Reinforcement Learning

basic introduction

丁钊翰

Content

MDP
Q Learning
DQN
Policy Gradient
Actor-Critic
SAC
TRPO
PPO
DDPG

README.md

Tianshou (天授) is a reinforcement learning platform based on pure PyTorch. Unlike existing reinforcement learning libraries, which are mainly based on TensorFlow, have many nested classes, unfriendly API, or slow-speed, Tianshou provides a fast-speed framework and pythonic API for building the deep reinforcement learning agent with the least number of lines of code. The supported interface algorithms currently include:

- Policy Gradient (PG)
- Deep Q-Network (DQN)
- Double DQN (DDQN) with n-step returns
- Prioritized DQN (PDQN)
- Advantage Actor-Critic (A2C)
- Deep Deterministic Policy Gradient (DDPG)
- Proximal Policy Optimization (PPO)
- Twin Delayed DDPG (TD3)
- Soft Actor-Critic (SAC)
- Vanilla Imitation Learning
- Generalized Advantage Estimation (GAE)

Tianshou supports parallel workers for all algorithms as well. All of these algorithms are reformatted as replay-buffer based algorithms. Our team is working on supporting more algorithms and more scenarios on Tianshou in this period of development.

In Chinese, Tianshou means divinely ordained and is derived to the gift of being born with. Tianshou is a reinforcement learning platform, and the RL algorithm does not learn from humans. So taking "Tianshou" means that there is no teacher to study with, but rather to learn by themselves through constant interaction with the environment.

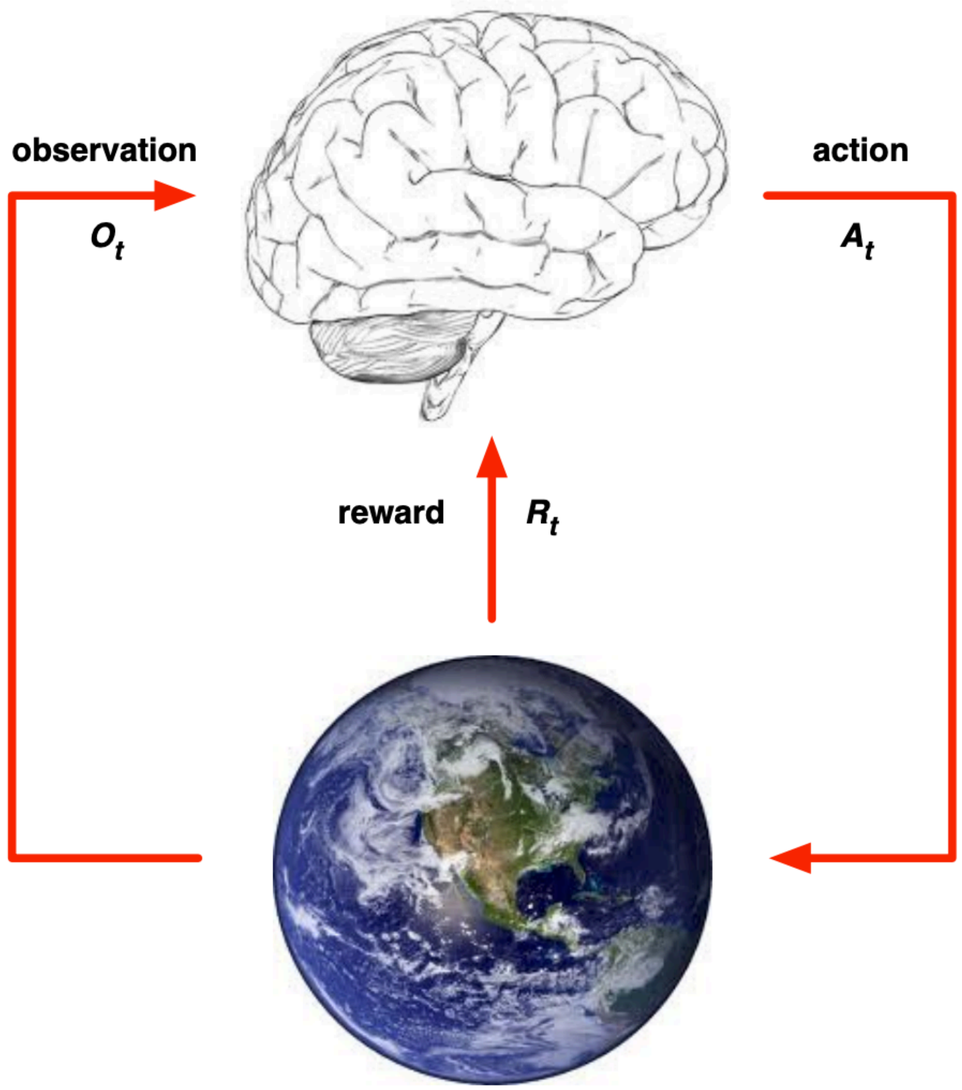
agent, env, reward, action, state, policy, model, value

state & observation

reward & return

exploration & exploitation

prediction & control



- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

Markov Decision Processes(MDP)

A State S_t is Markov if and only if

$$P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, \ldots, S_t)$$

A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all states are Markov.

