Supplement to the submission titled "Neural Network Compatible Off-Policy Natural Actor-Critic Algorithm"

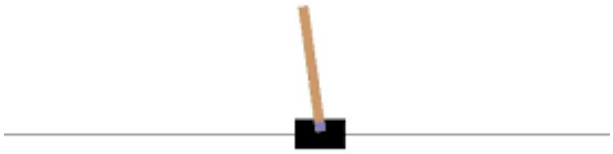
Contents

1 Description of RL tasks and Hyperparameters setting

In this section, we describe the benchmark RL tasks on which we test the algorithms. Further, we also discuss the hyperparameter setting that we set in our experiments.

1.1 CartPole

It consists of a pole attached by an unactuated joint to a cart. The cart can move on a frictionless track. Episode starts with an upright pole and the task is to balance the pole for maximum number of timesteps by applying either positive or negative force on the cart at each timestep. A **reward of +1** is received at each timestep until the episode ends. The episode ends if pole is more than 15° from vertical or cart moves more than 2.4 units from center. The maximum length of episode is **500 timesteps** and thus the maximum total reward that can be obtained in an episode would be +500.



State Space:

1. Cart Position
2. Cart Velocity
3. Pole Angle
4. Pole Angular Velocity

Actions:

1. +1 Force on Cart
2. -1 Force on Cart

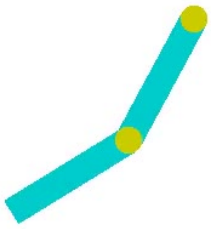
Algorithm	Hidden Layer Neurons(FC Layers)				Learning Rate				
	Actor	Value Critic	w	$\hat{w}$	Actor	Advantage Critic	Value Critic	w	$\hat{w}$
Deep AC	[16]	[64, 64]	-	-	0.001	-	0.005	-	-
Deep NAC	[16]	[64, 64]	-	-	0.001	0.001	0.01	-	-
Deep OffAC	[16]	[64, 64]	[16]	[16]	0.0005	-	0.01	0.001	0.001
Deep OffNAC	[16]	[64, 64]	[16]	[16]	0.0005	0.01	0.01	0.01	0.01

Table 1: Hyperparameters for Cart Pole

1.2 Acrobot

In the Acrobot task, we have two links attached to each other. One end of the first link is attached by an unactuated joint to the wall and the other end is attached to the second link by an actuated joint. Episode starts with the links hanging downwards. The task is to swing the lower end of second

link at a height atleast the length of one link above the un-actuated joint marked by a black line in minimum number of timesteps by applying torque on the actuated joint. Both the links can swing freely and can pass by each other. A **reward of -1** is received at each timestep until the episode ends. The maximum length of episode is **500 timesteps** and the maximum reward that can be obtained in an episode lies in the range between -90 to -80.



State Space:

1. $cos(\theta_1)$
2. $sin(\theta_1)$
3. $cos(\theta_2)$
4. $sin(\theta_2)$
5. Angular velocity of the first link
6. Angular velocity of second link with respect to first link

θ_1 is the angle made by first link with respect to vertically downwards direction. θ_2 is the angle made by second link with respect to first link.

Actions:

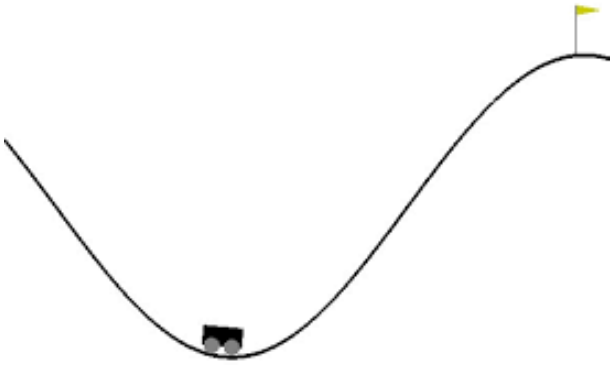
1. +1 Torque on actuated joint
2. 0 Torque on actuated joint
3. -1 Torque on actuated joint

