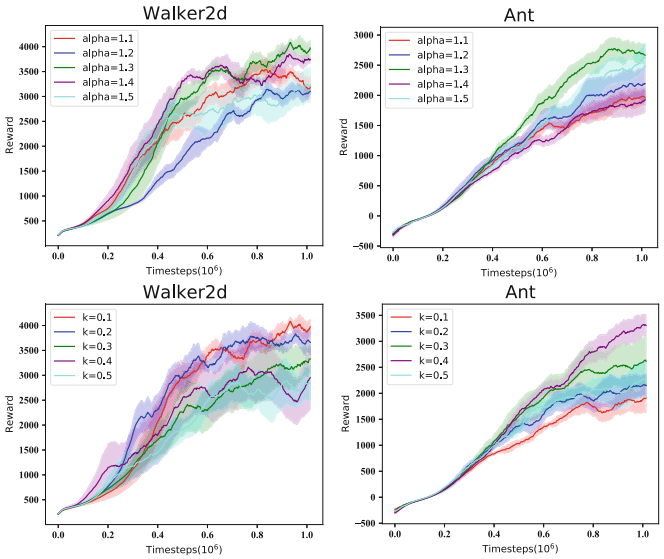

Fig. 6. Max average return over 5 trials of 1 million steps with different α and different k. The ranges of α and k are $[1.1, 1.5]$ and $[0.1, 0.5]$ respectively. And the intervals of two hyper-parameters are both 0.1.

Table 2. Used values for 6 different MuJoCo tasks respectively.

	CartPole	Acrobot	MountainCar	Hopper	Walker2d	HalfCheetah	Reacher	Swimmer	Ant
α	1.3	1.3	1.3	1.3	1.3	1.3	1.1	1.1	1.3
k	0.4	0.4	0.4	0.2	0.1	0.4	0.4	0.1	0.4

In this experiment we vary the hyper-parameter α to drop in the range $[1.1, 1.5]$, and the interval is set to 0.1. The hyper-parameter k varies from 0 to 0.5, and the interval is 0.1. The benchmarks we tested on are Walker2d and Ant, which own huge action spaces and observation spaces. From Fig. 6, we can observe that the performance of the algorithm varies with the change of α and k, which proves that the accelerating operation is essential. Furthermore, it is not true that the larger α and k are, the better performance will be achieved. Actually, the ratio may exceed the upper bound after a few epochs, although it is used to be lower than the lower bound. Thus just as depicted in Table 2, there exists the best value for α and k respectively.

6 Conclusion

Although PPO and its various variants have achieved impressive performance, these methods have failed to make a difference in improving the *escape speed*. However, the low *escape speed* may result in the loss of the sample efficiency as well as the degradation of the performance. Based on this observation, we proposed two different acceleration tricks to correct the gradient of the overall object across a batch of samples in few epochs. Both these two techniques significantly improve speed of escaping *negative optimization* and the sample efficiency. Extensive results prove the effectiveness of our method.

In conclusion, our results highlight the necessity to improve the *escape speed*, leading to the improvement in sample efficiency and performance of the policy. To our knowledge, this is the first work to focus on *negative optimization* and *escape speed*. We found that it is essential to increase the gradient of the samples trapped in *negative optimization*. We propose to study more on the *negative optimization* and exploit more methods to increase the escape speed.

References

1. Brockman, G., et al.: OpenAI Gym (2016)
2. Duan, Y., Chen, X., Houthooft, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. In: International Conference on Machine Learning, pp. 1329–1338. PMLR (2016)
3. Engstrom, L., et al.: Implementation matters in deep policy gradients: a case study on PPO and TRPO. arXiv preprint arXiv:2005.12729 (2020)
4. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning, pp. 1587–1596. PMLR (2018)
5. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning, pp. 1861–1870. PMLR (2018)
6. Hu, Y., Wang, W., Liu, H., Liu, L.: Reinforcement learning tracking control for robotic manipulator with kernel-based dynamic model. IEEE Trans. Neural Netw. Learn. Syst. **31**(9), 3570–3578 (2019)
7. Ilyas, A., et al.: Are deep policy gradient algorithms truly policy gradient algorithms? (2018)
8. Jiang, H., Qian, J., Xie, J., Yang, J.: Planning with learned dynamic model for unsupervised point cloud registration (2021)
9. Jiang, H., Qian, J., Xie, J., Yang, J.: Episode-experience replay based tree-backup method for off-policy actor-critic algorithm. In: Lai, J.-H., et al. (eds.) PRCV 2018. LNCS, vol. 11256, pp. 562–573. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03398-9_48
10. Jiang, H., Shen, Y., Xie, J., Li, J., Qian, J., Yang, J.: Sampling network guided cross-entropy method for unsupervised point cloud registration. arXiv preprint arXiv:2109.06619 (2021)
11. Jiang, H., Xie, J., Yang, J.: Action candidate based clipped double q-learning for discrete and continuous action tasks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 7979–7986 (2021)
12. Kakade, S.M.: A natural policy gradient. In: Advances in Neural Information Processing Systems, vol. 14 (2001)
13. Kempka, M., Wydmuch, M., Runc, G., Toczek, J., Jaśkowski, W.: ViZDoom: a doom-based AI research platform for visual reinforcement learning. In: 2016 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1–8. IEEE (2016)

14. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. J. Mach. Learn. Res. **17**(1), 1334–1373 (2016)
15. Li, T., Geyer, H., Atkeson, C.G., Rai, A.: Using deep reinforcement learning to learn high-level policies on the ATRIAS biped. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 263–269. IEEE (2019)
16. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
17. Mnih, V., et al.: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
18. Mnih, V., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
19. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning, pp. 1889–1897. PMLR (2015)
20. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
21. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484–489 (2016)
22. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y., et al.: Policy gradient methods for reinforcement learning with function approximation. In: NIPS, vol. 99, pp. 1057–1063. Citeseer (1999)
23. Todorov, E., Erez, T., Tassa, Y.: MuJoCo: a physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026–5033. IEEE (2012)
24. Wang, Y., He, H., Tan, X.: Truly proximal policy optimization. In: Uncertainty in Artificial Intelligence, pp. 113–122. PMLR (2020)
25. Wang, Y., He, H., Tan, X., Gan, Y.: Trust region-guided proximal policy optimization. arXiv preprint arXiv:1901.10314 (2019)
26. Wu, Y., Mansimov, E., Liao, S., Grosse, R., Ba, J.: Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. arXiv preprint arXiv:1708.05144 (2017)
27. Zhu, W., Rosendo, A.: Proximal policy optimization smoothed algorithm. arXiv preprint arXiv:2012.02439 (2020)