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^{a}_{s} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.

Markov Decision Processes(MDP)

Reward

Return: the total discounted reward from time-step t

$$G_t = R_{t+1} + \lambda R_{t+2} + \ldots = \sum_{k=0}^{\infty} \lambda^k R_{t+k+1}$$

Value Function: $v(s) = E[G_t \mid S_t = s]$

Bellman Equaltion

$$G_t = R_{t+1} + \lambda R_{t+2} + \ldots = \sum_{k=0}^{\infty} \lambda^k R_{t+k+1}$$

$$\begin{aligned} v(s) &= E[G_t \,|\, S_t = s] \\ &= E[R_{t+1} + \lambda R_{t+2} + \lambda^2 R_{t+3} + \ldots \,|\, S_t = s] \\ &= E[R_{t+1} + \lambda (R_{t+2} + \lambda R_{t+3} + \ldots) \,|\, S_t = s] \\ &= E[R_{t+1} + \lambda G_{t+1} \,|\, S_t = s] \\ &= E[R_{t+1} + \lambda v(S_{t+1}) \,|\, S_t = s] \end{aligned}$$

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Q Learning

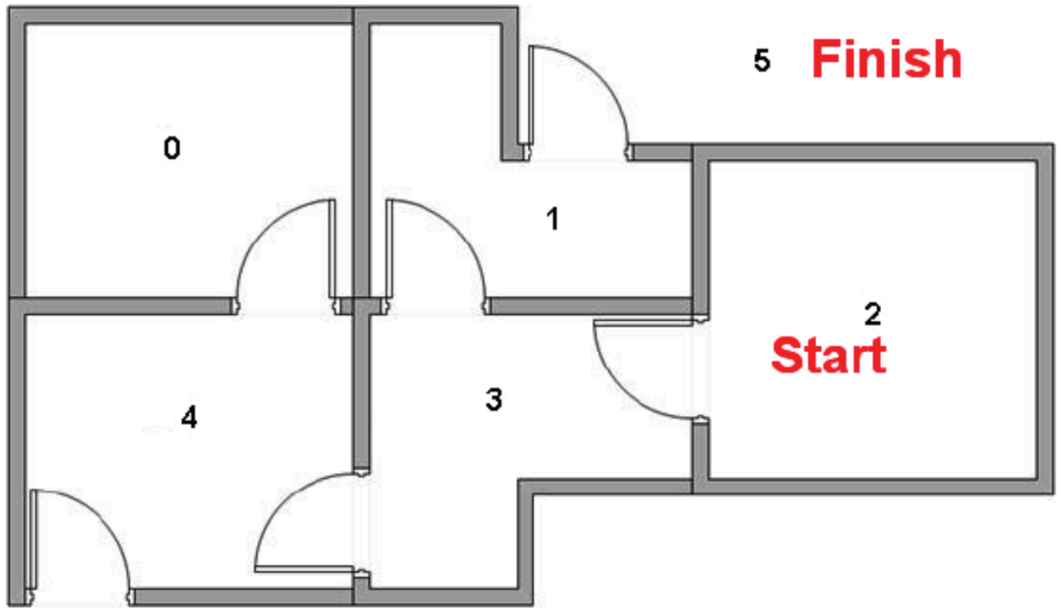


图 4

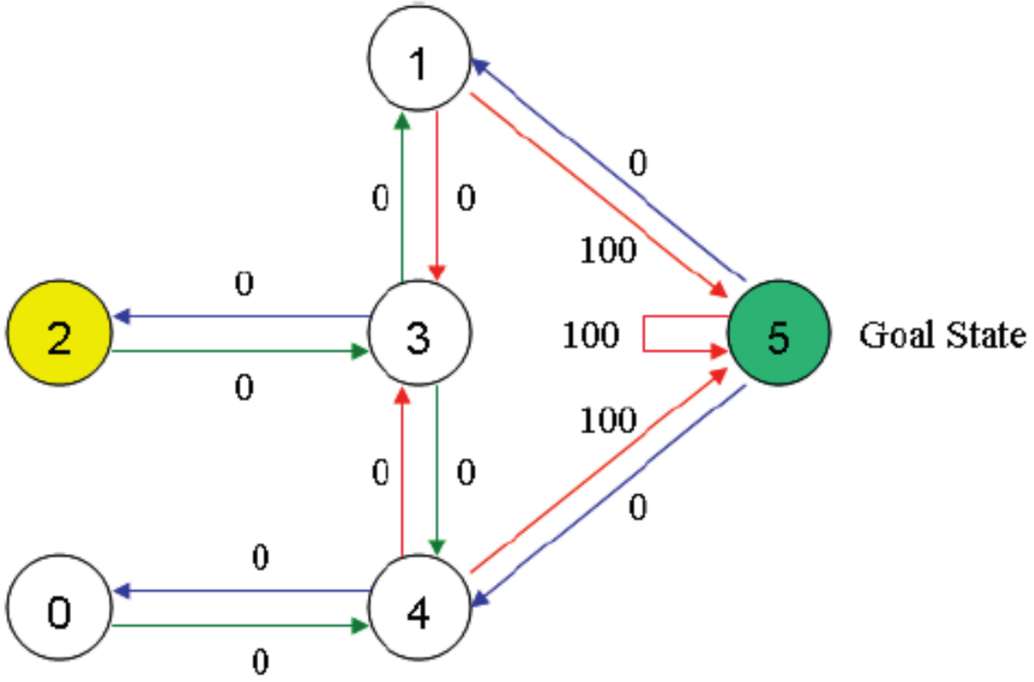


图 5

Q Learning

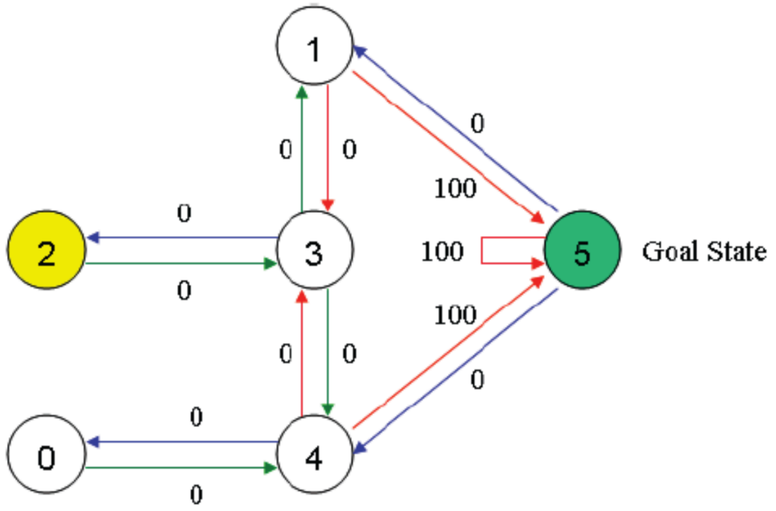


