Combining PPO and Evolutionary Strategies for Better Policy Search

Jennifer She [1]

Abstract

A good policy search algorithm needs to strike a balance between being able to explore candidate policies and being able to zero-in on good ones. In this project we propose and implement hybrid policy search algorithms inspired by Proximal Policy Optimization (PPO) and Natural Evolutionary Strategies (ES) in order to leverge their individual strengths. We compare these methods against PPO and ES in two OpenAI environments: CartPole and BipedalWalker.

1. Introduction

The standard reinforcement learning framework is modelled by a Markov Decision Process $M = (S, A, P, R, \gamma)$, where at each time step t, the agent takes an action $a_t \in A$ at a state $s_t \in S$, and as a result, transitions to a new state s_{t+1} according to P and receives a reward r_t according to R.

The objective of policy search is to determine a policy $\pi : S \times A \to [0, 1]$, parameterized by θ in our case, that specifies how the agent should act at each state s, ie.

$$\pi_\theta(a|s) = \Pr(a_t = a|s_t = s).$$

We want π_θ to maximize the expected return

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}[R(\tau)], \tag{1}$$

where $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$ is the return from following a specific trajectory τ under π_θ.

Two types of policy search algorithms are policy gradients like Proximal Policy Optimization (PPO), and evolutionary strategies or derivative-free optimization (ES). Policy gradient methods leverage the problem structure by estimating $\nabla_\theta J(\theta)$, and incorporates it into stochastic gradient descent methods in order to arrive at a potential solution quickly. However, they are said to face a lack of exploration in the space of policies due the greediness of their updates. Evolutionary strategies in contrast, are able to exhibit better exploration by directly injecting randomness into the space of policies via sampling θ. However, they make use of less information and thus require more time and samples to perform well. A natural extension is to construct a hybrid method that leverages the strengths of both types of methods. We test out 3 hybrid methods combining PPO and ES, that make use of the gradient, and involve stochastic sampling of θ. We compare these methods to the original PPO and ES in CartPole (CP) and BipedalWalker (BW). The code for this project is available at `https://github.com/jshe/CS229Project.git`.

[1]Computer Science, Stanford University. Correspondence to: Jennifer She <jenshe@stanford.edu>.

2. Related Work

The family of policy gradient methods stems from the original REINFORCE algorithm by Williams (1992). Since then, there have been many new variants that improve REINFORCE in many aspects: encouraging training stability by adding an advantage term, decreasing sample complexity by using off-policy sampling, and improving computation efficiency by using parallelism. Notable methods include Asynchronous Advantage Actor-Critic (Mnih et al., 2016), Trust-Region Policy Optimization (Schulman et al., 2015), and PPO (Schulman et al., 2017).

Evolutionary strategies in contrast have traditionally been used outside of reinforcement learning. A recent paper by Salimans et al. (2017) has brought them to the attention of the reinforcement learning community, as a competitive alternative to policy gradients.

A recent paper (Hämäläinen et al., 2018) proposes an algorithm that combines PPO and ES, however the algorithm still only incorporates stochasticity in the action space, rather than θ.

3. Methods

3.1. Baselines

3.1.1. Proximal Policy Optimization (PPO)

PPO stems from the REINFORCE algorithm (Williams, 1992), which we briefly discuss. REINFORCE involves

rewriting $\nabla_\theta J(\theta)$ as

$$\mathbb{E}_\tau[\nabla_\theta \log p(\tau;\theta) R(\tau)] \tag{2}$$

which can be approximated by taking the gradient of samples

$$\log p(\tau;\theta) R(\tau) = \sum_{t=0}^{T} \log \pi_\theta(a_t|s_t) R(\tau). \tag{3}$$

PPO increases sample efficiency by reusing trajectories from past policies $\pi_{\theta_{\text{old}}}$, and improves training stability by ensuring that π_θ updates at every iteration are small. At each iteration, trajectories are sampled under $\pi_{\theta_{\text{old}}}$ for a total of H state-action pairs. We then use mini-batch samples of these pairs (s_t, a_t) of to update π_θ. π_θ is updated using a modification of (3), where $\log \pi_\theta(a_t|s_t)$ is replaced by a ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ to allow for this type of sampling, and the ratio is clipped if it falls outside of some range $[1-\epsilon, 1+\epsilon]$ to increase stability. This results in the objective

$$\sum_{t=0}^{T} \min\left\{ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t(\tau), \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \right]_{\text{clip}} \hat{A}_i(\tau) \right\}, \tag{4}$$

where τ is sampled using $\pi_{\theta_{\text{old}}}$.

