rlpack Documentation

0.1

x

2020 08 02

Contents

CHAPTER 1

rlpack **tensorflow**

- TensorFlow Numpy
-
-

1.1

rlpack MuJoCo *PPO*

```
import argparse
import time
from collections import namedtuple

import gym
import numpy as np
import tensorflow as tf

from rlpack.algos import PPO
from rlpack.utils import mlp, mlp_gaussian_policy

parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('--env',  type=str, default="Reacher-v2")
args = parser.parse_args()

Transition = namedtuple('Transition', ('state', 'action', 'reward', 'done', 'early_stop',
↪ 'next_state'))
```

( )

( )

```
class Memory(object):
    def __init__(self):
        self.memory = []

    def push(self, *args):
        self.memory.append(Transition(*args))

    def sample(self):
        return Transition(*zip(*self.memory))


def policy_fn(x, a):
    return mlp_gaussian_policy(x, a, hidden_sizes=[64, 64], activation=tf.tanh)


def value_fn(x):
    v = mlp(x, [64, 64, 1])
    return tf.squeeze(v, axis=1)


def run_main():
    env = gym.make(args.env)
    dim_obs = env.observation_space.shape[0]
    dim_act = env.action_space.shape[0]
    max_ep_len = 1000

    agent = PPO(dim_act=dim_act, dim_obs=dim_obs, policy_fn=policy_fn, value_fn=value_fn,
↪ save_path="./log/ppo")

    start_time = time.time()
    o, ep_ret, ep_len = env.reset(), 0, 0
    for epoch in range(50):
        memory, ep_ret_list, ep_len_list = Memory(), [], []
        for t in range(1000):
            a = agent.get_action(o[np.newaxis, :])[0]
            nexto, r, d, _ = env.step(a)
            ep_ret += r
            ep_len += 1

            memory.push(o, a, r, int(d), int(ep_len == max_ep_len or t == 1000-1), nexto)

            o = nexto

            terminal = d or (ep_len == max_ep_len)
            if terminal or (t == 1000-1):
                if not(terminal):
                    print('Warning: trajectory cut off by epoch at %d steps.' % ep_len)
                if terminal:
                    #
                    ep_ret_list.append(ep_ret)
                    ep_len_list.append(ep_len)
```

( )

( )

```
                o, ep_ret, ep_len = env.reset(), 0, 0

        print(f"{epoch}th epoch. average_return={np.mean(ep_ret_list)}, average_len={np.
↪mean(ep_len_list)}")

        #
        batch = memory.sample()
        agent.update([np.array(x) for x in batch])

    elapsed_time = time.time() - start_time
    print("elapsed time:", elapsed_time)


if __name__ == "__main__":
    run_main()
```

1.2

Python3.6+ is required.

1.

environment.yml. Anaconda *python*

```
$ git clone https://github.com/liber145/rlpack
$ cd rlpack
$ conda env create -f environment.yml
$ conda activate py36
```

2. *rlpack*

```
$ python setup.py install
```

gym. MuJoCo gym

CHAPTER 2

Benchmarks

2.1 Mujoco game

	DDPG	TRPO	PPO	TD3
Ant-v2		609.61	969.08	1769.52
HalfCheetah-v2		667.06	2607.94	6108.17
Hopper-v2		1460.93	2100.74	2515.44
Humanoid-v2		339.35	458.59	278.14
HumanoidStandup-v2		58715.82	81282.21	84551.70 (1M: 90523.85)
InvertedDoublePendulum-v2		8131.25	6606.13	8342.53 (1M: 8925.03)
InvertedPendulum-v2		900.08	943.31	940.33 (1M: 972.17)
Reacher-v2	-13.96	-10.08	-10.34	-9.94
Swimmer-v2		38.05	44.17	43.55
Walker2d-v2		493.14 (1M: 1373.30)	1138.25	1008.70 (1M: 3394.72)

Performance on 500,000 sample steps.

CHAPTER 3

DQN

DQN is an off-policy algorithm.

3.1 Quick Review

DQN lighted the fire of reinforcement learning. It introduces deep learning to Q-learning and proposes two key ideas to make the learning not divergent [1].

The optimization objective of DQN can be formated by:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

The two key ingredients are in the above equation:

1. $U(D)$ means to uniform sample experienced transitions (s, a, r, s') from an experience replay buffer D. This alleviates the correlations in the observed sequence and smoothes over changes in the data distribution.
2. $Q(s', a'; \theta_i^-)$ is a target action value function, which helps reducing correlations with the target.

