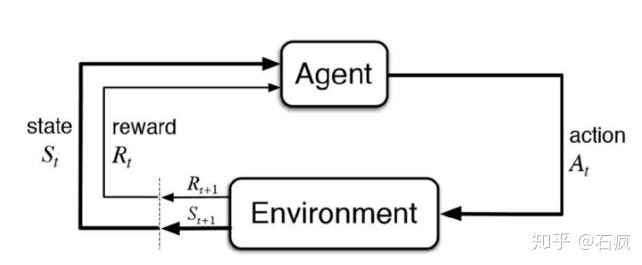
Reinforcement Learning (RL) is a way of machine learning (ML) which makes an entity be rewarded (assessment result of last time-step movement) at the next time-step.

从什么都不懂，通过不断的尝试，找出事件的规律，学习到到达目标的方法

Mark each action

the setup of the RL combines two: （schematic）



Environment: the situation where the Agent acts

Agent: RL itself

Environment send a state signal to Agent 1st, and then the Agent acknowledge (responses to the state) it based on the algorithms.

Next, the Environment sent another state and the reward. Agent update the algorithms based on the reward, the assessment result of last action. Loop until the end state signal has been sent to the Agent.

Terms:

Action (A): possible actions by Agent

State (S): Return of current situation from environment

Reward(R): return value, the results of last action took by Agent

Strategy (pi): Next action of Agent according to current situation

Value (V): long-term return value based on discount, different from R

[Vpi(s): with strategy pi, s’s E(long term return value) under the current situation]

Q: similar to value but got 1 more parameter current action a

[Qpi(s,a): the long term return when s take action a with strategy pi]

Model free vs Model based

|  |  |
| --- | --- |
| Model-Free | Model-based |
| No need to understand the environment (model)  Q-learning, sarsa, policy gradients  Got return from environment and then to learn | Understand environment and create a model  Q-learning, sarsa, policy gradients  Model the environment (virtual)  想象 ， 预判断 |

On-policy vs off-policy (experience learing)

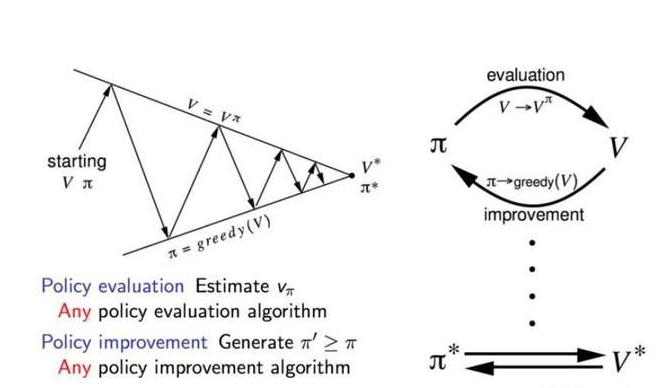
|  |  |
| --- | --- |
| On-policy | off-policy |
| Learning when playing  Sarsa sarsa(lamda) | Learning when playing and watching others playing  Q-learning, deep Q network |

|  |  |
| --- | --- |
| Policy-based | Value-based |
| Analysis environment  calculate the probabilities of each actions | Calculate the value of the action  Choose the action with highest weights  Contious acion can not apply |

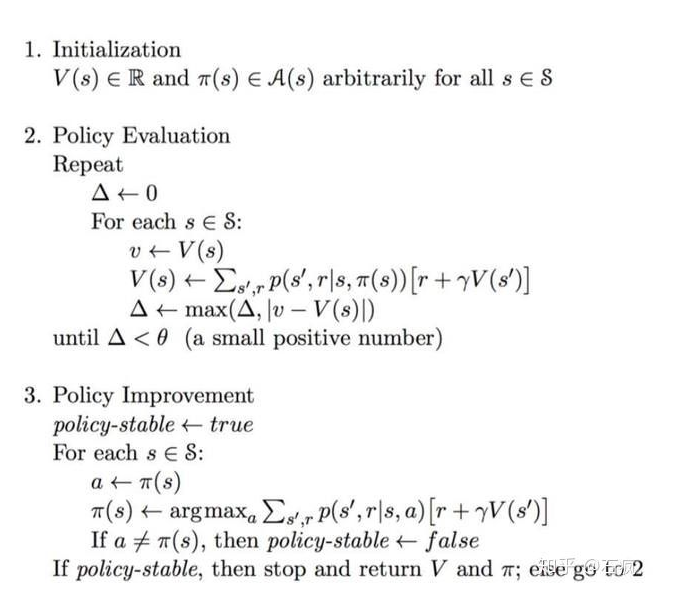
Actor-critic: add the study for the original policy gradients

Algorithms:

Q-learning: Bellman Equation , off-policy, model-free



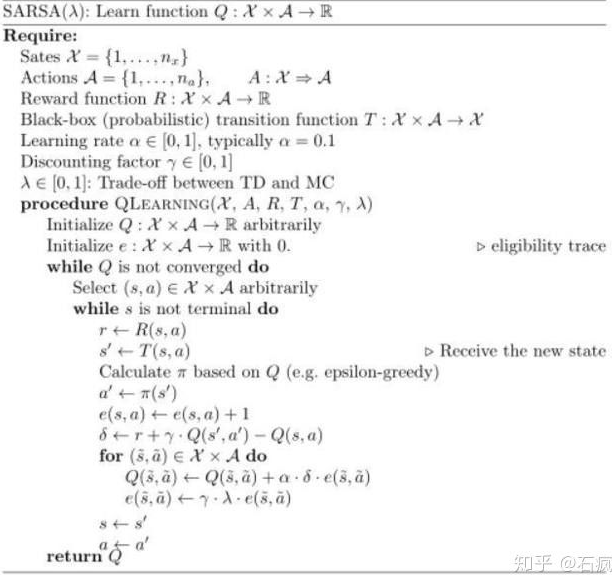
the expectations are subscripted by to indicate that they are conditional on being followed.