The advantage function $\hat{A}_t(\tau) = R_t(\tau) - v_\theta(s_t)$, where $R_t(\tau) = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ is a modification of $R(\tau)$ calculated using a learned value function v_θ. v_θ is updated along with π_θ using an additional loss

$$\sum_{t=1}^{T} (v_\theta(s_t) - R_t(\tau))^2. \tag{5}$$

An entropy term

$$c_{\text{ent}} \cdot H(\pi_\theta(a_t|s_t)) \tag{6}$$

can also be optionally added to the objective to encourage exploration in a_t. Combining (4), (5) and (6) results in the final PPO objective.

3.1.2. Evolutionary Strategies (ES)

In evolutionary strategies, or derivative-free policy optimization, the function $J(\theta)$ is treated as a black-box.

The general framework of evolutionary strategies involves at each step, sampling candidate parameters $\{\theta^{(1)}, ..., \theta^{(k)}\}$ from some distribution Θ, and using these $\theta^{(i)}$'s based on their performance in terms of $J(\theta)$, to update Θ.

A recent variant called Natural Evolutionary Strategies (Salimans et al., 2017) scales to problems in reinforcement learning. This variant represents Θ using a Gaussian distribution

$$\Theta = \bar{\theta} + \sigma\epsilon$$

where $\epsilon \sim \mathcal{N}(0, I)$. The objective is

$$\max_{\bar{\theta}} \mathbb{E}_{\theta\sim\Theta}[J(\theta)], \tag{7}$$

where $\bar{\theta}$ is updated using an approximation of the gradient $\nabla_{\bar{\theta}} \mathbb{E}_{\theta\sim\Theta}[J(\theta)]$, derived by rewriting it as $\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[\nabla_{\bar{\theta}} \log p(\theta;\bar{\theta}) J(\theta)]$ and approximating this using the samples $\{\theta^{(1)}, ..., \theta^{(k)}\}$ by

$$\frac{1}{k} \sum_{i=1}^{k} [\nabla_{\bar{\theta}} \log p(\theta;\bar{\theta}) R(\tau_t)] = \frac{1}{k\sigma} \sum_{i=1}^{k} [\epsilon_t R(\tau_t)]. \tag{8}$$

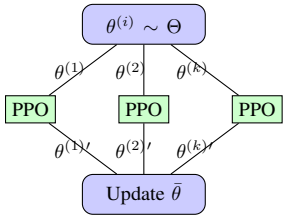


Figure 1. Diagram of ES-PPO and MAX-PPO.

3.2. Our Approaches

3.2.1. ES-PPO

Instead of sampling $\theta^{(i)}$ naively as in ES, we propose running PPO with each of these samples as initializations to obtain new samples $\theta^{(i)\prime}$. We then update π_θ by (8) with returns from these new samples and modified perturbations

$$\epsilon'_t = \frac{1}{\sigma}(\theta^{(i)\prime} - \bar{\theta}).$$

The general idea of this algorithm is summarized in Figure 1.

3.2.2. MAX-PPO

Instead of using the update (8) as in ES-PPO, we directly set $\bar{\theta}$ to $\theta^{(i)\prime}$ with the highest return

$$\text{argmax}_i R(\tau_t)|_{\tau\sim p(\tau;\theta^{(i)\prime})}.$$

We conjecture that this method would work well despite its apparent greediness because $\theta^{(i)\prime}$ are likely to be decent solutions as a result of the PPO updates.

3.2.3. ALT-PPO

We alternate between ES and PPO iterations by running ES every j iterations with $\bar{\theta} = \theta$ in order to inject stochasticity.

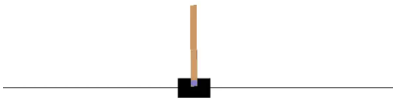


Figure 2. Initial state of CP.

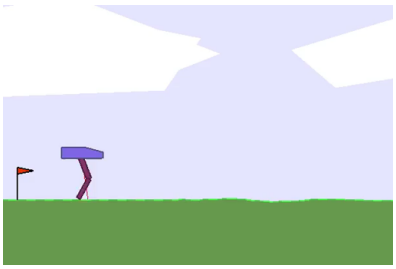


Figure 3. Initial state of BW.

4. Experiments/ Results

4.1. CartPole-v0 (CP)

The objective of CP (see Figure. 3) is to move the cart left and right in order to keep the pole upright. The agent receives $+1$ reward for every time step, for a maximum of 200 points. The episode ends when the pole falls, the cart goes off screen, or it reaches a max of 200 time steps. $S \subset \mathbb{R}^4$ represent the position/velocity of the cart, and $A = \{0, 1\}$ represent the actions left and right.

