Policy Gradient 策略梯度

- Stochastic Policy Gradient (SPG)
- Deterministic Policy Gradient (DPG)

SPG formula:

$$\nabla_\theta J(\pi_\theta) = \int_S \rho^\pi(s) \int_A \nabla_\theta \pi_\theta(a|s) \, Q_\pi(s,a) \, da \, ds$$

$\nabla_\theta J(\pi_\theta)$: policy gradient

$\int_S \rho^\pi(s)$: 对状态 s 积分

$\int_A \nabla_\theta \pi_\theta(a|s) \, Q_\pi(s,a) \, da$: like Bellman Equation

$$= E_{s\sim\rho^\pi,\, a\sim\pi_\theta}\left[\nabla_\theta \log(\pi_\theta(a|s)) \cdot Q^\pi(s,a)\right]$$

SPG 相比 DPG 多了一个 log().

本质源于 SPG 需要加一层对策略 π 的期望，而 DPG 中 π 确定，为 u，因此不用.

期望变换后得 log()

DPG formula:

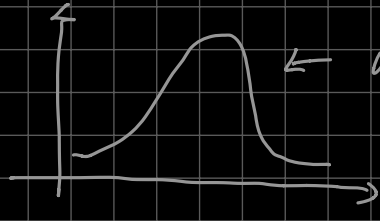
$$\nabla_\theta J(\mu_\theta) = \int_S \rho^u(s) \cdot \left(\nabla_\theta u_\theta(s)\right) \cdot \left(\nabla_a Q_u(s,a)\big|_{a=u_\theta(s)}\right) ds$$

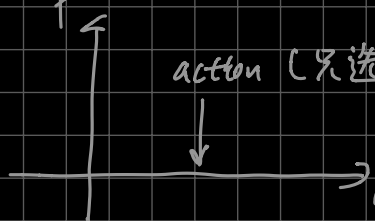
在确定型策略中，$a = u_\theta(s)$，π 的确定形式

因此没有了内层的积分

$$= E_{s\sim\rho^u}\left[\left(\nabla_\theta u_\theta(s)\right) \cdot \left(\nabla_a Q^u(s,a)\big|_{a=u_\theta(s)}\right)\right]$$

SPG:





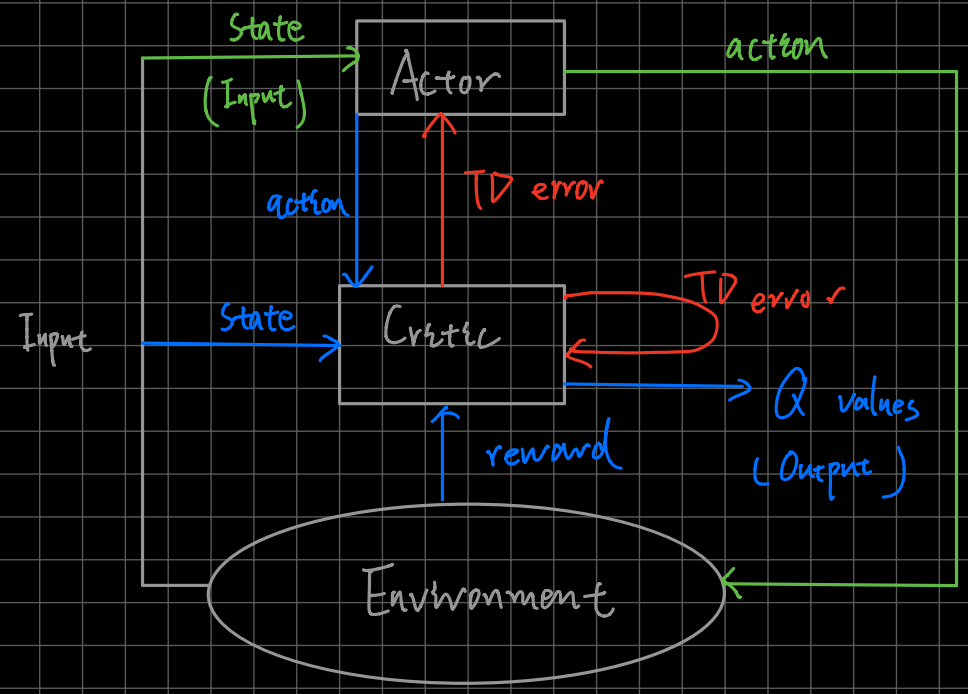
AC(Actor - Critic) Algorithm

Actor: Policy Q-network, used to choose action

Critic: Value Q-network, used to estimate action is good or not. And generates TD-error to guide updating.

