简易强化学习

Sisyphes

2020 年 6 月 6 日

目录

第一节 一些废话 **3**
1.1 废话补充 2023.3.11 3

第二节 基本认识 **5**
2.1 什么是强化学习 5
2.2 符号定义 6
2.3 小技巧 6
2.4 基本定理 7

第三节 Markov Models **7**
3.1 MRP 7
3.2 MDP 8
3.3 评估和优化 9

第四节 Markov Models Free **12**
4.1 评估 (预测) 12
4.1.1 MC 12
4.1.2 TD 13
4.2 优化 (控制) 13
4.2.1 Sarsa 14
4.2.2 Q-Learning 15
4.3 重要采样的应用 15
4.4 DP 和 TD 的差异 16

第五节 函数参数化 **16**
5.1 线性组合 17
5.2 DQN 18

第六节 策略参数化 **19**
6.1 基础建模 19
6.1.1 演员评论家 21
6.2 策略梯度的改进 22
6.2.1 Policy Gradient→TRPO→ACKTR→PPO 23
6.2.2 Q-learning→DDPG→TD3→SAC 25

第七节 Model Based **25**

第八节 模仿学习 **25**

第九节 分布式系统 **25**

第十节 多智能体 **26**

第十一节 Code **26**

第十二节 RLHF **26**

第一节　一些废话

RL 的学习应该会比较漫长。一个假想的研究员的学习情况，也许首先会看 sutton 的 complete draft，同时实现一些经典算法，然后紧跟学术潮流，看一些来自 openAI, DeepMind 等的论文，这中间也许会选一门课程，比如 David Silver，伯克利 RL 或周博磊等，偶尔也许会翻阅一些博客。我的初略体验 (一周) 是，这里面会存在一些不可避免的问题，比如流行算法的快速更新，新算法实验成本高，公式细节上似乎不太严格等。为了较大限度的减少以上问题，扎实经典基础理论，消化新论文核心思想 (不必推导每个细节)，加强代码实践，侧重建模能力练习，也许还不错。

本小册子来源于zhoubolei RL，之所以有以上设想，也源于周老师的讲义，比如在其中会存在 (列表可见**??**)，同样的符号，定义有微差 (一些公式来自 sutton，一些公式来自新论文，在约定上的细节差异)，下标不严谨，期望下标省略，这些会增加一些理解负担，通过做实验来确定会存在更多其他问题。另外，算法的快速迭代，去推导每一步会显得荒谬，而实际建模有近似需求，黑盒化，因此着重练习建模能力，在某一具体问题上深入挖掘，或许会不错。周老师的讲课非常棒，提纲挈领，简明扼要，深入浅出。

