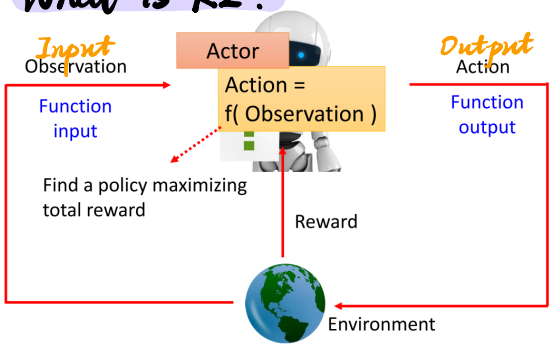
What is RL?



目标：找到一个最大化 Reward 的 Actor

围棋只有赢了才有 Reward=1，输了 Reward =-1，中间大部分 Action 的 reward=0

Step 1: function with unknown → **Step 2: define loss from training data** → **Step 3: optimization**

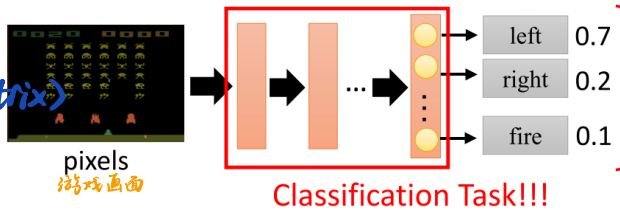
Step 1：Function with Unknown

→ RL中的 Actor 通常为 Policy Network

- Input：Observation（向量化表示为 Vector/Matrix）
- Output：每个 Action 所得的分数

→ 按几率从 Action 中 Sample 出行为（有面临相同情况时作出不同动作）

→ 有时也会采用 Softmax



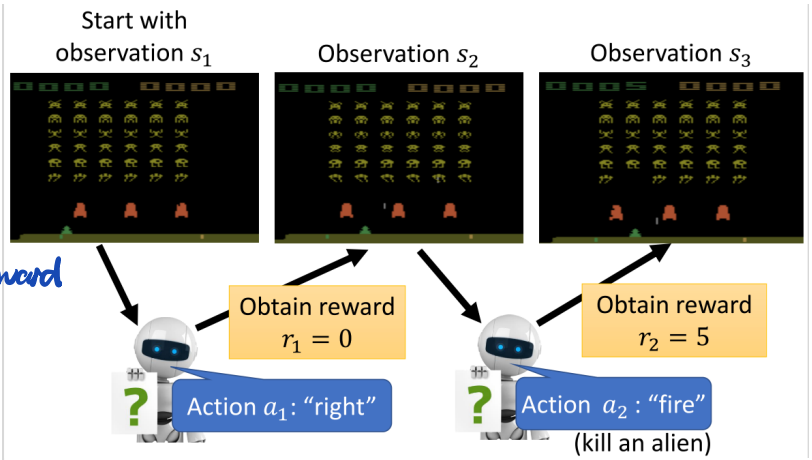
Classification Task!!!

Step 2：Define "Loss"

- 一局游戏结束 = 1 episode（满足结束条件）
- Total Reward ⇔ Return：1个Action获得的奖励
- Loss = − Total Reward：一局游戏获得的总 reward