图 5

$$Q(s,a) = R(s,a) + \gamma \cdot \max_{\widetilde{a}}\{Q(\widetilde{s},\widetilde{a})\},$$

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

图 6 reward 值矩阵

The Q-Learning algorithm goes as follows:

1. Set the gamma parameter, and environment rewards in matrix R.
2. Initialize matrix Q to zero.
3. For each episode:
   (a) Select a random initial state.
   (b) Do While the goal state hasn't been reached.
      i. Select one among all possible actions for the current state.
      ii. Using this possible action, consider going to the next state.
      iii. Get maximum Q value for this next state based on all possible actions.
      iv. Compute: Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
      v. Set the next state as the current state.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

图 14 规范化后的矩阵 Q

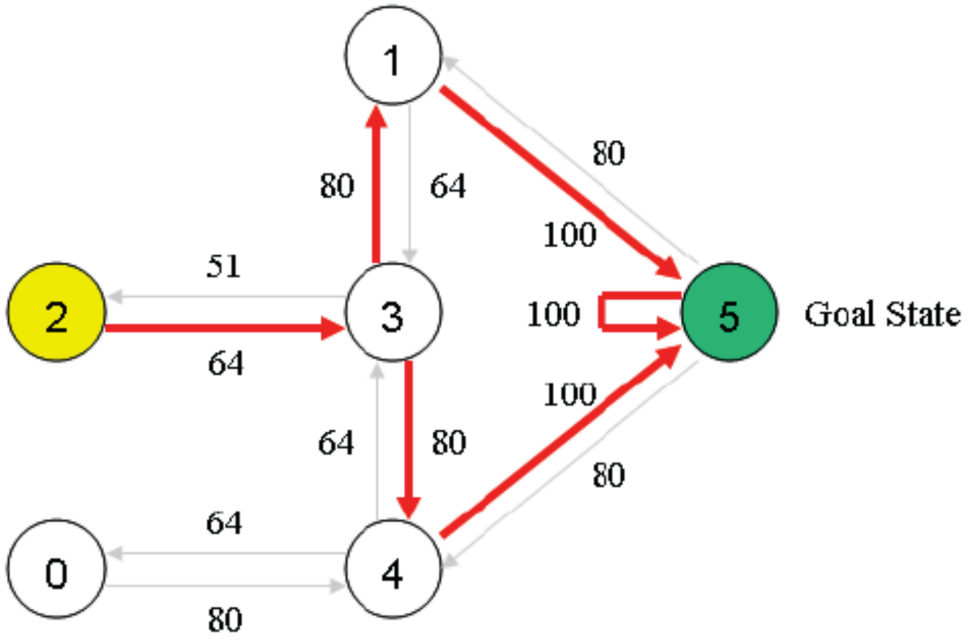


图 15

DQN

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[R_t | s_t = s, a_t = a\right]$$

