Reward Constrained Policy Optimization

Boris Meinardus, Tuan Anh Le

Preliminaries

MARKOV DECISION PROCESSES

$$(S, A, R, P, \mu, \gamma)$$

$$V_R^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_t \gamma^t \, r(s_t, a_t) | s_0 = s\right]$$

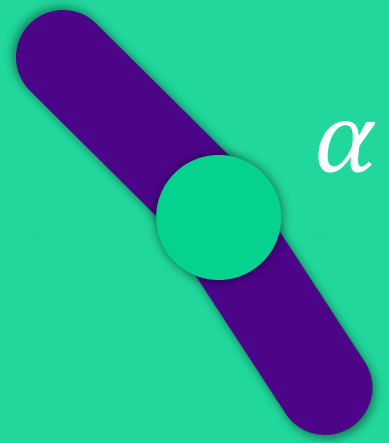
$$\max_{\pi \in \Pi} J_R^{\pi} \;, where \; J_R^{\pi} = \mathbb{E}_{s \sim \mu}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] = \sum_{s \in S} \mu(s) V_R^{\pi}(s)$$

Preliminaries

CONSTRAINED MDPS

$$J_C^{\pi} = \mathbb{E}_{s\sim\mu}^{\pi}\left[C(s)\right]$$

$$C(s_t) = F(c(s_t, a_t), \ldots, c(s_N, a_N))$$



Preliminaries

CONSTRAINED MDPS

$$\max_{\pi \in \Pi} J_R^{\pi} \ , s.t. J_C^{\pi} \leq \alpha$$

Constrained Policy Optimization

LAGRANGE MULTIPLIERS

$$\max_{\theta}[J_R^{\pi_\theta} - \lambda(J_C^{\pi_\theta} - \alpha)]$$

Constrained Policy Optimization

LAGRANGE MULTIPLIERS

$$\min_{\lambda \geq 0} \max_{\theta} [J_R^{\pi_\theta} - \lambda (J_C^{\pi_\theta} - \alpha)]$$

Constrained Policy Optimization

LAGRANGE MULTIPLIERS

$$\min_{\lambda \geq 0} \max_{\theta} L(\lambda, \theta) =$$

$$\min_{\lambda \geq 0} \max_{\theta} [J_R^{\pi_\theta} - \lambda (J_C^{\pi_\theta} - \alpha)]$$

Constrained Policy Optimization

ESTIMATING THE GRADIENT

$$\nabla_\theta L(\lambda, \theta) = \nabla_\theta \mathbb{E}^{\pi_\theta}_{s\sim\mu}[\log \pi(s, a; \theta)[R(s) - \lambda C(s)]]$$

$$\nabla_\lambda L(\lambda, \theta) = -(\mathbb{E}^{\pi_\theta}_{s\sim\mu}[C(s)] - \alpha)$$

Constrained Policy Optimization

UPDATING PARAMETERS

$$\lambda_{k+1} = \Gamma[\lambda_k - \eta_1(k)\nabla_\lambda L(\lambda_k, \theta_k)]$$

$$\theta_{k+1} = \Gamma[\theta_k + \eta_2(k)\nabla_\theta L(\lambda_k, \theta_k)]$$

Reward Constrained Policy Optimization

ACTOR CRITIC REQUIREMENTS

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

Abstract

We propose a new family of policy gradient methods for reinforcement learning, which alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Whereas standard policy gradient methods perform one gradient update per data sample, we propose a novel objective function that enables multiple epochs of minibatch updates. The new methods, which we call proximal policy optimization (PPO), have some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity (empirically). Our experiments test PPO on a collection of benchmark tasks, including simulated robotic locomotion and Atari game playing, and we show that PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time.