Total reward (return): $R = \sum_{t=1}^{T} r_t$

What we want to maximize

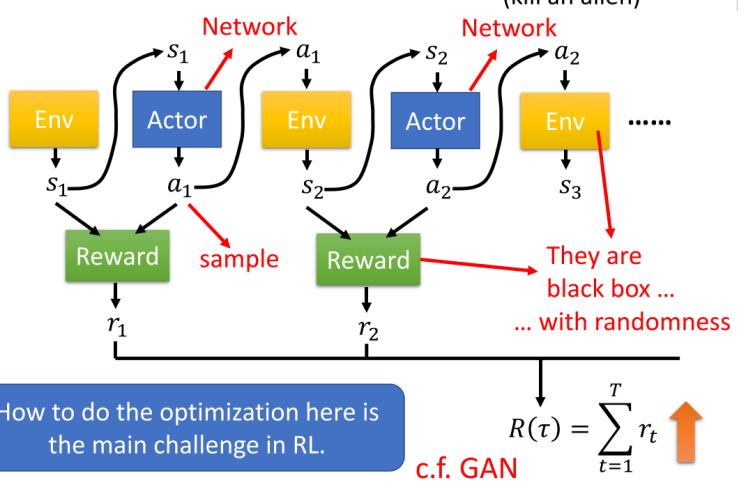


Step 3：Optimization

- Trajectory $\tau = \{s_1, a_1, s_2, a_2, \cdots\}$
- Reward Function 需要同时考虑 s 和 a
- Actor 的 Output 具有随机性：同样的 s 每次产生的 a 不一定相同
- Environment 和 Reward 都是 Black Box 且具有随机性：不是 Network，无法采用 Gradient Descent 调整网络参数

Random Seed 的随机性：训练时采用不同的初始化参数

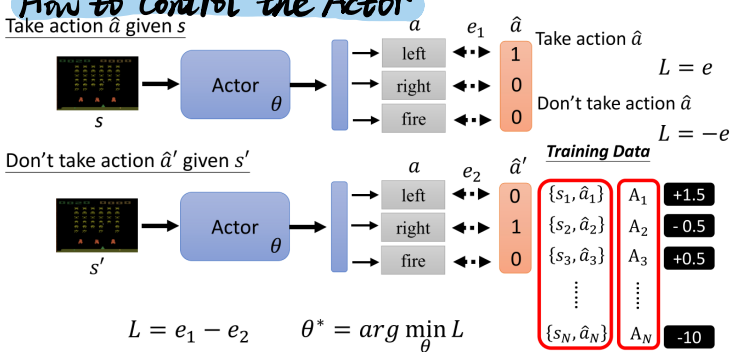
RL 的随机性：即使给定相同的 Input，同一个 agent 的 Output 也可能不同

