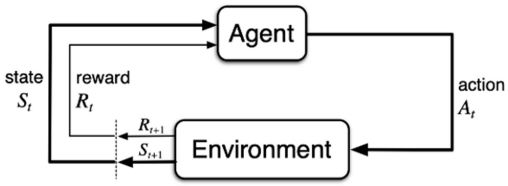
Policy

2021年2月13日 19:26

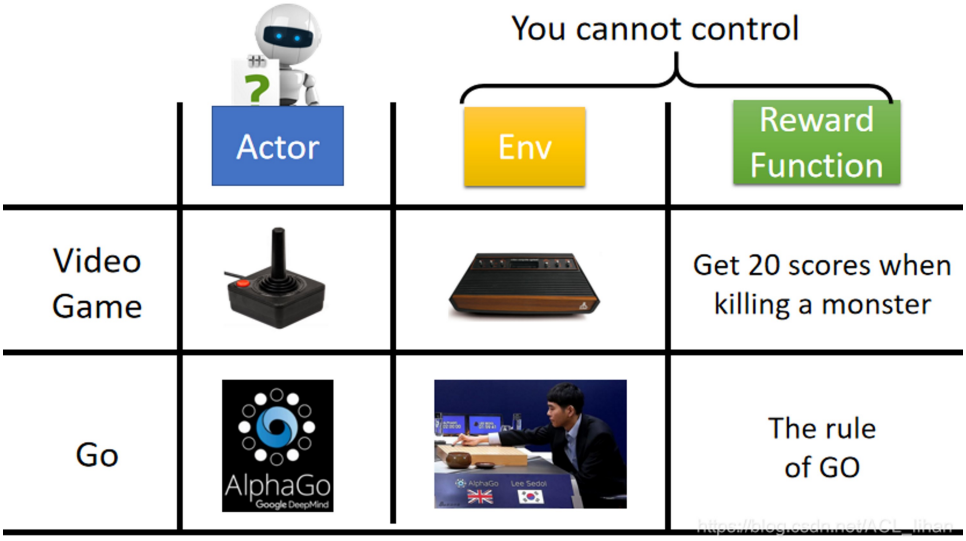
- Reinforcement Learning



a computational approach to learning whereby an agent tries to maximize the total amount of reward it receives while interacting with a complex and uncertain environment.

- Sutton and Barto

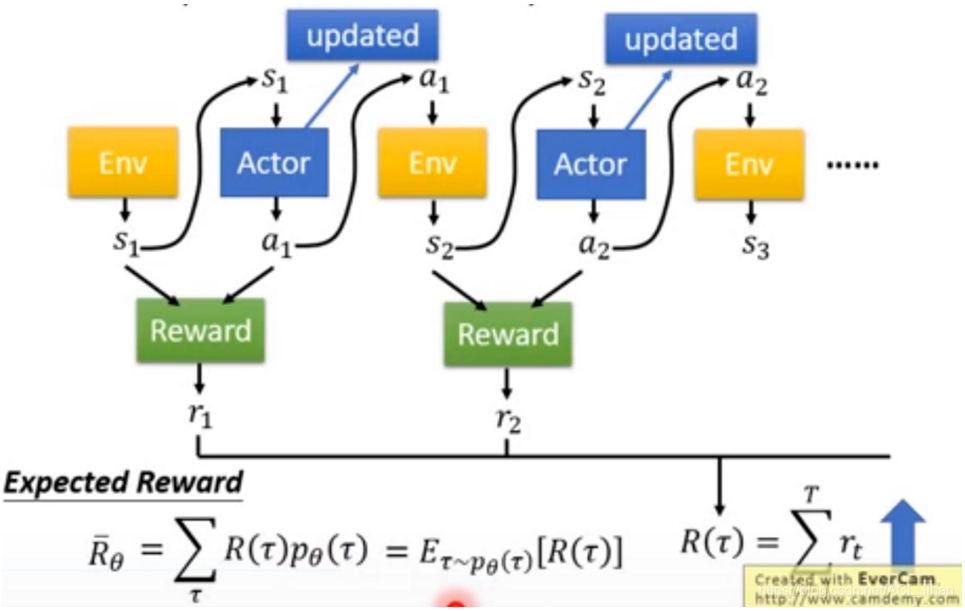
- Three components: Actor, environment, reward function.
  The Env and Reward function are predetermined and you cannot control them. The only thing that can be adjusted is the actor's policy so that the actor can get the maximum reward.



