Policy Search by Target Distribution Learning for Continuous Control

Chuheng Zhang
IIIS, Tsinghua University
zhangchuheng123@live.com

Yuanqi Li
IIIS, Tsinghua University
timezerolyq@gmail.com

Jian Li
IIIS, Tsinghua University
lapordge@gmail.com

Abstract

It is known that existing policy gradient methods (such as vanilla policy gradient, PPO, A2C) may suffer from overly large gradients when the current policy is close to deterministic, leading to an unstable training process. We show that such instability can happen even in a very simple environment. To address this issue, we propose a new method, called *target distribution learning* (TDL), for policy improvement in reinforcement learning. TDL alternates between proposing a target distribution and training the policy network to approach the target distribution. TDL is more effective in constraining the KL divergence between updated policies, and hence leads to more stable policy improvements over iterations. Our experiments show that TDL algorithms perform comparably to (or better than) state-of-the-art algorithms for most continuous control tasks in the MuJoCo environment while being more stable in training.

1 Introduction

Reinforcement learning (RL) algorithms can be broadly divided into value-based methods and policy search methods. When applied to continuous control tasks, value-based methods, such as (Mnih et al. 2015; Schaul et al. 2015; Wang et al. 2016; Van Hasselt, Guez, and Silver 2016; Dabney et al. 2018), need additional treatments to convert the learned value function to executable policies (Gu et al. 2016; Novati and Koumoutsakos 2019). On the other hand, policy search methods directly improve a policy for continuous control. Among others, policy gradient-based methods have been shown to be quite effective in searching good policies, e.g., (Williams 1992; Sutton, Barto, and others 1998; Silver et al. 2014; Lillicrap et al. 2015; Mnih et al. 2016). These methods first compute the gradient of the performance measure with respect to the parameters of the policy network and then update the parameters via stochastic gradient ascent. In order to ensure that the policy improves in terms of the performance measure over the iterations, the policy improvement theorem (Kakade and Langford 2002) suggests that the policy update should not be too large over one iteration. Specifically, policy improvement requires a regularization on the *state-action space* to avoid destructively large updates, i.e., the probability distributions over the action space of the old and new policies conditioned on a state should not vary too much.

Influenced by the policy improvement theorem, several effective algorithms have been proposed in the literature. TRPO (Schulman et al. 2015a) and ACKTR (Wu et al. 2017) both update the policy subject to a constraint in the state-action space (trust region). ACER (Wang et al. 2017) adopts a trust region optimization method that clips the policy gradient in the state-action space to constrain the policy update. PPO (Schulman et al. 2017) designs a clipped surrogate objective that approximates the regularization. PPO has been proven to be quite effective and is relatively simple-to-implement, thus becomes a quite popular method. In a very simple environment (see Section 3), we observe some weakness of PPO in our experiment: when the policy is near-deterministic, the gradient may explode which leads to instability. Moreover, PPO performs multiple epochs of minibatch updates on the same samples to fully utilize the samples. We observe significant performance degradation when increasing the sample reuse (see Section 6).

We propose a new policy search method, called *target distribution learning (TDL)*, that improves the policy over iterations. In each iteration, the *action distribution* conditioned on each encountered state is produced by the policy network. The actions are drawn from these distributions and used to interact with the environment. Then, TDL proposes better action distributions (called *target distributions*) to optimize the performance measure and updates the policy network to approach the target distributions.

The contributions of our work are summarized as follows:

1. (Section 3) We show experimentally that PPO (even combined with some commonly used remedies) suffers from an instability issue during the training even in a very simple environment.
2. (Section 4) We propose a new policy improvement method TDL that retains the simplicity of PPO while avoids the instability issue. We also propose three algorithms based on TDL, all of which set target distributions within a trust region and thus ensure the policy improvement. We provide theoretical guarantee that

the target distribution is close to the old distribution in terms of KL divergence (Appendix A). Two algorithms set the target distributions following an update rule of evolutionary strategy (ES) (Rechenberg 1973). Unlike previous work (such as (Salimans et al. 2017; Mania, Guy, and Recht 2018; Liu et al. 2019)) which used ES to search over the parameter space directly, we incorporate the idea in ES to propose better action distributions. Moreover, we show that the target distributions proposed by one of our algorithms based on ES indicate a desirable direction and illustrate that the algorithm can better prevent premature convergence (Appendix B).

3. (Section 6) We conduct several experiments to show that our algorithms perform comparably to (or better than) several state-of-the-art algorithms on benchmark tasks. Moreover, we show that our algorithms are more effective to realize a regularization in the state-action space than TRPO and PPO, and can increase the sample reuse without significant performance degradation.

2 Preliminaries

A Markov Decision Process (MDP) for continuous control is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ specifying the state space $\mathcal{S}$, the continuous action space $\mathcal{A} \subseteq \mathbb{R}^d$, the state transition probability $P(s_{t+1}|s_t, a_t)$, the reward $R(r_t|s_t, a_t)$ and the discount factor γ. Let π denote a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. In this paper, it is specified by a probability distribution whose statistical parameter is given by a neural network with parameter θ, i.e., $\pi(a_t|\phi_\theta(s_t))$ where $\phi_\theta(s_t)$ denotes the statistical parameter (e.g., the mean and standard deviation of a Gaussian). We call this probability distribution *action distribution*. The value function is defined as $V^\pi(s) := \mathbb{E}[\sum_{t=0}^{\infty} \sum_{s'} p(s_t = s'|s_0 = s, \pi_\theta)\gamma^t r_t]$ for each $s \in \mathcal{S}$. The corresponding Q-function is defined as $Q^\pi(s, a) := \mathbb{E}[\sum_{t=0}^{\infty} \sum_{s'} p(s_t = s'|s_0 = s, a_0 = a, \pi_\theta)\gamma^t r_t]$ for each $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The advantage function for each action a in state s is defined as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. The goal is to maximize the expected cumulative reward from an initial state distribution, i.e., $\max_\pi \eta(\pi) := \mathbb{E}_{s_0}[V^\pi(s_0)]$.

