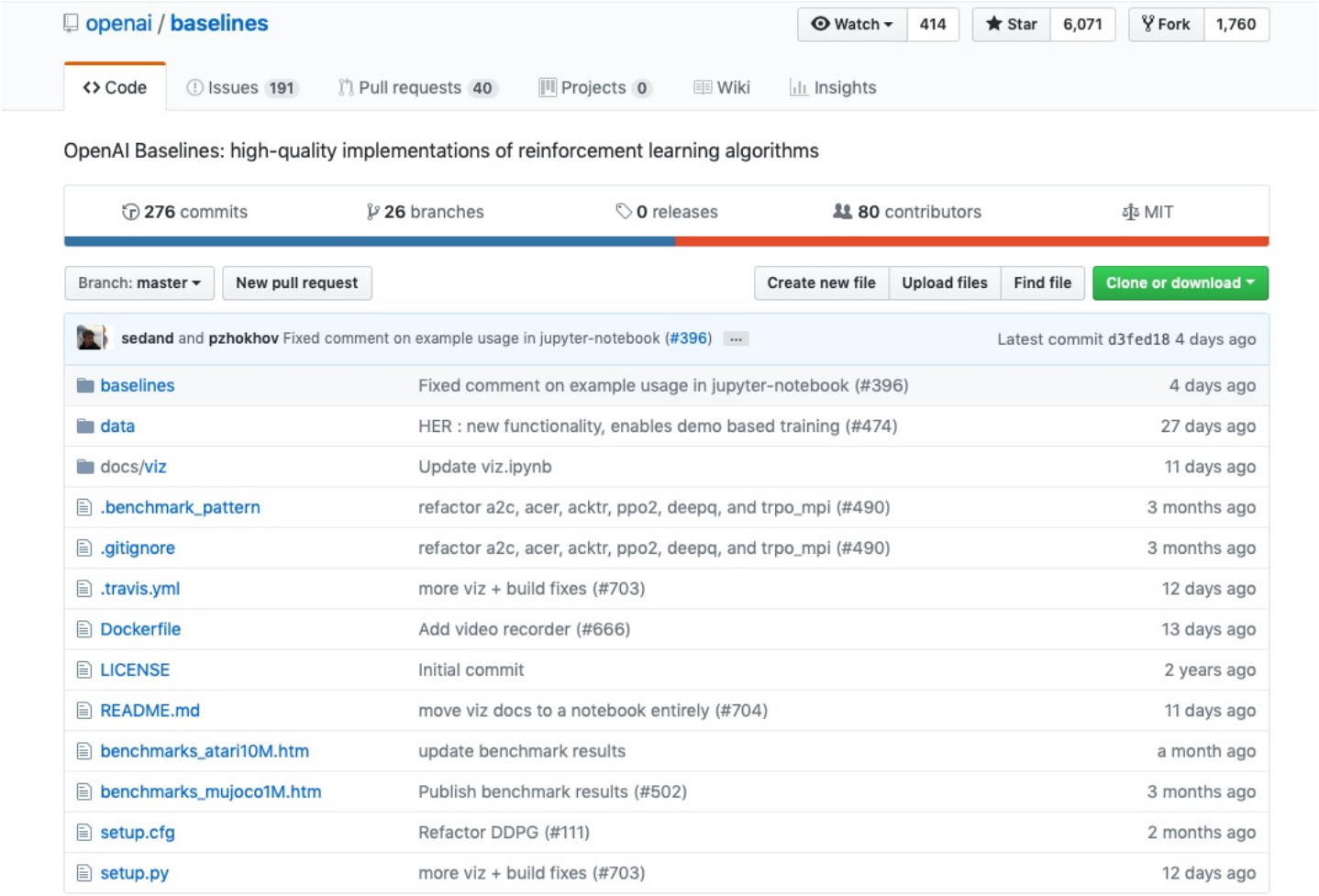


openai / baselines

Watch 414 Star 6,071 Fork 1,760

Code Issues 191 Pull requests 40 Projects 0 Wiki Insights

OpenAI Baselines: high-quality implementations of reinforcement learning algorithms