1 Introduction

In recent years, several different approaches have been proposed for reinforcement learning with neural network function approximators. The leading contenders are deep Q-learning [Mni+15], "vanilla" policy gradient methods [Mni+16], and trust region / natural policy gradient methods [Sch+15b]. 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). Q-learning (with function approximation) fails on many simple problems[1] and is poorly understood, vanilla policy gradient methods have poor data effiency and robustness; and trust region policy optimization (TRPO) is relatively complicated,

Reward Constrained Policy Optimization

PENALIZED REWARD FUNCTIONS

$$V_{C_\gamma}^{\pi}(s) \triangleq \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) | s_0 = s\right]$$

Reward Constrained Policy Optimization

PENALIZED REWARD FUNCTIONS

$$V_{C_\gamma}^{\pi}(s) \triangleq \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) | s_0 = s\right]$$

$$\hat{r}(\lambda, s, a) \triangleq r(s, a) - \lambda c(s, a)$$

Reward Constrained Policy Optimization

PENALIZED REWARD FUNCTIONS

$$V_{C_\gamma}^{\pi}(s) \triangleq \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) | s_0 = s\right]$$

$$\hat{r}(\lambda, s, a) \triangleq r(s, a) - \lambda c(s, a)$$

$$\hat{V}^{\pi}(\lambda, s) \triangleq \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \hat{r}(s_t, a_t) | s_0 = s\right]$$

Reward Constrained Policy Optimization

PENALIZED REWARD FUNCTIONS

$$V_{C_\gamma}^{\pi}(s) \triangleq \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) | s_0 = s\right]$$

$$\hat{r}(\lambda, s, a) \triangleq r(s, a) - \lambda c(s, a)$$

$$\hat{V}^{\pi}(\lambda, s) \triangleq V_R^{\pi}(s) - \lambda V_{C_\gamma}^{\pi}(s)$$

Reward Constrained Policy Optimization

ALGORITHM

Algorithm 2 RCPO Advantage Actor Critic

The original Advantage Actor Critic algorithm is in gray, whereas our additions are highlighted in black.

Input: penalty function $C(\cdot)$, threshold α and learning rates η_1, η_2, η_3
Initialize actor $\pi(\cdot|\cdot;\theta_p)$ and critic $V(\cdot;\theta_v)$ with random weights
Initialize $\lambda = 0$, $t = 0$, $s_0 \sim \mu$ ▷ Restart
for $T = 1, 2, ..., T_{max}$ **do**
  Reset gradients $d\theta_v \leftarrow 0$, $d\theta_p \leftarrow 0$ and $\forall i : d\lambda_i \leftarrow 0$
  $t_{start} = t$
  while s_t `not terminal and` $t - t_{start} < t_{max}$ **do**
    Perform a_t according to policy $\pi(a_t|s_t;\theta_p)$
    Receive r_t, s_{t+1} and penalty score $\hat{C}_t$
    $t \leftarrow t + 1$
  $R = \begin{cases} 0 & , \text{ for terminal } s_t \\ V(s_t, \theta_v) & , \text{ otherwise} \end{cases}$
  for $\tau = t-1, t-2, ..., t_{start}$ **do**
    $R \leftarrow r_\tau - \lambda \cdot \hat{C}_\tau + \gamma R$ ▷ Equation 10
    $d\theta_p \leftarrow d\theta_p + \nabla_{\theta_p} \log \pi(a_\tau|s_\tau;\theta_p)(R - V(s_\tau;\theta_v))$
    $d\theta_v \leftarrow d\theta_v + \partial(R - V(s_\tau;\theta_v))^2/\partial\theta_v$
  if s_t is terminal state **then**
    $d\lambda \leftarrow -(C - \alpha)$ ▷ Equation 8
    $t \leftarrow 0$
    $s_0 \sim \mu$
  Update θ_v, θ_p and λ
  Set $\lambda = \max(\lambda, 0)$ ▷ Ensure weights are non-negative (Equation 4)

Reward Constrained Policy Optimization

ALGORITHM

Algorithm 2 RCPO Advantage Actor Critic

The original Advantage Actor Critic algorithm is in gray, whereas our additions are highlighted in black.