In this paper, we use multivariate Gaussian distributions with diagonal covariance matrices as the action distributions for the stochastic policy. In this case, the statistical parameter $\phi_\theta(s)$ has two components, the action mean $\mu_\theta(s) \in \mathbb{R}^d$ and the diagonal elements of covariance matrix (variance) $\sigma^2_\theta(s) \in \mathbb{R}^d$. In each iteration, the new policy $\pi(a|s) = \mathcal{N}(a|\mu_\theta(s), \sigma_\theta(s))$ is updated from an old policy $\pi^{\text{old}}(a|s) = \mathcal{N}(a|\mu^{\text{old}}(s), \sigma^{\text{old}}(s))$.

In each iteration, the policy network is updated to maximize the following surrogate objective subject to a Kullback–Leibler (KL) divergence (Kullback and Leibler 1951) constraint to prevent destructively large updates. The formulation is first proposed by (Schulman et al. 2015a) based on (Kakade and Langford 2002).

$$L(\theta) = \mathbb{E}_{s\sim\rho_{\pi^{\text{old}}}}\left[\sum_a \mathcal{N}(a|\mu_\theta(s), \sigma_\theta(s))A^{\pi^{\text{old}}}(s, a)\right]$$
$$s.t. \max_{s\in\mathcal{S}} KL\left(\mathcal{N}(\mu^{\text{old}}(s), \sigma^{\text{old}}(s))||\mathcal{N}(\mu_\theta(s), \sigma_\theta(s))\right) \le \delta, \tag{1}$$

where $\rho_\pi(s) = \mathbb{E}_{s_0}[\sum_{t=0}^{\infty} \gamma^t p(s_t = s|s_0, \pi)]$ is the state visitation frequency and $\text{KL}(\cdot||\cdot)$ indicates the KL divergence between two probability distributions. When δ is small, a solution of the above optimization problem can guarantee a policy improvement over the iteration, i.e., $\eta(\pi) > \eta(\pi^{\text{old}})$ (Schulman et al. 2015a).

The above optimization objective can be approximated using Monte Carlo samples as follows:

$$\hat{L}(\theta) = \frac{1}{T}\sum_{t=1}^{T}\left[\hat{A}_t \frac{\mathcal{N}(a_t|\mu_\theta(s_t), \sigma_\theta(s_t))}{\mathcal{N}(a_t|\mu^{\text{old}}(s_t), \sigma^{\text{old}}(s_t))}\right], \tag{2}$$

where s_t and a_t are the samples of the state and the action, respectively, at timestep t following π^{old}. $\hat{A}_t := \hat{A}^{\pi^{\text{old}}}(s_t, a_t)$ is an estimator of the advantage function at timestep t. One popular choice is to use generalized advantage estimator (GAE) (Schulman et al. 2015b) as the advantage estimator. We note that TRPO and PPO are based on the above formulation.

3 Instability Issue of Previous Methods

In this section, we show that the gradient of the objective $\hat{L}(\theta)$ (in (2)) with respect to θ may explode when the policy is near-deterministic, i.e., $\sigma_\theta(\cdot)$ is small, which may lead to instability in training.

Let us consider a case where the standard deviation of the action distribution σ is state independent and thus itself is a parameter of the policy network. Define $\hat{L}_t(\theta) = \hat{A}_t \frac{\mathcal{N}(a_t|\mu_\theta(s_t), \sigma)}{\mathcal{N}(a_t|\mu^{\text{old}}(s_t), \sigma^{\text{old}})}$. By the standard chain rule, one can see that the gradient with respect to θ is as follows:

$$\frac{\partial\hat{L}(\theta)}{\partial\theta} = \frac{1}{T}\sum_{t=1}^{T}\left[\hat{L}_t(\theta)\frac{\partial \log\mathcal{N}(a_t|\mu_\theta(s_t), \sigma)}{\partial\mu_\theta(s_t)}\frac{\partial\mu_\theta(s_t)}{\partial\theta}\right]. \tag{3}$$

Moreover, the gradient of the logarithm of the probability density with respect to the mean is

$$\frac{\partial \log\mathcal{N}(a_t|\mu_\theta(s_t), \sigma)}{\partial\mu_\theta(s_t)} = \frac{a_t - \mu_\theta(s_t)}{\sigma^2}. \tag{4}$$

Therefore, the gradient with respect to θ is inversely proportional to σ for a typical sample a_t. When the policy is near-deterministic, i.e., σ is small, the gradient with respect to θ becomes large. So, it is likely that, given a state s, the mean of the action distribution conditioned on this state $\mu_\theta(s)$ is updated to a place far away from the previous mean $\mu^{\text{old}}(s)$, which may already be close to optimal. This thus leads to a "bad" action in the next iteration. Notice that other policy gradient-based algorithms involving the gradient of a probability density function (such as vanilla policy gradient, A2C) may suffer from the same issue. [1] (Zhao et al. 2011) has similar results that the variance of the policy gradient update is inversely proportional to the square of the standard deviation in two policy gradient algorithms, REINFORCE (Williams 1992) and PGPE (Grüttner et al. 2010).

[1] TRPO does not suffer from such issue since it performs a line search for the step size.

