Trust Region Evolution Strategies

Guoqing Liu[†][*], Li Zhao[‡], Feidiao Yang[‡][§], Jiang Bian[‡], Tao Qin[‡], Nenghai Yu[†], Tie-Yan Liu[‡]
[†]University of Science and Technology of China
[‡]Microsoft Research
[§]Institute of Computing Technology, Chinese Academy of Sciences
lgq1001@mail.ustc.edu.cn; {lizo, jiang.bian, taoqin, tyliu}@microsoft.com; yangfeidiao@ict.ac.cn

Abstract

Evolution Strategies (ES), a class of black-box optimization algorithms, has recently been demonstrated to be a viable alternative to popular MDP-based RL techniques such as Q-learning and Policy Gradients. ES achieves fairly good performance on challenging reinforcement learning problems and is easier to scale in a distributed setting. However, standard ES algorithms perform one gradient update per data sample, which is not very efficient. In this paper, with the purpose of more efficient using of sampled data, we propose a novel iterative procedure that optimizes a surrogate objective function, enabling to reuse data sample for multiple epochs of updates. We prove monotonic improvement guarantee for such procedure. By making several approximations to the theoretically-justified procedure, we further develop a practical algorithm called Trust Region Evolution Strategies (TRES). Our experiments demonstrate the effectiveness of TRES on a range of popular MuJoCo locomotion tasks in the OpenAI Gym, achieving better performance than ES algorithm.

Introduction

Developing agents that can accomplish challenging tasks in complex, uncertain environments is a primary goal of Reinforcement Learning (RL). The most popular paradigm for analyzing such problem has been using a class of MDP-based algorithms, such as DQN (Mnih et al. 2015), DDPG (Lillicrap et al. 2015), and TRPO (Schulman et al. 2015), which have gained significant achievements after applying to a variety of applications, including Atari games (Mnih et al. 2015; Van Hasselt, Guez, and Silver 2016; Mnih et al. 2016), robot locomotion tasks (Schulman et al. 2015; 2017) and the game of Go (Silver et al. 2016; 2017).

Recently, as a class of black-box optimization algorithms, Evolution Strategies (ES) (Salimans et al. 2017) has been risen to be a viable alternative to popular MDP-based RL techniques. As comparable to state-of-the-art policy gradient methods, ES has demonstrated its success in training policies on a variety of simulated robotics tasks in MuJoCo and on Atari games. With competitive performance and good parallelization, ES algorithms are receiving more and more attention as a scalable alternative to popular MDP-based RL techniques.

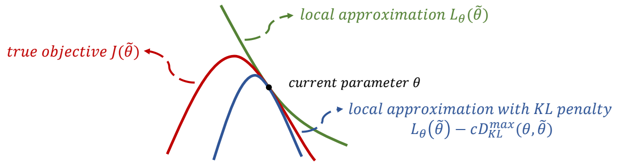

local approximation $L_\theta(\tilde{\theta})$
true objective $J(\tilde{\theta})$
current parameter θ
local approximation with KL penalty
$L_\theta(\tilde{\theta}) - cD_{KL}^{max}(\theta,\tilde{\theta})$

Figure 1: Red curve represents true objective. Green curve represents local approximation. Blue curve represents local approximation with KL penalty, which is the surrogate objective function. The surrogate objective function forms a lower bound of the true objective, so optimizing this surrogate objective is guaranteed to improve the true objective.

Compared to MDP-based RL techniques, ES algorithms employ a different path of techniques by directly searching optimal policy in parameter space. To be specific, the ES algorithm in (Salimans et al. 2017) proposed to optimize the *Gaussian smoothing* of objective function,

$$J(\theta) = \mathbb{E}_{X\sim\mathcal{N}(\theta,\sigma^2 I)}[F(X)], \tag{1}$$

where $F(X)$ is the objective function representing the expected total reward. In each training iteration of above ES algorithm, two phrases are executed sequentially: 1) Sampling search directions ϵ_i from Gaussian distribution $\mathcal{N}(0,\sigma^2 I)$ and evaluating these search directions by sampling data using policy with parameter $(\theta + \epsilon_i)$ in the environment. 2) Using sampled data to estimate the gradient which are then applied to update the corresponding parameter. Given such procedure, in order to enhance ES algorithms, one vital direction is to improve sample efficiency, which is not that high in current ES algorithm since the sampled data is only used for one gradient update.

To boost sample efficiency, it is essentially comprised of two major aspects. One aspect concerns how to sample better or more diverse search directions, and the other one corresponds to how to make more efficient use of existing sampled data. Some recent works (Choromanski et al. 2018; Maheswaranathan et al. 2018) have been proposed for improving sample efficiency of ES algorithms from the first

[*]This work was done when the first author was on an internship with Microsoft Research.

aspect. However, to the best of our knowledge, there is no existing work exploring another perspective: given the sampled data, how can we make more efficient use of them?

To take deeper investigation on the other aspect of sample efficiency, in this paper, we propose a novel iterative procedure that introduces a surrogate objective, by optimizing which can give rise to efficiently reusing sample data for multiple epochs of updates. In particular, we first introduce a local approximation $L_\theta(\tilde{\theta})$ to the *Gaussian smoothing* objective $J(\tilde{\theta})$ around current parameter θ where we sample data, with the purpose to approximately optimize $J(\tilde{\theta})$ by reusing sampled data from θ instead of sampling new data from $\tilde{\theta}$. However, this local approximation only works well when $\tilde{\theta}$ is close enough to current parameter θ, as shown in Figure 1. It's still not clear that how much can we update when optimizing the local approximation $L_\theta(\tilde{\theta})$. To address this issue, we propose a novel procedure that optimizes a surrogate objective function, namely a local approximation with a KL divergence penalty, which forms a lower bound of the true objective $J(\tilde{\theta})$. We prove theoretical monotonic improvement guarantee for such procedure. At last, we make a series of approximations to the theoretically-justified procedure, yielding a practical algorithm, which leads to better performance than ES algorithm, called Trust Region Evolution Strategies (TRES).