$$L(w) = \mathbb{E}[(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{Target} - Q(s, a, w))^2]$$

https://www.tensorflow.org/agents/tutorials/0_intro_rl

Policy Gradient

策略搜索方法和值函数方法比较

- 直接接策略搜索方法是对策略π进行参数化表示，与值函数方中对值函数进行参数化表示相比，策略参数化更简单，有更好的收敛性。

- 利用值函数方法求解最优策略时，策略改进需要求解$argmax_a Q_\theta(s,a)$,当要解决的问题动作空间很大或者动作为连续集时，该式无法有效求解。

- 直接策略搜索方法经常采用的随机策略，能够学习随机策略。可以将探索直接集成到策略之中。

- 策略搜索的方法容易收敛到局部最小值。

- 评估单个策略时并不充分，方差较大。

- Goal: given policy $\pi_\theta(s, a)$ with parameters θ, find best θ
- But how do we measure the quality of a policy π_θ?
- In episodic environments we can use the start value

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- In continuing environments we can use the average value

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

- Or the average reward per time-step

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

- where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ

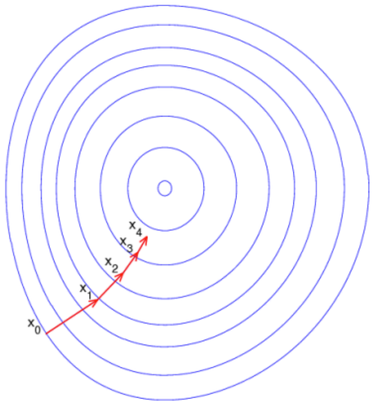
- Let $J(\theta)$ be any policy objective function
- Policy gradient algorithms search for a *local* maximum in $J(\theta)$ by ascending the gradient of the policy, w.r.t. parameters θ

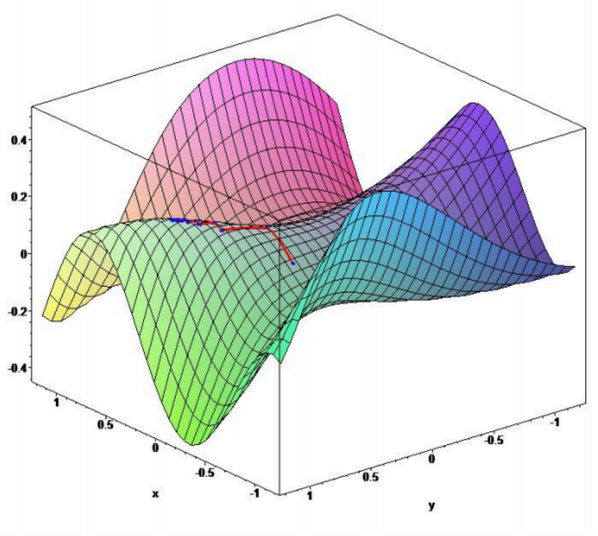
$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

- Where $\nabla_\theta J(\theta)$ is the policy gradient

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

- and α is a step-size parameter





- We now compute the policy gradient *analytically*
- Assume policy π_θ is differentiable whenever it is non-zero
- and we know the gradient $\nabla_\theta \pi_\theta(s, a)$
- Likelihood ratios exploit the following identity

$$\begin{aligned}\nabla_\theta \pi_\theta(s, a) &= \pi_\theta(s, a) \frac{\nabla_\theta \pi_\theta(s, a)}{\pi_\theta(s, a)} \\ &= \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a)\end{aligned}$$

- The score function is $\nabla_\theta \log \pi_\theta(s, a)$

Theorem

For any differentiable policy $\pi_\theta(s, a)$,
for any of the policy objective functions $J = J_1, J_{avR}$, *or* $\frac{1}{1-\gamma} J_{avV}$,
the policy gradient is