3.2 Reference

[1] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529.

CHAPTER 4

A2C

Advantage Actor Critic is an off-policy algorithm.

4.1 Quick Review

First, let's look at REINFORCE algorithm, which is a Monte Carlo policy gradient aglrothm.

$$\begin{aligned}\nabla\eta(\theta) &= \sum_s d_\pi(s)\sum_a q_\pi(s,a)\nabla_\theta\pi(a|s;\theta)\\ &= \mathbb{E}_{(s_t,a_t)\sim\pi}\gamma^t q_\pi(s_t,a_t)\nabla_\theta\pi(a_t|s_t;\theta)\end{aligned}$$

REINFORCE algorithm uses Monte Carlo method to estimate the expected $q(s_t, a_t)$.

Note that $\mathbb{E}_{(s_t,a_t)\sim\pi}b(s_t)\nabla_\theta\pi(a_t|s_t;\theta) = 0$, we have

$$\nabla\eta(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi}\gamma^t[q_\pi(s_t,a_t) - b(s_t)]\nabla_\theta\pi(a_t|s_t;\theta)$$

The term $b(s_t)$ called *baseline* is usually estimated by state value $v(s_t)$. The residual term $q(s_t, a_t) - v(s_t)$ is called *advantage*. In general, the baseline leaves the expected value of the update unchanged, but it can have a large effect on reducing its variance [1].

Now, let's go to advantage actor critic (A2C). Instead, A2C uses a state value approximate function to estimate $v(s_t;\theta)$. The action value can be derived as $q(s_t, a_t) = r_t + \gamma v(s_{t+1};\theta)$. The critic part updates the value function from TD error. The actor part updates the policy function by policy gradient.

4.2 Reference

[1] Sutton, Richard S., and Andrew G. Barto. "Reinforcement Learning: An Introduction." (1998).

CHAPTER 5

TRPO

TRPO Trust Region Policy Optimization TRPO

-
-

5.1

π $J(\pi) = \mathbb{E}_{s_0,a_0,\ldots\sim\pi} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$. Sham Kakade 2012 $\tilde{\pi}$ π

$$\begin{aligned} J(\tilde{\pi}) - J(\pi) &= \mathbb{E}_{s_0,a_0,\ldots\sim\tilde{\pi}} \sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \\ &= \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a). \end{aligned}$$

$A_\pi(s_t, a_t)$ $A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$.

π :math:‘tilde{pi}(a|s)‘ importance sampling

$$\sum_a \pi(a|s) \frac{\tilde{\pi}(a|s)}{\pi(a|s)} A_\pi(s, a),$$

$\rho_{\tilde{\pi}}(s)$ TRPO $\rho_\pi(s)$

$$L_\pi(\tilde{\pi}) = \sum_s \rho_\pi(s) \sum_a \pi(a|s) \frac{\tilde{\pi}(a|s)}{\pi(a|s)} A_\pi(s, a)$$

Actor Critic Actor Critic TRPO KL

$$\begin{aligned} &\max_{\tilde{\pi}} \sum_s \rho_\pi(s) \sum_a \pi(a|s) \frac{\tilde{\pi}(a|s)}{\pi(a|s)} A_\pi(s, a) \\ &s.t. \quad \mathbb{E}_{s\sim\rho_\pi} D_{KL}(\pi(\cdot|s) \| \tilde{\pi}(\cdot|s)) \leq \epsilon \end{aligned}$$

5.2

-
-

TRPO

$$J(\tilde{\pi}) - J(\pi) \geq L_{\pi}(\tilde{\pi}) - CD_{KL}^{\max(\pi,\tilde{\pi})}$$
$$C = \frac{4\gamma\epsilon}{(1-\gamma)^2}, \epsilon = \max_{s,a} |A(s,a)|.$$

$J(\tilde{\pi}) - J(\pi)$ $L_{\pi}(\tilde{\pi})$ KL

5.3

$$\mathbb{E}_{s\sim\rho_{\pi_{\theta_{old}}}(\cdot),a\sim\pi_{\theta_{old}}(\cdot|s)} \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} A_{\pi_{\theta_{old}}}(s,a) = g^{\top}(\theta - \theta_{old}) + K_0,$$