To summarize, our contributions are as follows:

- Make more efficient use of sampled data by optimizing surrogate objective function for multiple epochs of updates, instead for just one gradient update.
- Prove theoretical monotonic improvement guarantee for this optimization procedure.
- Develop practical algorithm TRES after making several approximations. Experiments on five popular physical locomotion tasks in the OpenAI Gym demonstrate the effectiveness of TRES.

Background

The goal of reinforcement learning (RL) is to find, through trial and error, a feedback policy that prescribes how an agent should optimally act in a dynamic, uncertain environment. ES address RL problem by directly searching for optimal policy in parameter space, casted as a maximization problem of the form:

$$\max_{\theta \in \mathbb{R}^d} F(\theta), \tag{2}$$

where a family of deterministic policies mapping states to actions is parameterized by $\theta \in \mathbb{R}^d$, and the objective $F : \mathbb{R}^d \to \mathbb{R}$ measures the expected total reward of policy parameter θ.

However, when the environment is stochastic and implemented in black-box physics simulators, the objective function F is only accessible via noisy and expensive function evaluations, and may not be smooth. So the gradient of F cannot be computed with a backpropagation-like algorithm. That means we cannot directly use standard gradient-based optimization methods to find a good solution for θ.

In order to both make the problem smooth and to find a way to estimate its gradients, it is necessary to add noise. In this way, we turn to optimize the *Gaussian Smoothing* of the objective function,

$$\max_{\theta \in \mathbb{R}^d} J(\theta) = \mathbb{E}_{X \sim p_\theta(x)}[F(X)], \tag{3}$$

where $p_\theta(x)$ denotes the probability density function of multivariate Gaussian distribution with diagonal covariance matrix $\mathcal{N}(\theta, \sigma^2 I)$.

Such process can be interpreted as adding a Gaussian blur to the original objective, which results in a smooth, differentiable objective $J(\theta)$ that can be solve with stochastic gradient ascent.

Estimating Gradients of Gaussian Smoothing

The gradient of $J(\theta)$ is given by

$$\nabla J(\theta) = \frac{1}{\sigma^2} \mathbb{E}_{X \sim p_\theta(x)} \left[F(X)(X - \theta) \right]. \tag{4}$$

In practice, this gradient is intractable, and must be estimated. The gradient $\nabla J(\theta)$ can be estimated via a variety of Monte Carlo estimators. In this paper, we investigate two fundamental estimators:

Vanilla ES gradient estimator. The first estimator is derived via the Monte Carlo REINFORCE (Williams 1992) estimator, and coincides with a standard Monte Carlo estimator of the expectation appearing in Equation (4):

$$\hat{\nabla}_N^V J(\theta) = \frac{1}{N\sigma^2} \sum_{i=1}^{N} F(x_i)(x_i - \theta), \tag{5}$$

where $(x_i)_{i=1}^N \overset{\text{i.i.d}}{\sim} \mathcal{N}(\theta, \sigma^2 I)$ can be interpreted as search parameters sampled from search distribution $\mathcal{N}(\theta, \sigma^2 I)$. We refer to this gradient estimator as the vanilla ES gradient estimator.

Antithetic ES gradient estimator. Secondly, we consider the version of vanilla ES gradient estimator augmented with antithetic variables, as in (Salimans et al. 2017), given by

$$\hat{\nabla}_N^A J(\theta) = \frac{1}{2N\sigma^2} \sum_{i=1}^{N} ((F(x_i) - F(2 \cdot \theta - x_i))(x_i - \theta)), \tag{6}$$

where again $(x_i)_{i=1}^N \overset{\text{i.i.d}}{\sim} \mathcal{N}(\theta, \sigma^2 I)$. The two terms appearing in the summand are contributed by search parameter $x_i \sim \mathcal{N}(\theta, \sigma^2 I)$ and its antithetic counterpart $(2 \cdot \theta - x_i)$, We refer to this gradient estimator as the antithetic ES gradient estimator, owing to its use of antithetic Monte Carlo samples.

In summary, Evolution Strategies (ES) introduced by (Salimans et al. 2017) aims to optimize *Gaussian Smoothing* objective by directly using stochastic gradient ascent, with the antithetic ES gradient estimator, and each gradient update step of ES algorithm could be divided into two sub-steps: 1) Sampling search parameters $\{x_1, ..., x_N\}$ from search distribution $\mathcal{N}(\theta, \sigma^2 I)$ and evaluating these search parameters via doing rollouts in the environment. 3) Using sampled data $\{F(x_1), ..., F(x_N)\}$ to estimate the gradient and updating current parameter with the estimated gradient. For each parameter update, ES needs to sample new data to estimate the gradient, which is not very efficient.