Input: penalty function $C(\cdot)$, threshold α and learning rates η_1, η_2, η_3
Initialize actor $\pi(\cdot|\cdot;\theta_p)$ and critic $V(\cdot;\theta_v)$ with random weights
Initialize $\lambda = 0$, $t = 0$, $s_0 \sim \mu$ ▷ Restart
for $T = 1, 2, ..., T_{max}$ **do**
    Reset gradients $d\theta_v \leftarrow 0$, $d\theta_p \leftarrow 0$ and $\forall i : d\lambda_i \leftarrow 0$
    $t_{start} = t$
    while s_t `not terminal and` $t - t_{start} < t_{max}$ **do**
        Perform a_t according to policy $\pi(a_t|s_t;\theta_p)$
        Receive r_t, s_{t+1} and penalty score $\hat{C}_t$
        $t \leftarrow t + 1$
    $R = \begin{cases} 0 & , \text{ for terminal } s_t \\ V(s_t, \theta_v) & , \text{ otherwise} \end{cases}$
    for $\tau = t-1, t-2, ..., t_{start}$ **do**
        $R \leftarrow r_\tau - \lambda \cdot \hat{C}_\tau + \gamma R$ ▷ Equation 10
        $d\theta_p \leftarrow d\theta_p + \nabla_{\theta_p} \log \pi(a_\tau|s_\tau;\theta_p)(R - V(s_\tau;\theta_v))$
        $d\theta_v \leftarrow d\theta_v + \partial(R - V(s_\tau;\theta_v))^2/\partial\theta_v$
    if s_t is terminal state **then**
        $d\lambda \leftarrow -(C - \alpha)$ ▷ Equation 8
        $t \leftarrow 0$
        $s_0 \sim \mu$
    Update θ_v, θ_p and λ
    Set $\lambda = \max(\lambda, 0)$ ▷ Ensure weights are non-negative (Equation 4)

Reward Constrained Policy Optimization

ALGORITHM

Algorithm 2 RCPO Advantage Actor Critic

The original Advantage Actor Critic algorithm is in gray, whereas our additions are highlighted in black.

Input: penalty function $C(\cdot)$, threshold α and learning rates η_1, η_2, η_3
Initialize actor $\pi(\cdot|\cdot;\theta_p)$ and critic $V(\cdot;\theta_v)$ with random weights
Initialize $\lambda = 0$, $t = 0$, $s_0 \sim \mu$ ▷ Restart
for $T = 1, 2, ..., T_{max}$ **do**
    Reset gradients $d\theta_v \leftarrow 0$, $d\theta_p \leftarrow 0$ and $\forall i : d\lambda_i \leftarrow 0$
    $t_{start} = t$
    while s_t `not terminal and` $t - t_{start} < t_{max}$ **do**
        Perform a_t according to policy $\pi(a_t|s_t;\theta_p)$
        Receive r_t, s_{t+1} and penalty score $\hat{C}_t$
        $t \leftarrow t + 1$
        $R = \begin{cases} 0 & , \text{ for terminal } s_t \\ V(s_t, \theta_v) & , \text{ otherwise} \end{cases}$
    for $\tau = t-1, t-2, ..., t_{start}$ **do**
        $R \leftarrow r_\tau - \lambda \cdot \hat{C}_\tau + \gamma R$ ▷ Equation 10
        $d\theta_p \leftarrow d\theta_p + \nabla_{\theta_p} \log \pi(a_\tau|s_\tau;\theta_p)(R - V(s_\tau;\theta_v))$
        $d\theta_v \leftarrow d\theta_v + \partial(R - V(s_\tau;\theta_v))^2/\partial\theta_v$
    if s_t is terminal state **then**
        $d\lambda \leftarrow -(C - \alpha)$ ▷ Equation 8
        $t \leftarrow 0$
        $s_0 \sim \mu$
    Update θ_v, θ_p and λ
    Set $\lambda = \max(\lambda, 0)$ ▷ Ensure weights are non-negative (Equation 4)