Policy Q-network && Value Q-network

Actor, Critic 均使用 GD 来 update，而 Actor 产生的 TD-error 相当于表明此次的 ascent 是否正确（若正确，就多 ascent，错误就少 ascent）



Why propose AC algorithm? ⇒ (Google DeepMind)

- DQN, Q-learning, SARSA (SARSA-λ) 等 Value based method 无法处理连续动作空间的情形。
- 传统的 Policy Gradient 为回合更新制，model 学习效率低下，学习速度缓慢。

→ Combine: AC algorithm

优：AC算法结合了 Value-based 与 Policy based 算法的优势，可处理连续、离散问题，又可单步更新，提升学习效率。

缺：难收敛 ⇒ 收敛性取决于 critic 的价值判断，而 critic 依赖的 actor 不断更新，导致更难收敛。

How to solve? ⇒ Deep Deterministic Policy Gradient

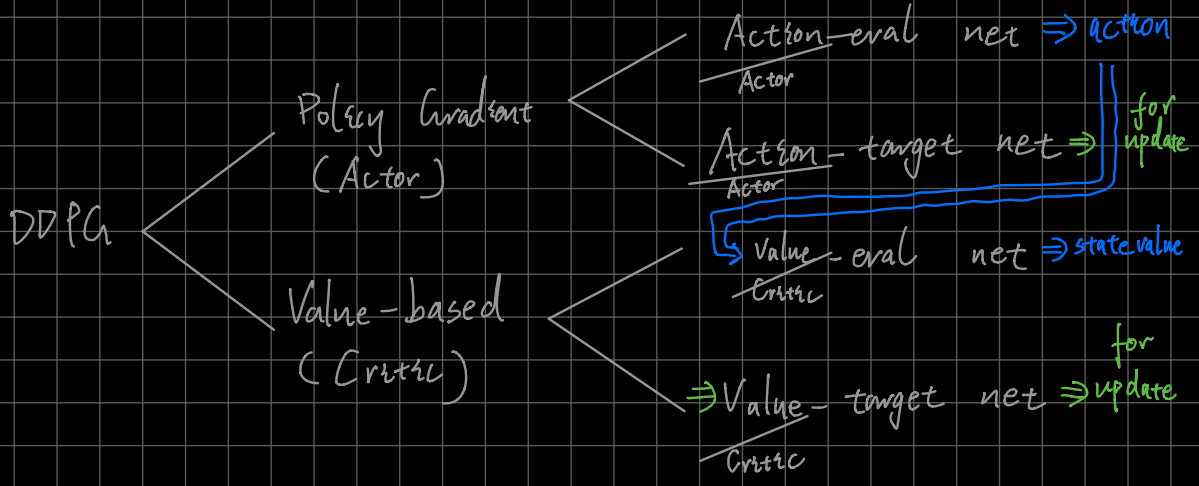
采用 DQN 相似策略，用 memory 储存 transitions，并通过 采样 来降低数据间相关性，提高 收敛性、稳定性

△ 参数更新的相关性越低，算法收敛性越高

⇓

DDPG, A3C 均是为降低 AC 算法参数间相关性做的改进（采样、并行更新）

Famous DPG Algorithm: DDPG (Deep DPG)



DDPG 即 DQN + AC，前半部分（①）很像 DQN，后半部分（②）像 AC。

DDPG 实际上算 Value-based：因为 policy 更新的目标实际上是找最优的 Q。

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
① Initialize a random process $\mathcal{N}$ for action exploration
Receive initial observation state s_1
for t = 1, T **do**
Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise (noise sample)
⇒ 解决连续空间 action 探索能力差的问题
Execute action a_t and observe reward r_t and observe new state s_{t+1}
Store transition (s_t, a_t, r_t, s_{t+1}) in R
Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R (experience replay)
Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2)$
Update the actor policy using the sampled gradient:

