Proximal Policy Optimization Algorithms, Schulman et al, 2017

옥찬호
utilForever@gmail.com

Introduction

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- In recent years, several different approaches have been proposed for reinforcement learning with neural network function approximators. The leading contenders are deep Q-learning, "vanilla" policy gradient methods, and trust region / natural policy gradient methods.
- However, there is room for improvement in developing a method that is **scalable** (to large models and parallel implementations), **data efficient**, and **robust** (i.e., successful on a variety of problems without hyperparameter tuning).

Introduction

- DQN
  - fails on many simple problems and is poorly understood
- A3C – "Vanilla" policy gradient methods
  - have poor data efficiency and robustness
- TRPO
  - relatively complicated
  - is not compatible with architectures that include noise (such as dropout) or parameter sharing (between the policy and value function, or with auxiliary tasks)

Introduction

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- We propose a novel objective with **clipped probability ratios**, which forms a pessimistic estimate (i.e., lower bound) of the performance of the policy.
- This paper seeks to improve the current state of affairs by introducing an algorithm that attains the data efficiency and reliable performance of TRPO, while using only first-order optimization.

Introduction

- To optimize policies, we alternate between sampling data from the policy and performing several epochs of optimization on the sampled data.

Introduction

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- Our experiments compare the performance of various different versions of the surrogate objective, and find that the version with the **<u>clipped probability ratios performs best</u>**.
- We also compare PPO to several previous algorithms from the literature.
  - On continuous control tasks, <u>it performs better than the algorithms we compare against</u>.
  - On Atari, <u>it performs significantly better (in terms of sample complexity) than A2C and similarly to ACER though it is much simpler</u>.

Background: Policy Optimization

PPO, Schulman et al, 2017

- Policy gradient methods work by computing an estimator of the policy gradient and plugging it into a stochastic gradient ascent algorithm. The most commonly used gradient estimator has the form

$$\hat{g} = \hat{\mathbb{E}}_t\left[\nabla_\theta \log \pi_\theta\left(a_t|s_t\right)\hat{A}_t\right]$$

  - where π_θ is a stochastic policy and $\hat{A}_t$ is an estimator of the advantage function at timestep t. Here, the expectation $\hat{\mathbb{E}}_t[\ldots]$ indicates the empirical average over a finite batch of samples, in an algorithm that alternates between sampling and optimization.

Background: Policy Optimization

PPO, Schulman et al, 2017

- Implementations that use automatic differentiation software work by constructing an objective function whose gradient is the policy gradient estimator; the estimator $\hat{g}$ is obtained by differentiating the objective

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\left[\log \pi_\theta (a_t|s_t)\hat{A}_t\right]$$

Background: Policy Optimization

PPO, Schulman et al, 2017

- While it is appealing to perform multiple steps of optimization on this loss L^{PG} using the same trajectory, doing so is not well-justified, and empirically it often leads to **destructively large policy updates**.

Background: Policy Optimization

PPO, Schulman et al, 2017

- In TRPO, an objective function (the "surrogate" objective) is **<u>maximized subject to a constraint on the size of the policy update</u>**. Specifically,

$$\underset{\theta}{\text{maximize}}\ \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right]$$

$$\text{subject to}\ \hat{\mathbb{E}}_t\left[KL[\pi_{\theta_{\text{old}}}(\cdot\,|s_t),\pi_\theta(\cdot\,|s_t)]\right]\le\delta$$

  - Here, θ_{old} is the vector of policy parameters before the update.

Background: Policy Optimization

PPO, Schulman et al, 2017

- This problem can efficiently be approximately solved using the conjugate gradient algorithm, after making a linear approximation to the objective and a quadratic approximation to the constraint.

Background: Policy Optimization

PPO, Schulman et al, 2017

- The theory justifying TRPO actually suggests **using a penalty instead of a constraint**, i.e., solving the unconstrained optimization problem for some coefficient β.

$$\underset{\theta}{\text{maximize}}\ \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t - \beta KL[\pi_{\theta_{\text{old}}}(\cdot\,|s_t), \pi_\theta(\cdot\,|s_t)]\right]$$

Background: Policy Optimization

PPO, Schulman et al, 2017

- This follows from the fact that a certain surrogate objective (which computes the max KL over states instead of the mean) forms a lower bound (i.e., a pessimistic bound) on the performance of the policy π.
- TRPO uses a hard constraint rather than a penalty because **it is hard to choose a single value of β that performs well across different problems**—or even within a single problem, where the characteristics change over the course of learning.

