Multi Objective PPO

git@github.com:Yaishnavi/MultiObjective-PPO.git

Contents

Introduction

Multi-Objective RL problem with k different reward functions. Find a policy that performs well for all k reward functions.

Implemented the following algorithm:

Use PPO to find policy pi_1 with respect to rewards r_1

For k = 2 to K

Use PPO to find policy pi_k where reward is r_k + a*KL(pi_{k-1}|| pi_k)

The **Kullback-Leibler Divergence** score, or KL divergence score, quantifies how much one probability distribution differs from another probability distribution.

The KL divergence between two distributions Q and P is often stated using the following notation: KL(P || Q)

Where the "||" operator indicates "divergence" or Ps divergence from Q.

PPO is a policy gradient method for reinforcement learning, which alternates 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.

Proximal Policy Optimization

PPO can be viewed as an approximation of TRPO, but unlike TRPO, which uses a second-order Taylor expansion, PPO uses only a first-order approximation, which makes PPO very effective in RNN networks and in a wide distribution space.

for $i \in 1, 2, \ldots, N$ do:
    Run policy π_θ for T timesteps, collecting s_t, a_t, r_t
    $\pi_{\text{old}} \leftarrow \pi_\theta$
    for $j \in 1, 2, \ldots, M$ do:

$$J_{\text{PPO}}(\theta) = \sum_{t=1}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)} \hat{A}_t - \lambda \text{KL}[\pi_{\text{old}}|\pi_\theta]$$

        Update θ by a gradient method w.r.t. $J_{\text{PPO}}(\theta)$
    end for
    for $j \in 1, 2, \ldots, B$ do:

$$L_{\text{BL}}(\phi) = -\sum_{t=1}^{T} (\sum_{t'>t} \gamma^{t'-t} r_{t'} - V_\phi(s_t))^2$$

        Update ϕ by a gradient methor w.r.t. $L_{\text{BL}}(\phi)$
    end for
    if $\text{KL}[\pi_{\text{old}}|\pi_\theta] > \beta_{\text{high}} \text{KL}_{\text{target}}$ then
        $\lambda \leftarrow \alpha\lambda$
    if $\text{KL}[\pi_{\text{old}}|\pi_\theta] < \beta_{\text{high}} \text{KL}_{\text{target}}$ then
        $\lambda \leftarrow \alpha/\lambda$

The first half of Estimate Advantage is obtained through the rollout strategy, and the second half of V is obtained from a value network. (Value network can be trained by the data obtained by rollout, where the mean square error is used).

There is a clipped surrogate objective,

$$L^{\text{CLIP}}(\theta) = \hat{E}_t[r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$
$$\text{where } r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

Dataset

Fruit API is a universal deep reinforcement learning framework, which is designed meticulously to provide a friendly user interface, a fast algorithm prototyping tool, and a multi-purpose library for the RL research community. External environments can be integrated into the framework easily by plugging into FruitEnvironment. Finally, we developed 5 extra environments as a testbed to examine different disciplines in deep RL:

- Mountain car (multi-objective environment/graphical support)
- Deep sea treasure (multi-objective environment/graphical support)

- Tank battle (multi-agent/multi-objective/human-agent cooperation environment)
- Food collector (multi-objective environment)
- Other environments used but not evaluated for this task are Milk factory and fruit tree.

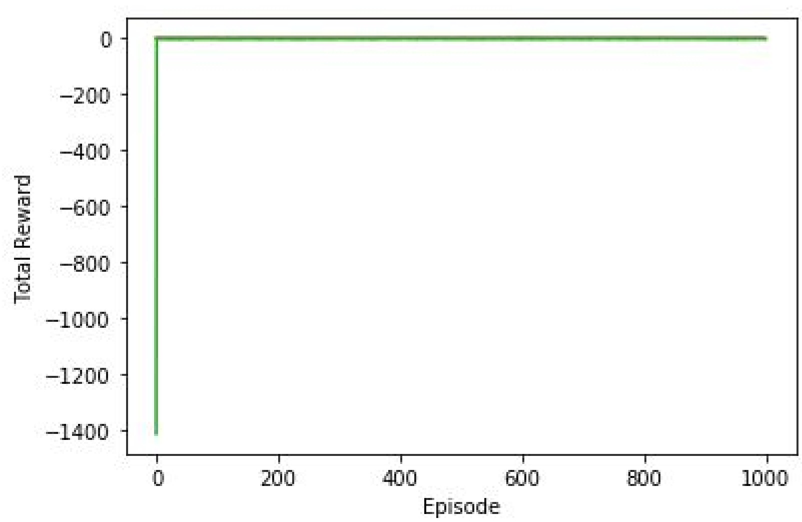
	Number of rewards/objectives	Number of actions	State Dimension
Mountain Car	3	3	1
Deep Sea Treasure	2	4	2
Tank Battle	2	5	650x650x3
Food Collector	2	6	300x325x3
Mountain Car 2	2	3	1