Now, we experimentally show that PPO suffers from such instability issue even in a simple environment as follows: In each round, the environment samples a state $s \sim U([0,1]^1)$. The agent receives the state, performs an action $a \in \mathbb{R}^1$ and suffers a cost (negative reward) $c(a) = a^2$. The objective is to minimize the one-step cost, i.e., $\min_\theta \mathbb{E}[\sum_a \pi_\theta(a|s)c(a)]$. Notice that the cost is independent of the state but the state is still fed as an input to the policy network. It is obvious that the optimal policy should play $a = 0$ with probability 1 for any state, which is a deterministic policy. Our experiment shows that PPO suffers from the aforementioned instability issue, resulting in an oscillating and diverging behavior. On the other hand, our new method TDL (see Section 4) circumvents the computation of the gradient of a probability density function, hence does not suffer from such instability issue. In practice, there are some common tricks that attempt to solve the instability issue, such as adding an entropy term in the loss function, setting a minimum variance threshold below which the variance of the action distribution is not allowed to decrease further, or using a small clip constant in PPO. We also adopt these tricks for PPO and compare them with our method. The entropy term can stabilize the training but leads to a worse asymptotic performance. The minimum variance constraint can achieve a smaller cost but is still unstable. A small clip constant may delay but does not prevent the instability. We show the result in Figure 1.

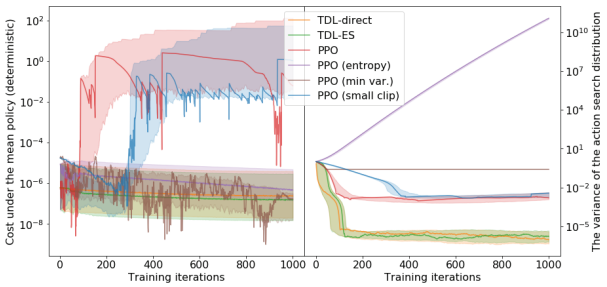

Figure 1: Median performance out of 100 independent runs for each algorithm in the simple environment described in Section 3. The shaded areas indicate the 10% and 90% quantiles. Left. The cost from executing the mean of the action distribution along the training. Right. The variance of the action distribution along the training.

4 Target Distribution Learning

Instead of optimizing the objective $L(\theta)$ with respect θ directly, TDL solves the constrained optimization problem in two steps: TDL first proposes statistical parameters that specify action distributions (targets) under which the expected advantage function of the old policy is improved. Then, TDL trains the policy network to match the targets.

In the first step, for each state sample s_t, TDL proposes a target distribution whose statistical parameters attempt to maximize $L_{t,1}(\mu, \sigma)$, the surrogate objective on state s_t (cf. equation (1)), where

$$L_{t,1}(\mu,\sigma) = \mathbb{E}_{a\sim\mathcal{N}(\mu,\sigma)}\left[A^{\pi^{\text{old}}}(s_t,a)\right] \tag{5}$$

or maximize $L_{t,2}(\mu, \sigma)$, the probability that the target policy is improved over the old value function, where

$$L_{t,2}(\mu,\sigma) = \mathbb{E}_{a\sim\mathcal{N}(\mu,\sigma)}\left[\mathbb{I}\{A^{\pi^{\text{old}}}(s_t,a) > 0\}\right] \tag{6}$$

while being subject to the following constraint:

$$KL(\mathcal{N}(\cdot|\mu^{\text{old}}(s_t),\sigma^{\text{old}}(s_t))||\mathcal{N}(\cdot|\mu,\sigma)) \leq \delta. \tag{7}$$

In the second step, the policy network learns to match the proposed targets by minimizing the mean squared error with respect to these target statistical parameters.

Notice that typically only one estimate $\hat{A}_t := \hat{A}^{\pi^{\text{old}}}(s_t, a_t)$ is known. Therefore, the above optimization problems cannot be solved exactly and there is a tradeoff between exploitation and exploration for the target distribution, i.e., we can move the target distribution to the "best" area indicated by the estimate and shrink the variance of the distribution (exploit) or we can increase the variance for a better estimation of the advantage function (explore). In policy gradient methods, the mean and the variance of the action distribution are updated jointly subjected to the law given by the gradient of the probability density function. However, in TDL, the mean and the variance can be updated independently to implement different action search strategies.

Next, we propose three algorithms based on TDL, *TDL-direct*, *TDL-ES* and *TDL-ESr*. The three algorithms differ in the way they propose target distributions. The pseudocode for TDL is shown in Algorithm 1, the details of which are described in the following paragraphs.

Target variance

The three algorithms update the variance in the same way described as follows. The update rule allows an adaptation for exploration while changes the variance slowly which prevents premature convergence and violation of the constraint.

Inspired by the self-adaption technique (Hansen and Ostermeier 2001; Hansen 2000), given one state sample, action sample and the corresponding advantage estimate $(s_t, a_t, \hat{A}_t)$, the target variance is proposed as:

$$\hat{\sigma}_t^2 = (a_t - \mu^{\text{old}}(s_t))^2\mathbb{I}\{\hat{A}_t > 0\} + (\sigma^{\text{old}}(s_t))^2\mathbb{I}\{\hat{A}_t \leq 0\}, \tag{8}$$

where $\mu^{\text{old}}(\cdot)$ and $\sigma^{\text{old}}(\cdot)$ are the values of $\mu_\theta(\cdot)$ and $\sigma_\theta(\cdot)$ in the last iteration. When the advantage estimate is positive, if the action sample is within the one-sigma range, the target variance will be smaller than the old variance, otherwise it gets larger than the old variance for further exploration. When the advantage estimate is negative, the target variance remains the same as the old variance. Notice that when the advantage estimator is non-informative, the target variance remains the same as the old in expectation, i.e., $\mathbb{E}_{a_t\sim\mathcal{N}(\mu^{\text{old}}(s_t),\sigma^{\text{old}}(s_t))}[\hat{\sigma}_t^2] = \sigma^{\text{old}}(s_t)^2$. This prevents a drift of the action distribution when the critic is not well learned.