Methodology

In this section, we present a new algorithm, Trust Region Evolution Strategies (TRES), with the purpose of more efficient using of sampled data. We will start with an introduction of a local approximation $L_\theta(\tilde{\theta})$ to the true objective $J(\tilde{\theta})$, aiming to approximately optimize $J(\tilde{\theta})$ by reusing sampled data from θ instead of sampling new data from $\tilde{\theta}$. Then, we derive that the difference between local approximation $L_\theta(\tilde{\theta})$ and true objective $J(\tilde{\theta})$ is bounded by a KL divergence term $D_{KL}^{max}(\theta, \tilde{\theta})$. Based on this bound, we propose to optimize a surrogate objective function, namely local approximation with a KL divergence penalty, which is a lower bound of the true objective $J(\tilde{\theta})$ and thus can provide explicit lower bounds on the improvement of $J(\tilde{\theta})$. After making a series of approximations to the theoretical procedure, we finally develop the practical TRES algorithm and propose a distributed implementation of it.

Local Approximation of Gaussian Smoothing

In this subsection, we introduce a local approximation to the true objective $J(\tilde{\theta})$ around current parameter θ where we sample data.

We start by introducing the difference of the true objective between new parameter $\tilde{\theta}$ and current parameter θ :

$$J(\tilde{\theta}) - J(\theta) = \mathbb{E}_{X\sim p_{\tilde{\theta}}(x)}[F(X)] - \mathbb{E}_{X\sim p_\theta(x)}[F(X)]. \quad (7)$$

Given that X stands for a d-dimension search parameter, we can rewrite it in vector form $[X^1, ..., X^d]$, where the superscript denotes the corresponding dimension. Thus Equation (7) becomes:

$$J(\tilde{\theta}) - J(\theta) = \mathbb{E}_{X^{1\cdots d}\sim\tilde{\theta}}[F(X)] - \mathbb{E}_{X^{1\cdots d}\sim\theta}[F(X)]. \quad (8)$$

For compactness, we use the notation $X^{1\cdots i}$ to represent the first i dimensions of X, namely $[X^1, ..., X^i]$ and the notation $p_\theta(x^{1\cdots i})$ to represent the marginal distribution of $X^{1\cdots i}$, for $i \in \{1, ..., d\}$. Besides, we further compress $p_\theta(x^{1\cdots i})$ down to θ in expectation forms.

We decompose this difference as a sum of per-dimension difference:

$$\sum_{i=0}^{d-1}\left[\mathbb{E}_{\substack{X^{1\cdots i+1}\sim\tilde{\theta}\\ X^{i+2\cdots d}\sim\theta}}[F(X)] - \mathbb{E}_{\substack{X^{1\cdots i}\sim\tilde{\theta}\\ X^{i+1\cdots d}\sim\theta}}[F(X)]\right]. \quad (9)$$

For convenience, we will give the following definitions of the value function V_θ, and the advantage function A_θ in our framework:

$$V_\theta(x^{1\cdots i}) = \mathbb{E}_{X^{i+1\cdots d}\sim\theta}\Big[F(X)|X^{1\cdots i} = x^{1\cdots i}\Big], \forall i = 0, ..., d.$$
$$A_\theta(x^{1\cdots i+1}) = V_\theta(x^{1\cdots i+1}) - V_\theta(x^{1\cdots i}), \forall i = 0, ..., d-1. \quad (10)$$

Specifically, when $i = 0$, $V_\theta(x^{1\cdots 0}) = \mathbb{E}_{X^{1\cdots d}\sim\theta}[F(X)] = J(\theta)$; when $i = d$, $V_\theta(x^{1\cdots d}) = F(x^{1\cdots d}) = F(x)$.

In other words, value function $V_\theta(x^{1\cdots i})$ represents the expectation of $F(X)$ when first i dimensions of X are given and remaining dimensions are sampled from $p_\theta(x^{i+1\cdots d})$, and advantage function $A_\theta(x^{1\cdots i+1})$ represents the expectation difference in whether $(i+1)$th dimension of X is given or sampled from $p_\theta(x^{i+1})$. We can rewrite Equation (9) with the notations described above:

$$\begin{aligned} J(\tilde{\theta}) &= J(\theta) + \mathbb{E}_{X\sim\tilde{\theta}}\left[\sum_{i=0}^{d-1}(V_\theta(X^{1\cdots i+1}) - V_\theta(X^{1\cdots i}))\right] \\ &= J(\theta) + \mathbb{E}_{X\sim\tilde{\theta}}\left[\sum_{i=0}^{d-1}A_\theta(X^{1\cdots i+1})\right]. \end{aligned} \quad (11)$$

Then we proceed to expand Equation (11):

$$\begin{aligned} J(\tilde{\theta}) &= J(\theta) + \sum_{i=0}^{d-1}\mathbb{E}_{X\sim\tilde{\theta}}\Big[A_\theta(X^{1\cdots i+1})\Big] \\ &= J(\theta) + \sum_{i=0}^{d-1}\sum_{x^{1\cdots i+1}}p_{\tilde{\theta}}(x^{1\cdots i+1})A_\theta(x^{1\cdots i+1}) \\ &= J(\theta) + \sum_{i=0}^{d-1}\sum_{x^{1\cdots i}}p_{\tilde{\theta}}(x^{1\cdots i})\sum_{x_{i+1}}p_{\tilde{\theta}}(x^{i+1})A_\theta(x^{1\cdots i+1}). \end{aligned} \quad (12)$$