Reward Constrained Policy Optimization

ALGORITHM

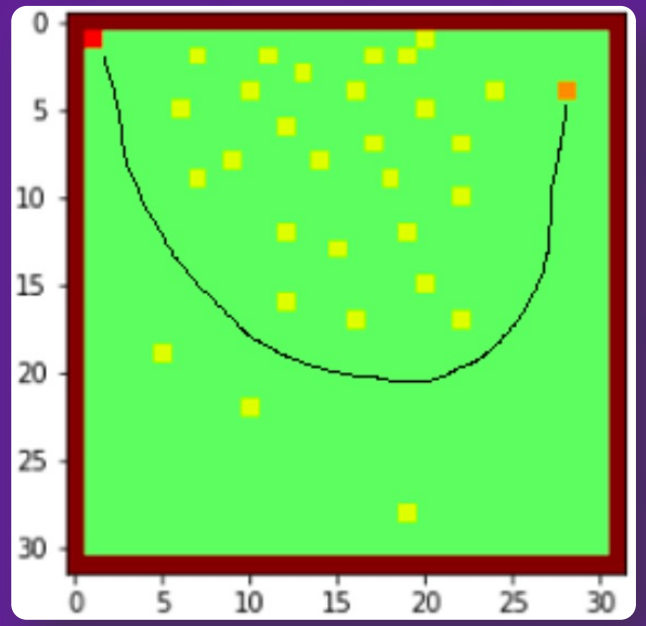
Algorithm 2 RCPO Advantage Actor Critic

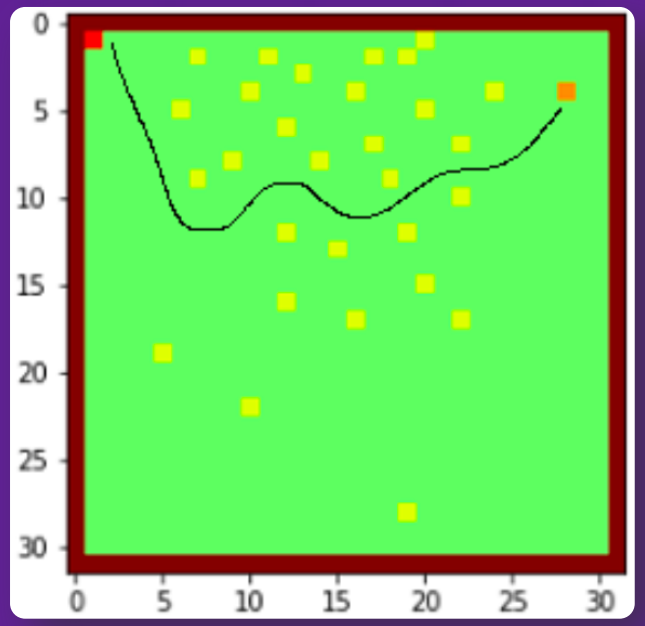
The original Advantage Actor Critic algorithm is in gray, whereas our additions are highlighted in black.