Algorithm	Hidden Layer Neurons(FC Layers)				Learning Rate				
	Actor	Value Critic	w	$\hat{w}$	Actor	Advantage Critic	Value Critic	w	$\hat{w}$
Deep AC	[32]	[32, 32]	-	-	0.0005	-	0.001	-	-
Deep NAC	[32]	[32, 32]	-	-	0.0001	0.001	0.005	-	-
Deep OffAC	[32]	[32, 32]	[16]	[16]	0.0001	-	0.005	0.0001	0.0001
Deep OffNAC	[32]	[32, 32]	[16]	[16]	0.00005	0.0001	0.005	0.0001	0.0001

Table 2: Hyperparameters for Acrobot

1.3 Mountain Car

In this task, a car is positioned on a one-dimensional track positioned between two mountains. Episode starts with car in between the mountains. The task is to drive the car to the top of right mountain which is marked by a flag in minimum number of timesteps using car's engine. The car's engine is not strong enough to scale the mountain in a single pass. A **reward of -1** is received at each timestep until the episode ends. The maximum length of episode is **10000 timesteps** and the maximum reward that can be obtained in an episode lies in the range -110 to -120.



State Space:

1. Car Position
2. Car Velocity

Actions:

1. Accelerate to the left
2. Don't Accelerate
3. Accelerate to the right

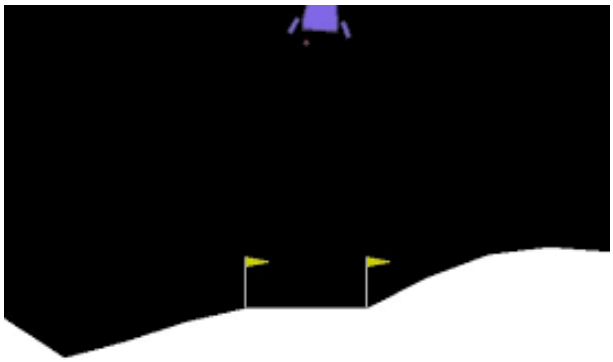
Algorithm	Hidden Layer Neurons(FC Layers)				Learning Rate				
	Actor	Value Critic	w	$\hat{w}$	Actor	Advantage Critic	Value Critic	w	$\hat{w}$
Deep AC	[32]	[32, 32]	-	-	0.001	-	0.005	-	-
Deep NAC	[32]	[32, 32]	-	-	0.00001	0.0001	0.005	-	-
Deep OffAC	[32]	[32, 32]	[16]	[16]	0.0001	-	0.005	0.01	0.01
Deep OffNAC	[32]	[32, 32]	[16]	[16]	0.000001	0.0001	0.005	0.01	0.01

Algorithm	Hidden Layer Neurons(FC Layers)				Learning Rate					λ
	Actor	Value Critic	w	$\hat{w}$	Actor	Advantage Critic	Value Critic	w	$\hat{w}$	
Deep AC TD(λ)	[32]	[32, 32]	-	-	0.005	-	0.05	-	-	0.7
Deep NAC TD(λ)	[32]	[32, 32]	-	-	0.0001	0.001	0.05	-	-	1
Deep OffAC TD(λ)	[32]	[32, 32]	[16]	[16]	0.0001	-	0.005	0.01	0.01	0.7
Deep OffNAC TD(λ)	[32]	[32, 32]	[16]	[16]	0.000001	0.0001	0.005	0.01	0.01	1

Table 3: Hyperparameters for Mountain Car

1.4 LunarLander

This task consists of lander and a landing pad marked by two flags. Episode starts with the lander moving downwards due to gravity. The task is to land the lander safely using different engines available on lander with zero speed on the landing pad as quickly and fuel efficiently as possible. Fuel is infinite and landing outside the landing pad is also possible. Reward for moving from the top of the screen and landing on landing pad with zero speed is between **100 to 140 points**. This reward is taken back if lander moves away from landing pad. Each leg ground contact yields a reward of **10 points**. Firing main engine leads to a reward of **-0.3 points** in each frame. Firing the side engine leads to a reward of **-0.03 points** in each frame. An additional reward of **-100 or +100 points** is received if the lander crashes or comes to rest respectively which also leads to end of the episode. The maximum length of episode is **1000 timesteps** and the maximum reward that can be obtained in an episode lies in the range +150 to +200.