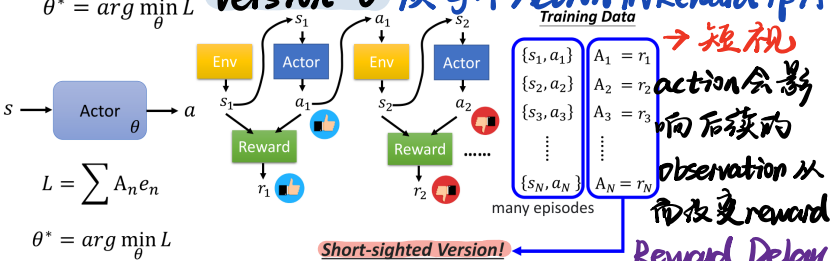


Policy Gradient

To learn more about policy gradient: https://youtu.be/W8XF3ME8G2I

How to Control the Actor

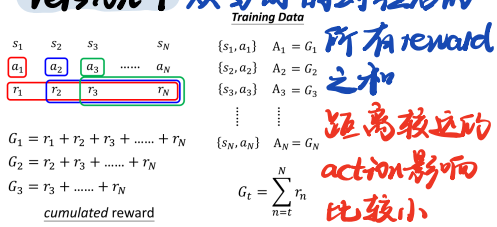




Version 0 以每个 Action 的 Reward 作 A → 短视 action 会影响后续的 observation 从而改变 reward

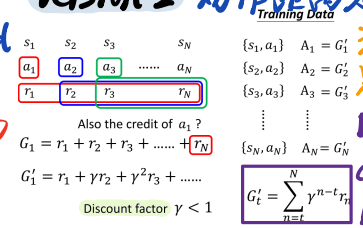
Reward Delay

Version 1 从当时时刻往后的所有 reward 之和



距离较远的 action 影响比较小

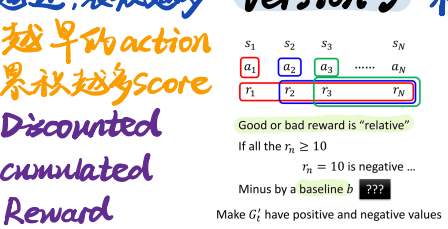
Version 2 动作距离越远，衰减越多



越早的 action 累积越多 score

Discounted cumulated Reward

Version 3 标准化 Reward



设定 baseline 使 G_t' 有正有负

Baseline b 平均实力

Policy Gradient

- Initialize actor network parameters θ^0
- For training iteration $i = 1$ to T
  - Using actor θ^{i-1} to interact
  - Obtain data $\{s_1, a_1\}, \{s_2, a_2\}, \ldots, \{s_N, a_N\}$
  - Compute $A_1, A_2, \ldots, A_N$
  - Compute loss L
  - $\theta^i \leftarrow \theta^{i-1} - \eta \nabla L$

Data collection is in the "for loop" of training iterations.

Trajectory of θ^{i-1}

s_1	s_2	s_3		s_N
a_1	a_2	a_3	……	a_N
r_1	r_2	r_3		r_N

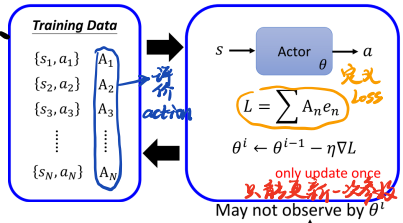
→ To train θ^i

一般训练是先收集 Data 再进入循环

R2 是跑一次循环收集一次 Data

→ 训练过程十分耗时

第 i-1 次的资料是 θi 的经验，对于 θi 的训练不一定有帮助



(被训练)

→ 收集资料的 Actor 跟训练的和执行的相同 (和 Environment 互动) → On-policy

Trajectory of θ^{i-1}

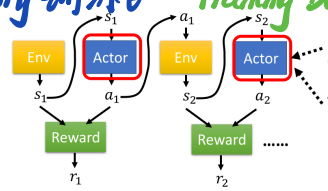
s_1	s_2	s_3		s_N
a_1	a_2	a_3	……	a_N
r_1	r_2	r_3		r_N

Actor to train 和 Actor for Interacting 不相同 → Off-policy 不用每次重新收集

→ 用 θi-1 的 Trajectory 训练 θi

但被训练的 Actor 要知道自己和示范的不同

Training Data



Enlarge output entropy

Add noises onto parameters

How to Exploration: 扩大 Output 的 Entropy 或参数中加入 Noise

Exploration

Collection Training Data

Suppose your actor always takes "left".

We never know what would happen if taking "fire".

Data Collection 时需要有一些随机性

否则无法得知 Action 可能获得的 Reward

→ Sample Actions 的主要原因

Actor-Critic

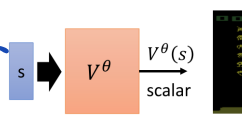
Critic

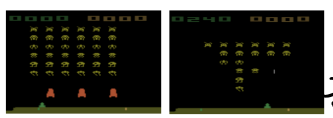
Actor θ 观察 State s 和采取 action a 的好坏

Value Function $V^\theta(s)$

$G'_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots\ldots$

使用 actor θ 时，在 state s 以后能获得的 G' 的期望





$V^\theta(s)$ is large $V^\theta(s)$ is smaller

Critic 的 Output 取决于正在评估的 actor

How to Estimate $V^\theta(s)$ 训练 Critic 的 Value Function

→ Monte-Carlo (MC) based approach 蒙特卡洛法 → Temporal-difference (TD) Approach 时序差分法，不用等到整个 Episode 结束

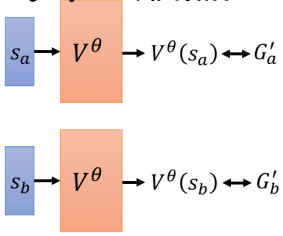
Critic 观察 actor θ 和 Environment 互动

After seeing s_a,

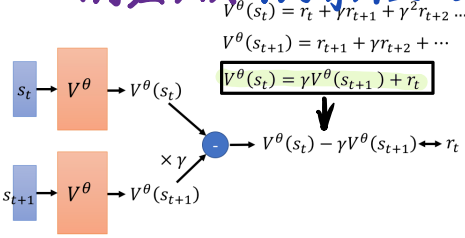
Until the end of the episode, the cumulated reward is G'_a