The complex dependency of $p_{\tilde{\theta}}(x^{1\cdots i})$ on $\tilde{\theta}$ makes Equation (12) difficult to optimize directly without new sampled data over $\tilde{\theta}$. With the purpose to approximately optimize $J(\tilde{\theta})$ by reusing sampled data from θ instead of sampling new data from $\tilde{\theta}$, we introduce the following local approximation to $J(\tilde{\theta})$,

$$L_\theta(\tilde{\theta}) = J(\theta) + \sum_{i=0}^{d-1}\sum_{x^{1\cdots i}}p_\theta(x^{1\cdots i})\sum_{x^{i+1}}p_{\tilde{\theta}}(x^{i+1})A_\theta(x^{1\cdots i+1}). \quad (13)$$

Although $L_\theta(\tilde{\theta})$ uses $p_\theta(x^{1\cdots i})$ rather than $p_{\tilde{\theta}}(x^{1\cdots i})$, we can find that $L_\theta(\tilde{\theta})$ matches $J(\tilde{\theta})$ to zero order and first order. That is, for any parameter value θ,

$$\begin{aligned} L_\theta(\tilde{\theta})|_{\tilde{\theta}=\theta} &= J(\tilde{\theta})|_{\tilde{\theta}=\theta}, \\ \nabla_{\tilde{\theta}}L_\theta(\tilde{\theta})|_{\tilde{\theta}=\theta} &= \nabla_{\tilde{\theta}}J(\tilde{\theta})|_{\tilde{\theta}=\theta}. \end{aligned} \quad (14)$$

which implies that a sufficiently small step $\theta \to \tilde{\theta}$ that improves $L_\theta(\tilde{\theta})$ will also improve $J(\tilde{\theta})$.

Monotonic Improvement Guarantee

In this subsection, we first derive KL divergence bound of the difference between local approximation $L_\theta(\tilde{\theta})$ and true objective $J(\tilde{\theta})$. Based on this bound, we propose to optimize a surrogate objective function, namely local approximation with a KL divergence penalty, with theoretical monotonic improvement guarantee.

The particular distance measure we use here is the total variation divergence, which is defined by $D_{TV}(p||q) = \frac{1}{2}\sum_i |p_i - q_i|$ for discrete probability distributions p, q. Define $D_{TV}^{max}(\theta, \tilde{\theta})$ as

$$D_{TV}^{max}(\theta, \tilde{\theta}) = \max_{1\leq i\leq d} D_{TV}(\theta^i, \tilde{\theta}^i). \quad (15)$$

For simplicity, we use the notation $D_{TV}(\theta^i, \tilde{\theta}^i)$ to represent $D_{TV}(\mathcal{N}(\theta^i, \sigma^2), \mathcal{N}(\tilde{\theta}^i, \sigma^2))$.

Theorem 1 *Let $\alpha = D_{TV}^{max}(\theta, \tilde{\theta})$. Then the following bound holds:*

$$
\begin{aligned}
&|J(\tilde{\theta}) - L_\theta(\tilde{\theta})| \leq 2\epsilon d \cdot (d+1)\alpha^2 \\
&\quad where\, \epsilon = \max_{\substack{x^{1\ldots i} \\ 1\leq i \leq d}} |A_\theta(x^{1\ldots i})|.
\end{aligned} \tag{16}
$$

Theorem 1 provides a TV divergence bound of the difference between true objective $J(\theta)$ and local approximation $L_\theta(\tilde{\theta})$, and we defer the proof of Theorem 1 to supplementary material.

In consideration of the following relationship between the total variation divergence and the KL divergence: $D_{TV}(p||q)^2 \leq D_{KL}(p||q)$, we define $D_{KL}^{max}(\theta, \tilde{\theta}) = \max_{1\leq i\leq d} D_{KL}(\theta^i, \tilde{\theta}^i)$, and then the KL divergence bound follows directly from Theorem 1:

$$
\begin{aligned}
&J(\tilde{\theta}) \geq L_\theta(\tilde{\theta}) - c \cdot D_{KL}^{max}(\theta, \tilde{\theta}), \\
&\quad where\ c = 2\epsilon d \cdot (d+1).
\end{aligned} \tag{17}
$$

where we also use $D_{KL}(\theta^i, \tilde{\theta}^i)$ to represent $D_{KL}(\mathcal{N}(\theta^i, \sigma^2 I), \mathcal{N}(\tilde{\theta}^i, \sigma^2 I))$, and we define $D_{KL}^{max}(\theta, \tilde{\theta})$ as $\max_{1\leq i\leq d} D_{KL}(\theta^i, \tilde{\theta}^i)$.

Based on Equation (17), we further define $M_\theta(\tilde{\theta}) = L_\theta(\tilde{\theta}) - c \cdot D_{KL}^{max}(\theta, \tilde{\theta})$. Then

$$
\begin{aligned}
&J(\tilde{\theta}) \geq M_\theta(\tilde{\theta}) \text{ by Equation (17)}, \\
&J(\theta) = M_\theta(\theta) \text{ by definition}, \\
\Rightarrow &J(\tilde{\theta}) - J(\theta) \geq M_\theta(\tilde{\theta}) - M_\theta(\theta).
\end{aligned} \tag{18}
$$

Thus, by maximizing $M_\theta(\tilde{\theta})$ at each iteration, we guarantee that the true objective $J(\tilde{\theta})$ is non-decreasing. This algorithm is a type of minorization-maximization (MM) algorithm (Hunter and Lange 2004). In the terminology of MM algorithms, $M_\theta(\tilde{\theta})$ is called the surrogate objective function that minorizes $J(\tilde{\theta})$ with equality at θ.

Practical Algorithm

After justifying the effectiveness of optimizing a surrogate objective function, we now describe how to derive a practical algorithm from these theoretical foundations.