State Space:

1. Horizontal Position
2. Vertical Position
3. Horizontal Velocity
4. Vertical Velocity
5. Angle
6. Angular Velocity
7. Left Leg Contact
8. Right Leg Contact

Actions:

1. Do Nothing
2. Fire Main Engine
3. Fire Left Engine
4. Fire Right Engine

Algorithm	Hidden Layer Neurons(FC Layers)				Learning Rate				
	Actor	Value Critic	w	$\hat{w}$	Actor	Advantage Critic	Value Critic	w	$\hat{w}$
Deep AC	[16]	[64, 64]	-	-	0.001	-	0.005	-	-
Deep NAC	[16]	[64, 64]	-	-	0.0001	0.001	0.005	-	-
Deep OffAC	[128]	[128, 128]	[16]	[16]	0.001	-	0.01	0.0001	0.0001
Deep OffNAC	[128]	[128, 128]	[16]	[16]	0.00001	0.005	0.01	0.001	0.001

Table 4: Hyperparameters for Lunar Lander

2 Natural Actor-Critic with TD(λ) Critic

In this section, we propose a family of algorithms parameterized by λ, where we replace TD(0) prediction used in the value critic by the TD(λ) prediction [2]. We refer to the new family of algorithms as "Deep NAC(λ)". The parameter λ helps in trading the bias and variance in the value function estimation of the policy. By choosing a suitable value of λ (based on nature of the task), we can improve the prediction of the value function. This in turn can help the actor to compute a better policy. The complete description of the on-policy algorithm is provided in Algorithm 1 and the off-policy algorithm is provided in Algorithm 2.

Algorithm 1 On-Policy Deep Natural Actor-Critic with TD(λ) prediction (Deep NAC(λ))

Input: $0 \leq \lambda \leq 1$.
Initialize the policy network parameter θ.
Initialize the value Function network parameter ψ.
Initialize the advantage value function parameter x.

for $n = 0, \ldots, \infty$ **do**
  Initialise $s \sim d_0(.)$
  $z \leftarrow 0$
  while the trajectory has not terminated **do**
    Obtain an action $a \sim \pi_\theta(s, .)$.
    Obtain next state and reward from the environment $(s', r) \sim \mathcal{E}$.
    $z \leftarrow \gamma\lambda z + \nabla_\psi V_\psi(s)$
    $\psi \leftarrow \psi + \alpha_n \big(r + \gamma V_\psi(s') - V_\psi(s)\big) z$
    $x \leftarrow x + \alpha_n \big(r + \gamma V_\psi(s') - V_\psi(s) - x^T \nabla_\theta \log \pi_{\theta_t}(s, a)\big) \nabla_\theta \log \pi_\theta(s, a)$
    $\theta \leftarrow \theta + \beta_n x$
    $s \leftarrow s'$

Algorithm 2 Off-Policy Deep Natural Actor-Critic with TD(λ) prediction (Deep OffNAC(λ))

Input: $0 \leq \lambda \leq 1$.
μ : Behavior policy.
Initialize the policy network parameter θ.
Initialize the value function network parameter ψ.
Initialize the advantage value function parameter x.

for $n = 0, \ldots, \infty$ **do**
  Initialize $s \sim d_0(.)$
  $z \leftarrow 0$
  while the trajectory has not terminated **do**
    Estimate $\hat{w}_\theta$ and w_θ using Algorithms 1 and 2, respectively, of [1].
    Obtain an action $a \sim \mu(s, .)$.
    Obtain next state and reward from the environment $(s', r) \sim \mathcal{E}$.
    Set $\rho(s, a) = \frac{\pi_\theta(s,a)}{\mu(s,a)}$.
    $z \leftarrow \gamma\lambda z + \nabla_\psi V_\psi(s)$
    $\psi \leftarrow \psi + \alpha_n \Big(\hat{w}_\theta(s)\rho(s, a)\big(r + \gamma V_\psi(s') - V_\psi(s)\big)z \Big)$
    $x \leftarrow x + \alpha_n \Big(w_\theta(s)\rho(s, a)\big(r + \gamma V_\psi(s') - V_\psi(s) - x^T \nabla_\theta \log \pi_{\theta_t}(s, a)\big) \nabla_\theta \log \pi_\theta(s, a) \Big)$
    $\theta \leftarrow \theta + \beta_n x$
    $s \leftarrow s'$

References

[1] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. *arXiv preprint arXiv:1810.12429*, 2018.

[2] Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. MIT press Cambridge, 2018.