To prevent large variance update that may lead to a violation of the KL divergence constraint, the variance on a state s_t is designed to be the average of a state independent component and a state dependent component as follows:

$$\sigma_\theta(s_t) = \sigma^{1/(\varphi+1)}\tilde{\sigma}_\theta(s_t)^{\varphi/(\varphi+1)}, \tag{9}$$

Algorithm 1 Target learning

Number of timesteps in one iteration T, minibatch size M, number of epochs E
Initialize the action distribution of the policy $\mathcal{N}(\mu_\theta(\cdot), \sigma_\theta(\cdot) = \sigma^{1/(\varphi+1)}\tilde{\sigma}_\theta(\cdot)^{\varphi/(\varphi+1)})$
Initialize the critic network $V_\phi(\cdot)$
for $i = 0, 1, 2, \cdots$ **do**
  Interact with the environment and obtain T on-policy transitions $\{(s_t, a_t, r_t, s_{t+1})\}$
  Calculate the Monte Carlo return for each transition $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
  Calculate the advantage function estimate $\hat{A}_t$ for each transition (by GAE)
  Calculate the target standard deviation $\hat{\sigma}_t$ following (8)
  if TDL-direct **then** calculate the target means $\hat{\mu}_t$ following (11)
  if TDL-ESr **then** revise the sampled actions following (15)
  if TDL-ES **or** TDL-ESr **then** calculate the target means $\hat{\mu}_t$ following (12)
  for $j = 1 : ET/M$ **do**
    Sample a minibatch that contains M transitions
    Update the policy network to minimize $\frac{1}{M}\sum_{t=1}^{M}(\hat{\mu}_t - \mu_\theta(s_t))^2$ and $\frac{1}{M}\sum_{t=1}^{M}(\hat{\sigma}_t - \tilde{\sigma}_\theta(s_t))^2$ on the minibatch
    Update the critic network to minimize $\frac{1}{M}\sum_{t=1}^{M}(R_t - V_\phi(s_t))^2$ on the minibatch
  Update σ to $\hat{\sigma}$ defined in (10) and $\sigma_\theta(\cdot)$ following (9)

where $\varphi > 0$ is a hyperparameter that controls the degree to which the variances on different states are updated independently. $\tilde{\sigma}_\theta(s_t)$ is the state dependent component which is a neural network trained to minimized the MSE w.r.t. $\hat{\sigma}_t$. σ is the state independent component and directly updated to $\hat{\sigma}$ in each iteration, where

$$\hat{\sigma}^2 = \frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_t^2. \tag{10}$$

The state independent component is based on all the T samples in the iteration and therefore allows a global adaptation for exploration while changes slowly. With the state independent component, the variance changes slowly in one iteration. Empirically in our later experiments on Mujoco (Todorov, Erez, and Tassa 2012), with $\varphi = 1$ and $T = 2048$, each dimension of $\sigma_\theta(s_t)/\sigma^{\text{old}}(s_t)$ for any state falls within $[1-\epsilon, 1+\epsilon]$ for a $\epsilon \leq 0.01$.

TDL-direct algorithm

Consider a state sample s_t, an action sample a_t and its advantage estimate $\hat{A}_t$. Given that the update of the variance is small in each iteration, TDL-direct sets the target mean $\hat{\mu}_t$ to μ that maximizes $\mathcal{N}(a_t\,|\mu, \sigma^{\text{old}}(s_t))\hat{A}_t$ (i.e., the Monte Carlo estimate of $L_{t,1}$) subject to the constraint in (7).

Recall that the action a_t is sampled from $\mathcal{N}(\mu^{\text{old}}(s_t), \sigma^{\text{old}}(s_t))$. Hence, we can write $a_t = \mu^{\text{old}}(s_t) + y_t\sigma^{\text{old}}(s_t)$, where $y_t \sim \mathcal{N}(0, I)$. Let $\alpha > 0$ be a hyperparameter controlling the size of the trust region. The target mean for the sample at timestep t is proposed as

$$\hat{\mu}_t = \mu^{\text{old}}(s_t) + \text{sign}(\hat{A}_t)\min\left(1, \frac{\sqrt{2\alpha}}{\|y_t\|_2}\right) y_t\sigma^{\text{old}}(s_t), \tag{11}$$

where $\text{sign}(\cdot)$ is the sign function. When $\hat{A}_t > 0$, $\mathcal{N}(a_t\,|\mu, \sigma)\hat{A}_t$ can be maximized by setting $\mu = a_t$. When $\hat{A}_t \leq 0$, it is preferred that μ should be as far away from a_t as possible. However, this may violate the constraint in (7). Thus, we clip the amount of the change from $\mu^{\text{old}}(s_t)$ to $\hat{\mu}_t$ such that $KL(\mathcal{N}(\cdot|\mu^{\text{old}}(s_t), \sigma^{\text{old}}(s_t))||\mathcal{N}(\cdot|\hat{\mu}_t, \sigma(s_t))) \leq d\alpha(1+2\epsilon) + o(\epsilon^2) \approx d\alpha$, recalling that the standard deviation changes within $[1-\epsilon, 1+\epsilon]$ in each iteration. In Appendix A, we show that the above clip operation can guarantee that the KL constraint is satisfied (by leveraging the fact that the KL divergence between two Gaussian distributions has a closed-form formula).

TDL-ES algorithm

$(1+1)$-ES, one of the evolutionary strategies (ES) (Beyer and Schwefel 2002), can be used to maximize $L_{t,1}$. It is a family of optimization techniques for finding a distribution $\mathcal{D}$ that maximizes $\mathbb{E}_{x\sim\mathcal{D}}[f(x)]$, for a blackbox *fitness* function f. Natural evolutionary strategy (NES) (Wierstra et al. 2014) provides an algorithm based on $(1+1)$-ES that iteratively updates a Gaussian distribution to optimize the objective along a natural gradient direction. Specifically, in each iteration, an *offspring* x is sampled from the Gaussian distribution $\mathcal{N}(\mu^{\text{old}}, \sigma^{\text{old}})$ centering at the *parent* μ^{old} and the distribution is updated based on the comparison between the fitness of the offspring $f(x)$ and that of the parent $f(\mu^{\text{old}})$.