Constrained Optimization As indicated by Equation (18), by performing the following maximization, we are guaranteed to improve the true objective $J(\tilde{\theta})$:

$$
\max_{\tilde{\theta}} \Big[L_\theta(\tilde{\theta}) - c \cdot D_{KL}^{max}(\theta, \tilde{\theta}) \Big]. \tag{19}
$$

In practice, considering $c = 2\epsilon d \cdot (d+1)$, the penalty coefficient increases accordingly to the dimension size of the policy parameter, and then the update step size would be very small. A natural way to take larger steps in a robust way is to use a constraint on the $D_{KL}^{max}(\theta, \tilde{\theta})$, i.e., a trust region constraint, so we can transform Equation (19) to the following constraint optimization problem:

$$
\begin{aligned}
&\max_{\tilde{\theta}} L_\theta(\tilde{\theta}) \\
&\text{subject to } D_{KL}^{max}(\theta, \tilde{\theta}) \leq \delta.
\end{aligned} \tag{20}
$$

Sampled-Based Estimation We seek to solve the following optimization problem, obtained by expanding $L_\theta(\tilde{\theta})$ in Equation (20):

$$
\begin{aligned}
&max_{\tilde{\theta}} \sum_{i=0}^{d-1} \sum_{x^{1\ldots i}} p_\theta(x^{1\ldots i}) \sum_{x^{i+1}} p_{\tilde{\theta}}(x^{i+1}) A_\theta(x^{1\ldots i+1}) \\
&\text{subject to } D_{KL}^{max}(\theta, \tilde{\theta}) \leq \delta.
\end{aligned} \tag{21}
$$

We first replace the sum over first $i+1$ dimensions of x in this objective by the expectation:

$$
\begin{aligned}
&max_{\tilde{\theta}} \sum_{i=0}^{d-1} \mathbb{E}_{\substack{X^{1\ldots i} \sim \theta \\ X^{i+1} \sim \tilde{\theta}}} \left[A_\theta(X^{1\ldots i+1}) \right] \\
&\text{subject to } D_{KL}^{max}(\theta, \tilde{\theta}) \leq \delta.
\end{aligned} \tag{22}
$$

Note that in Equation (22), X^{i+1} still follows the distribution $p_{\tilde{\theta}}(X^{i+1})$, rather than $p_\theta(X^{i+1})$. we solve this problem by using an importance sampling estimator:

$$
\begin{aligned}
&max_{\tilde{\theta}} \sum_{i=0}^{d-1} \mathbb{E}_{X^{1\ldots i+1} \sim \theta} \left[\frac{p_{\tilde{\theta}}(X^{i+1})}{p_\theta(X^{i+1})} A_\theta(X^{1\ldots i+1}) \right] \\
&\text{subject to } D_{KL}^{max}(\theta, \tilde{\theta}) \leq \delta.
\end{aligned} \tag{23}
$$

Next we replace the advantage function $A_\theta(x^{1\ldots i+1})$ by the value function $V_\theta(x^{1\ldots i+1})$, which only changes the objective by a constant.

At last, our optimization problem in Equation (23) is exactly equivalent to the following one:

$$
\begin{aligned}
&\max_{\tilde{\theta}} \mathbb{E}_{X \sim \theta} \left[\sum_{i=0}^{d-1} \frac{p_{\tilde{\theta}}(X^{i+1})}{p_\theta(X^{i+1})} V_\theta(X^{1\ldots i+1}) \right] \\
&\text{subject to } D_{KL}^{max}(\theta, \tilde{\theta}) \leq \delta.
\end{aligned} \tag{24}
$$

All that remains is to replace the expectation by sample averages and replace the value function by an empirical estimate.

Clipped Surrogate Objective However, the optimization of Equation (24) is exactly a constrained optimization problem with box constraint, which is complicated and hard to solve precisely. Inspired by PPO (Schulman et al. 2017), we propose a novel surrogate objective with clipped probability ratio, which forms a lower bound of the local approximation $L_\theta(\tilde{\theta})$, and we can optimize this clipped objective via multiple epochs of gradient updates. Let $r_i(\tilde{\theta})$ denotes the probability ratio of ith dimension, namely, $r_i(\tilde{\theta}) = \frac{p_{\tilde{\theta}}(X^i)}{p_\theta(X^i)}$, the main objective $L_\theta^{CLIP}(\tilde{\theta})$ is the following:

$$
\begin{aligned}
&L_\theta^{CLIP}(\tilde{\theta}) = \mathbb{E}_{X \sim \theta} \left[\sum_{i=1}^{d} min(l_i(\tilde{\theta}),\ l_i^{CLIP}(\tilde{\theta})) \right] \\
&where\ l_i(\tilde{\theta}) = r_i(\tilde{\theta}) V_\theta(X^{1\ldots i}), \\
&l_i^{CLIP}(\tilde{\theta}) = clip(r_i(\tilde{\theta}), 1-\lambda, 1+\lambda) V_\theta(X^{1\ldots i}).
\end{aligned} \tag{25}
$$

The motivation for this objective is as follows: the first term $l_i(\tilde{\theta})$ is the ith dimension's contribution to local approximation $L_\theta(\tilde{\theta})$, and the second term $l_i^{CLIP}(\tilde{\theta})$ modifies $l_i(\tilde{\theta})$ by clipping the probability ratio, which removes the search parameters for moving r_i outside of the interval $[1-\lambda, 1+\lambda]$ and thus constrains the update step during training process. Finally, we take the minimum of the clipped and unclipped objective, so the final objective is a lower bound on the local approximation, which ensures that the improvement of local approximation $L_\theta(\tilde{\theta})$ when we optimize $L_\theta^{CLIP}(\tilde{\theta})$. Although this objective is not exactly constrained in trust region, we still call our algorithm Trust Region Evolution Strategies since in spirit it encourages the parameter to stay in trust region.