Experiment

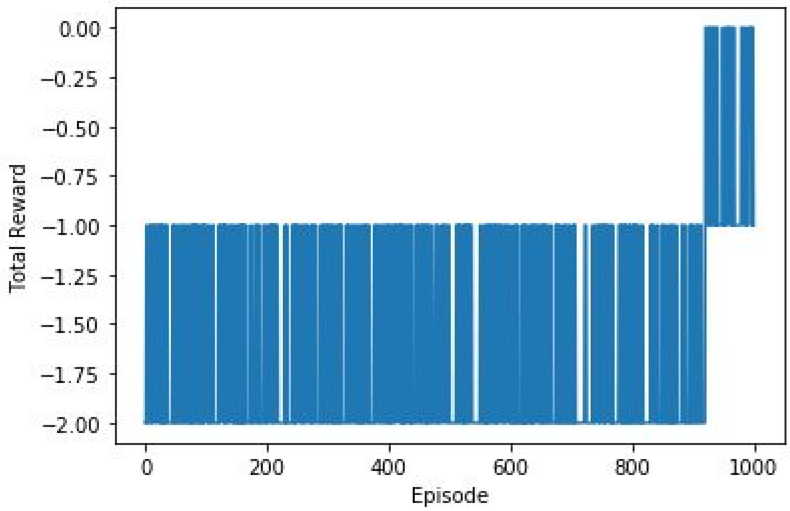
PPO algorithm is used to compute the policy for each objective/reward function. This objective is updated with the KL divergence of the previous policy. The baseline using a linear combination of the rewards to determine the best policy. Here dense neural network model is used for all of the tasks mentioned above.