Clipped Surrogate Objective

PPO, Schulman et al, 2017

- Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, so $r_t(\theta_{\text{old}}) = 1$. TRPO maximizes a "surrogate" objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

- The superscript CPI refers to conservative policy iteration, where this objective was proposed.

Clipped Surrogate Objective

PPO, Schulman et al, 2017

- Without a constraint, maximization of L^{CPI} would lead to **an excessively large policy update**; hence, we now consider how to modify the objective, to penalize changes to the policy that move $r_t(\theta)$ away from 1.
- The main objective we propose is the following:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\,\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

  - where epsilon is a hyperparameter, say, $\epsilon = 0.2$.

Clipped Surrogate Objective

PPO, Schulman et al, 2017

- The motivation for this objective is as follows.
  - The first term inside the min is $L^{CPI}(\theta)$.
  - The second term, $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$, modifies the surrogate objective by **clipping the probability ratio**, which removes the incentive for moving $r_t(\theta)$ outside of the interval $[1-\epsilon, 1+\epsilon]$.
  - Finally, we take the minimum of the clipped and unclipped objective, so the final objective is **a lower bound (i.e., a pessimistic bound) on the unclipped objective**.

Clipped Surrogate Objective

PPO, Schulman et al, 2017

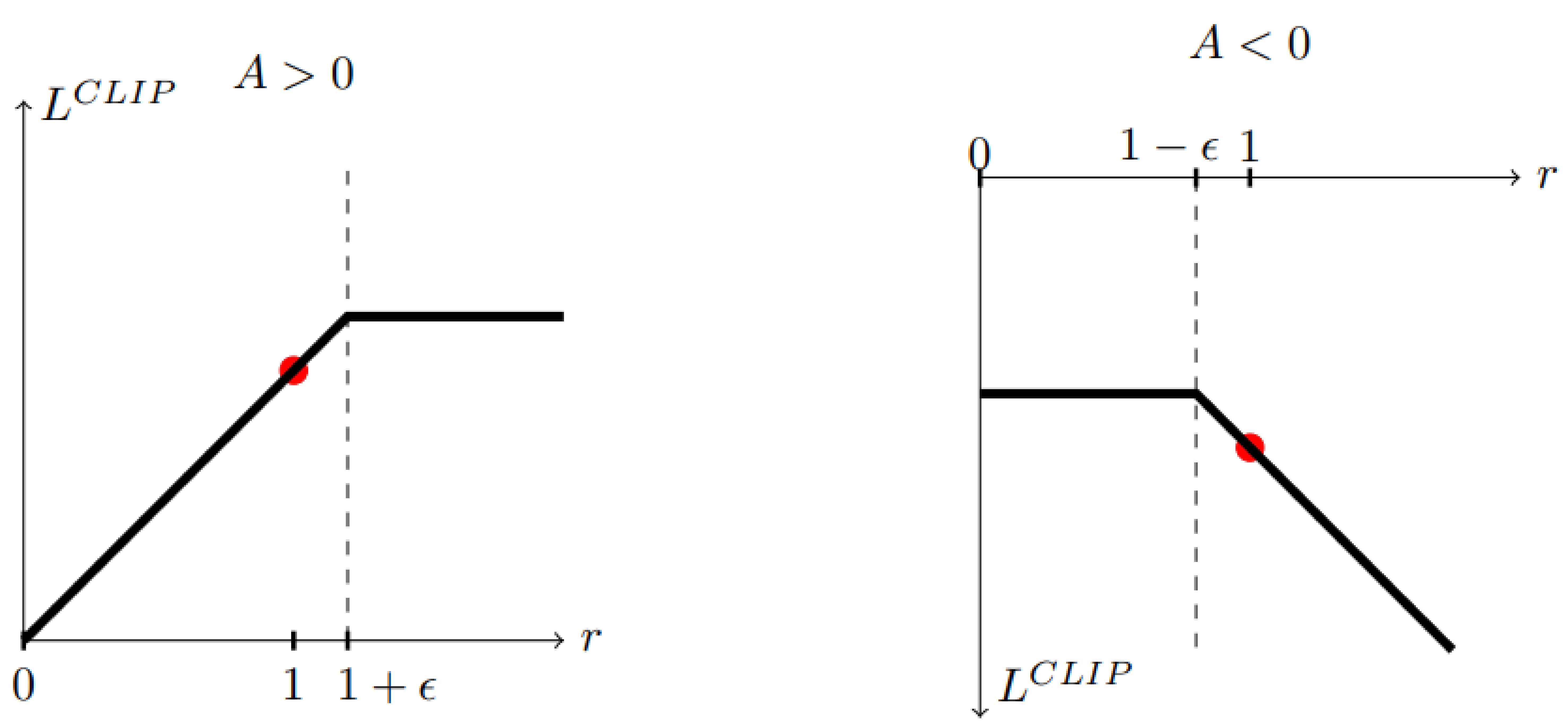


Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP} sums many of these terms.

Clipped Surrogate Objective

PPO, Schulman et al, 2017

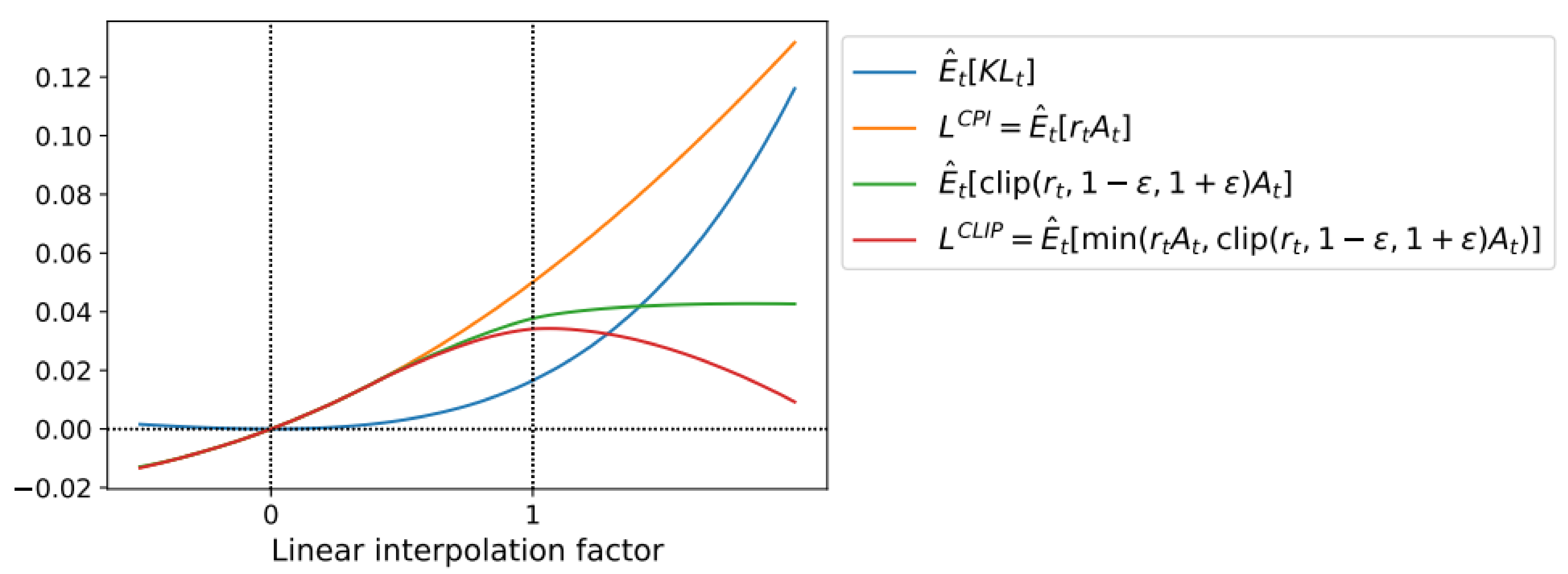


Figure 2: Surrogate objectives, as we interpolate between the initial policy parameter θ_{old}, and the updated policy parameter, which we compute after one iteration of PPO. The updated policy has a KL divergence of about 0.02 from the initial policy, and this is the point at which L^{CLIP} is maximal.