Distributed Implementation The resulting practical algorithm repeatedly executes two phases: 1) Sampling search parameters $\{x_1, ..., x_N\}$ from search distribution $\mathcal{N}(\theta, \sigma^2 I)$ and evaluating sampled search parameters by doing rollouts in the environment, and 2) Reusing sampled data $\{F(x_1), ..., F(x_N)\}$ and performing K epochs of gradient updates to optimize the clipped surrogate objective.

The evaluation of various search parameters in the first phase is well suited to be scaled up to many parallel workers: the rollout process of each search parameter operates independent of each other. Following the efficient communication strategy introduced by (Salimans et al. 2017), we share random seeds to sample search parameters between workers and each worker knows what search parameter the other workers used, so each worker only needs to communicate a single scalar to and from each other worker to agree on a parameter update, TRES thus requires extremely low bandwidth, in sharp contrast to policy gradient methods, which require workers to communicate entire gradients. We give the parallel version of TRES in Algorithm 1.

Algorithm 1 Trust Region Evolution Strategies

```
1: **Input:** noise standard deviation $\sigma$, clip factor $\lambda$, epoch number $K$, learning rate $\alpha$
2: **Initialize:** $N$ workers with known random seeds, and initial policy parameters $\theta_0$
3: for each iteration $t = 0, 1, 2, ...$ do
4:     for each worker $i = 1, ..., N$ do
5:         Sample $x_i \sim \mathcal{N}(\theta_t, \sigma^2 I)$
6:         Compute rollout returns $F(x_i)$
7:     end for
8:     Send all rollout returns to every worker
9:     for each worker $i = 1, .., N$ do
10:        Reconstruct all search parameters
11:        Compute value function with rollout returns
12:        Let $\theta_{t,1} = \theta_t$
13:        for each epoch $j = 1, ..., K$ do
14:            Reusing sampled data from $\theta_t$
15:            $\theta_{t,j+1} = \theta_{t,j} + \alpha \nabla L_{\theta_t}^{CLIP}(\theta_{t,j})$
16:        end for
17:        Update policy parameter via $\theta_{t+1} = \theta_{t,K+1}$
18:    end for
19: end for
```

Experiments

To demonstrate the effectiveness of TRES, we conducted experiments on the continuous MuJoCo locomotion tasks from the OpenAI Gym (Brockman et al. 2016). These tasks have been widely studied in the reinforcement learning community (Duan et al. 2016; Gu et al. 2016; 2017; Rajeswaran et al. 2017). In these tasks, the states of the robots are their generalized positions and velocities, and the controls are joint torques. And, these tasks are challenging due to under-actuation, high dimensionality, and non-smooth dynamics.

Experimental Settings

The empirical result from (Mania, Guy, and Recht 2018) shows that linear policies are sufficiently expressive to capture diverse behaviors in MuJoCo; thus, we use linear policy structure in all tasks we evaluated. Thus, the policy parameter size is equal to the product of state space dimension and action space dimension in the environment. We uniformly initialize policies with zero matrix before training in all tasks.

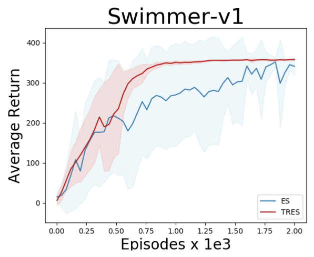
To reduce variance, We use the antithetic sampling (also known as mirrored sampling) technique following (Salimans et al. 2017). For some of tasks, such as Walker-v1 and Ant-v1, the default reward functions include a survival bonus, which rewards RL agents with a constant reward at each timestep as long as a terminal condition has not been reached. This may cause that finding policies are more likely to be local optimal. To resolve this problem, we subtract the the survival bonus from the rewards outputted as in (Mania, Guy, and Recht 2018).

To remove the influence of outlier individuals in sampled parameters and avoid falling into local optima, we perform fitness shaping (Sun et al. 2009; Wierstra et al. 2011; 2014; Salimans et al. 2017) by applying a rank transformation to the rollout returns before computing each parameter update. We also perform state normalization trick, which is widely used in various RL related works (Schulman et al. 2015; 2017; Wu et al. 2017; Mania, Guy, and Recht 2018) and is similar to data whiting used in regression tasks. Intuitively, it ensures that policies put equal weight on the different components of the states.

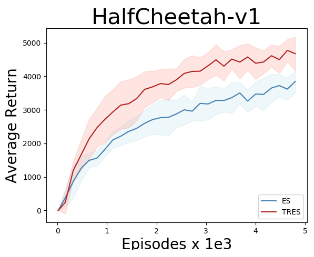
Our algorithms are built based on the implementation of OpenAI evolution-strategies-starter code (Salimans et al. 2017). Each algorithm has been evaluated on 5 environments, with 6 random seeds on each. For fair comparison, we sample six random seeds uniformly from the interval [0, 1000) and share them in all environments.