- Policy (π) is a network with parameter θ
  - Input: the observation of machine represented as a vector or a matrix
  - Output: each action corresponds to a neuron in output laye

$$p_\theta(\tau)$$
$$= p(s_1)p_\theta(a_1|s_1)p(s_2|s_1,a_1)p_\theta(a_2|s_2)p(s_3|s_2,a_2)\cdots$$
$$= p(s_1)\prod_{t=1}^{T} p_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)$$

- The process in one episode



In order to maximize the expected reward, we can take the policy gradient

Policy Gradient $\bar{R}_\theta = \sum_\tau R(\tau)p_\theta(\tau)$ $\nabla\bar{R}_\theta = ?$

$$\nabla\bar{R}_\theta = \sum_\tau R(\tau)\nabla p_\theta(\tau) = \sum_\tau R(\tau)p_\theta(\tau)\frac{\nabla p_\theta(\tau)}{p_\theta(\tau)}$$

$R(\tau)$ do not have to be differentiable

It can even be a black box.

$$= \sum_\tau R(\tau)p_\theta(\tau)\nabla log p_\theta(\tau)$$

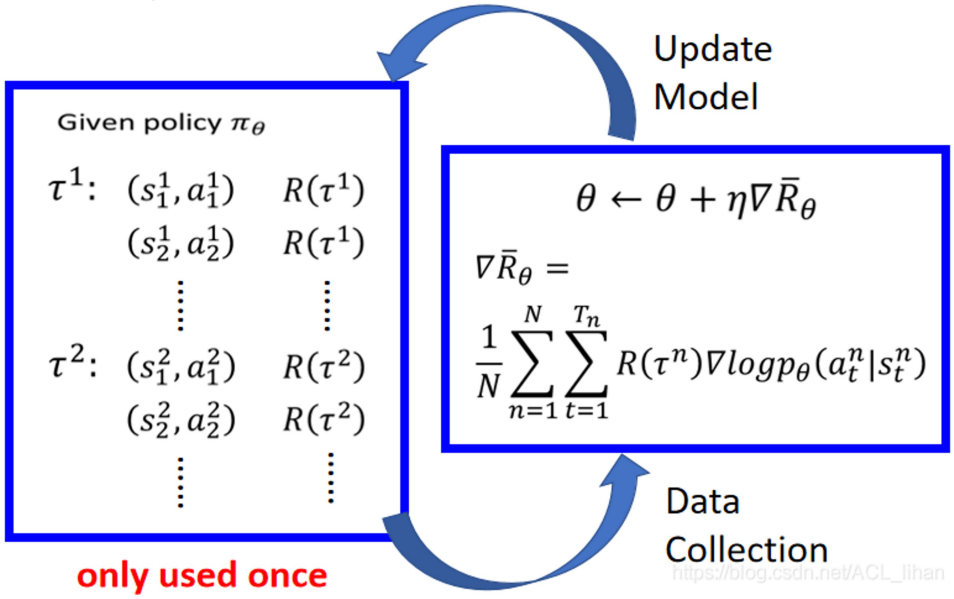
$$\nabla f(x) = f(x)\nabla log f(x)$$

$$= E_{\tau\sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)] \approx \frac{1}{N}\sum_{n=1}^{N} R(\tau^n)\nabla log p_\theta(\tau^n)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla log p_\theta(a_t^n|s_t^n)$$

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)]$$

Policy Gradient



PPO

2021年2月14日 7:15

- On-policy V.S .Off-policy
  - On-policy: The agent learned and the agent interacting with the environment is the same. (learn from its own interactions)

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)]$$

    - Use π_θ to collect data. When θ is updated, we have to sample training data again.
    - Goal: Using the sample from $\pi_{\theta'}$ to train θ . θ' is fixed, so we can re-use the sample data.
  - Off-policy: The agent learned and the agent interacting with the environment is the different.(learn from others interactions)

$$\nabla \bar{R}_\theta = E_{\tau \sim p_{\theta'}(\tau)}\left[\frac{p_\theta(\tau)}{p_{\theta'}(\tau)} R(\tau)\nabla log p_\theta(\tau)\right]$$

    - Sample the data from θ'.
    - Use the data to train θ many times.