所以，本小册子，以基本概念，基本定理，问题建模，代码实现，新论文的近似阅读为逻辑展开。当前版本 0.1，将来若有时间，会逐渐完善。

1.1　废话补充 2023.3.11

三年过去了。

列举一些基本认识，和新的 (相对于 2020 年) 比较清晰的入门教程。

- 动手学强化学习包含比较全面的介绍，以及代码。代码存在版本问题，我更改了一些细节，适应了现在的版本 gym=0.26.x。

强化学习基本工具的认识：

- Issac Gym:GPU 采样
- Isaac Gym 的经验介绍
- gym
- ml-agents

- 各种基本库的经验介绍。

链接形式补充了一些遗留的基础，别人写的不错，我也懒得滥竽充数了。个人主要集中在一些基础代码，项目实战，RLHF，角色动画，图像生成的精准控制相关的内容上。

第二节　基本认识

2.1　什么是强化学习

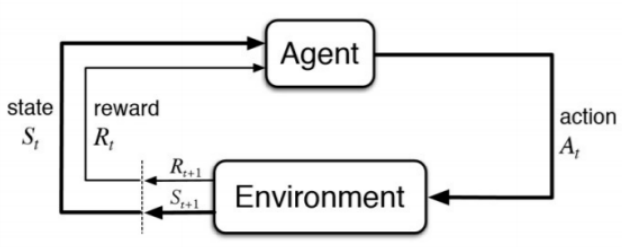


图 1: RL Prime

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

- 基本要素:
  - Agent(智能体)
  - model(模型)
  - value(价值函数)
  - policy(策略函数)
- 特点：
  - 试错探索
  - 延迟回报
  - 时间问题 (序列数据，无既定分布)
  - Agent 的动作会影响接下来的输入数据，并改变环境的状态
- 例子：
  - 策略游戏，跑跑卡丁车，纸片游戏，围棋等
  - 机器人走路，穿衣，抓取东西，拧魔方
  - 路径规划，机器控制

2.2　符号定义

2.3　小技巧

增量平均：一个简单的变换将序列平均值的计算转化为前一个平均值和当前值与前平均值的“差”的和。

$$\begin{aligned}\mu_t &= \frac{1}{t}\sum_{j=1}^{t} x_j \\ &= \frac{1}{t}\left(x_t + \sum_{j=1}^{t-1} x_j\right) \\ &= \frac{1}{t}\left(x_t + (t-1)\mu_{t-1}\right) \\ &= \mu_{t-1} + \frac{1}{t}\left(x_t - \mu_{t-1}\right)\end{aligned} \tag{1}$$

重要采样: 分布 P 不好采样，用更好采样的 Q 替换它，只是需要同时乘以 P 相对于 Q 的权重因子。

$$E_{x\sim P}[f(x)] = \int f(x)P(x)dx \approx \frac{1}{n}\sum_i f\left(x_i\right)$$

$$\begin{aligned}\mathbb{E}_{x\sim P}[f(x)] &= \int P(x)f(x)dx \\ &= \int Q(x)\frac{P(x)}{Q(x)}f(x)dx \\ &= \mathbb{E}_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right] \approx \frac{1}{n}\sum_i \frac{P\left(x_i\right)}{Q\left(x_i\right)}f\left(x_i\right)\end{aligned}$$

对数求导：利用对数导数为其倒数的特点，可以将连乘函数的导数和化(注意其和重要采样的联合使用)。

$$\begin{aligned}\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] &= \nabla_\theta \int f(x)p_\theta(x)dx \\ &= \int f(x)\nabla_\theta p_\theta(x)dx \\ &= \int f(x)p_\theta(x)\nabla_\theta \log p_\theta(x)dx \\ &= \mathbb{E}_{x\sim p_\theta(x)}\left[f(x)\nabla_\theta \log p_\theta(x)\right] \\ &\approx \frac{1}{N}\sum_{i=1}^{N} f\left(x_i\right)\nabla_\theta \log p_\theta\left(x_i\right)\end{aligned}$$

重参数化：利用分布的映射关系，将复杂函数的采样转化为从简单分布采样，然后映射到复杂分布上去，从而达到解决了复杂分布采样难问题。

$$\begin{aligned}
\varepsilon &\sim q(\varepsilon) \\
x &= g_\theta(\varepsilon) \\
\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[f(x)] &= \nabla_\theta \mathbb{E}_{\varepsilon\sim q(\varepsilon)}\left[f\left(g_\theta(\varepsilon)\right)\right] \\
&= \mathbb{E}_{\varepsilon\sim q(\varepsilon)}\left[\nabla_\theta f\left(g_\theta(\varepsilon)\right)\right] \\
&\approx \frac{1}{N}\sum_{i=1}^{N}\left(\nabla_\theta f\left(g_\theta\left(\varepsilon_i\right)\right)\right)
\end{aligned}$$

共轭梯度：使梯度的方向保持共轭关系 (垂直)，极大加快优化速度，Conjugate gradient method

2.4 基本定理

第三节 Markov Models

3.1 MRP

“Markov chain因俄国数学家安德烈·马尔可夫得名，为状态空间中经过从一个状态到另一个状态的转换的随机过程。该过程要求具备“无记忆”的性质：下一状态的概率分布只能由当前状态决定，在时间序列中它前面的事件均与之无关。”

