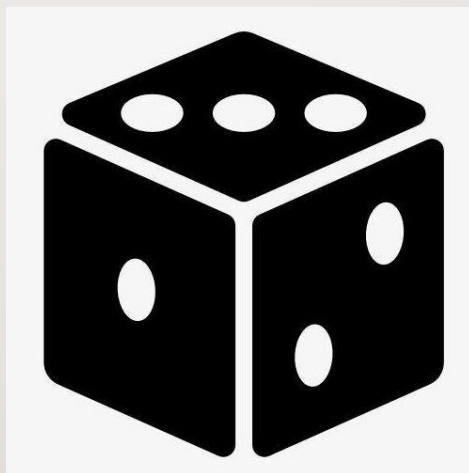


真-极简爬坡式强化学习入门

作者: 骰子AI

2022-3



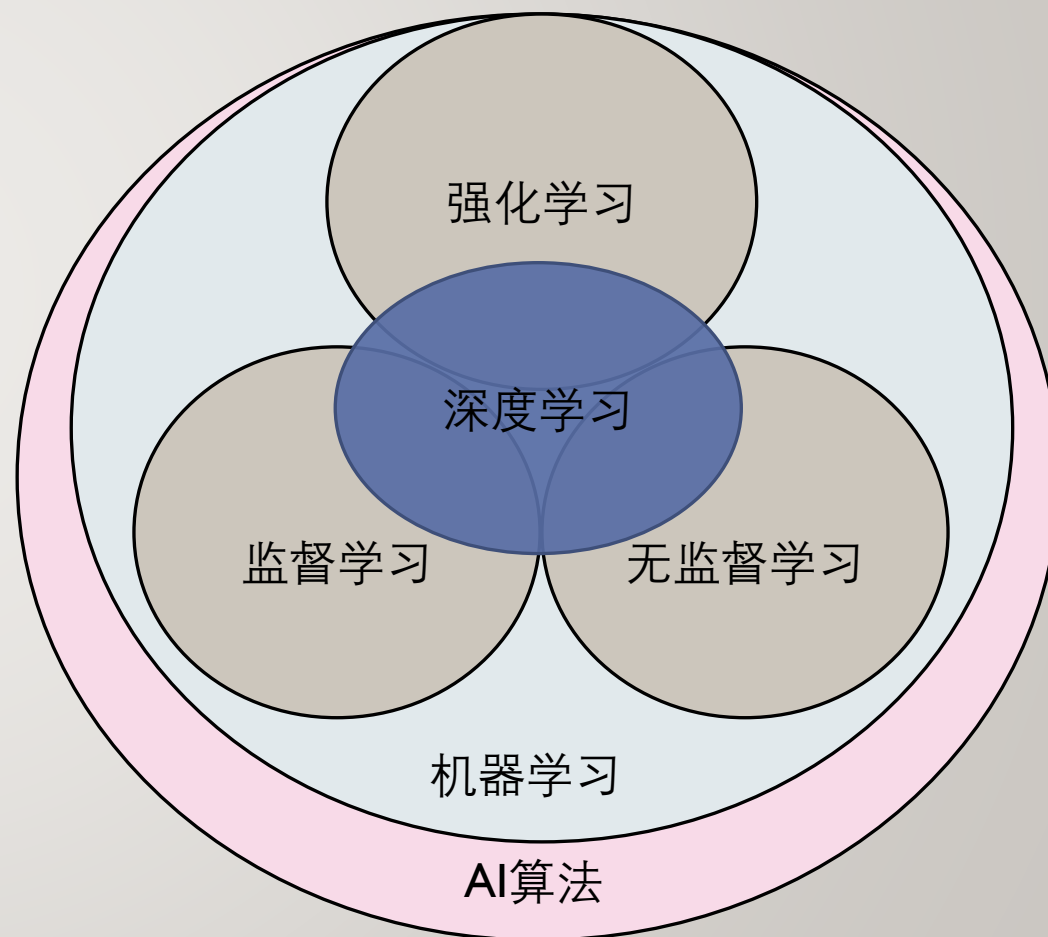
大纲

- 代码地址: [rexrex9/reinforcement_torch_pfrl](https://github.com/rexrex9/reinforcement_torch_pfrl): 真-极简强化学习(基于torch的强化学习框架pfrl) ([github.com](https://github.com/rexrex9/reinforcement_torch_pfrl))
- 上半部分: 前置知识, 基础的机器学习的Sense
 1. 强化学习初印象(3-7)
 - ① 什么是强化学习(3-4)
 - ② TD时序差分(5)
 - ③ 探索与利用(6)
 - ④ 基于表格的强化学习(7)
 2. SARSA算法(8-10)
 3. Q-learning算法(11)
 4. On-policy与Off-policy(12)
- 下半部分: 前置知识, 真-极简神经网络入门
 5. 为何需要神经网络(13)
 6. DQN算法(14-15)
 7. 经验回放(16)
 8. 固定Q目标(17)
 9. 探索概率衰减(18)
 10. 使用pfrl(19)