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

⇒ 往最大 Q 值的 action 方向更新 policy
⇒ Deterministic policy，因此只有对 state 积分，action 随便

此时 update 为 Q-network，不是 target
Critic target Network
Actor target Network
Critic-eval network
update Critic Network
action eval network
update Actor Network

② Update the target networks:
soft-update 软更新

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

update Actor & Critic Target Network
TAQ factor

end for
end for

A3C (Asynchronous Advantage Actor-Critic) Algorithm

Brief Introduction: 将AC放于多个thread中同步训练.

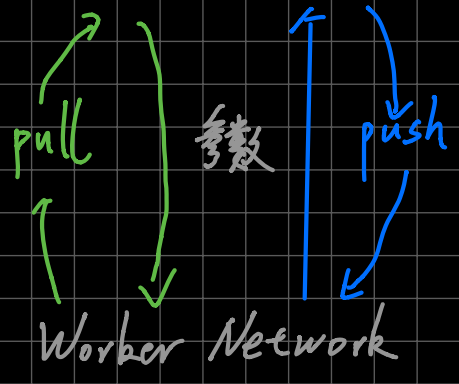
Advantages: ① 提升训练速度

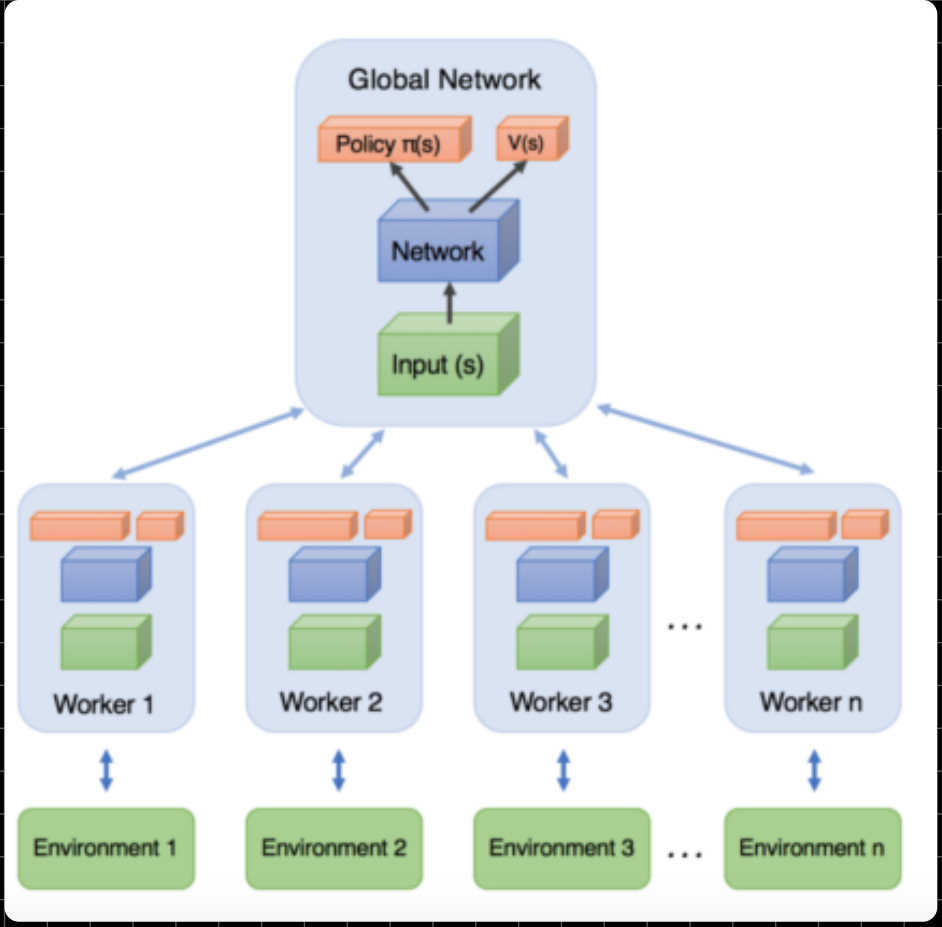
② 减弱训练相关性, 加速收敛

Features: ① Global Network and Worker Network have same structure.