For ES in this setting, we represent π_θ by

$$\pi_\theta(a|s) = \mathbf{1}[a = f_\theta(s)] \tag{9}$$

where f_θ is a fully-connected neural network: FC(4×100) + ReLU + FC(100×1) + Sigmoid + **1**[]. For all other methods, we use

$$\pi_\theta(a|s) \sim \text{Bernoulli}(g_\theta(s))$$

where g_θ is a fully-connected neural network: FC(4×100) + ReLU + FC(100×100) + ReLU + FC(100×1) + Sigmoid. We also parameterize v_θ by: FC(4×100) + ReLU + FC(100×100) + ReLU + FC(100×1), where the first fully-connected layer is tied with g_θ. We perform hyperparameter search for each method to obtain the best configuration, and describe the details below. The results are shown in Figure 4 and Table 1.

4.1.1. PPO

We use an ADAM optimizer with learning rate $= 0.0001$ chosen among $\{0.0001, 0.00025, 0.001\}$ to be the most stable. We choose $H = 256$ and batch size 32, and iterate over the entire 256 samples $l = 3$ times on each iteration (which seem sufficicent for CP). We set $c_{\text{ent}} = 0.0$ among $\{0.0, 0.0001\}$ because the entropy term seems unnecessary. We also set clipping $\epsilon = 0.2$ among $\{0.01, 0.02, 0.2\}$ to give the largest signal. Finally, we fix $\gamma = 0.99$.

	Episodic Rewards	Training Time
ES	200.0	60.59
PPO	200.0	53.74
ES-PPO	200.0	515.03
MAX-PPO	200.0	363.52
ALT-PPO	200.0	131.24

Table 1. Final results from CP averaged across 5 trials.

4.1.2. ES

We choose $k = 5$ among $\{5, 10, 20\}$ as it seems to be sufficient for CP and results in the fastest training time. We also set $\sigma^2 = 0.1$ among $\{0.1, 0.001, 0.0001\}$ with learning rate 0.001 among $\{0.0001, 0.0025, 0.001\}$.

4.1.3. ES-PPO

We use the same hyperparameters as in ES and PPO, with the exception that each PPO subcall runs for 3 iterations with $l = 1$ instead of 3, as we found this helps decrease variance in the updates, despite resulting in worse sample efficiency.

4.1.4. MAX-PPO

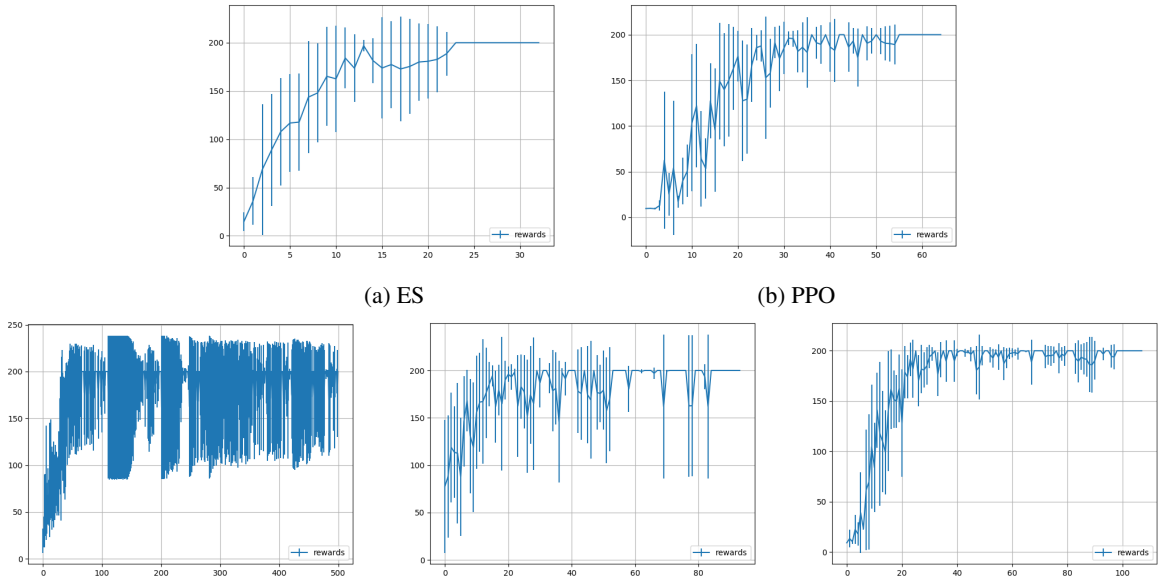
We use the same hyperparameters as ES-PPO, except we find that $\sigma^2 = 0.001$ works better, as it also reduces variance in the updates.