Adaptive KL Penalty Coefficient

PPO, Schulman et al, 2017

- Another approach, which can be used as an alternative to the clipped surrogate objective, or in addition to it, is to use a penalty on KL divergence, and to **adapt the penalty coefficient** so that we achieve some target value of the KL divergence d_{targ} each policy update.
  - In our experiments, we found that the KL penalty performed worse than the clipped surrogate objective, however, we've included it here because it's an important baseline.

Adaptive KL Penalty Coefficient

PPO, Schulman et al, 2017

- In the simplest instantiation of this algorithm, we perform the following steps in each policy update:
  - Using several epochs of minibatch SGD, <u>**optimize the KL-penalized objective**</u>

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t - \beta KL[\pi_{\theta_{\text{old}}}(\cdot\,|s_t), \pi_\theta(\cdot\,|s_t)]\right]$$

  - <u>**Compute $\boldsymbol{d} = \hat{\mathbb{E}}_t\left[\boldsymbol{KL}[\boldsymbol{\pi_{\theta_{old}}}(\cdot\,|\boldsymbol{s_t}), \boldsymbol{\pi_\theta}(\cdot\,|\boldsymbol{s_t})]\right]$**</u>
    - If $d < d_{targ}/1.5$, $\beta \leftarrow \beta/2$
    - If $d > d_{targ} \times 1.5, \beta \leftarrow \beta \times 2$
    - The updated β is used for the next policy update.

Algorithm

- Most techniques for computing variance-reduced advantage-function estimators make use a **learned state-value function** $V(s)$
  - Generalized advantage estimation
  - The finite-horizon estimators

Algorithm

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- If using a neural network architecture that shares parameters between the policy and value function, we must use a loss function that combines the policy surrogate and a value function error term.
- This objective can further be augmented by adding an entropy bonus to ensure sufficient exploration, as suggested in past work (A3C).

Algorithm

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- Combining these terms, we obtain the following objective, which is (approximately) maximized each iteration:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$$

  - where c_1, c_2 are coefficients, and S denotes an entropy bonus, and L_t^{VF} is a squared-error loss $\left(V_\theta(s_t) - V_t^{\text{targ}}\right)^2$.

Algorithm

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- One style of policy gradient implementation, popularized in A3C and well-suited for use with recurrent neural networks, **runs the policy for T timesteps (where T is much less than the episode length), and uses the collected samples for an update**.

Algorithm

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- This style **requires an advantage estimator that does not look beyond timestep T**. The estimator used by A3C is

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$$

  - where t specifies the time index in $[0, T]$,
    within a given length-T trajectory segment.

Algorithm

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- Generalizing this choice, we can use a truncated version of **generalized advantage estimation**, which reduces to previous equation when $\lambda = 1$:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}$$

$$\text{where } \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

Algorithm

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- A proximal policy optimization (PPO) algorithm that uses fixed-length trajectory segments is shown below.
  - Each iteration, each of N (parallel) actors collect T timesteps of data.
  - Then we construct the surrogate loss on these NT timesteps of data, and optimize it with minibatch SGD (or usually for better performance, Adam), for K epochs.

Algorithm 1 PPO, Actor-Critic Style

for iteration$=1,2,\ldots$ **do**
  for actor$=1,2,\ldots,N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1,\ldots,\hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

Experiments

- First, we compare several different surrogate objectives under different hyperparameters. Here, we compare the surrogate objective L^{CLIP} to several natural variations and ablated versions.
  - No clipping or penalty: $L_t(\theta) = r_t(\theta)\hat{A}_t$
  - Clipping: $L_t(\theta) = \min(r_t(\theta)\,\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)$
  - KL penalty (fixed or adaptive): $L_t(\theta) = r_t(\theta)\hat{A}_t - \beta KL[\pi_{\theta_{\mathrm{old}}}, \pi_\theta]$