:math‘g‘ $A_{\pi_{\theta_{old}}}(s,a)\pi_{\theta}(a|s)/\pi_{\theta_{old}}(a|s)$ $\theta = \theta_{old}$ K_0 θ

$$\mathbb{E}_{s\sim\rho_{\pi_{\theta_{old}}}(\cdot)} D_{\alpha}(\pi_{\theta_{old}}(\cdot|s) \| \pi_{\theta}(\cdot|s)) = \frac{1}{2}(\theta - \theta_{old})^{\top} H (\theta - \theta_{old}) + K_1,$$

H $\theta = \theta_{old}$ K_1 θ $\theta = \theta_{old}$

θ

$$\min_{\theta} \; -g^{\top}(\theta - \theta_{old})$$
$$s.t. \; \frac{1}{2}(\theta - \theta_{old})^{\top} H (\theta - \theta_{old}) \leq \epsilon.$$

$$\min_{\theta} \max_{\lambda \geq 0} \; L(\theta, \lambda) = -g^{\top}(\theta - \theta_{old}) + \lambda \cdot (\frac{1}{2}(\theta - \theta_{old})^{\top} H (\theta - \theta_{old}) - \epsilon).$$

KKT $L(\theta, \lambda)$ $\partial L/\partial\theta = 0$ $\theta = \theta_{old} + \lambda^{-1}H^{-1}g.$ $\partial L/\partial\lambda = 0$ $\lambda = \sqrt{(g^{\top}H^{-1}g)/(2\epsilon)}.$ $\theta = \theta_{old} + \sqrt{2\epsilon(g^{\top}H^{-1}g)^{-1}}H^{-1}g.$

5.4

[1] Schulman, John, et al. ”Trust region policy optimization.” International Conference on Machine Learning. 2015.

CHAPTER 6

PPO

PPO Proximal Policy Optimization PPO TRPO

6.1

PPO TRPO

$$\max_{\theta} \mathbb{E}_{s\sim\rho_{\pi_{\theta_{old}}}(\cdot),a\sim\pi_{\theta_{old}}(\cdot|s)} \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}A_{\pi_{\theta_{old}}}(s,a), clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, 1-\epsilon, 1+\epsilon\right)A_{\pi_{\theta_{old}}}(s,a)\right)$$

TRPO clip min

6.2

$$A_{\pi_{old}}(s,a) = Q_{\pi_{old}}(s,a) - V_{\pi_{old}}(s)$$

- $A > 0$ min clip $1+\epsilon$ $1+\epsilon$
- $A < 0$ $-\max(r, clip(r, 1-\epsilon, 1+\epsilon))A$ r min clip $1-\epsilon$ $1-\epsilon$

6.3

[1] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

CHAPTER 7

DDPG

DDPG is an off-policy algorithm.

7.1 Quick Review

DDPG is the deep learning vergion of deterministic policy gradient (DPG) algorithm [2]. DPG consider policy gradient algorithm in the context of deterministic policy.

Simliar to policy gradient theorem, [2] gives a deterministic policy gradient theorem,

$$\begin{aligned}\nabla_\theta J(\mu_\theta) &= \sum_s \rho^\mu(s)\nabla_\theta\mu_\theta(s)\nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^\mu}\nabla_\theta\mu_\theta(s)\nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\end{aligned}$$

The action value udpate is to minimize TD error between target value and current value as usual.

7.2 Implementation

The policy update can be rewritten to $\nabla_\theta Q(s,\mu_\theta(s))$. We can write the policy loss as $\mathbb{E}_s[-Q(s,\mu_\theta(s))]$, then pick an optimizer to do gradient descent on policy loss and value loss iteratively.

Given a state, straightforward action inference by old policy makes no exploration. [1] introduces an Ornstein-Uhlenbeck process to generate temporally correlated exploration.

7.3 Reference

[1] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

[2] Silver, David, et al. "Deterministic policy gradient algorithms." ICML. 2014.

CHAPTER 8

rlpack.algos package

Relevant tutorials: *PPO*.

Relevant tutorials: *TRPO*.

Relevant tutorials: *A2C*.

Relevant tutorials: *DQN*.

Relevant tutorials: *DDPG*.

CHAPTER 9

rlpack.environment package

CHAPTER 10

Indices and tables

- genindex
- modindex
- search