什么是强化学习

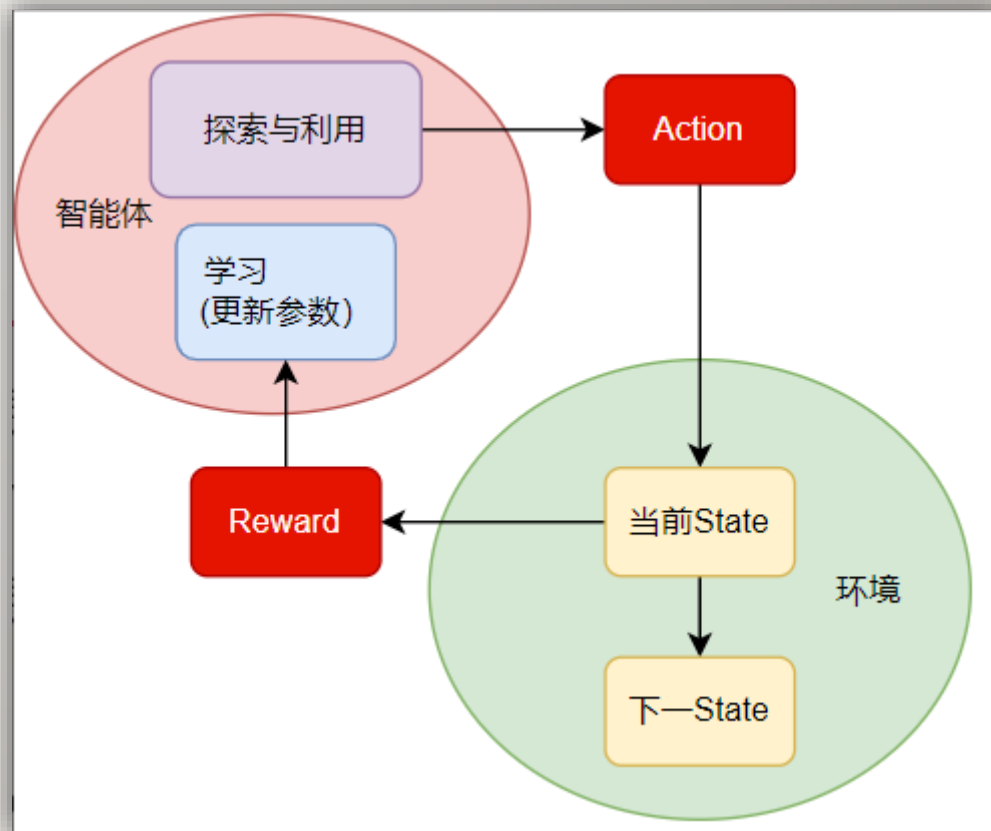
- 监督学习：根据输入数据与标注之间的关系从而建立模型进行标注的预测。
- 无监督学习：无标注，仅根据数据自身的规律寻找关系。
- 强化学习：AI不断与环境进行交互，并根据环境的反馈进行学习。

注意：强化学习不等于半监督学习。



什么是强化学习

- 中心思想：智能体Agent在环境Environment中，根据环境的状态State，进行动作Action，并根据Reward奖励反馈，不断优化动作。
- 英文名：**Reinforcement Learning**
- 两部分：
 1. Agent智能体
 2. Environment环境
- 三要素：
 1. State状态/Observation观察值
 2. Action动作
 3. Reward奖励