After seeing s_b,

Until the end of the episode, the cumulated reward is G'_b



一个 Episod 结束后所得的 DCR G' 作为训练的 GT



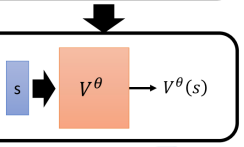
$\cdots s_t, a_t, r_t, s_{t+1} \cdots$ 都是 reward 的期望值，在此处省略

Version 3.5

用 Value Function 评估的基于 (s_t, a_t) 的 G_t' 的期望作为 Version 3 的 Baseline

$\{s_t, a_t\}$ $A_t = G'_t - V^\theta(s_t)$

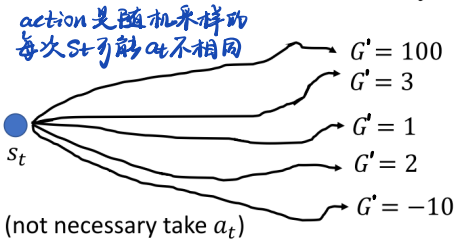
s_1	s_2	s_3		s_N
a_1	a_2	a_3	……	a_N
r_1	r_2	r_3		r_N



Training Data

$\{s_1, a_1\}$	$A_1 = G'_1 - V^\theta(s_1)$
$\{s_2, a_2\}$	$A_2 = G'_2 - V^\theta(s_2)$
$\{s_3, a_3\}$	$A_3 = G'_3 - V^\theta(s_3)$
⋮	⋮
$\{s_N, a_N\}$	$A_N = G'_N - V^\theta(s_N)$

action 是随机采样的，每次 St 可能 at 不相同



$V^\theta(s_t)$ 多次 G_t' 的平均

$A_t > 0$ a_t is better than average.

$A_t < 0$ a_t is worse than average.

G_t' 为 St 在采取一个 at 后最终所得的 Discounted Cumulated Reward (一个 Sample 的结果)

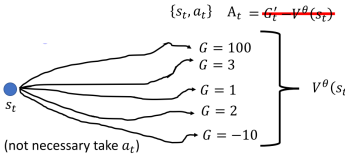
→ 可能偏差较大

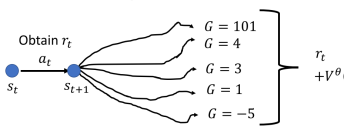
Version 4.0

平均减平均代替 Sample 减平均

$r_t + V^\theta(s_{t+1}) - V^\theta(s_t)$

$\{s_t, a_t\}$ $A_t = G'_t - V^\theta(s_t)$





→ 在 St 下执行完 at 以后，直至 Episode 结束，期望获得的 G' 的平均值

→ $G_t' = r_{t+} V^\theta(S_{t+1})$

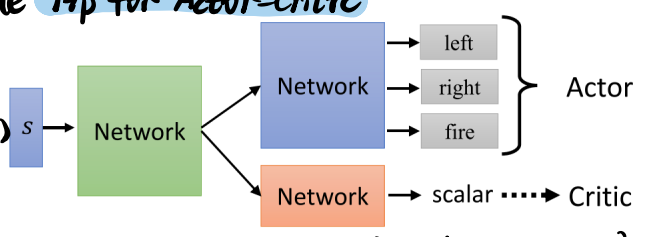
$A_t = G_t' - V^\theta(S_t) = r_t + V^\theta(S_{t+1}) - V^\theta(S_t)$

→ Advantage Actor-Critic (AAC)

直接用 Critic 决定采用的 action

→ DQN https://arxiv.org/abs/1710.02298 Rainbow

Tip for Actor-Critic



Actor 和 Critic 两个 Network 的部分参数可以共享

Reward Shaping

Sparse Reward reward 在多数情况下都是 0，每个 action 的执行结果区别不大 (eg. 机械手臂拧螺丝)