We observe that the objective $L_{t,1}$ is essentially the same as the objective for $(1+1)$-ES. By letting $\mu^{\text{old}} = \mu^{\text{old}}(s_t)$ and $x = a_t$, $\hat{A}_t$, which is an estimate of $Q^{\pi^{\text{old}}}(s_t, a_t) - V^{\pi^{\text{old}}}(s_t)$, can be used to indicate $f(x) - f(\mu^{\text{old}})$. In this way, the optimization problem defined in (5) can be solved by $(1+1)$-ES. Therefore, the target mean can be proposed by the update rule for NES, as follows:

$$\hat{\mu}_t = \mu^{\text{old}}(s_t) + \nu\mathbb{I}\{\hat{A}_t > 0\}(a_t - \mu^{\text{old}}(s_t)), \tag{12}$$

where $\nu \in (0, 1]$ is the step size. For the update of the variance, we still use the aforementioned update rule. [2]

[2] We found that the variance easily explodes following the NES rules to set the target variance in RL context. Hence, we choose to keep using the update rules described in (8) and (10), where (8)

TDL-ES algorithm with target statistical parameters defined in (8), (10) and (12) has the following properties.

First, for a typical action sample a_t, the proposed target statistical parameters satisfy the constraint in (7), i.e., $\mathbb{E}[KL(\mathcal{N}(\cdot|\mu^{\text{old}}(s_t), \sigma^{\text{old}}(s_t))||\mathcal{N}(\cdot|\hat{\mu}_t, \sigma(s_t)))] \leq \frac{1}{2}d\nu^2(1+2\epsilon) + o(\epsilon^2) \approx \frac{1}{2}d\nu^2$ (cf. Appendix A).

Second, (12) and (8) can be regarded as a stochastic gradient ascent step w.r.t. $L_{t,2}$ for some step sizes λ_μ and λ_σ (cf. Appendix B), i.e.,

$$\hat{\mu}_t = \mu^{\text{old}}(s_t) + \lambda_\mu \left.\frac{\partial L_{t,2}(\mu, \sigma^{\text{old}}(s_t))}{\partial \mu}\right|_{\mu=\mu^{\text{old}}(s_t)}, \quad (13)$$

$$\hat{\sigma}_t^2 = (\sigma^{\text{old}}(s_t))^2 + \lambda_\sigma \left.\frac{\partial L_{t,2}(\mu^{\text{old}}(s_t), \sigma)}{\partial \sigma^2}\right|_{\sigma=\sigma^{\text{old}}(s_t)}. \quad (14)$$

Third, consider one policy improvement step in TDL-ES and denote $D := \{a|Q^{\pi^{\text{old}}}(s_t, a) > V^{\pi^{\text{old}}}(s_t)\}$ for a state s_t which represents the "good" areas in the action space indicated by the value functions of the old policy. TDL-ES updates the standard deviation of the action distribution towards the (truncated) "radius" of D and the mean of the action distribution towards the "center" of D. This is appealing, since when the actor performs poorly (leading to a small $V(s_t)$ and a large D), it keeps exploring. In addition, when the critic estimate is overly large or small (leading to very large or very small D), the action distribution remains the same in expectation. In contrast, a vanilla policy gradient method under a similar setting updates the variance of the action distribution towards zero and the mean of the action distribution towards $arg\max_a Q^{\pi^{\text{old}}}(s_t, a)$. This may lead to premature convergence. See the detail in Appendix B.

TDL-ESr algorithm

Both TDL-direct and TDL-ES propose the target mean based solely on the action sample a_t from the state s_t and ignore the temporal structure of MDP. According to the observation that the state representation does not change too fast in adjacent steps, we can revise the formulation for the target mean in TDL-ES (i.e., (12)) by the information of $2N+1$ adjacent samples $(a_{t+t'}, \hat{A}_{t+t'}), t' \in [-N, N]$, resulting in a revised version of TDL-ES which we call TDL-ESr. The revised formula is the same as (12), except that we substitute a_t with $\tilde{a}_t$. Suppose $a_t = \mu^{\text{old}}(s_t) + y_t\sigma^{\text{old}}(s_t)$ is obtained by sampling $y_t \sim \mathcal{N}(0, I)$. For a revising ratio $r \in [0, 1]$, $\tilde{a}_t$ can be defined as $\tilde{a}_t = \mu^{\text{old}}(s_t) + \tilde{y}_t\sigma^{\text{old}}(s_t)$, where

$$\begin{aligned} \tilde{y}_t &= (1-r)y_t + ry'_t, \\ y'_t &= \frac{\sum_{t'=-N}^{N} y_t \max(0, \hat{A}_{t+t'})}{\sum_{t'=-N}^{N} \max(0, \hat{A}_{t+t'})}. \end{aligned} \quad (15)$$

Recall that, in TDL-ES, the mean of the action distribution moves to the direction indicated by y_t. This revision makes the mean update tilt to a direction y'_t indicated by adjacent "good" samples, i.e., samples with $\hat{A}_{t+t'} > 0$. Consider a case where an action sample a_t yields a large reward and results in a large $\hat{A}_t$, indicating that this is potentially a good direction. In TDL-ESr, the mean updates on the adjacent states will tilt towards this direction. This yields a directional exploration.

can be regarded as the first order Taylor approximation of the NES rules for the variance update.