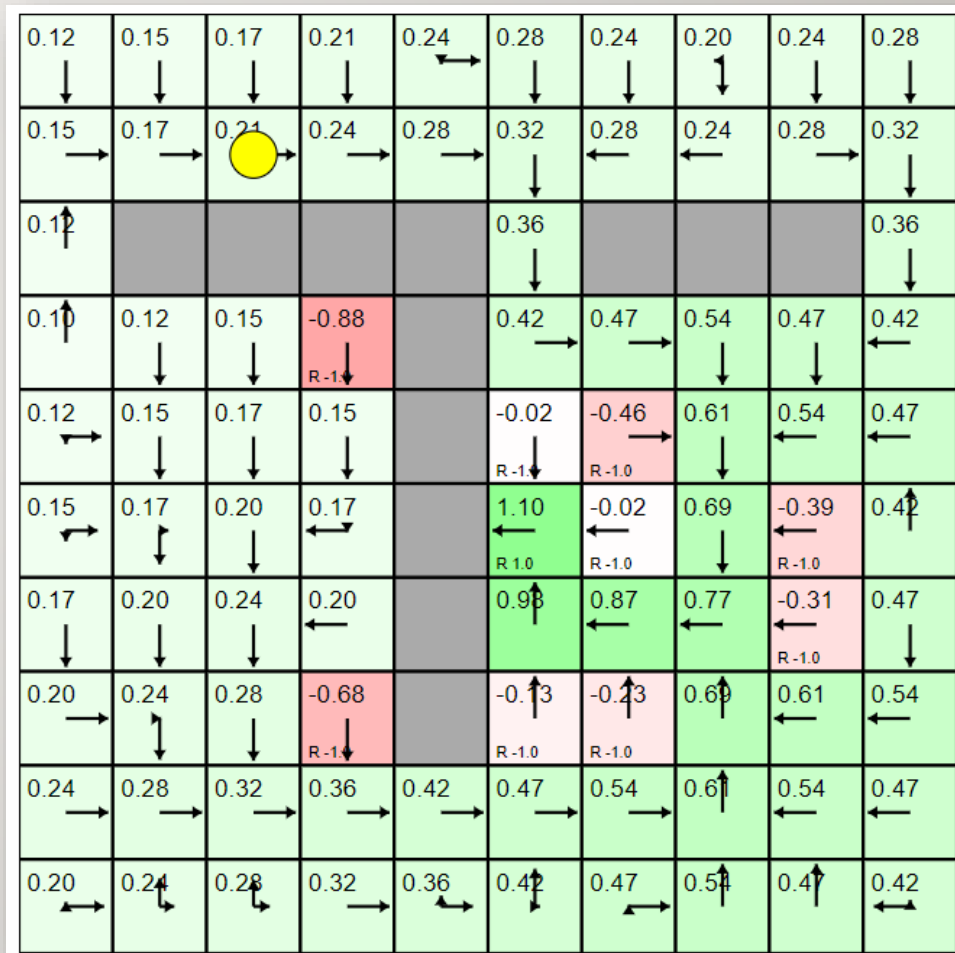


学习算法 - TD 时序差分

- 游戏模拟网址: [REINFORCEjs: Gridworld with Dynamic Programming \(stanford.edu\)](https://reinformejs.com/)
- 时序差分 Temporal Difference, TD

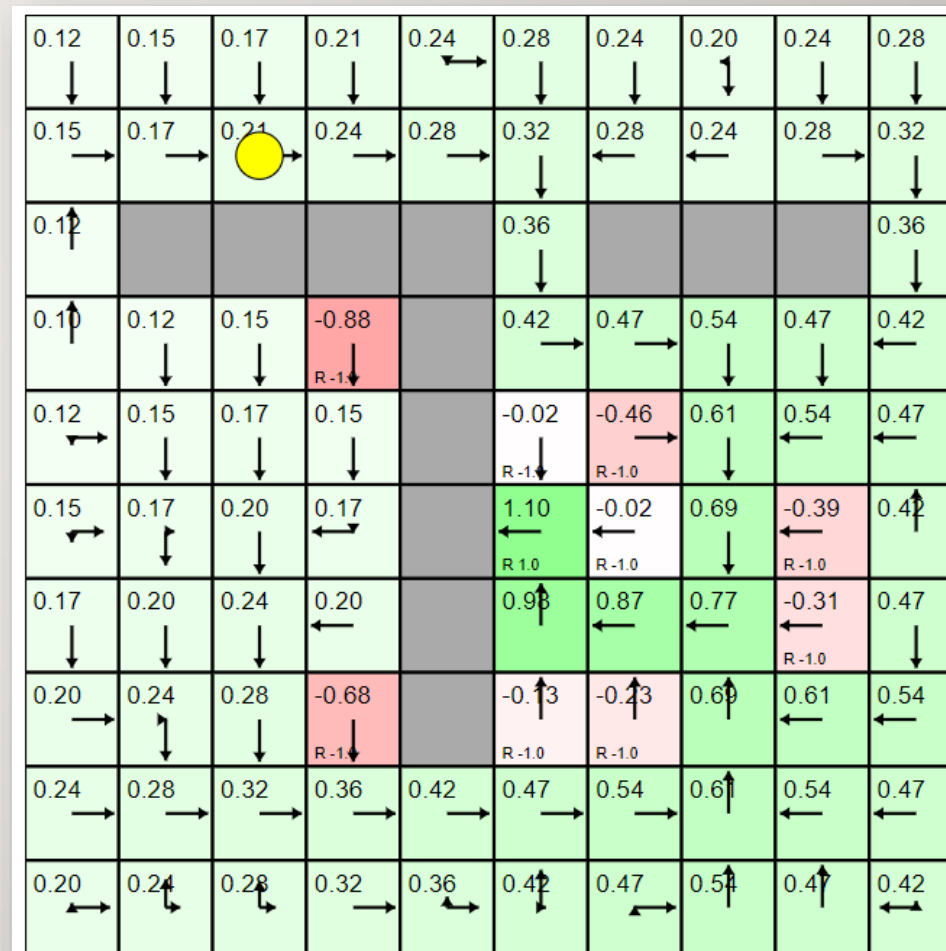
$$V(S_t) \leftarrow V(S_t) + \alpha (\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{目标值}} - \underbrace{V(S_t)}_{\text{当前值}})$$

