A Joint Imitation-Reinforcement Learning Framework for Reduced Baseline Regret

Technical Report

A detailed description of the experimental settings and hyperparameters used to obtain the results reported in our paper has been presented in this report.

I. Reward Functions

The reward function used in each of the domains is listed below.

Inverted pendulum: The reward at time step t, r_t, is given by,

$$r_t = -\theta_t^2 - 0.1 \times \dot{\theta}_t^2 - 0.01 \times u_t^2$$

where θ_t is the normalized angle the pendulum makes with the vertical axis, $\dot{\theta}_t$ is the angular velocity, and u_t is the torque applied on the pendulum.

Lunar lander: We used the default reward function provided in the OpenAI gym.

Lane following (CARLA): The reward at time step t, r_t, is given by,

$$r_t = \begin{cases} -10, & \text{if } s_t = \text{terminal} \\ 1 + 0.1 \times \text{throttle}_t, & \text{otherwise} \end{cases}$$

A living reward of 1 was assigned for every time-step it stayed inside the lane. The actual throttle for the car was restricted in the range $0.4 - 0.6$. Restricting the range of the throttle value helped in reducing the training time as the car takes a long time to learn to move forward in the early stages of the training. But, it came at a cost of the car not being able to learn to brake or move at the maximum possible speed.

Lane following (JetRacer): The reward at time step t, r_t, is given by,

$$r_t = 1 + 0.1 \times \text{throttle}_t$$

A living reward of 1 was assigned for every time-step it stayed inside the lane. The actual throttle for the car was restricted in the range $0.45 - 0.6$.

Walker-2D: We used the default reward function provided in the PyBullet environment.

II. Baseline Policies

Sub-optimal baseline policies were defined in order to demonstrate that JIRL can learn from and outperform such a demonstrator. To obtain these baselines, we followed the procedure as described below for each domain.

Inverted pendulum: The baseline policy was obtained after training an RL agent using vanilla SAC for only $1,000$ environment steps. Additionally, the reward function was tweaked to encourage the agent to swing the pendulum using lower torque values than what is optimal.

Lunar lander: The baseline policy was obtained after training an RL agent using vanilla SAC for only $400,000$ environment steps.

Lane following (CARLA): The baseline policy was trained on $40,000$ state-action pairs collected by manually driving around a custom track using supervised learning (behavior cloning). The agent was also trained on action sequences of recovering from unsafe states.

Lane following (JetRacer): The baseline policy was trained on $10,000$ state-action pairs collected by manually driving on a custom track using supervised learning (behavioural cloning). We also used a human expert during the training phase of JIRL as a method to provide action corrections manually when the car tried to venture out of track. This strategy was adopted due to the inability of the baseline policy to make corrections when the car approached the edge of track in a few instances. Access to a more robust baseline policy can obviate the necessity of a human expert to oversee the training process.

Walker-2D: The baseline policy was obtained after training an RL agent using vanilla SAC for only 1 million environment steps.

III. Hyperparameters

A. JIRL Hyperparameters

Table I lists the domain-wise hyperparameters used in the JIRL framework to obtain the reported results.

Parameter	IP	LL	LF (CARLA)	LF (JetRacer)	W-2D
K	1	1	5	10	1
σ^2	0.01	0.1	0.01	0.01	0.01

TABLE I: JIRL Domain Specific Hyperparameters for Inverted pendulum (IP), Lunar lander (LL), Lane following (LF), Walker-2D (W-2D).

In our experiments, we observed that K plays an important role in reducing the baseline regret in the Lane following and the Walker-2D domains. Increasing the value of K results in a lower baseline regret. We observed that values of K between $5 - 10$ and $1 - 3$ achieve an acceptable trade-off between reducing the baseline regret and training time for the Lane following domain and the Walker-2D domain respectively. For the Inverted pendulum and Lunar lander domains, all values of K between $1 - 5$ gave similar results in terms of reducing the baseline regret. σ^2 may be set in the range $0.01 - 0.1$ depending on how close the baseline's policy is to the optimal policy.

B. SAC Hyperparameters

Table II lists the hyperparameters shared across all the tasks in the JIRL+SAC framework and in vanilla SAC. Table III contains the domain-specific SAC hyperparameters.

Parameter	Value
Optimizer	`Adam`
Learning rate	3×10^{-4}
Discount (γ)	0.99
Target smoothing coefficient (τ)	0.005
Target update interval	1
Number of hidden layers (all networks)	2
Nonlinearity	`ReLU`

TABLE II: SAC Shared Hyperparameters

Parameter	IP	LL	LF (CARLA)	LF (JetRacer)	W-2D
Number of actions	1	2	2	2	6
Number of hidden units per layer	64	64	64	64	256
Entropy coefficient	0.05	0.05	0.01	0.01	0.01
Replay buffer size	50,000	10^5	30,000	10,000	10^6
Time steps per update	1	1	1	400	1
Gradient steps per update	1	1	1	200	1

TABLE III: SAC Domain Specific Hyperparameters

C. TRPO Hyperparameters

Table IV contains the domain-specific TRPO hyperparameters.

Parameter	IP	LL	W-2D
Number of actions	1	2	6
Number of hidden units per layer	64	64	256
KL loss threshold	1.93×10^{-4}	0.01	0.01
Time steps per batch	1024	1024	2048
Iterations for conjugate gradient	10	10	20
Entropy coefficient	0.01118	0	0.001
Compute gradient damping factor	2.35×10^{-5}	0.1	0.1
Value function step size	0.00428	0.001	0.001
Value function learning iterations	10	5	5
GAE factor	0.9	0.98	0.95
Optimizer	`Adam`	`Adam`	`Adam`
Discount (γ)	0.99	0.99	0.99
Number of hidden layers	2	2	2
Nonlinearity	`ReLU`	`ReLU`	`ReLU`

TABLE IV: TRPO Domain Specific Hyperparameters