Input: penalty function $C(\cdot)$, threshold α and learning rates η_1, η_2, η_3
Initialize actor $\pi(\cdot|\cdot;\theta_p)$ and critic $V(\cdot;\theta_v)$ with random weights
Initialize $\lambda = 0$, $t = 0$, $s_0 \sim \mu$ ▷ Restart
for $T = 1, 2, ..., T_{max}$ **do**
    Reset gradients $d\theta_v \leftarrow 0$, $d\theta_p \leftarrow 0$ and $\forall i : d\lambda_i \leftarrow 0$
    $t_{start} = t$
    while s_t `not terminal and` $t - t_{start} < t_{max}$ **do**
        Perform a_t according to policy $\pi(a_t|s_t;\theta_p)$
        Receive r_t, s_{t+1} and penalty score $\hat{C}_t$
        $t \leftarrow t + 1$
        $R = \begin{cases} 0 & , \text{ for terminal } s_t \\ V(s_t, \theta_v) & , \text{ otherwise} \end{cases}$
    for $\tau = t - 1, t - 2, ..., t_{start}$ **do**
        $R \leftarrow r_\tau - \lambda \cdot \hat{C}_\tau + \gamma R$ ▷ Equation 10
        $d\theta_p \leftarrow d\theta_p + \nabla_{\theta_p} \log \pi(a_\tau|s_\tau;\theta_p)(R - V(s_\tau;\theta_v))$
        $d\theta_v \leftarrow d\theta_v + \partial(R - V(s_\tau;\theta_v))^2/\partial\theta_v$
    if s_t is terminal state **then**
        $d\lambda \leftarrow -(C - \alpha)$ ▷ Equation 8
        $t \leftarrow 0$
        $s_0 \sim \mu$
    Update θ_v, θ_p and λ
    Set $\lambda = \max(\lambda, 0)$ ▷ Ensure weights are non-negative (Equation 4)