4.1.5. ALT-PPO

We use the same hyperparameters as ES-PPO, and additionally set $j = 5$ between $\{5, 20\}$.

We note that CP is a very simple setting, likely with no evident local minima. Thus with the stopping condition being that each method must achieve the maximum episodic reward of 200 for 10 consecutive iterations (with the maximum number of iterations capped at 3000), all methods are able to achieve 200 at the end of training across all 5 trials.

However, based on training time, it seems like ES and PPO take the shortest amount of time to converge. In contrast, ALT-PPO takes a factor of $2 - 3$ as much time, which we conjecture is due to the randomness from ES sometimes perturbing θ away from a good solution achieved by PPO. This can be seen by the long tail of plot (c) in Figure 4. Lastly, ES-PPO and MAX-PPO both take around a factor of



(a) ES (b) PPO

(c) ES-PPO (d) MAX-PPO (e) ALT-PPO

Figure 4. Episodic/sum of rewards over training CP (in 10's of iterations) across 5 trials each. Vertical bars are the standard deviation.

5 as much time as ALT-PPO, with MAX-PPO performing slightly better than ALT-PPO. This is likely due to the high computation of running 5 instances of PPO at a time.

4.2. BipedalWalker-v2 (BW)

The objective of BW is to maneuver the walker to the rightmost side of the environment without falling. The agent receives $+\epsilon$ for moving forward for a total of 300 on the walker reaching its destination. The agent also receives -100 for falling. The episode ends when the walker reaches its destination or falls. $S \subset \mathbb{R}^{24}$, and $A = [-1, 1]^4$ represent the various states and actions of the walker and its components (hips, knees etc).

For ES in this setting, we again represent π_θ by (9) where f_θ is a fully-connected neural network: FC(24×100) + ReLU + FC(100×4) + Tanh. For all other methods, we use

$$\pi_\theta(a|s) \sim \mathcal{N}(g_\theta(s), \sigma^2)$$

where g_θ is a fully-connected neural network: FC(24×100) + ReLU + FC(100×100) + ReLU + FC(100×4) + Tanh. We also parameterize v_θ by: FC(24×100) + ReLU + FC(100×100) + ReLU + FC(100×1), without tied layers this time (because we need more parameters in a more complex setting). We perform hyperparameter search for each method to obtain the best configuration under the constraints of our compute, which we detail below. The results are shown in Figure 5.

4.2.1. PPO

We use the same setting as PPO in the case of CP, except we increase H to 2048 and batch size to 64, and make use of the entropy term with $c_{\text{ent}} = 0.0001$.

4.2.2. ES

We set $\sigma = 0.1$, but increase population size to 20, which we find helps performance significantly here, and we modify learning rate to 0.01 to allow for larger updates corresponding to better directions as a result of larger k.

4.2.3. ES-PPO

We use the same setting as ES-PPO in the case of CP, with the modification that $H = 2048$, batch size $= 64$, and $c_{\text{ent}} = 0.0001$ as in PPO. $k = 20$ is too computationally slow, so we stick with $k = 5$.