5 Related Work

Conservative policy iteration. TDL follows conservative policy iteration (CPI) (Kakade and Langford 2002; Scherrer 2014; Agarwal et al. 2019), which suggests a small policy update in each iteration to ensure monotonic policy improvement and avoid oscillations. Several previous methods (Schulman et al. 2015a; 2017; Wu et al. 2017; Novati and Koumoutsakos 2019) also follow CPI and constrain the *parameters* of the policy directly. Instead, ACER (Wang et al. 2017) clips the gradient on the *statistical parameters* produced by the policy network. Similarly, TDL also constrains the policy update in the statistical parameter space but proposes target distributions that satisfy the constraint. Safe policy iteration (Pirotta et al. 2013) proposes to update the policy with different step sizes on different states to ensure policy improvement and is characterized by faster convergence and guaranteed policy improvement. TDL benefits from the similar idea but with a simpler rule to determine the step sizes on different states.

Setting targets. Setting target policies is an effective way to guide the learning of the policy network. In large discrete action domain, AlphaGo Zero (Silver et al. 2017) uses MCTS to propose a categorical distribution as the target for the policy network to learn. In continuous action domain, MPO (Abdolmaleki et al. 2018b) and Relative Entropy Regularized Policy Iteration (Abdolmaleki et al. 2018a) set the target action distribution to be an improvement over the estimated Q-function. TDL uses a state value function which is typically easier to estimate than a Q-function, and perform in more robust manner upon near-deterministic policies, as shown in our experiments.

Instability problem. The analysis on the variance of the policy gradient update in (Zhao et al. 2011) implies the instability problem on near-deterministic policies. In addition, (Van Hasselt and Wiering 2007; Hämäläinen et al. 2018) state that updating along the negative direction upon a "bad" action sample may cause instability. We empirically show that this may lead to instability but can accelerate the learning process when the action space dimension or the step size is small (cf. Appendix D).

6 Experiments

We conduct several experiments in order to demonstrate the following: 1) The performance of our algorithms on continuous control benchmark tasks is comparable with (or better than) the state-of-the-art algorithms; 2) We can safely increase the on-policy sample reuse without damaging the performance of our algorithms; 3) Our algorithms can constrain the maximum KL divergence across the state space more effectively than TRPO and PPO.

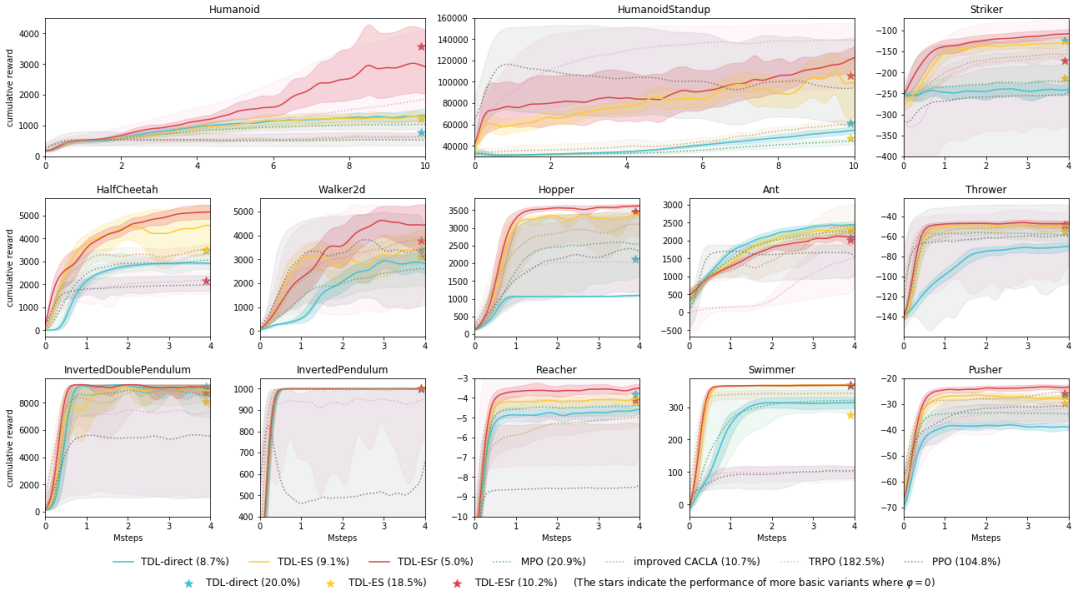

Figure 2: Comparison of several algorithms on MuJoCo tasks. The lines indicate the moving average across five independent runs and the shaded areas indicate the 10% and 90% quantiles. The percentage numbers in the legend indicate the normalized fluctuation of the scores in the last 100 iterations averaged over all the tasks.

Performance on continuous control benchmarks

We implemented TDL-direct, TDL-ES and TDL-ESr for the continuous control tasks provided by OpenAI Gym (Brockman et al. 2016) using MuJoCo simulator (Todorov, Erez, and Tassa 2012). Due to space limit, the detailed setting of hyperparameters can be found in Appendix G. In our experiments, we compare our algorithms against TRPO (Schulman et al. 2015a) and PPO (Schulman et al. 2017) (the clipped version) which are two popular policy gradient-based algorithms. We also compare our algorithms with MPO (Abdolmaleki et al. 2018b) and an improved version of CACLA (Van Hasselt and Wiering 2007). Our algorithms differ from MPO in the way we set target distributions. Specifically, we set target statistical parameters, whereas MPO sets target probability density values on action samples based on an estimated Q-function. The improved version of CACLA is actually an ablated version of TDL-ES without setting targets, i.e., the policy network is directly updated by stochastic gradient ascent on $L_{t,2}$. We show the results in Figure 2.

We can see that at least one of our algorithms outperform the previous algorithms in most tasks. TDL-ES performs better than the improved CACLA as setting targets prevents destructively large updates. Our algorithms generally outperform MPO which illustrates the effectiveness of the way we set target distributions. Moreover, the performance fluctuation during the training (and especially at the end of the training) for our algorithms is typically small (see also the normalized fluctuation in the legend of Figure 2), which indicates that the training processes of our algorithms are more stable and steadily improved over iterations. This also indicates that our algorithms are robust across different seeds. For tasks that require a precise control such as `Reacher` and `InvertedPendulum`, our algorithms result in a higher average cumulative reward than TRPO and PPO. This is due to the fact that our algorithms address the instability issue illustrated in Section 3. In particular, TDL-ESr performs the best on most tasks, especially the ones with large action space dimensions (such as `Humanoid`).