- When using off-policy, using gradient for parameter update

On-policy → Off-policy

Gradient for update $\nabla f(x) = f(x)\nabla log f(x)$

$$= E_{(s_t,a_t)\sim\pi_\theta}[A^\theta(s_t,a_t)\nabla log p_\theta(a_t^n|s_t^n)]$$

$A^{\theta'}(s_t,a_t)$ This term is from sampled data.

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{P_\theta(s_t,a_t)}{P_{\theta'}(s_t,a_t)} A^{\theta'}(s_t,a_t)\nabla log p_\theta(a_t^n|s_t^n)]$$

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}\frac{p_\theta(s_t)}{p_{\theta'}(s_t)} A^{\theta'}(s_t,a_t)\nabla log p_\theta(a_t^n|s_t^n)]$$

$$J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t,a_t)\right]$$

We can get new Objective function:

$$J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t,a_t)\right]$$

- PPO algorithm

If θ and θ'have large difference, the result of Important sampling will be wrong. **In order to**

prevent too much difference between θ and θ', PPO algorithm can be used.

Proximal Policy Optimization (PPO)

$$\nabla f(x) = f(x)\nabla log f(x)$$

$$J_{PPO}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta KL(\theta, \theta')$$

$$J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t)\right]$$

In PPO, there are two terms. The first term is the origianl objective function which is what we want to optimize. The Second term is constrain which is similar with Regularization term in ML. This term represents the difference between the two distributions. **The greater the difference, the greater the value. The penalty imposed on the objective function will be larger,** so the difference between the two distributions should be as small as possible to ensure a larger objective function.(**The Difference between θ and $\theta'$is on behavior not parameters.** The change of parameters and the change of action may not be exactly the same, the parameters have changed a lot, but the behavior of output may not change much, so what we really care about is the gap in the behavior of actors)

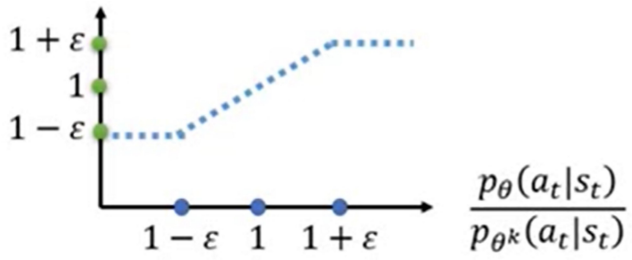
How to set β ?
In PPO, β is somewhat similar to the learning rate and needs to be set manually. We can set two thresholds, KL_{MAX} and KL_{Min}. After a parameter update, check the value of KL. If KL(θ , θ') is greater than KL_{MAX}, then the difference between θ and θ' is too much, so β need to be increase, which means the penalty imposed on the objective function will be larger, otherwise decrease β and penatly.

PPO2

PPO2 algorithm

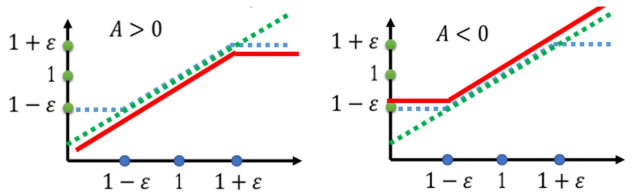
$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t),\right.$$
$$\left. clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}(s_t, a_t)\right)$$

Function : Min(a,b) is to take the minimum between a and b
Function: clip(X, minimum value A, Maximum value B), when X is greater than B,Result = B. When X is smaller than A,Result = A. otherwish, Result= X.