数学符号语言为：历史状态：$h_t = \{s_1, s_2, s_3, \ldots, s_t\}$，状态 s_t 为 Markovian 当且仅当

$$\begin{aligned}
p\left(s_{t+1}|s_t\right) &= p\left(s_{t+1}|h_t\right) \\
p\left(s_{t+1}|s_t, a_t\right) &= p\left(s_{t+1}|h_t, a_t\right)
\end{aligned}$$

若状态转移过程中有奖励 (reward)，则称之为 MRP(Markov Reward Process)。其基本元素有：

S: 有限状态集 ($s \in S$)

P: 状态转移概率 $P\left(S_{t+1} = s'|s_t = s\right)$

R: 奖励函数 (reward)$R\left(s_t = s\right) = \mathbb{E}\left[r_t|s_t = s\right]$

γ: 折扣因子 $\gamma \in [0, 1]$

在实际的 RL 交互环境中，还需要定义三个变量：

Horizon: 不同 episode(一个探索周期) 的时间步长的最大值
Return: 时间 t 到 Horizon 的折扣回报和 $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \ldots + \gamma^{T-t-1} R_T$
状态价值函数：状态 s 在 t 时刻得到的回报的期望值 $V_t(s) = \mathbb{E}[G_t | s_t = s]$

根据定义,对价值函数做变换:$V(s) = \mathbb{E}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \ldots] | s_t = s]$ 容易得出 Bellman equation

$$V(s) = \underbrace{R(s)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s' \in S} P(s'|s) V(s')}_{\text{Discounted sum of future reward}} \tag{2}$$

于是对于有限状态的 MRP，有：

$$\begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_N) \end{bmatrix} = \begin{bmatrix} R(s_1) \\ R(s_2) \\ \vdots \\ R(s_N) \end{bmatrix} + \gamma \begin{bmatrix} P(s_1|s_1) & P(s_2|s_1) & \ldots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \ldots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \ldots & P(s_N|s_N) \end{bmatrix} \begin{bmatrix} V(s_1) \\ V(s_2) \\ \vdots \\ V(s_N) \end{bmatrix}$$

即 $V = R + \gamma P V$，其代数解为 $V = (I - \gamma P)^{-1} R$。不过实际应用中因 P 过大，求逆运算复杂度 $O(N^3)$ 过高，会选择迭代方式求解。包含动态规划，蒙特卡洛估计，Temporal-Difference 学习。

3.2 MDP

MDP：在 MRP 上增加一个动作项，可用 (S, A, P, R, γ) 表示。其中 A 有限的动作集,$P^a, P(s_{t+1} = s' | s_t = s, a_t = a), R(s_t = s, a_t = a) = \mathbb{E}[r_t | s_t = s, a_t = a]$

MDP 是对决策过程的建模，目标是在决策集 (Policies) 中寻找最优决策。其中决策函数是动作在状态空间的概率分布,用 $\pi(a|s) = P(a_t = a | s_t = s)$ 表示状态 s 执行动作 a 的概率，决策函数是时间独立的，即对任意 $t > 0$，$A_t \sim \pi(a|s)$。

给定 MDP(S, A, P, R, γ) 以及决策 π，状态序列 $S_1, S_2, \ldots$ 是一个马尔科夫过程 (S, P^π)，状态、奖励序列 $S_1, R_2, S_2, R_3, \ldots$ 是马尔科夫奖励过程

$(S, P^\pi, R^\pi, \gamma)$ 其中

$$P^\pi(s'|s) = \sum_{a\in A} \pi(a|s) P(s'|s,a)$$
$$R^\pi(s) = \sum_{a\in A} \pi(a|s) R(s,a)$$