276 commits 26 branches 0 releases 80 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

sedand and pzhokhov Fixed comment on example usage in jupyter-notebook (#396) ... Latest commit d3fed18 4 days ago

baselines	Fixed comment on example usage in jupyter-notebook (#396)	4 days ago
data	HER : new functionality, enables demo based training (#474)	27 days ago
docs/viz	Update viz.ipynb	11 days ago
.benchmark_pattern	refactor a2c, acer, acktr, ppo2, deepq, and trpo_mpi (#490)	3 months ago
.gitignore	refactor a2c, acer, acktr, ppo2, deepq, and trpo_mpi (#490)	3 months ago
.travis.yml	more viz + build fixes (#703)	12 days ago
Dockerfile	Add video recorder (#666)	13 days ago
LICENSE	Initial commit	2 years ago
README.md	move viz docs to a notebook entirely (#704)	11 days ago
benchmarks_atari10M.htm	update benchmark results	a month ago
benchmarks_mujoco1M.htm	Publish benchmark results (#502)	3 months ago
setup.cfg	Refactor DDPG (#111)	2 months ago
setup.py	more viz + build fixes (#703)	12 days ago

【强化学习实践 30】复现PPO

张楚珩
清华大学 交叉信息院博士在读

序言

讲了好多强化学习的paper了，下面是时候仔细考虑一下Implementation matters这句话了。这些常见的算法都有已经调好参数的程序，大家可以直接跑，就在[github]openai.baselines。如果只是为了跑一个对照试验的话，直接使用别人实现的代码就可以了。但是我们对于更多的东西感兴趣，比如，究竟要如何实现一个高性能的算法？有哪些（文章里面不会提到的）小技巧导致我们实现的算法性能低甚至崩溃？

各种实现小技巧的调研

首先要知己知彼才可以，我们先通读baselines代码，看看究竟这里面包含了哪些文章里面没有提到的内容。另外也参考了文献[1]里面提到的一些技巧。（注，现在baselines里面的ppo1是马上要弃用了的，但是我当时不知道，主要调研的ppo1，现在更新了多进程版本的ppo2；不过最后实践表明我实现的算法与ppo2跑出来的效果差不多）

1. 超参数设置

```
# 最多采样多少步，可以用户设定
max_timesteps=2e7
# 每个batch采样多少步，里面可能包含很多回合的终点，然后起点开始继续采样，直到凑够
timesteps_per_actorbatch=2048
# PPO的clip参数
clip_param=0.2
# 交叉熵项系数
entcoeff=0.0
# 每次采集一个batch的样本之后过几遍
optim_epochs=10
# Adam学习率
optim_stepsize=3e-4
# 优化用的minibatch size
optim_batchsize=64
# discount rate
gamma=0.99
# GAE parameter
lam=0.95
adam_epsilon=1e-5
schedule='linear'
```

2. observation normalization and clipping

```
tf.clip_by_value((ob - self.ob_rms.mean) / self.ob_rms.std, -5.0, 5.0)
```

即归一化到(0,1)高斯分布，然后再clip掉偏离比较远的。

3. value function clipping

在参考资料[1]和[2]里面都提到了要类似策略的损失函数一样对价值函数的损失函数做类似的clip。但是我在openai.baselines.ppo1里面并没有找到相应的实现。况且参考资料[1]里面也说了这个技巧没啥效果。

4. reward scaling / clipping

同样，在文献[1]里面提到了这个技巧，不过我没有在openai.baselines.ppo1里找到。

5. orthogonal initialization and layer scaling

一般的神经网络实现（TensorFlow、PyTorch等）一般默认xavier初始化[3]，但是在强化学习里面一般都在使用正交初始化[4]。

在openai.baselines.ppo1里面是独立的正态分布生成权重，然后每一行做归一化

```
def normc_initializer(std=1.0, axis=0):
    def _initializer(shape, dtype=None, partition_info=None):  # pylint: disable=W061
        out = np.random.randn(*shape).astype(dtype.as_numpy_dtype)
        out *= std / np.sqrt(np.square(out).sum(axis=axis, keepdims=True))
        return tf.constant(out)
    return _initializer
```

在openai.baselines.ppo2里面是利用SVD分解来做的

```
w = tf.get_variable("w", [nin, nh], initializer=ortho_init(init_scale))
b = tf.get_variable("b", [nh], initializer=tf.constant_initializer(init_bias))


def ortho_init(scale=1.0):
    def _ortho_init(shape, dtype, partition_info=None):
        #Lasagne ortho init for tf
        shape = tuple(shape)
        if len(shape) == 2:
            flat_shape = shape
        elif len(shape) == 4: # assumes NHWC
            flat_shape = (np.prod(shape[:-1]), shape[-1])
        else:
            raise NotImplementedError
        a = np.random.normal(0.0, 1.0, flat_shape)
        u, _, v = np.linalg.svd(a, full_matrices=False)
        q = u if u.shape == flat_shape else v # pick the one with the correct shape
        q = q.reshape(shape)
        return (scale * q[:shape[0], :shape[1]]).astype(np.float32)
    return _ortho_init
```