Experiments

MARS ROVER

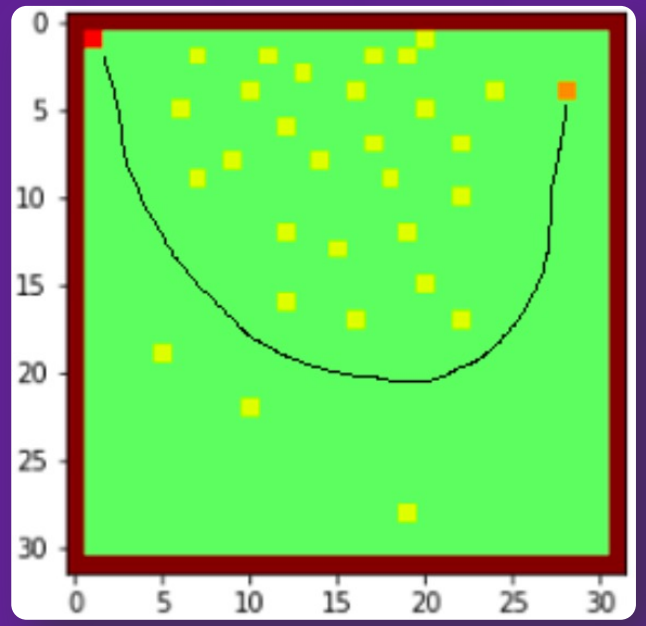




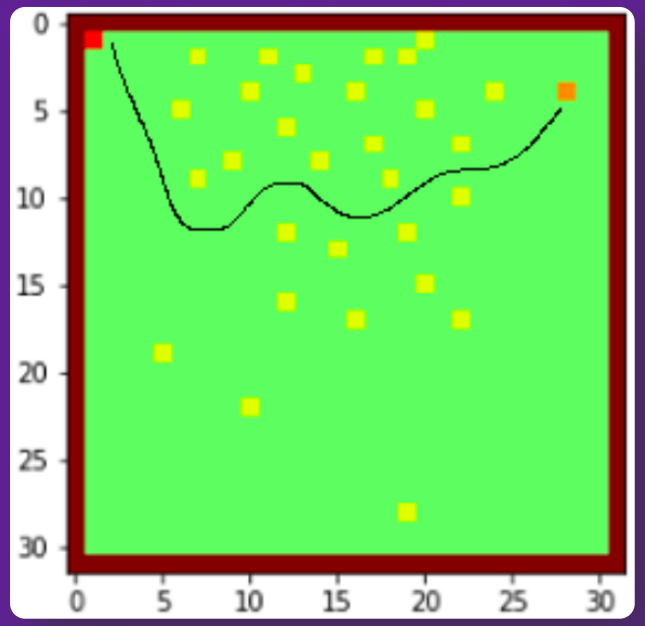
Experiments

MARS ROVER

$$\mathbb{P}_{\mu}^{\pi_\theta}(failure) \leq \alpha$$



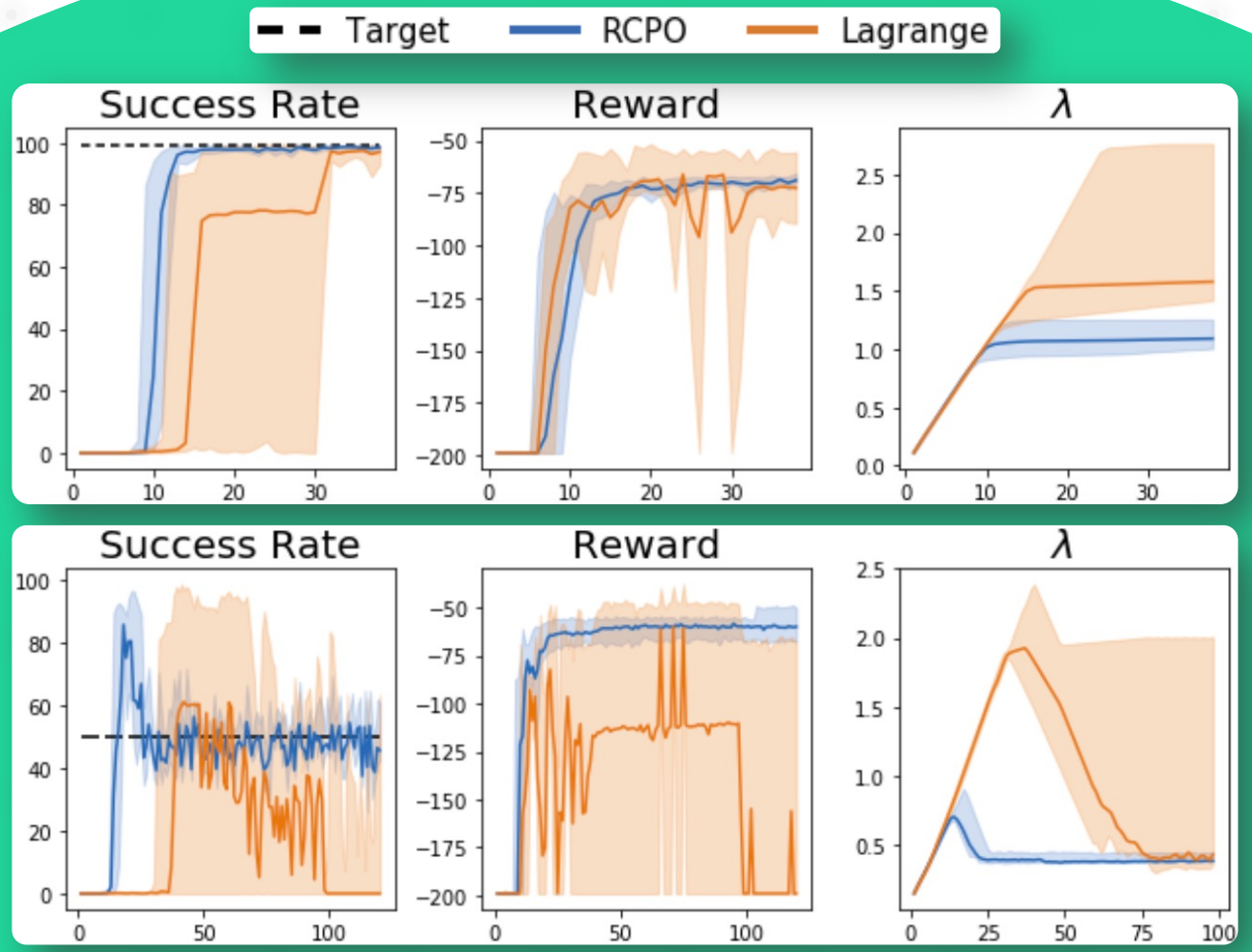
$\alpha = 0.01$



$\alpha = 0.5$

Experiments