Results

1. Mountain Car

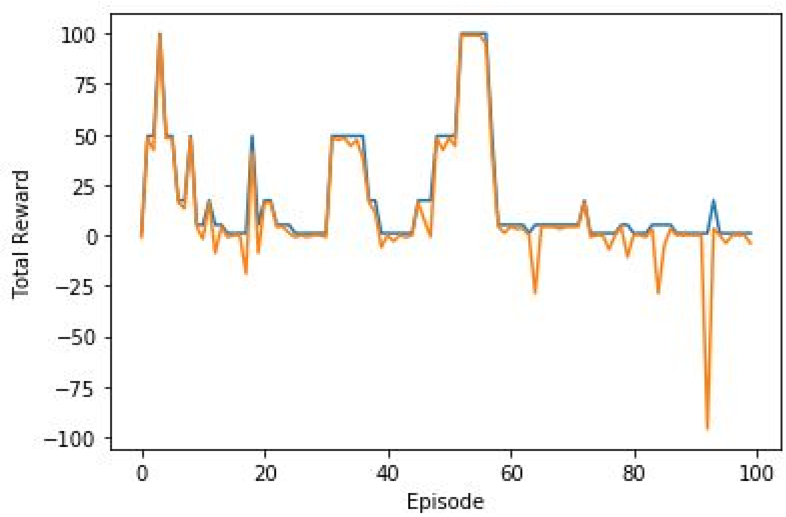


Proposed method

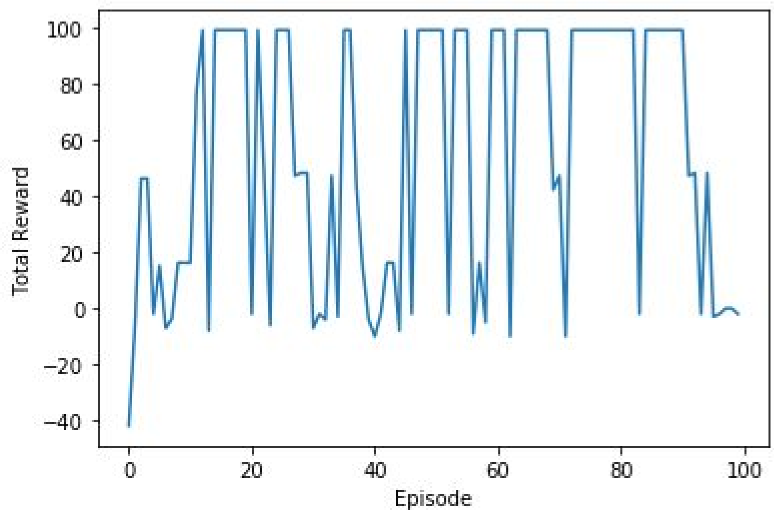


Baseline

2. Deep Sea Treasure

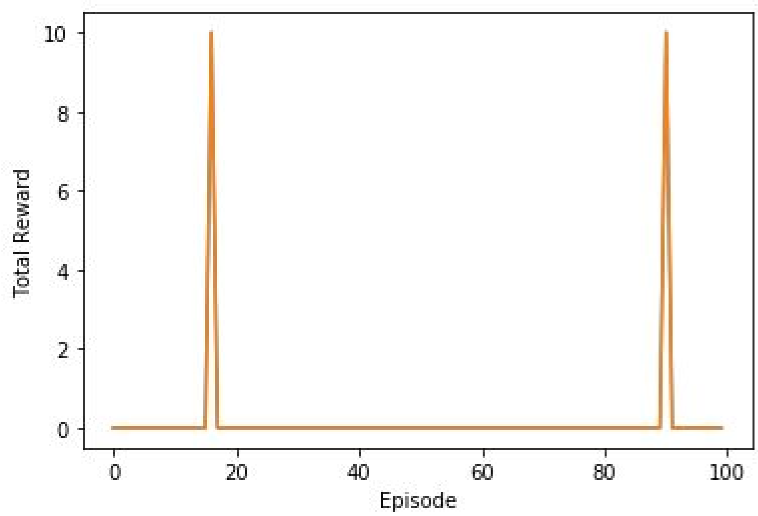


Proposed method

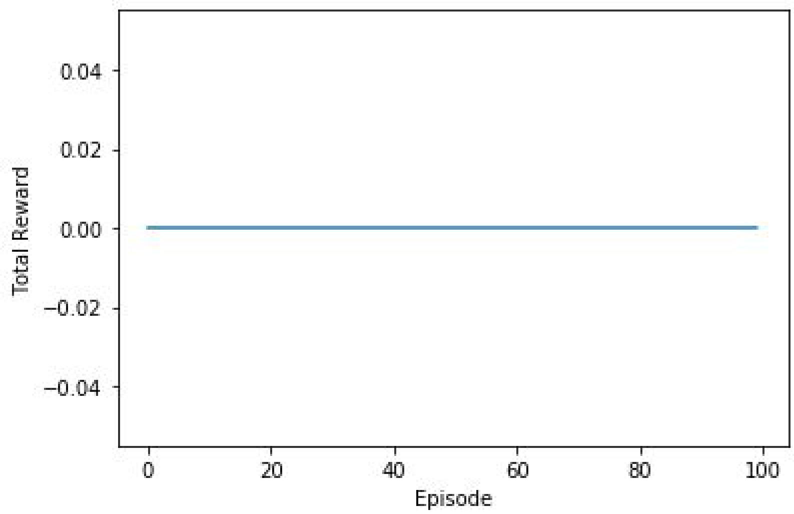


baseline

3. Tank Battle

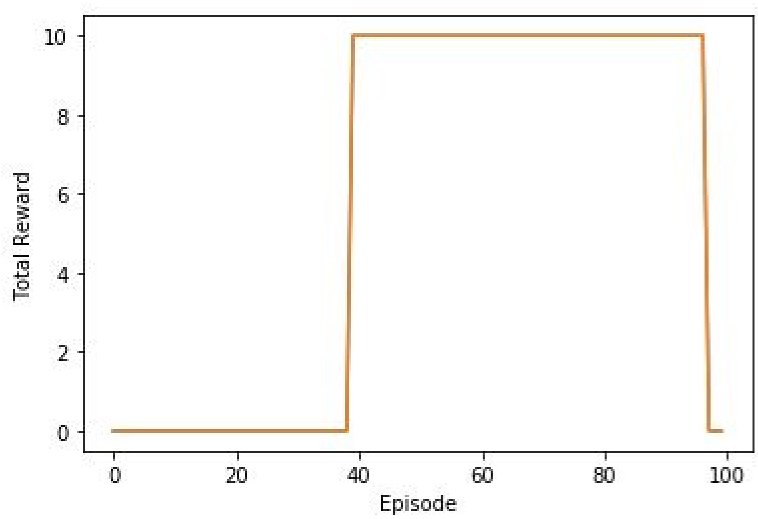


Proposed Method

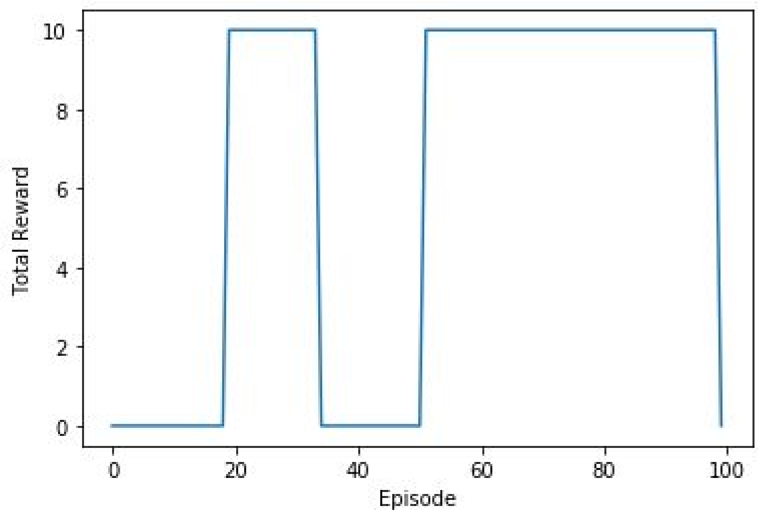


baseline

4. Food Collector

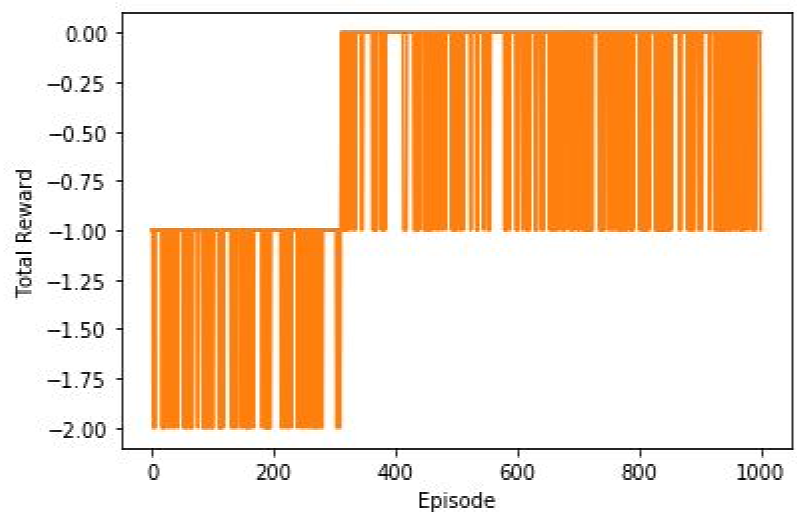


Proposed method

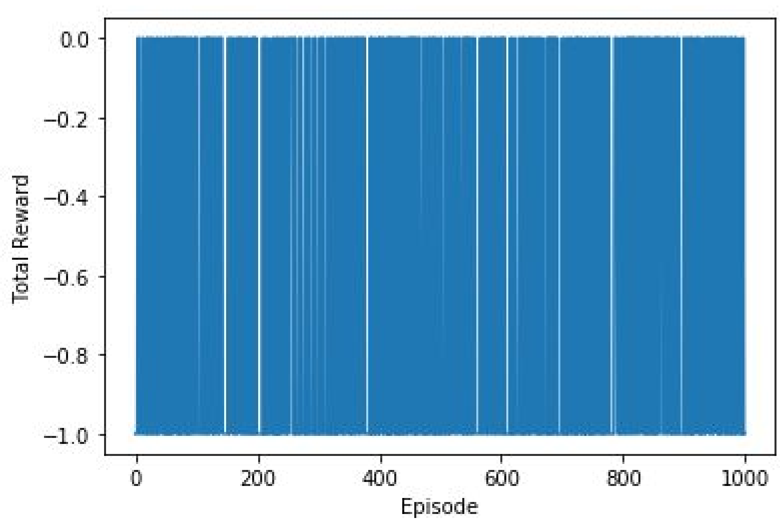


baseline

5. Mountain Car2



Proposed Method



baseline

Installation

1. Install fruit api to access environments
   Git clone https://github.com/garlicdevs/Fruit-API.git
   Open install path and run python setup.py install
2. Other packages needed: Python==3.6, Pytorch, Matplotlib, Numpy

References

- Dataset: FruitAPI https://fruitlab.org/
- Generalizing across Multi-Objective Reward Functions in Deep Reinforcement Learning https://arxiv.org/pdf/1809.06364.pdf
- Proximal Policy Optimization Algorithms https://arxiv.org/pdf/1707.06347.pdf
- Multi Objective Deep Reinforcement Learning https://arxiv.org/pdf/1610.02707.pdf