Main Results

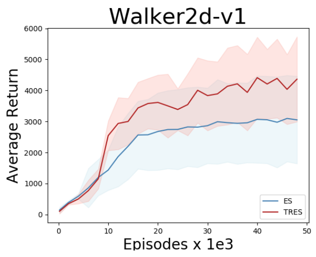
To compare the performance of TRES with those of ES, TRPO and PPO, Figure 2 shows the training curves of TRES and ES, respectively, and Table 1 illustrates the number of timesteps required to reach a prescribed reward threshold by TRES, ES, TRPO and PPO, respectively. From the figure and this table, we can see that TRES outperforms ES across all the tasks and provides competitive results with TRPO and



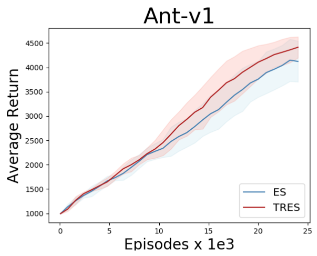
(a)



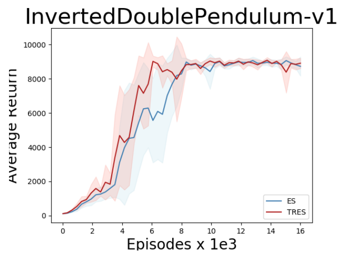
(b)



(c)



(d)



(e)

Figure 2: Training curves of TRES (red) versus ES (blue) on the MuJoCo locomotion tasks. For each run, after every ten training iterations, we evaluate current policy parameter via average returns of 100 independent rollouts. The curves are averaged over six random seeds, and the shaded region shows the standard deviation of these seeds.

Environment	Threshold	TRPO timesteps	PPO timesteps	ES timesteps	TRES timesteps
Swimmer-v1	128	4.6e+6	4.4e+5	4.5e+5	**2.4e+5**
HalfCheetah-v1	2385	2.6e+6	1.2e+6	1.7e+6	**9.8e+5**
Walker2d-v1	2872	2.9e+6	**1.9e+6**	1.2e+7	3.6e+6
Ant-v1	3000	N/A	**1.0e+7**	1.5e+7	1.3e+7
InvertedDoublePendulum-v1	9104	4.4e+6	1.9e+6	8.8e+5	**6e+5**

Table 1: A comparison of TRES, ES, TRPO and PPO on the MuJoCo locomotion tasks. For each task we show the average number of timesteps required to reach a prescribed reward threshold, averaged over six random seeds.

PPO in most tasks. This indicates that TRES can achieve robust and consistent performance for various tasks. A particular notable case is the walker2d-v1 task, which is known for its difficulty and falling into local optimal easily. In such task, our TRES converges much faster than ES and achieves better final performance, without falling into the local optima that ES stuck into. In Figure 2, we also find that the shaded region of TRES is smaller than that of ES in most tasks. This suggests that TRES achieves stronger robustness across different random seeds. Especially in Swimmer-v1 task, ES has a very high variance during training, while TRES can obtain a stable training process with low variance.

Impact of hyper-parameters

There are two main hyper-parameters λ and K in TRES algorithms. In this subsection, we conducted several experiments to investigate their impact.

Impact of clip factor λ Clip factor λ is introduced to constrain policy parameter update in each iteration. We conducted several experiments on HalfCheetah-v1 task to study the impact of λ. We plot the training curves of different λ in Figure 3. From Figure 3, we can see that $\lambda \in [8.0, 20.0]$ can improve sample efficiency consistently against baseline, and $\lambda = 14.0$ has the best performance. Reducing or increasing λ from 14.0 hurts the performance. Intuitively, too small clip factor means that policy parameter update is constrained strictly, which may slow down the training process, while too big clip factor means that policy parameter update is completely unconstrained, which may make the training process unstable. Therefore, a medium clip factor is a better choice to obtain a fast and stable training process. These results well suggest our intuitive understanding.

Impact of epoch number K By reusing the sampled data, we perform K gradient steps over clip surrogate objective in each iteration. We conducted some experiments on HalfCheetah-v1 task to study the impact of epoch number K. Intuitively, larger epoch number contributes to higher reuse frequency of sampled data, but if the updated parameter is too far away from current policy parameter where we sample data, it would be no longer precise to improve true objective by using the approximated gradient from clipped surrogate objective. To investigate the balance between reuse frequency and gradient precision, we try different epoch numbers with fixed λ. Figure 4 shows the training curves of various settings of K. From Figure 4, we can observe that a small K does not make the best of sampled data, and a large K causes a performance drop. We observe that $K = 15$ can gain best performance.

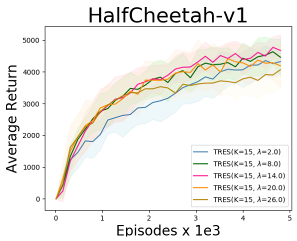


Figure 3: Impact of λ on HalfCheetah-v1 task.

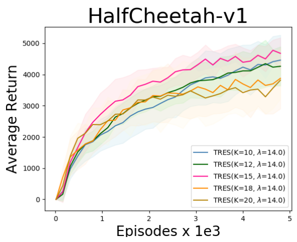
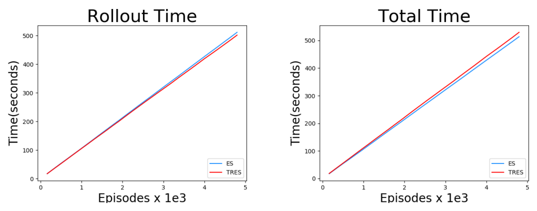


Figure 4: Impact of K on HalfCheetah-v1 task.

Parallel version

The good parallelizability is one of the most important characteristics of ES algorithm, which allows a very fast training over distributed computing environment. Our TRES retains this valuable property.