- $V(S_t)$ 表示状态 S 在 t 时刻的价值, R_{t+1} 代表奖励。
- 公式表示对当前状态 S_t 的价值更新过程。
- α 与 γ 是超参, α 是学习步长, γ 是价值衰减率。



探索与利用算法 - ϵ - greedy

- 探索：指随机选择一个Action。
- 利用：指根据经验选择一个Action, 当前例子中即选择价值收益最大的Action即可。
- 设一个探索概率阈值 ϵ ，每一次选择Action时有 ϵ 选择探索方式，有 $(1 - \epsilon)$ 的概率选择利用方式。



基于表格的强化学习

- Q表格：状态数量 × 动作数量的表格。

状态 (State)	上	下	左	右
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

0.12 ↓	0.15 ↓	0.17 ↓	0.21 ↓	0.24 ↗	0.28 ↓	0.24 ↓	0.20 ↓	0.24 ↓	0.28 ↓
0.15 →	0.17 →	0.21 →	0.24 →	0.28 →	0.32 ↓	0.28 ←	0.24 ←	0.28 →	0.32 ↓
0.12 ↑					0.36 ↓				0.36 ↓
0.10 ↑	0.12 ↓	0.15 ↓	-0.88 R-1.0		0.42 →	0.47 →	0.54 ↓	0.47 ↓	0.42 ←
0.12 ↗	0.15 ↓	0.17 ↓	0.15 ↓		-0.02 R-1.0	-0.46 R-1.0	0.61 ↓	0.54 ←	0.47 ←
0.15 ↗	0.17 ↓	0.20 ↓	0.17 ←		1.10 R-1.0	-0.02 R-1.0	0.69 ↓	-0.39 R-1.0	0.42 ↑
0.17 ↓	0.20 ↓	0.24 ↓	0.20 ←		0.98 ↑	0.87 ←	0.77 ←	-0.31 R-1.0	0.47 ↓
0.20 →	0.24 ↓	0.28 ↓	-0.68 R-1.0		-0.13 R-1.0	-0.13 R-1.0	0.69 ↑	0.61 ←	0.54 ←
0.24 →	0.28 →	0.32 →	0.36 →	0.42 →	0.47 →	0.54 →	0.61 ↑	0.54 ←	0.47 ←
0.20 ↖	0.24 ↑	0.28 ↑	0.32 →	0.36 ↖	0.42 ↑	0.47 ↖	0.54 ↑	0.47 ↑	0.42 ↖