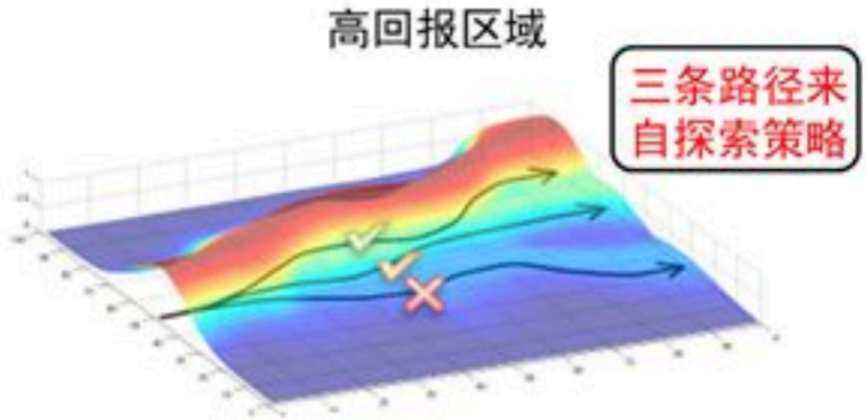
$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s, a) \; Q^{\pi_\theta}(s, a) \right]$$

$$\cdot R(\tau)=\sum_{t=0}^{H} R\left(s_t, u_t\right)$$

$$U(\theta)=E\left(\sum_{t=0}^{H} R\left(s_t, u_t\right) ; \pi_\theta\right)=\sum_\tau P(\tau ; \theta) R(\tau)$$

$$\begin{aligned}
\nabla_\theta U(\theta) & =\nabla_\theta \sum_\tau P(\tau ; \theta) R(\tau) \\
& =\sum_\tau \nabla_\theta P(\tau ; \theta) R(\tau) \\
& =\sum_\tau \frac{P(\tau ; \theta)}{P(\tau ; \theta)} \nabla_\theta P(\tau ; \theta) R(\tau) \\
& =\sum_\tau P(\tau ; \theta) \frac{\nabla_\theta P(\tau ; \theta) R(\tau)}{P(\tau ; \theta)} \\
& =\sum_\tau P(\tau ; \theta) \nabla_\theta \log P(\tau ; \theta) R(\tau)
\end{aligned}$$

$$\nabla_\theta U(\theta) \approx \hat{g}=\frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau ; \theta) R(\tau)$$



Policy Gradient

$$\nabla_\theta U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau;\theta)\, R(\tau)$$

$$\nabla_\theta \log P(\tau;\theta) \, ?$$

$$P\left(\tau^{(i)};\theta\right) = \prod_{t=0}^{H} P\left(s_{t+1}^{(i)} | s_t^{(i)}, u_t^{(i)}\right) \cdot \pi_\theta\left(u_t^{(i)} | s_t^{(i)}\right)$$

$$\nabla_\theta \log P\left(\tau^{(i)};\theta\right) = \nabla_\theta \log \left[\prod_{t=0}^{H} P\left(s_{t+1}^{(i)} | s_t^{(i)}, u_t^{(i)}\right) \cdot \pi_\theta\left(u_t^{(i)} | s_t^{(i)}\right)\right]$$

$$= \nabla_\theta \left[\sum_{t=0}^{H} \log P\left(s_{t+1}^{(i)} | s_t^{(i)}, u_t^{(i)}\right) + \sum_{t=0}^{H} \log \pi_\theta\left(u_t^{(i)} | s_t^{(i)}\right)\right]$$

$$= \nabla_\theta \left[\sum_{t=0}^{H} \log \pi_\theta\left(u_t^{(i)} | s_t^{(i)}\right)\right]$$

$$= \sum_{t=0}^{H} \nabla_\theta \log \pi_\theta\left(u_t^{(i)} | s_t^{(i)}\right)$$

Policy Gradient

$$\pi_\theta = \mu_\theta + \varepsilon$$

$$\mu(s) = \phi(s)^T \theta$$

$$\varepsilon \sim N\left(0, \sigma^2\right)$$

$$\pi(u|s) \sim \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(u - \phi(s)^T\theta\right)^2}{2\sigma^2}\right)$$

Sample s_t, u_t

$$\nabla_\theta \log \pi_\theta\left(u_t^{(i)} | s_t^{(i)}\right) = \frac{\left(u_t^{(i)} - \phi\left(s_t^{(i)}\right)^T \theta\right) \phi\left(s_t^{(i)}\right)}{\sigma^2}$$

Actor-Critic

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta\left(a_t \middle| s_t\right)\right] \quad (1.3)$$

Ψ_t 可以是下列任何一个：

1.$\Sigma_{t=0}^{\infty} r_t$ 轨迹的总回报，

2.$\Sigma_{t'=t}^{\infty} r_{t'}$ 动作后的回报

3.$\Sigma_{t'=t}^{\infty} r_{t'} - b\left(s_t\right)$ 加入基线的形式

4. $Q^\pi\left(s_t, a_t\right)$ 状态-行为值函数

5. $A^\pi\left(s_t, a_t\right)$ ， 优势函数

6.$r_t + V^\pi\left(s_{t+1}\right) - V^\pi\left(s_t\right)$: TD残差

Actor-Critic

当 Ψ_t 取TD残差，并且值函数 $V^{\pi}(s_t)$ 由参数为 w 的神经网络进行逼近时。AC算法的更新步骤为：

$$\delta \leftarrow G_t - \hat{v}(S_t, w)$$
$$w \leftarrow w + \beta \delta \nabla_w \hat{v}(S_t, w)$$

$$\theta \leftarrow \theta + \alpha \delta \nabla_\theta \log \pi(A_t | S_t, \theta)$$