On-policy sample reuse

Unlike off-policy algorithms that use past experiences to improve sample efficiency, on-policy algorithms can improve the sample efficiency by learning more epochs on the same on-policy samples. We compare our algorithms against PPO with different level of sample reuse and show the result in Figure 3. Notice that PPO is quite similar to our algorithms and the main difference is that PPO updates along the policy gradient and ours update to match the target distributions. We see that, in PPO, although the sample efficiency improves from the increase of the sample reuse, the performance gets damaged. In contrast, TDL methods avoid this issue and we can safely increase the sample reuse. This is due to the fact that the policy network in TDL learns to match the fixed target distributions, whereas the policy network in PPO updates along the policy gradient of a clipped surrogate objective. In PPO, when iteratively optimizing this objective, more samples are masked by the clipping and the action distributions conditioned on the corresponding state samples may stray away.

KL divergence constraints

Our algorithms, like TRPO and PPO, rely on a conservative policy iteration that requires a constraint in the state-action space. This experiment is designed to evaluate how effective

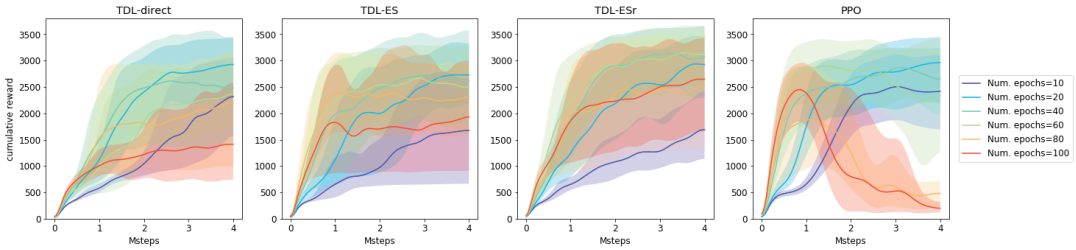

Figure 3: Performance of TDL-direct, TDL-ES, TDL-ESr and PPO on `Hopper-v2` task with different levels of sample reuse. `Num. epochs` denotes the average number of times that a sample is used the for the neural network update. The lines indicate the moving average across five independent runs and the shaded areas indicate the 10% and 90% quantiles.

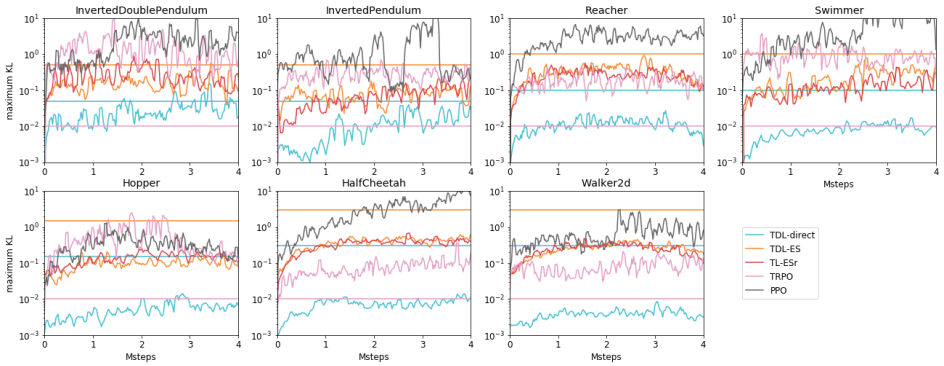

Figure 4: The maximum KL divergence during the training on different tasks. The mean KL constraint in TRPO is set to $\delta = 0.01$ and the level is indicated by the straight pink lines (bottom). The maximum KL bounds on samples for TDL-direct and TDL-ES(r) are indicated by the straight blue lines (middle) and the straight orange lines (top) respectively.

these algorithms can enforce such a constraint. TRPO, PPO and our algorithms aim to constrain the maximum KL divergence across the state space. In this experiment, we approximate the maximum KL divergence across the state space by first sampling a holdout set of 2048 transitions in each iteration and then recording the maximum KL divergence of the action distributions conditioned on each state sample in the holdout set over the iteration.

We show the results in Figure 4. First, we observe that in our algorithms the maximum KL divergences are effectively bounded by the limits we set while in TRPO the maximum KL divergences can be up to two orders of magnitude larger than the prescribed limits. Second, the maximum KL divergences in our algorithms (especially in TDL-direct) are generally smaller than those of TRPO and PPO, indicating a more conservative policy update in our algorithms. Thus, our algorithms result in a more stably improved performance during the training while achieving a comparable asymptotic performance and sample efficiency. The result indicates that the sample-wise constraint in TDL is more effective in enforcing a global constraint in the state-action space than previous methods.

7 Conclusion

We proposed a new method, called *target distribution learning*, to optimize stochastic policies for continuous control. This method proposes target distributions in each iteration and then trains the policy network to match these distributions. It enables a safe increase in the sample reuse to improve the sample efficiency for an on-policy algorithm. We designed three algorithms via this method. These algorithms can effectively impose constraint on the state-action space and avoid the instability problem of some prior policy gradient-based methods. Empirically, our algorithms achieve comparable performances to some state-of-the-art algorithms on a set of continuous control benchmark tasks.

In this paper, we focus on on-policy algorithms and Gaussian distribution for the action distribution. However, *target distribution learning* can be readily extended to off-policy settings, other types of action distributions and other types of constraints in the state-action space. We leave the extension as an interesting future direction.