SARSA算法

- Q表格：状态数量 × 动作数量的表格。

状态 (State)	上	下	左	右
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

- TD公式：

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

- SARSA公式：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

➤ S_t : 当前的状态

➤ A_t : 当前的Action

➤ R_{t+1} : 下一状态的Reward

➤ S_{t+1} : 下一状态

R_{t+1} 与 S_{t+1} 是由 A_t 与环境交互得到

➤ A_{t+1} : 下一个的Action

A_{t+1} 是探索与利用得到的下一次Action

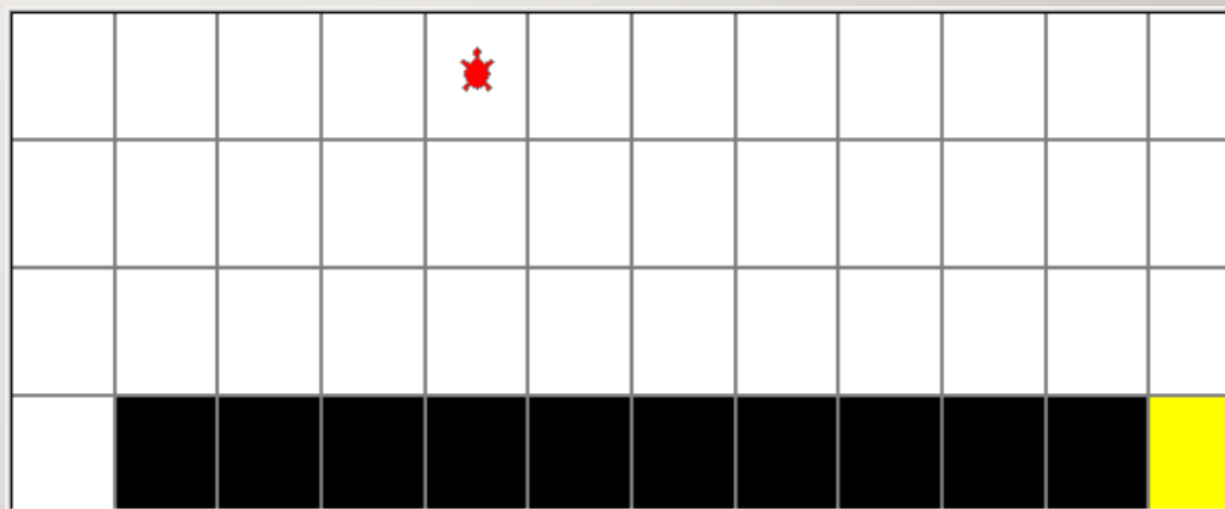
初步环境准备

- Python
- `pip install numpy`
- `pip install pygame`
- `pip install gym` #模拟练习强化学习时环境的工具库，官网地址[Gym \(openai.com\)](https://gym.openai.com/)

入门实战-悬崖行走

```
import gym
env = gym.make("CliffWalking-v0") #声明一个环境
state = env.reset() #重置环境, 返回最初的状态
state, reward, done, info = env.step(action) #输入一个action与环境交互, 通常action就是一个int类型的数字, 用以索引某个具体的动作
# state:交互后的状态
# reward:奖励
# done: 布尔值, 代表完成与否
# info: 一点信息
env.render() #渲染一帧动画
```

- 走到白格得到Reward -1。
- 走到黑格(悬崖)会回到原点且Reward-100。
- 走到黄点结束。



Q-LEARNING算法

- SARSA公式:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

- Q-learning公式:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max(Q(S_{t+1}, :)) - Q(S_t, A_t))$$

- 区别在于对于动作期望收益的获取，**SARSA**是由完整的探索与利用机制得到期望价值(On-policy)。而**Q-learning**直接采取利用的方式(Off-policy)得到期望价值。

ON-POLICY与OFF-POLICY

- On-policy:

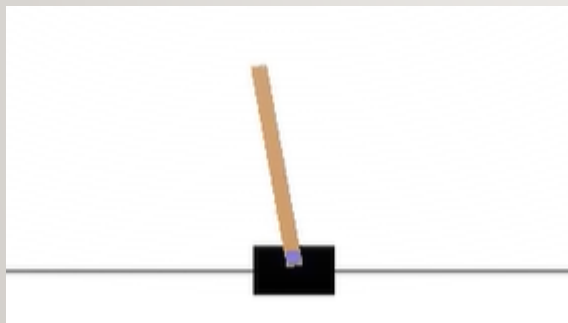
- 指仅采取一种策略 进行学习和交互 的算法。
- 在计算期望价值时假设的Action与下一次环境交互的Action是一样的。
- On-policy在学习时会兼顾探索，从而相对保守。

- Off-policy:

- 指学习策略(target policy)与环境交互策略(behavior policy)不同的算法。
- Behavior policy : 与环境交互时采用探索与利用机制得到的Action。
- Target policy: 在计算期望价值时假设的Action是最大收益的Action，相当于仅采取探索与利用中利用的方式得到预测的Action。
- Off-policy在学习时不会考虑探索，所以每一步都会选择最大的利益。