Soft Actor-Critic

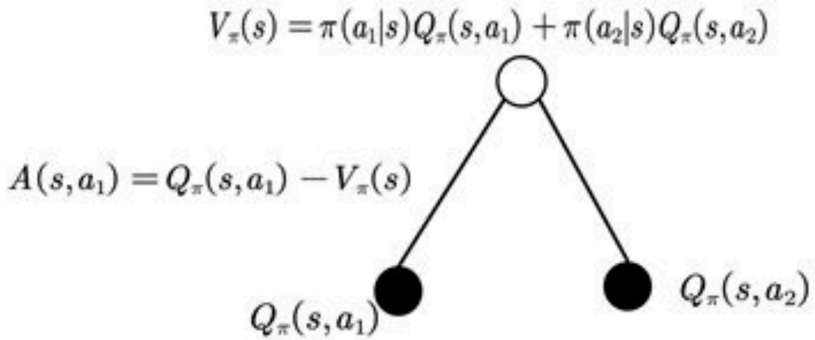
Algorithm 1 Soft Actor-Critic

Input: θ_1, θ_2, ϕ ▷ Initial parameters
$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target network weights
$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay pool
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$ ▷ Sample action from the policy
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ ▷ Sample transition from the environment
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ ▷ Store the transition in the replay pool
  end for
  for each gradient step **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ ▷ Update the Q-function parameters
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights
    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ▷ Adjust temperature
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target network weights
  end for
end for
Output: θ_1, θ_2, ϕ ▷ Optimized parameters

Trust region policy optimization(TRPO)

$$\eta(\tilde{\pi}) = E_{\tau|\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t\left(r\left(s_t\right)\right)\right]$$

$$\eta(\tilde{\pi}) = \eta(\pi) + E_{s_0,a_0,\cdots\tilde{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t A_\pi\left(s_t,a_t\right)\right]$$



$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s) = E_{s'\sim P(s'|s,a)}[r(s) + \gamma V^\pi(s') - V^\pi(s)]$$

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_{t=0}^{\infty}\sum_{s}P\left(s_t=s|\tilde{\pi}\right)\sum_{a}\tilde{\pi}\left(a|s\right)\gamma^t A_\pi\left(s,a\right)$$

其中 $P\left(s_t=s|\tilde{\pi}\right)\tilde{\pi}\left(a|s\right)$ 为 (s,a) 的联合概率，$\sum_{a}\tilde{\pi}\left(a|s\right)\gamma^t A_\pi\left(s,a\right)$ 为求对动作 a 的边际分布，也就是说在状态s对整个动作空间求和；$\sum_{s}P\left(s_t=s|\tilde{\pi}\right)$ 为求对状态 s 的边际分布，即对整个状态空间求和；$\sum_{t=0}^{\infty}\sum_{s}P\left(s_t=s|\tilde{\pi}\right)$ 求整个时间序列的和。

TRPO

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_{t=0}^{\infty} \sum_{s} P(s_t = s|\tilde{\pi}) \sum_{a} \tilde{\pi}(a|s) \gamma^t A_{\pi}(s,a)$$

$$\rho_{\pi}(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \cdots$$

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_{s} \rho_{\tilde{\pi}}(s) \sum_{a} \tilde{\pi}(a|s) A^{\pi}(s,a)$$

注意，这时状态s的分布由新的策略产生，对新的策略严重依赖。

Tricks

$$\eta\left(\tilde{\pi}\right)=\eta\left(\pi\right)+\sum_{s}\rho_{\tilde{\pi}}\left(s\right)\sum_{a}\tilde{\pi}\left(a|s\right)A^{\pi}\left(s,a\right)$$

$$L_{\pi}\left(\tilde{\pi}\right)=\eta\left(\pi\right)+\sum_{s}\rho_{\pi}\left(s\right)\sum_{a}\tilde{\pi}\left(a|s\right)A^{\pi}\left(s,a\right)$$

$$\sum_{a}\tilde{\pi}_{\theta}\left(a|s_{n}\right)A_{\theta_{old}}\left(s_{n},a\right)=E_{a\ q}\left[\frac{\tilde{\pi}_{\theta}\left(a|s_{n}\right)}{q\left(a|s_{n}\right)}A_{\theta_{old}}\left(s_{n},a\right)\right]$$

$$L_{\pi}\left(\tilde{\pi}\right)=\eta\left(\pi\right)+E_{s\ \rho_{\theta_{old}},a\ \pi_{\theta_{old}}}\left[\frac{\tilde{\pi}_{\theta}\left(a|s\right)}{\pi_{\theta_{old}}\left(a|s\right)}A_{\theta_{old}}\left(s,a\right)\right]$$