The whole algorithm:

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}A^{\theta^k}(s_t,a_t),\quad clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)},1-\varepsilon,1+\varepsilon\right)A^{\theta^k}(s_t,a_t)\right)$$



The green line is the first term, the red line is the Objective function.

When Action is positive, we hope $\frac{P_\theta}{P_{\theta'}}$ increase until $1+\varepsilon$, if greater than $1+\varepsilon$, there is no benifit.

When Action is nagetive, we hope $\frac{P_\theta}{P_{\theta'}}$ decrease until $1-\varepsilon$.

In this way, p_θ won't differ too much with $P_{\theta'}$

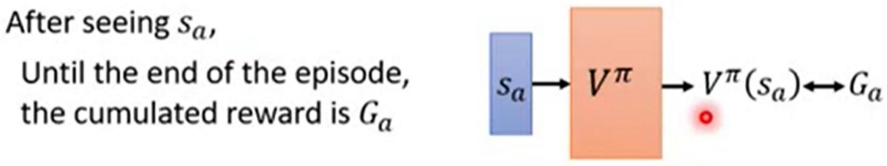
Q- Learning

2021年2月14日 21:29

- Introduction of Q- learning
  Q-learning ia a value-based method, in the value based method, what we learn is **critic** but policy
- Critic
  - A critic doesn't not directly determine the action.
  - Given an actor π, it evaluates how good the actor is
  - State Value function $V^{\pi}(s)$
    When using actor π, the cumulated reward expects to be obtained after visiting state s.

  The output values of a critic depend on the actor evaluated.
  Eg: $V_{5\text{yearsago}}(\text{do } Master's\ thesis) = bad;$
  $V_{now}(\text{do } Master's\ thesis) = good;$

- How to estimate $V^{\pi}(s)$
  a. Monte-Carlo(MC) based approach : The Critic watches π playing the game



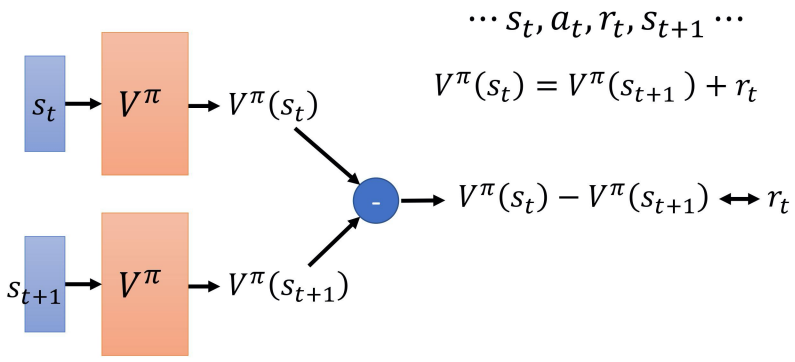
  CRITIC watches π play the entire game until the end of that game turn and then calculates the cumulative reward (**train CRITIC by comparing expected reward and actual reward**(G)

  However, this method of waiting until the end of a game turn to calculate the reward is slow to train, so another method, Temporal difference (TD), is introduced.

  b. Temporal-difference(TD) approach

How to estimate $V^{\pi}(s)$

- **Temporal-difference (TD) approach**



Some applications have very long episodes, so that delaying all learning until an episode's end is too slow.