MARS ROVER

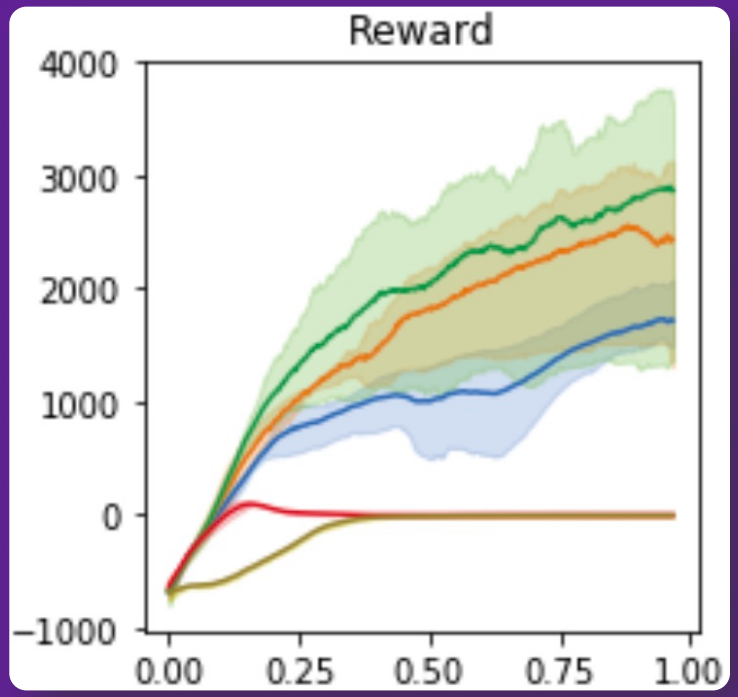


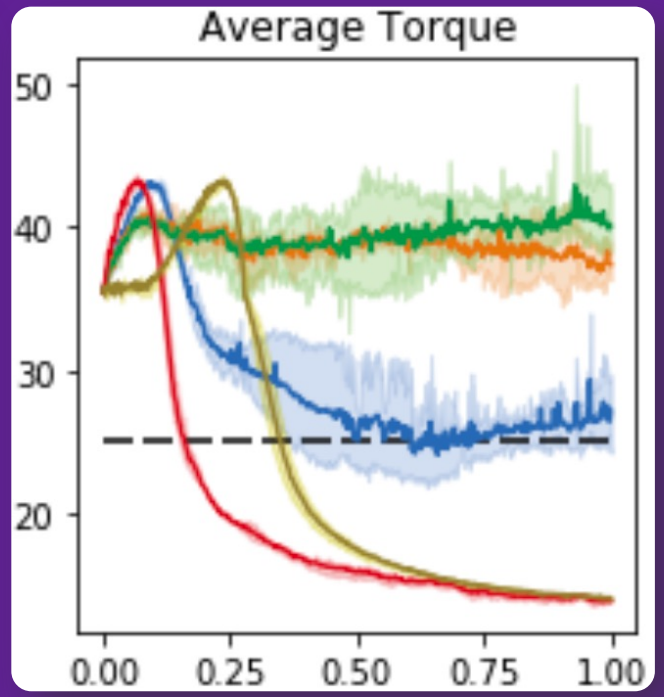
$\alpha = 0.01$

$\alpha = 0.5$

Experiments

ROBOTICS – HALFCHEETAH





Torque Threshold — RCPO — $\lambda = 0$ — $\lambda = 0.00001$ — $\lambda = 0.1$ — $\lambda = 100$

Discussion

QUESTIONS?