为何需要神经网络

- 基于Q表格强化学习算法的局限性是仅能处理可穷举环境状态的情况。
- 若环境状态不可穷举，则可由环境当前的观察值(Observation)替代当前的状态。
- 下图是gym中CartPole的小游戏，小车每次会选择向左或向右移动，若车子的位置或者杆子的角度超过一定数值则游戏结束，所以小车坚持的越久Reward就越高，最高为200。



观察值名称	Min	Max
车子位置	-4.8	4.8
车子速度	$-\infty$	∞
杆子角度	-0.418 (24°)	0.418 (24°)
杆子旋转速度	$-\infty$	∞

- 可用神经网络替代Q表格。该神经网络可称为Q函数，通常输入是当前环境的Observation观察值。输出是维度为动作数量的向量, 向量中每个元素对应每个动作的收益。

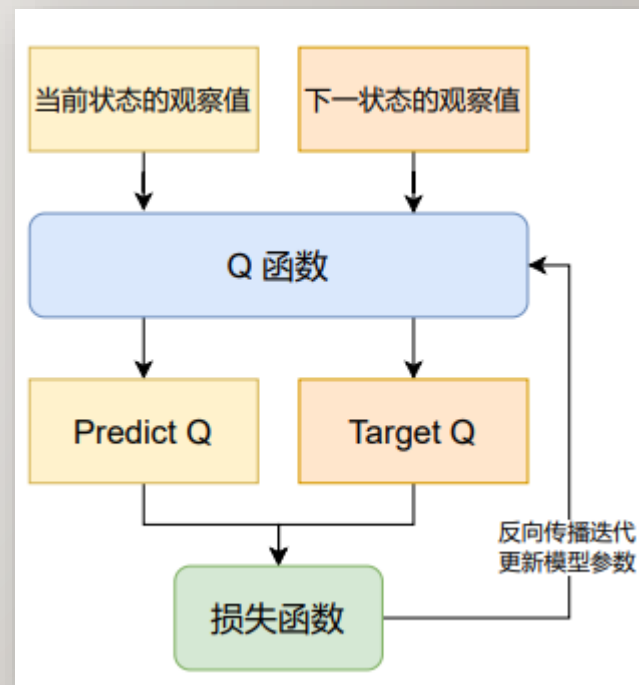
DQN

- Q-learning公式:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(\underbrace{R_{t+1} + \gamma \max(Q(S_{t+1}, :))}_{\text{目标Q}} - \underbrace{Q(S_t, A_t)}_{\text{当前Q}} \right)$$

- DQN(Deep Q-Network) 的进化:

1. 用神经网络替代Q表格。该神经网络可称为Q函数，通常输入是当前环境的Observation观察值。输出是维度为动作数量的向量，向量中每个元素对应每个动作的收益。
2. Q-learning公式中的 $Q(S_t, A_t)$ ，在DQN中可被认为 $Q(obs_t)[A_t]$ ，即通过Q函数预测得到的向量中对应 A_t 的收益。
3. 当前Q在DQN中被认为是预测Q(Predict Q)，也就是 $Q(obs_t)[A_t]$ 。
4. 目标Q(Target Q)在DQN，可被表示为 $R_{t+1} + \gamma \max(Q(obs_{t+1}))$ 。
5. 计算预测Q与目标Q之间的损失函数从而反向传播迭代更新Q函数的模型参数。



DQN环境准备

- pip install torch
- pip install pfrl : 基于PyTorch的强化学习算法库, [PFRL, a deep reinforcement learning library — PFRL 0.3.0 documentation](#)

经验回放

- 经验回放(Experience replay):