We show time analysis between TRES and ES on HalfCheetah-v1 task in Figure 5. From figure 5(a), we can see that TRES is the same with ES with respect to time cost of rollout process. From figure 5(b), we can see that although TRES needs to update parameter for multiple rounds in each iteration, which is more complex than ES, TRES curve is almost coincident with ES curve with respect to overall process. This makes sense because the time cost of rollout process is far greater than that of update process during the actual training process.



(a) Rollout time cost (b) Total time cost

Figure 5: Time Analysis for TRES and ES

Related work

Evolution Strategies (ES) is an age-old black-box optimization method (Rechenberg 1971; Sun et al. 2009; Schaul, Glasmachers, and Schmidhuber 2011; Wierstra et al. 2011; 2014), which generates a descent direction via finite differences over random sampled directions. Recently, (Salimans et al. 2017; Mania, Guy, and Recht 2018) applied it to reinforcement learning domain and empirically demonstrated that ES is comparable to state-of-the-art policy gradient algorithms, which has generated renewed interest in ES as a promising direction in reinforcement learning.

However, how to improve sample efficiency of ES algorithm is still a challenging problem. Recent methods proposed for this purpose could be divided into two categories: (1) Adapting the search distribution by encouraging the exploration diversity. (2) Adapting the search distribution with the help of extra gradient information. In the first category, (Choromanski et al. 2018) propose to enforce orthogonality conditions on the Gaussian perturbations for parameter exploration, and they demonstrate theoretically and empirically that random orthogonal and Quasi Monte Carlo (QMC) finite difference directions are much more effective for parameter exploration than random Gaussian directions used in (Salimans et al. 2017). (Conti et al. 2017) hybridize ES with some existing algorithms which promote directed exploration like novelty search (NS) and quality diversity (QD), improving ES performance on sparse or deceptive deep RL tasks. In the second category, (Maheswaranathan et al. 2018) proposes *Guided Evolution Strategies*, the main idea of such method is to use extra surrogate gradient information (directions that may be correlated with, but not necessarily identical to, the true gradient) along with ES random perturbations. Specifically, they define a new search distribution for ES that is elongated along a guiding subspace spanned by the surrogate gradients.

All of above mentioned works focus on sampling better or more diverse search directions to improve sample efficiency of ES algorithm. Quite different from these methods, our approach focuses on how to fully utilize the sampled data, no matter how they are sampled.

Conclusion and Future Work

In this paper, we have proposed a new algorithm, Trust Region Evolution Strategies (TRES) in the context of reinforcement learning. With the purpose of making more efficient use of sampled data, we propose a novel iterative procedure that optimizes a surrogate objective function, with guaranteed monotonic improvement. After making a series of approximations to the theoretically-justified procedure, we further develop the practical algorithm TRES. Experiments on five popular MuJoCo locomotion tasks in the OpenAI Gym show that our TRES has more efficient performance than ES algorithm.

For future work, we plan to combine our method with some prior works that focus on sampling better or more diverse search directions, it would be possible to substantially reduce sample complexity of ES algorithm.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China under Grant U1636201.

References

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

Choromanski, K.; Rowland, M.; Sindhwani, V.; E., T. R.; and Weller, A. 2018. Structured evolution with compact architectures for scalable policy optimization. In *International Conference on Machine Learning*.

Conti, E.; Madhavan, V.; Such, F. P.; Lehman, J.; Stanley, K. O.; and Clune, J. 2017. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *arXiv preprint arXiv:1712.06560*.

Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*.

Gu, S.; Lillicrap, T.; Sutskever, I.; and Levine, S. 2016. Continuous deep q-learning with model-based acceleration. In *Proceedings of The 33rd International Conference on Machine Learning*, 2829–2838.

Gu, S.; Lillicrap, T.; Ghahramani, Z.; Turner, R. E.; and Levine, S. 2017. Q-prop: Sample-efficient policy gradient with an off-policy critic. *ICLR*.

Hunter, D. R., and Lange, K. 2004. A tutorial on mm algorithms. *The American Statistician* 58(1):30–37.

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*.

Maheswaranathan, N.; Metz, L.; Tucker, G.; and Sohl-Dickstein, J. 2018. Guided evolutionary strategies: escaping the curse of dimensionality in random search. *arXiv preprint arXiv:1806.10230*.

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):529–533.

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 *International Conference on Machine Learning*, 1928–1937.

Rajeswaran, A.; Kumar, V.; Gupta, A.; Vezzani, G.; Schulman, J.; Todorov, E.; and Levine, S. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*.

Rechenberg, I. 1971. *Evolutionsstrategie–optimierung technisher systeme nach prinzipien der biologischen evolution*. Ph.D. Dissertation, Technical University of Berlin.

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.; Glasmachers, T.; and Schmidhuber, J. 2011. High dimensions and heavy tails for natural evolution strategies. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, 845–852. New York, NY, USA: ACM.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 1889–1897.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T.; Fan, H.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676):354–359.

Sun, Y.; Wierstra, D.; Schaul, T.; and Schmidhuber, J. 2009. Stochastic search using the natural gradient. In *International Conference on Machine Learning*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *AAAI*, 2094–2100.

Wierstra, D.; Schaul, T.; Glasmachers, T.; Sun, Y.; and Schmidhuber, J. 2011. Natural evolution strategies. *arXiv preprint arXiv:1106.4487*.

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

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 58(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 *Advances in neural information processing systems*, 5285–5294.