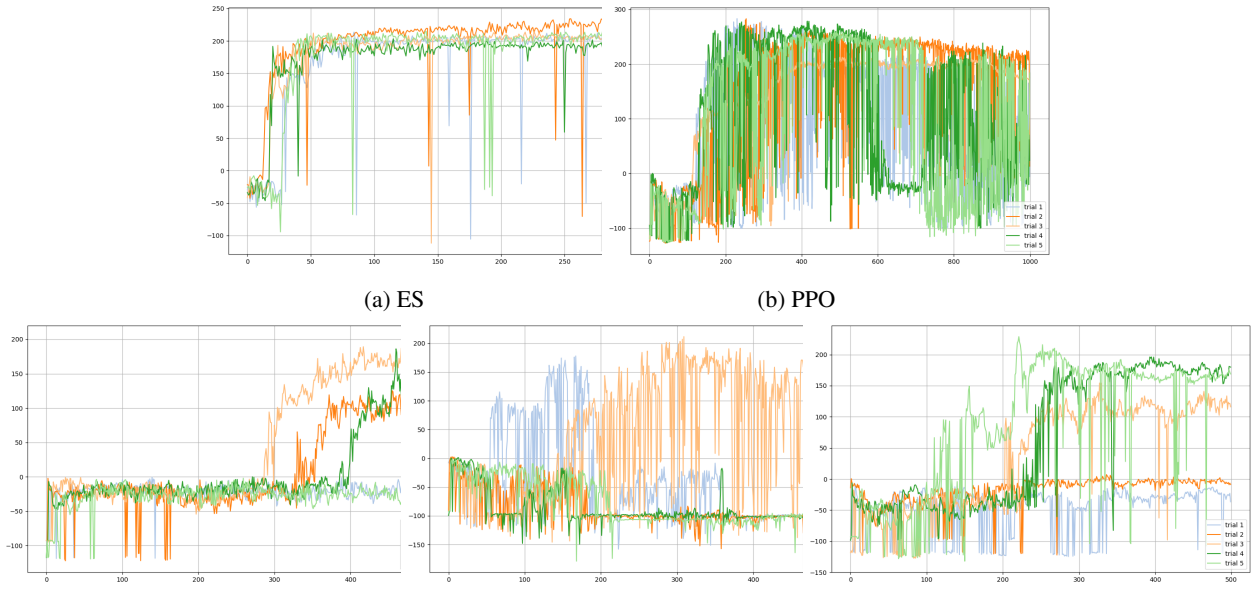
4.2.4. MAX-PPO

We use the same setting as ES-PPO, except with $\sigma^2 = 0.01$ as in the CP case.

4.2.5. ALT-PPO

We use the same setting as ES-PPO, and set $j = 5$ as in the CP case.

We note that none of the algorithms are able to achieve the maximum reward of 300 in BW, so all of the methods are terminated early.



(a) ES (b) PPO

(c) ES-PPO (d) MAX-PPO (e) ALT-PPO

Figure 5. Episodic/sum of rewards (in 10's of iterations) over training across 5 trials each.

ES and PPO achieve the highest episodic rewards. However the training curves of PPO is much more volatile than that of ES. We suspect that this is because PPO reuses samples from $\pi_{\theta_{\text{old}}}$ in order to encourage sample efficiency.

We also find that while using $k = 20$ for ES leads to much more stable training and greater improvements in episodic return, each iteration as a result becomes very slow. ES-PPO and MAX-PPO exponentiate this problem, and as a result, we choose a maximum sample size of $k = 5$ for ES-PPO and MAX-PPO.

One insight we have about the poor performance of ES-PPO in BW is that the PPO subcalls may drive $\theta^{(i)\prime}$ far from $\bar{\theta}$. Thus, a weighted average of the returns at $\theta^{(i)\prime}$ may no longer be a good predictor of the return at a weighted average of $\theta^{(i)\prime}$. This can cause misleading updates that result in $\bar{\theta}$ having a much lower return than the weighted average.

MAX-PPO mitigates this averaging problem of ES-PPO, as seen by the few trials in plot (d) in Figure 1 with returns that improve much earlier on during training. However, we find that it still has very high training instability. We believe this is because MAX-PPO is unable to stay at its current $\bar{\theta}$ unless the PPO subcalls are run for long enough. This means MAX-PPO can lead away from good solutions when all neighbouring $\theta^{(i)\prime}$ have low returns.

ALT-PPO also seems to have high variance, as demonstrated by some trials leading to much better returns than others. We believe that in order for a hybrid method to achieve good performance, we cannot simply inject stochasticity naively – this stochasticity needs to take into account the quality of the current $\bar{\theta}$.

5. Conclusion

From the results above, we believe that in order for a hybrid method to achieve better performance than PPO and ES, it needs to combine them in a more clever way. One potential idea to try is to somehow adaptively modify the variance of Θ using gradient information, so that only closer $\theta^{(i)}$ are sampled when $\bar{\theta}$ has a relatively good return.

Another potential direction is to investigate how we can better leverage large-scale parallel compute in order to speed up methods like ES-PPO and MAX-PPO.

In the case of PPO, it would also be interesting to look into the trade-offs between sample efficiency and training stability, and see whether sampling from π_θ instead of $\pi_{\theta_{\text{old}}}$ can reduce this instability.

Lastly, it may be more effective to compare these methods in more complex environments, where there exist obvious local minima, and PPO should fail.

Acknowledgements

We thank Mario Srouji for the project idea and help during the project.

References

Hämäläinen, Perttu, Babadi, Amin, Ma, Xiaoxiao, and Lehtinen, Jaakko. PPO-CMA: proximal policy optimization with covariance matrix adaptation. *CoRR*, abs/1810.02541, 2018. URL http://arxiv.org/abs/1810.02541.

Mnih, Volodymyr, Badia, Adria Puigdomenech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy, Harley, Tim, Silver, David, and Kavukcuoglu, Koray. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Salimans, Tim, Ho, Jonathan, Chen, Xi, Sidor, Szymon, and Sutskever, Ilya. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael, and Moritz, Philipp. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.