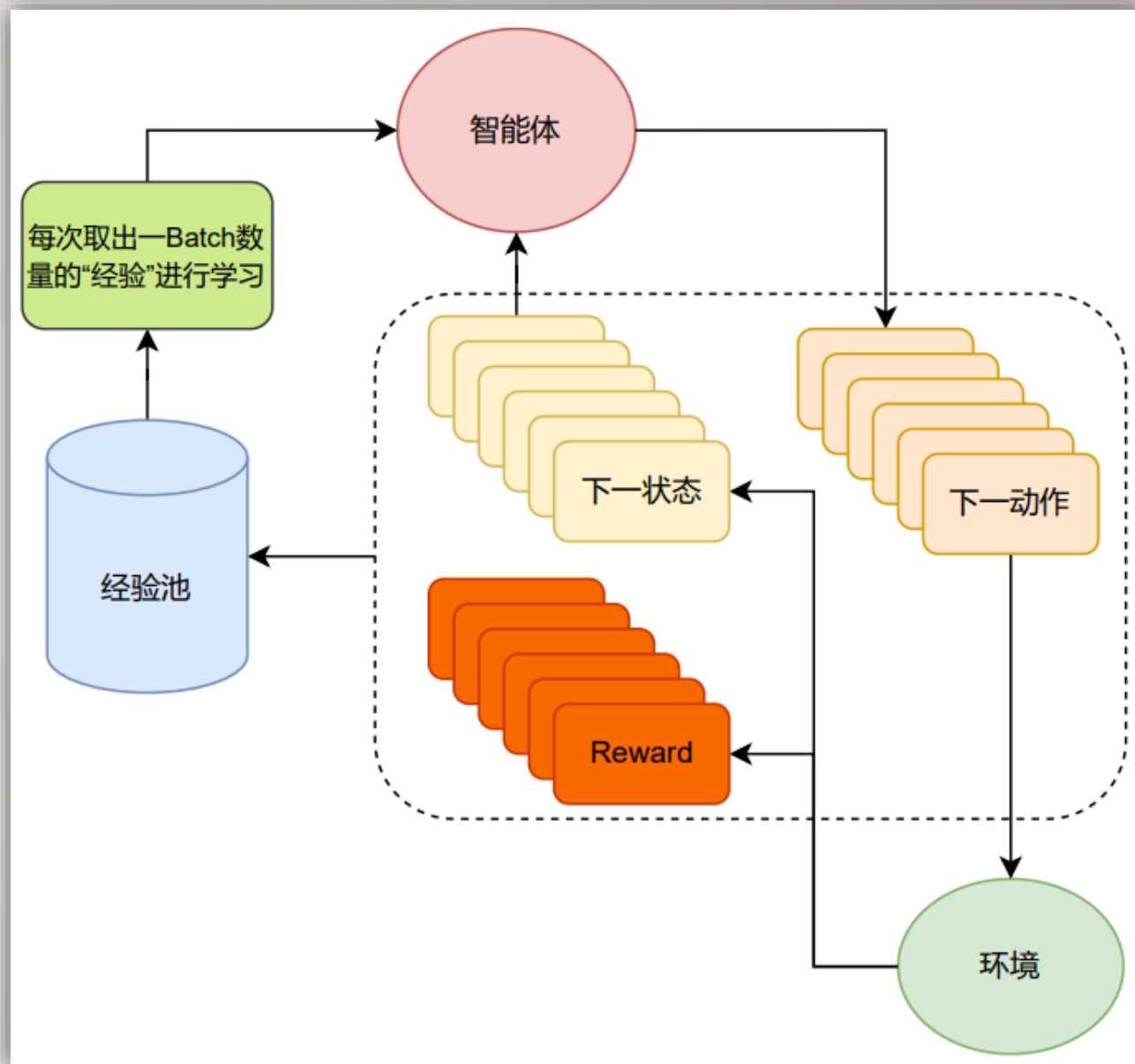
指设定一个经验池, 将每一步交互缓存进经验池, 积攒到一定程度后可以每一次取出Batch Size个“经验”从而进行批量学习。

- 其中注意的事项如下:

1. 频批分开: 学习频次与每次学习的“经验”数量(Batch Size)是不同的。例如可设定为每4轮交互进行一次学习, 每次学习从经验池中取出32轮交互经验。
2. 延迟启动: 前N轮的交互并不进行学习, 等经验池中的经验积攒到一定程度后再开始学习。