$A_t = r_t + V^\theta(s_{t+1}) - V^\theta(s_t)$ → 定义额外的 reward 帮助 agent 学习 Reward Shaping

Curiosity 当 agent 发现新的 (且有意义) 的事物则给予额外 reward

https://arxiv.org/abs/1705.05363

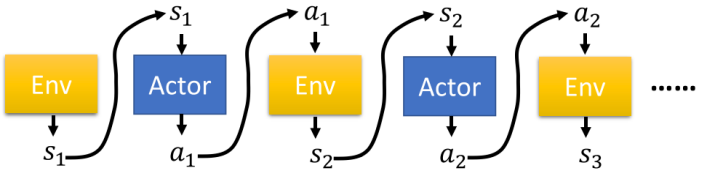
Curiosity Driven Exploration by Self-Supervised Prediction

s_1	s_2	s_3		s_N
a_1	a_2	a_3	……	a_N
r_1	r_2	r_3		r_N

Training Data

$\{s_1, a_1\}$	A_1
$\{s_2, a_2\}$	A_2
$\{s_3, a_3\}$	A_3
⋮	⋮
$\{s_N, a_N\}$	A_N

No Reward : Learning from Demonstration



Motivation

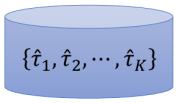
- 在真实环境中 reward 的定义可能都非常困难
- 人工定义的 reward 可能导致不可控的行为

Imitation Learning

Actor 可以和 Environment 互动，但没有 reward function

→ 使用 Demonstration of the expert 作为参照（人类的行为方式）→ Behavior Cloning (Supervised Learning)

We have demonstration of the expert.



Each $\hat{\tau}$ is a trajectory of the export.

Self driving: record human drivers

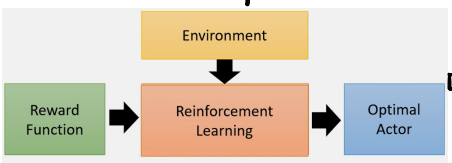
Robot: grab the arm of robot

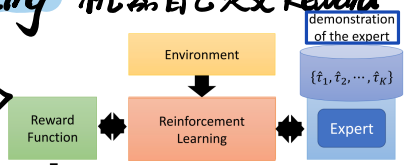
Problem：

- Experts 只会 sample 有限的 Observation（不会做过简单的）
- Agent 会学习一些无关的动作

Inverse Reinforcement Learning

机器自己定义 Reward





从 Expert Demonstration 反推 Reward Function 再训练 Actor 使用 Reward Function 找到 Optimal Actor

可以简单，也可能学出比较复杂的 Actor

Using the reward function to find the *optimal actor*.

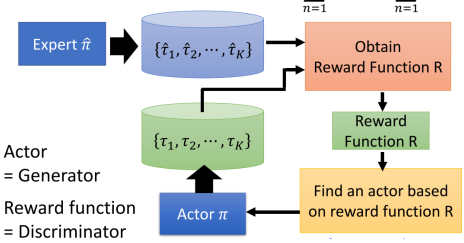
先定义 Reward Function 找到 Optimal Actor

Algorithm

- Principle: ***The teacher is always the best***.
- Basic idea:
  - Initialize an actor
  - In each iteration
    - The actor interacts with the environments to obtain some trajectories.
    - Define a reward function, which makes the trajectories of the teacher better than the actor.
    - The actor learns to maximize the reward based on the new reward function.
  - Output the reward function and the actor learned from the reward function

Framework of IRL

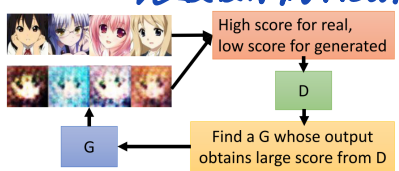
$$\sum_{n=1}^{K} R(\hat{\tau}_n) > \sum_{n=1}^{K} R(\tau)$$



Actor = Generator

Reward function = Discriminator

GAN



IRL