6. Adam learning rate annealing

在不同的learning epoch使用不断减小的学习率。

```
cur_lrmult =  max(1.0 - float(timesteps_so_far) / max_timesteps, 0)
adam.update(g, optim_stepsize * cur_lrmult)
```

7. PPO clip parameter annealing

和上面类似，对于PPO的参数也使用这样的clipping。

```
cur_lrmult =  max(1.0 - float(timesteps_so_far) / max_timesteps, 0)
clip_param = clip_param * lrmult
surr2 = tf.clip_by_value(ratio, 1.0 - clip_param, 1.0 + clip_param) * atarg
```

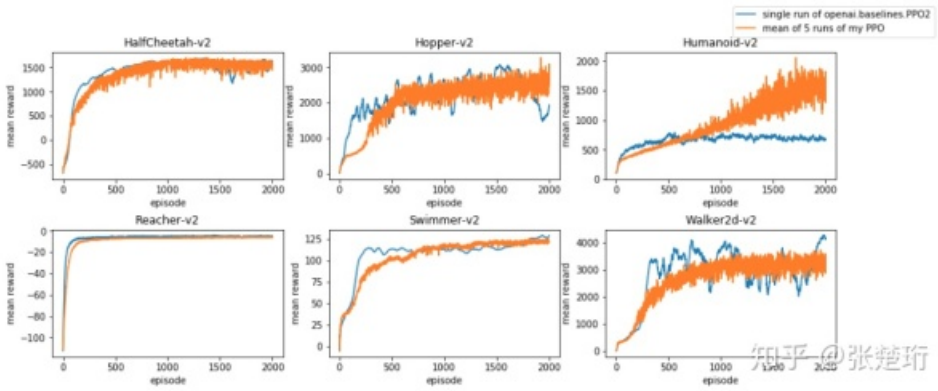
8. 神经网络结构

```
hid_size=64
num_hid_layers=2

value network:
    MLP1 tanh(obs_dim, hid_size) init=normc(1.0)
    MLP2 tanh(hid_size, hid_size) init=normc(1.0)
    MLP3 linear(hid_size, 1) init=norm(1.0)
policy network:
    MLP1 tanh(obs_dim, hid_size) init=normc(1.0)
    MLP2 tanh(hid_size, hid_size) init=normc(1.0)
    MLP3 tanh(hid_size, act_dim) init=normc(0.01)  --> mean
    Variable logstd init=zeros                     --> logstd
```

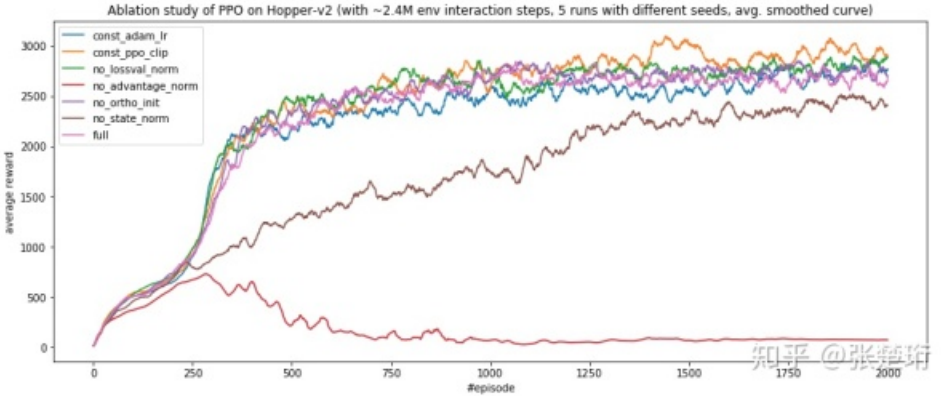
实现的结果

最后我实现的结果如下，其中蓝线是运行baselines算法得到的曲线（单次运行），橙线是运行我复现的PPO算法得到的曲线（用不同的随机数种子运行五次取平均）。



不同小技巧的效果对比（Ablation Study）

可以看到对于advantage和observation的归一化对于算法的性能影响是比较大的，其他方面影响倒不是很大，这一点和文献[1]里面的结果有一些区别。



代码

欢迎大家来围观我复现的代码，这里用不到400行代码达到了和baselines一样的性能。

https://github.com/zhangchuheng123/Reinforcement-Implementation/blob/master/code/ppo.py

github.com

参考资料

[1] Ilyas, Andrew, et al. "Are Deep Policy Gradient Algorithms Truly Policy Gradient Algorithms?."*arXiv preprint arXiv:1811.02553*(2018).