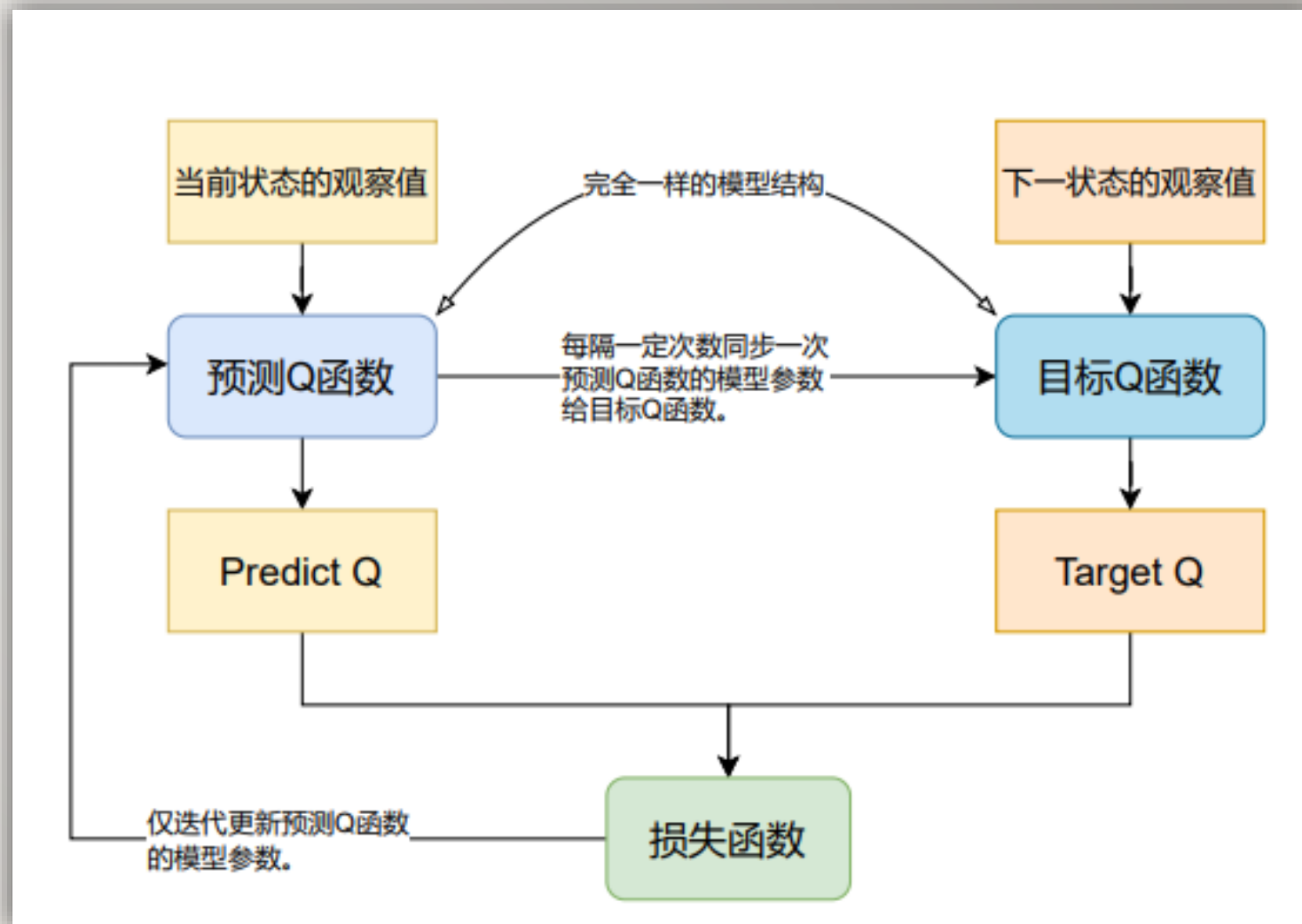
- 经验池的好处:

1. 提高样本利用率
2. 打乱样本关联性
 - 因为普通的机器学习样本之间的关系都是独立的。而智能体与环境交互产生的经验样本如果不经处理则存在序列样本关联性, 这对模型的更新不利。



固定Q目标

- 固定Q目标(Fixed Q Target):
 - 首先将Q函数复制一份作为目标Q函数。原先的Q函数则称为预测Q函数。
 - Predict Q由预测Q函数得到, Target Q由目标Q函数得到。
 - 模型学习的过程中仅迭代更新预测Q函数的模型参数。目标Q函数的模型参数固定不变。
 - 每隔一定次数将预测Q函数的模型参数同步给目标Q函数。
- 好处:
 - 减轻模型训练的难度。
 - 因为若目标Q函数每次同样会迭代更新。则意味着Target Q每次均会变化, 这就像是照着移动的靶子练习射箭, 会增加训练的难度。



探索概率衰减


- $\epsilon - greedy$ 原本公式:

$$Action = \begin{cases} \text{探索,} & p = \epsilon \\ \text{利用,} & p = 1 - \epsilon \end{cases}$$

- ϵ 的值可随着智能体与环境的交互次数增多而减少, 例如设定一个 ϵ 衰减值 ϵ_{decay} 。则每一次 ϵ 的更新可表达为:

$$\epsilon \leftarrow \epsilon - \epsilon_{decay}$$

使用PFRL

 PFRL
latest

Search docs

Installation

Quickstart Guide

API Reference

Action values

Agents

Experiments

Explorers

Modules

Policies

Q-functions

Replay Buffers

Docs » API Reference

Edit on GitHub

API Reference

- Action values
- Agents
- Experiments
- Explorers
- Modules
- Policies
- Q-functions
- Replay Buffers

Previous

Next

结束