Acknowledgement

The research is supported in part by the National Natural Science Foundation of China Grant 61822203, 61772297, 61632016, 61761146003, and the Zhongguancun Haihua Institute for Frontier Information Technology and Turing AI Institute of Nanjing.

References

Abdolmaleki, A.; Springenberg, J. T.; Degrave, J.; Bohez, S.; Tassa, Y.; Belov, D.; Heess, N.; and Riedmiller, M. 2018a. Relative Entropy Regularized Policy Iteration. *arXiv preprint arXiv:1812.02256*.

Abdolmaleki, A.; Springenberg, J. T.; Tassa, Y.; Munos, R.; Heess, N.; and Riedmiller, M. 2018b. Maximum a Posteriori Policy Optimisation. In *Proc. 6th ICLR*.

Agarwal, A.; Kakade, S. M.; Lee, J. D.; and Mahajan, G. 2019. Optimality and Approximation with Policy Gradient Methods in Markov Decision Processes. *arXiv preprint arXiv:1908.00261*.

Beyer, H.-G., and Schwefel, H.-P. 2002. Evolution Strategies: A Comprehensive Introduction. *Natural Computing* 1(1):3–52.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

Dabney, W.; Ostrovski, G.; Silver, D.; and Munos, R. 2018. Implicit Quantile Networks for Distributional Reinforcement Learning. *arXiv preprint arXiv:1806.06923*.

Grüttner, M.; Sehnke, F.; Schaul, T.; and Schmidhuber, J. 2010. Multi-dimensional Deep Memory Atari-Go Players for Parameter Exploring Policy Gradients. In *Proc. 19th ICANN*, 114–123.

Gu, S.; Lillicrap, T.; Sutskever, I.; and Levine, S. 2016. Continuous Deep Q-learning with Model-based Acceleration. In *Proc. 33rd ICML*, 2829–2838.

Hämäläinen, P.; Babadi, A.; Ma, X.; and Lehtinen, J. 2018. PPO-CMA: Proximal Policy Optimization with Covariance Matrix Adaptation. *arXiv preprint arXiv:1810.02541*.

Hansen, N., and Ostermeier, A. 2001. Completely Derandomized Self-adaptation in Evolution Strategies. *Evolutionary Computation* 9(2):159–195.

Hansen, N. 2000. Invariance, Self-adaptation and Correlated Mutations in Evolution Strategies. In *Proc. 6th PPSN*, 355–364. Springer.

Kakade, S., and Langford, J. 2002. Approximately Optimal Approximate Reinforcement Learning. In *Proc. 19th ICML*, 267–274.

Kullback, S., and Leibler, R. A. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22(1):79–86.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous Control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971*.

Liu, G.; Zhao, L.; Yang, F.; Bian, J.; Qin, T.; Yu, N.; and Liu, T.-Y. 2019. Trust Region Evolution Strategies. In *Proc. 33rd AAAI*, 4352–4359.

Mania, H.; Guy, A.; and Recht, B. 2018. Simple Random Search Provides a Competitive Approach to Reinforcement Learning. *arXiv preprint arXiv:1803.07055*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level Control through Deep Reinforcement Learning. *Nature* 518(7540):529.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proc. 33rd ICML*, 1928–1937.

Novati, G., and Koumoutsakos, P. 2019. Remember and Forget for Experience Replay. In *Proc. 36th ICML*, 4851–4860.

Pirotta, M.; Restelli, M.; Pecorino, A.; and Calandriello, D. 2013. Safe Policy Iteration. In *Proc. 30th ICML*, 307–315.

Rechenberg, I. 1973. Evolutionsstrategie–Optimierung Technischer Systeme nach Prinzipien der Biologischen Information. *Stuttgart-Bad Cannstatt: Friedrich Frommann Verlag*.

Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; and Sutskever, I. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *arXiv preprint arXiv:1703.03864*.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952*.

Scherrer, B. 2014. Approximate Policy Iteration Schemes: A Comparison. In *Proc. 31st ICML*, 1314–1322.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M. I.; and Moritz, P. 2015a. Trust Region Policy Optimization. In *Proc. 32nd ICML*, volume 37, 1889–1897.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015b. High-dimensional Continuous Control using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic Policy Gradient Algorithms. In *Proc. 31st ICML*, 387–395.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the Game of Go without Human Knowledge. *Nature* 550(7676):354.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Introduction to Reinforcement Learning*, volume 135. MIT press Cambridge.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A Physics Engine for Model-based Control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Van Hasselt, H., and Wiering, M. A. 2007. Reinforcement Learning in Continuous Action Spaces. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 272–279. IEEE.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-learning. In *Proc. 30th AAAI*, 2094–2100.

Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; and De Freitas, N. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *Proc. 33rd ICML*, 1995–2003.

Wang, Z.; Bapst, V.; Heess, N.; Mnih, V.; Munos, R.; Kavukcuoglu, K.; and de Freitas, N. 2017. Sample Efficient Actor-Critic with Experience Replay. In *Proc. 5th ICLR*.

Wierstra, D.; Schaul, T.; Glasmachers, T.; Sun, Y.; Peters, J.; and Schmidhuber, J. 2014. Natural Evolution Strategies. *The Journal of Machine Learning Research* 15(1):949–980.

Williams, R. J. 1992. Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning. *Machine learning* 8(3-4):229–256.

Wu, Y.; Mansimov, E.; Grosse, R. B.; Liao, S.; and Ba, J. 2017. Scalable Trust-region Method for Deep Reinforcement Learning using Kronecker-factored Approximation. In *Proc. 31st NeurIPS*, 5279–5288.

Zhao, T.; Hachiya, H.; Niu, G.; and Sugiyama, M. 2011. Analysis and Improvement of Policy Gradient Estimation. In *Proc. 25th NeurIPS*, 262–270.