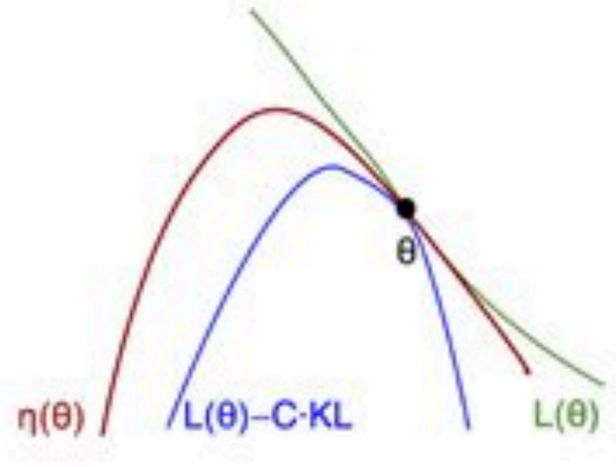
用 $\frac{1}{1-\gamma}E_{s\ \rho_{\theta_{old}}}\left[\cdots\right]$ 代替 $\sum_{s}\rho_{\theta_{old}}\left(s\right)\left[\cdots\right]$

$$q\left(a|s_{n}\right)=\pi_{\theta_{old}}\left(a|s_{n}\right)$$

重要性采样：https://zhuanlan.zhihu.com/p/41217212

TRPO

$$L_{\pi_{\theta_{old}}}\left(\pi_{\theta_{old}}\right) = \eta\left(\pi_{\theta_{old}}\right)$$
$$\nabla_\theta L_{\pi_{\theta_{old}}}\left(\pi_\theta\right)|_{\theta=\theta_{old}} = \nabla_\theta \eta\left(\pi_\theta\right)|_{\theta=\theta_{old}}$$



$$\eta\left(\tilde{\pi}\right) \geqslant L_\pi\left(\tilde{\pi}\right) - CD_{KL}^{\max}\left(\pi, \tilde{\pi}\right) \quad where\, C = \frac{2\varepsilon\gamma}{\left(1-\gamma\right)^2}$$

$$M_i\left(\pi\right) = L_{\pi_i}\left(\pi\right) - CD_{KL}^{\max}\left(\pi_i, \pi\right)$$

TRPO

$$\eta\left(\pi_{i+1}\right) \geqslant M_i\left(\pi_{i+1}\right) \ . \ \eta\left(\pi_i\right) = M_i\left(\pi_i\right)$$

$$\eta\left(\pi_{i+1}\right) - \eta\left(\pi_i\right) \geqslant M_i\left(\pi_{i+1}\right) - M\left(\pi_i\right) \ \geqslant 0$$

$$maximize_\theta \left[L_{\theta_{old}}\left(\theta\right) - CD_{KL}^{\max}\left(\theta_{old}, \theta\right)\right]$$

$$maximize_\theta E_{s\ \rho_{\theta_{old}}, a\ \pi_{\theta_{old}}}\left[\frac{\pi_\theta\left(a|s\right)}{\pi_{\theta_{old}}\left(a|s\right)} A_{\theta_{old}}\left(s,a\right)\right] \ \ subject\ to\ \ D_{KL}^{\max}\left(\theta_{old}, \theta\right) \leq \delta$$

Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO

Logan Engstrom*, Andrew Ilyas*, Shibani Santurkar[1], Dimitris Tsipras[1],
Firdaus Janoos[2], Larry Rudolph[1,2], and Aleksander Mądry[1]

[1]MIT [2]Two Sigma
{engstrom,ailyas,shibani,tsipras,madry}@mit.edu
rudolph@csail.mit.edu, firdaus.janoos@twosigma.com

$$maximize_\theta E_{s\ \pi_{\theta_{old}}, a\ \pi_{\theta_{old}}}\left[\frac{\pi_\theta\left(a|s\right)}{\pi_{\theta_{old}}\left(a|s\right)} A\right. \qquad \left.\right] \leq \delta$$

Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO

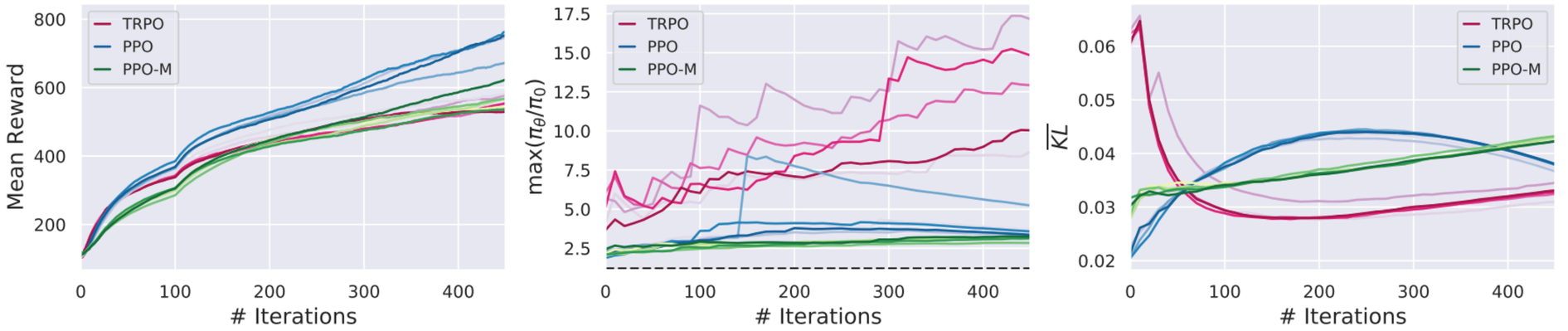
Logan Engstrom[*], Andrew Ilyas[*], Shibani Santurkar[1], Dimitris Tsipras[1],
Firdaus Janoos[2], Larry Rudolph[1,2], and Aleksander Mądry[1]

[1]MIT [2]Two Sigma
{engstrom,ailyas,shibani,tsipras,madry}@mit.edu
rudolph@csail.mit.edu, firdaus.janoos@twosigma.com

1. **Value function clipping:** Schulman et al. (2017) originally suggest fitting the value network via regression to target values:
$$L^V = (V_{\theta_t} - V_{targ})^2,$$
but the standard implementation instead fits the value network with a PPO-like objective:
$$L^V = \min\left[(V_{\theta_t} - V_{targ})^2, \left(\text{clip}\left(V_{\theta_t}, V_{\theta_{t-1}} - \varepsilon, V_{\theta_{t-1}} + \varepsilon\right) - V_{targ}\right)^2\right],$$
where V_θ is clipped around the previous value estimates (and ε is fixed to the same value as the value used in (2) to clip the probability ratios).
2. **Reward scaling:** Rather than feeding the rewards directly from the environment into the objective, the PPO implementation performs a certain discount-based scaling scheme. In this scheme, the rewards are divided through by the standard deviation of a rolling discounted sum of the rewards (without subtracting and re-adding the mean)—see Algorithm 1 in Appendix A.2.
3. **Orthogonal initialization and layer scaling:** Instead of using the default weight initialization scheme for the policy and value networks, the implementation uses an orthogonal initialization scheme with scaling that varies from layer to layer.
4. **Adam learning rate annealing:** Depending on the task, the implementation sometimes anneals the learning rate of Adam (Kingma & Ba, 2014) (an already adaptive method) for optimization.
5. **Reward Clipping**: The implementation also clips the rewards within a preset range (usually $[-5, 5]$ or $[-10, 10]$).
6. **Observation Normalization**: In a similar manner to the rewards, the raw states are also not fed into the optimizer. Instead, the states are first normalized to mean-zero, variance-one vectors.
7. **Observation Clipping**: Analagously to rewards, the observations are also clipped within a range, usually $[-10, 10]$.
8. **Hyperbolic tan activations**: As also observed by Henderson et al. (2017), implementations of policy gradient algorithms also also use hyperbolic tangent function activations between layers in the policy and value networks.
9. **Global Gradient Clipping**: After computing the gradient with respect to the policy and the value networks, the implementation clips the gradients such the "global ℓ_2 norm" (i.e. the norm of the concatenated gradients of all parameters) does not exceed 0.5.

Table 1: List of algorithms studied in this work, with their crucial properties. Step method refers to the method used to build each training step, PPO clipping refers to the use of clipping in the step (as in Equation (2)), and PPO optimizations refer to the optimizations listed in Section 3.

Algorithm	**Section**	**Step method**	**Uses PPO clipping?**	**Uses PPO optimizations?**
PPO	—	PPO	✓	As in (Dhariwal et al., 2017)
PPO-M	Sec. 3	PPO	✓	✗
PPO-NoClip	Sec. 4	PPO	✗	Found via grid search
TRPO	—	TRPO	—	✗
TRPO+	Sec. 5	TRPO	—	Found via grid search



Reference

- Implementation Matters in Deep RL: A Case Study on PPO and TRPO
- Soft Actor-Critic Algorithms and Applications
- Posterior Sampling for Multi-agent Reinforcement Learning: Solving Extensive Games with Imperfect Information