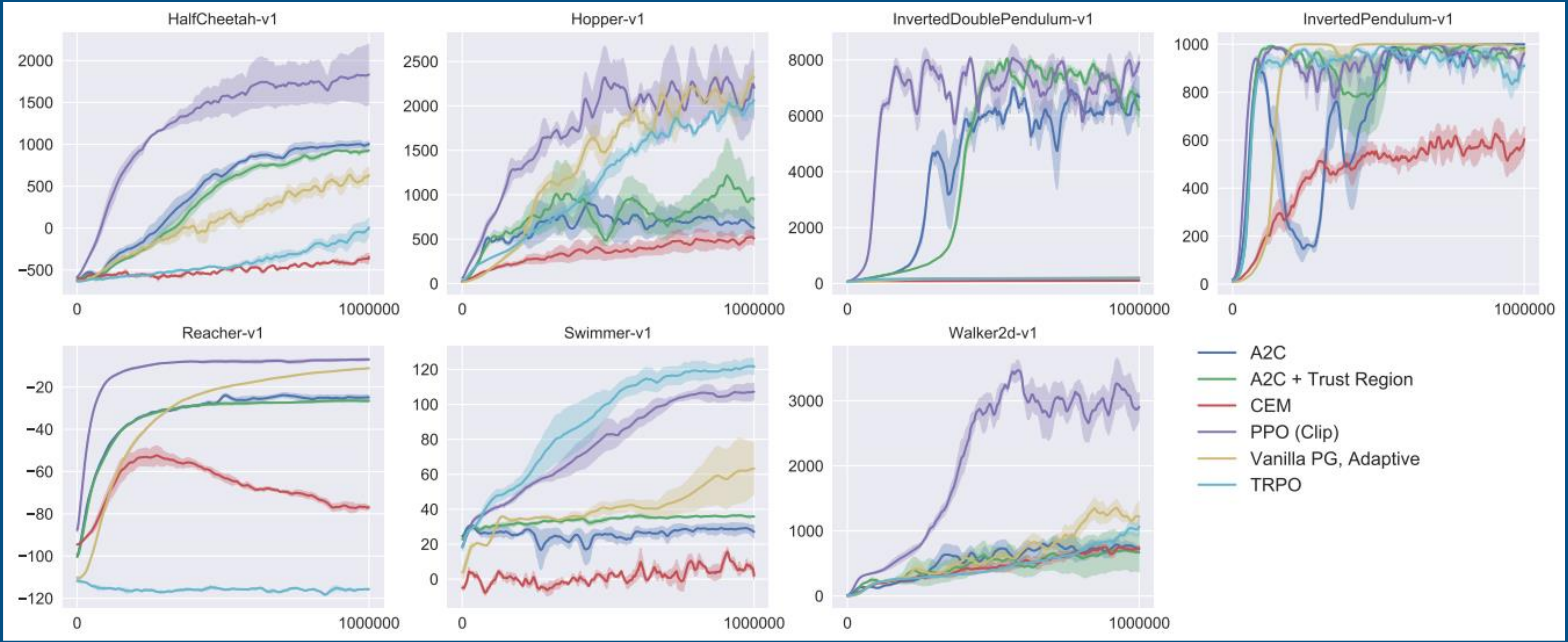
Experiments

Proximal Policy Optimization Algorithms, Schulman et al, 2017

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

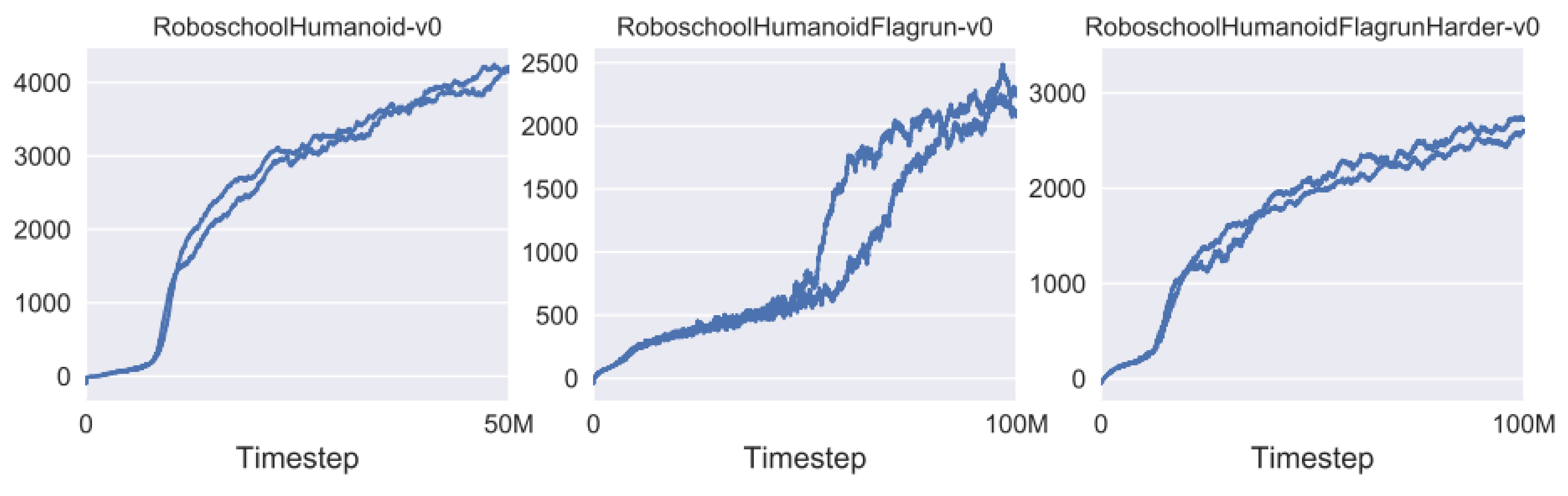
Experiments

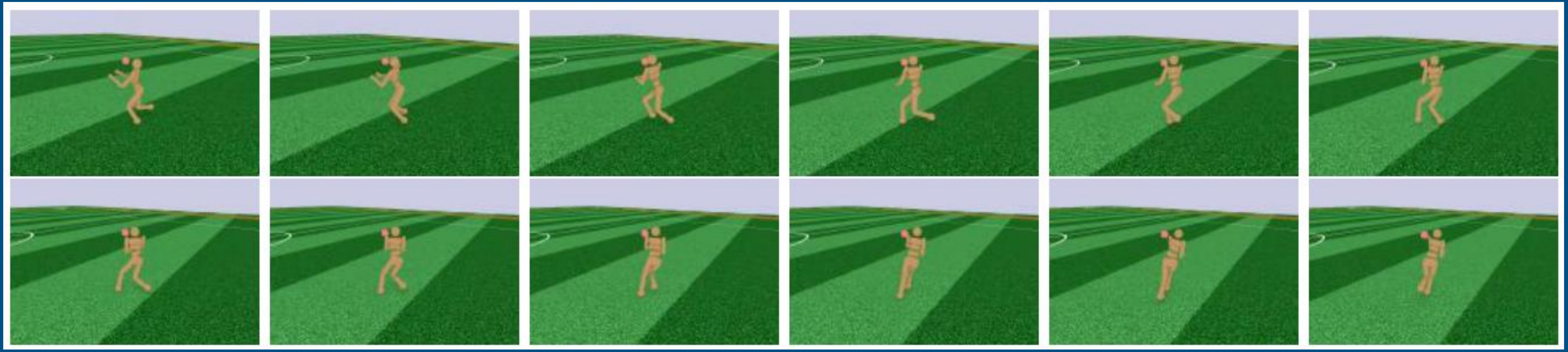
Proximal Policy Optimization Algorithms, Schulman et al, 2017



Experiments

Proximal Policy Optimization Algorithms, Schulman et al, 2017





Experiments

Proximal Policy Optimization Algorithms, Schulman et al, 2017

	A2C	ACER	PPO	Tie
(1) avg. episode reward over all of training	1	18	**30**	0
(2) avg. episode reward over last 100 episodes	1	**28**	19	1

Conclusion

- We have introduced **proximal policy optimization**, a family of policy optimization methods that **use multiple epochs of stochastic gradient ascent** to perform each policy update.

Conclusion

Proximal Policy Optimization Algorithms, Schulman et al, 2017

- These methods have **the stability and reliability of trust-region methods** but are much simpler to implement, requiring only few lines of code change to a vanilla policy gradient (A3C) implementation, applicable in more general settings (for example, when using a joint architecture for the policy and value function), and have better overall performance.

References

- https://reinforcement-learning-kr.github.io/2018/06/22/7_ppo/
- https://lynnn.tistory.com/73
- https://jay.tech.blog/2018/10/09/trpo%EC%99%80-ppo/
- https://talkingaboutme.tistory.com/entry/RL-Policy-Gradient-Algorithms

Thank you!