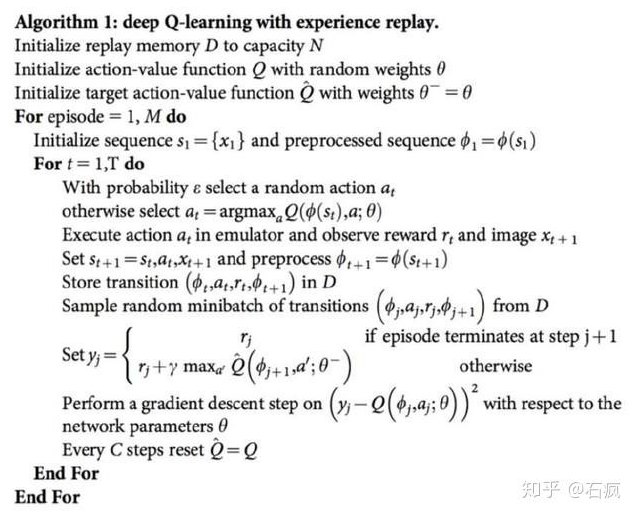
The strategy assessment evaluates the value function V of the greedy strategy obtained from the last policy improvement. On the other hand, policy improvement updates the strategy by acting to maximize the V value of each state. The update equation is based on the Bellman equation. It iterates until it converges

The next criterion for selecting a is to be able to maximize the Q value of the next state instead of following the current strategy.

SARSA: state-action-reward-state-action, on-policy



DQN



1. Experience Replay: Because the training samples in a typical reinforcement learning setup are highly correlated and the data efficiency is low, this will make the network more difficult to converge. One way to solve the sample distribution problem is to use empirical playback. Essentially, the sample conversion is stored and then randomly selected from the Conversion Pool to update the knowledge.

2. Separate Target Network: The target Q network is the same as the network structure used for valuation. According to the pseudo code above, the target network is reset to another in each C step. As a result, fluctuations become less severe and result in more stable training.

* [Numpy, Pandas](https://morvanzhou.github.io/tutorials/data-manipulation/np-pd/) (必学), 用于学习的数据处理
* [Matplotlib](https://morvanzhou.github.io/tutorials/data-manipulation/plt/) (可学), 偶尔会用来呈现误差曲线什么的
* [Tkinter](https://morvanzhou.github.io/tutorials/python-basic/tkinter/) (可学), 你可以自己用它来编写模拟环境， virtualization
* [Tensorflow](https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/) (可学), 后面实现神经网络与强化学习结合的时候用到
* [OpenAI gym](https://gym.openai.com/) (可学), 提供了很多现成的模拟环境

The learner and decision maker is called the agent. The thing it interacts with, comprising

everything outside the agent, is called the environment. These interact continually, the agent selecting

actions and the environment responding to these actions and presenting new situations to the agent.

The environment also gives rise to rewards, special numerical values that the agent seeks to maximize over time through its choice of actions. See actions can be any decisions we want to learn how to make, and the states can be anything we can know that might be useful in making them

The general rule we follow is that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment. We do not assume that everything in the environment is unknown to the agent. For example, the agent often knows quite a bit about how its rewards are computed as a function of its actions and the states in which they are taken. But we always consider the reward computation to be external to the agent because it denes the task facing the agent and thus must be beyond its ability to change arbitrarily.

Agent-environment boundary represents the limit of the agent's absolute control, not of its knowledge.

In practice, the agent{environment boundary is determined once one has selected particular states, actions, and rewards, and thus has identied a specic decision making task of interest.

any problem of learning goal-directed behavior can be reduced to three signals passing back and forth between an agent and its environment: one signal to represent the choices made by the agent (the actions), one signal to represent the basis on which the choices are made (the states), and one signal to de the agent's goal (the rewards).

These may not be sufficient but widely useful and applicable.

Solving a reinforcement learning task means, roughly, finding a policy that achieves a lot of reward over the long run

Exploration is needed because there is always uncertainty about the accuracy of the action-value estimates.

E-greedy action selection forces the non-greedy actions to be tried, but indiscriminately,with no preference for those that are nearly greedy or particularly uncertain.

If a model is not available, then it is particularly useful to estimate action values (the values of state{

action pairs) rather than state values

With a model, state values alone are su\_cient to determine a

policy; one simply looks ahead one step and chooses whichever action leads to the best combination of

reward and next state, as we did in the chapter on DP.

Chapter 5

On-policy methods attempt to evaluate or improve the policy that is used to make decisions, whereas o\_-policy methods evaluate or improve a policy di\_erent from that used to generate the data.

e-greedy policies, meaning that most of the time they choose an action that has maximal

estimated action value, but with probability " they instead select an action at random.

All nongreedy actions are given the minimal probability of selection e/A(s)