② Global Network stores parameters, while Worker Network updates parameters through training.

③ Global Network




Global Network
Policy π(s)
V(s)
Network
Input (s)
Worker 1
Worker 2
Worker 3
...
Worker n
Environment 1
Environment 2
Environment 3
...
Environment n

Comparison between Policy Gradient methods:

$$\text{Gradient} = \frac{1}{N_s}\sum_{s}\left[\nabla_\theta \log\pi(s;\theta)\cdot V(s,a)\right]$$

Policy Gradient
- ① Get $\log\pi(s;\theta)$ <主>
- ② Get $V(s,a)/Q(s,a)$ <辅>

① Get $\log\pi(s;\theta)$

(i) Policy Gradient of discrete problem (CartPole)

tf.nn.sparse_softmax_cross_entropy_with_logits(logits = act_outs, label = self.tf_as

logits = act_outs ⇐ From: PG network output

label = self.tf_as ⇐ choose_action()

act_probs = tf.nn.softmax(act_outs)

action = np.random.choice(range(act_probs.shape[1]), p=act_probs.ravel())

通过 action 的概率分布选择。

(ii) AC of discrete problem (CartPole)

tf.log(self.act_probs[0, self.a])

self.act_probs ⇐ From: Actor output

self.a ⇐ choose_action()

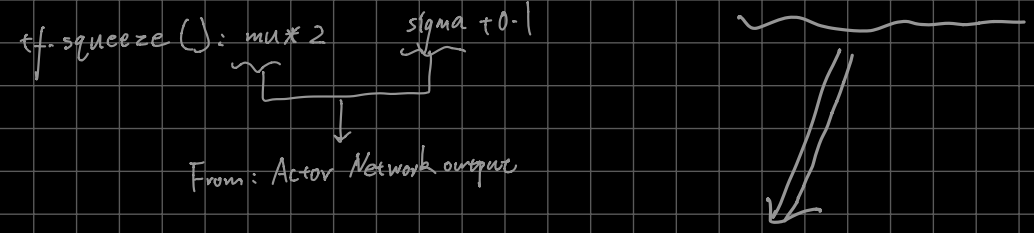
choose_action() ⇐ 与 (i) 中相同

(iii) AC of continuous problem (Pendulum)

self.normal_dist.log_prob(self.a)

self.normal_dist ⇐ tf.distributions.Normal(mu, sigma)

self.a ⇐ choose_action()



tf.clip_by_value(self.normal_dist.sample(1), (low, high))

从概率分布中采样1个action　　self.action_bound

(ii) $\nabla_{\theta^u} \underbrace{Q(s;\theta_Q)}_{s_i} = 1$

choose_action(), 通过 deterministic method 选取 ⇒

self.sess.run(self.a, feed_dict)[0]

可作为 action value 的表征

② Get $V(s,a)$ / td_error / $Q(s,a)$

(i) Policy Gradient of discrete problem (CartPole)

discount_and_norm (self.ep_rs)

experience reward in one turn (回合制新)

(ii) AC of discrete problem (CartPole)

td_error (From Critic)

$R + \gamma \cdot V_- - V$

From: env reward

Critic network output ⇒

V_- : S_

V : S

input

(iii) AC of continuous problem (Pendulum)

td_error (From Critic)

tf.reduce_mean ($R + \gamma \cdot V_- - V$)

(相比 discrete problem 不同之处)

(iv) DDPG (Pendulum)

tf. reduce_mean (Q) ⇒ (from: Critic/eval())

continuous problem 要加 $\frac{1}{N}$

原先为: $\frac{1}{N}\sum_i \nabla_a Q \cdot \nabla_\theta \mu$ ($\nabla_a Q$ ⇒ Q)

△ DDPG的 Critic 输出为 Q-value，而 AC的Critic 输出为 V-value

$Q(s,a;\theta)$ $V(s;\theta)$

更新的是 Critic network output it算 loss 进行更新.

$$\frac{1}{N}\cdot\sum_{i=1}^{s} \text{td-error}^2$$