[2] github.com/Jiankai-Sun/...

[3] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks."*Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

[4] Saxe, Andrew M., James L. McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks."*arXiv preprint arXiv:1312.6120*(2013).

不同Mujoco环境里面的奖励函数的计算方法

做实验的时候也不能把环境和任务当做一个黑盒子搞呀，还是要注意看看现在训练的究竟是个什么任务。而奖励的计算则尤为重要，因此，我还专门去统计了一个各个环境的奖励函数是如何算出来的。

Can be found at https://github.com/openai/gym/tree/master/gym/envs/mujoco

environment(v2)	main reward function	control penalty[1]	survive reward	contact penalty[2]	notice
Ant	`xposafter - xposbefore`	yes	yes	yes	
HalfCheetah	`xposafter - xposbefore`	yes	yes	no	
Hopper	`xposafter - xposbefore`	yes	yes	no	
Humanoid	`xposafter - xposbefore` [3]	yes	yes	yes	
HumanoidStandup	`zposafter`	yes	no[4]	yes	
InvertedDoublePendulum	`- square(dist) - sqaure(velocity)`	no	yes	no	
InvertedPendulum	none (only const survive reward)	no	yes	no	
Pusher	`- dist(obj, goal) - dist(arm, obj)`	yes	no	no	
Reacher	`- dist(obj, goal)`	yes	no	no	
Striker	`- dist(obj, goal) - dist(arm, obj)`	yes	no	no	no termination
Swimmer	`xposafter - xposbefore`	yes	no	no	
Thrower	`- dist(obj, goal)`	yes	no	no	no termination
Walker2d	`xposafter - xposbefore`	yes	yes	no	

[1] `- square(control)`

[2] `- clip(square(cfrc_ext).sum())`

[3] mass x location

[4] survive reward is actually included in the non-conservative main reward function

	env	state_space	state_space_range	action_space	action_space_range	reward_range
0	Ant-v2	Box(111,)	(-inf, inf)	Box(8,)	(-1.00, 1.00)	(-inf, inf)
1	HalfCheetah-v2	Box(17,)	(-inf, inf)	Box(6,)	(-1.00, 1.00)	(-inf, inf)
2	Humanoid-v2	Box(376,)	(-inf, inf)	Box(17,)	(-0.40, 0.40)	(-inf, inf)
3	Walker2d-v2	Box(17,)	(-inf, inf)	Box(6,)	(-1.00, 1.00)	(-inf, inf)
4	Swimmer-v2	Box(8,)	(-inf, inf)	Box(2,)	(-1.00, 1.00)	(-inf, inf)
5	Reacher-v2	Box(11,)	(-inf, inf)	Box(2,)	(-1.00, 1.00)	(-inf, inf)
6	InvertedPendulum-v2	Box(4,)	(-inf, inf)	Box(1,)	(-3.00, 3.00)	(-inf, inf)
7	InvertedDoublePendulum-v2	Box(11,)	(-inf, inf)	Box(1,)	(-1.00, 1.00)	(-inf, inf)
8	HumanoidStandup-v2	Box(376,)	(-inf, inf)	Box(17,)	(-0.40, 0.40)	(-inf, inf)
9	Hopper-v2	Box(11,)	(-inf, inf)	Box(3,)	(-1.00, 1.00)	(-inf, inf)
10	Pusher-v2	Box(23,)	(-inf, inf)	Box(7,)	(-2.00, 2.00)	(-inf, inf)
11	Striker-v2	Box(23,)	(-inf, inf)	Box(7,)	(-3.00, 3.00)	(-inf, inf)
12	Thrower-v2	Box(23,)	(-inf, inf)	Box(7,)	(-15.00, 15.00)	(-inf, inf)

编辑于 2018-12-08

算法 强化学习 (Reinforcement Learning)

文章被以下专栏收录