根据T 时刻的States$_t$, 采取policya_t, 得到reward r_t,

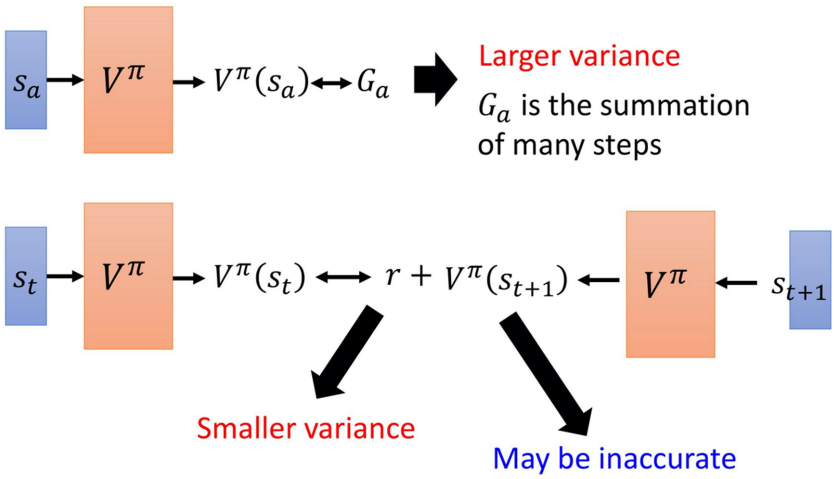
在进入下一个$State_{t+1}$ ， 这样 2个state之间差的Value 就是 r_t.

- MC vs. TD

$$Var[kX] = k^2 Var[X]$$

MC vs. TD

-



The biggest problem with MC is that it has very large Variance. When we're playing a game, it's inherently random, so is a random variable .

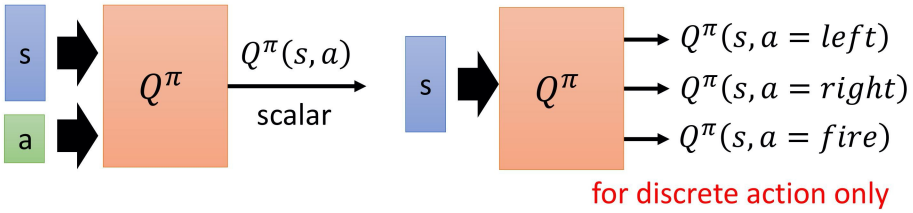
Even though everytime when we arrive at the same S_a, we still get the different G_a. And, there are huge difference between $G_a$we get each time.

In the TD, we need minmize formular, and the r is the random variable. Even though we take the same action in S_a, we may still get different r. However, the Variance of r is much smaller than the Variance of G_a, Since the G_a is combined by many r. But the V may be inaccurate. **So the biggest problem with TD is that it has inaccurate result.**

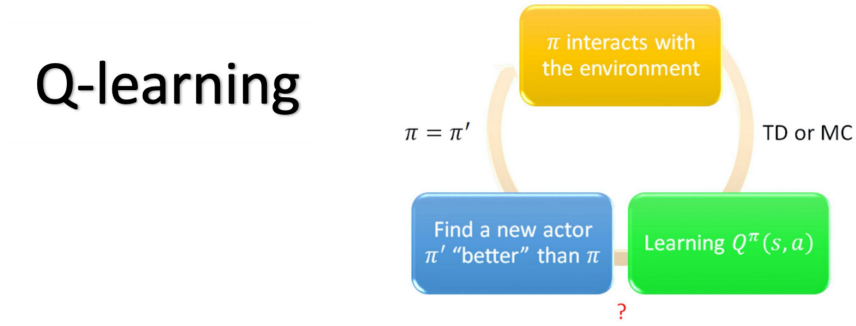
- Another Critic

Another Critic

- State-action value function $Q^{\pi}(s,a)$
  - When using actor π, the *cumulated* reward expects to be obtained after taking a at state s



- Policy Improvement
  With the Q-function, we can evaluate whether this action is good or bad. When use this Q function, we can achieve inforcement learning, find a better policy π'. Then use π' replace π, we can find a new Q function. As the cycle goes on, the policy will get better and better



- Given $Q^{\pi}(s,a)$, find a new actor π' "better" than π
  - "Better": $V^{\pi'}(s) \geq V^{\pi}(s)$, for all state s

$$\pi'(s) = arg\max_{a} Q^{\pi}(s,a)$$

- π' does not have extra parameters. It depends on Q
- Not suitable for continuous action a (solve it later)

"Better" means that in the same state s, If we continue to interact with the environment with π , we will definitely get less reward than if we use π'.

How to find π' ?

Suppose we have learned the q-function by poice π. Now, at a given state S, we take all possible actions a into the q-function and see which of them maximizes the value of the q-function. Then the action (π') will be taken.

- Target Network