策略 π 下开始状态为 s 的状态价值函数 $v^\pi(s) = \mathbb{E}_\pi[G_t|s_t = s]$，动作价值函数 $q^\pi(s,a) = \mathbb{E}_\pi[G_t|s_t = s, A_t = a]$，表示在状态 s 执行动作 a 后，按照策略 π 执行下去的回报期望。根据以上定义，容易得两者的关系：

$$v^\pi(s) = \sum_{a\in A} \pi(a|s) q^\pi(s,a) \tag{3}$$

$$q^\pi(s,a) = R_s^a + \gamma \sum_{s'\in S} P(s'|s,a)\, v^\pi(s') \tag{4}$$

其贝尔曼递归形式也容易写出：

$$v^\pi(s) = E_\pi[R_{t+1} + \gamma v^\pi(s_{t+1})|s_t = s] \tag{5}$$

$$q^\pi(s,a) = E_\pi[R_{t+1} + \gamma q^\pi(s_{t+1}, A_{t+1})|s_t = s, A_t = a] \tag{6}$$

分别把 (4) 带入 (3)，(3) 带入 (4) 得到：

$$\begin{aligned} v^\pi(s) &= \sum_{a\in A} \pi(a|s) \left(R(s,a) + \gamma \sum_{s'\in S} P(s'|s,a)\, v^\pi(s') \right) \\ q^\pi(s,a) &= R(s,a) + \gamma \sum_{s'\in S} P(s'|s,a) \sum_{a'\in A} \pi(a'|s')\, q^\pi(s',a') \end{aligned} \tag{7}$$

公式 (7) 的图 (Backup Diagram) 表示：

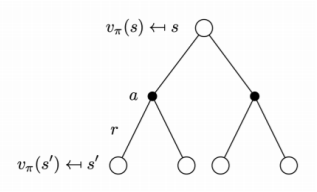
简言之，一个状态下可采取不同动作，得到状态动作价值；一个状态下执行不同的动作，得到奖励，并进入不同的状态；一个状态并执行了某一动作，得到即刻回报，进入不同状态，并执行不同动作。

建立好模型，并得到了一些关系式，**问题：如何计算？**

3.3 评估和优化

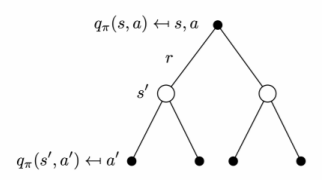
利用3, 2的关系式递推计算，如下迭代：

$$v_{t+1}(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \left(R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)\, v_t(s') \right), v_1 \to v_2 \to \ldots \to v^\pi$$



$$v^\pi(s) = \sum_{a \in A} \pi(a|s)(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v^\pi(s'))$$

图 2: V



$$q^\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \sum_{a' \in A} \pi(a'|s') q^\pi(s',a')$$

图 3: Q

即能得到给定策略 π 下的价值函数。这里递推关系和一般递关系比如 Fibonacci 数列顺序上是相反的，在实际优化中会一次更新所有值 (有限表格，和关键词 bootstrapping 吻合)，效率虽慢，但能得到最优解。

策略的最优价值函数：$v^*(s) = \max_\pi v^\pi(s)$

最优策略：$\pi^*(s) = \arg\max_\pi v^\pi(s)$ 或

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_{a \in A} q^*(s,a) \\ 0, & \text{otherwise} \end{cases}$$

在求得最优状态价值函数或动作价值函数后，根据如上两条定义，很容易得到最优策略。这个过程被称为值迭代。

这里还有另外一种方式得到最佳策略，一边 policy evaluation(V 值更新，划分不是很准确)，一边利用更新的价值函数计算出动作价值函数然后更新策略 (greedy)。即：

$$q^{\pi_i}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v^{\pi_i}(s')$$

$$\pi_{i+1}(s) = \arg\max_a q^{\pi_i}(s,a)$$

用图可表示为：

一个 MDP 模型存在唯一的最优价值函数 (证明见 Sutton)，但最优策略函数不唯一。

Table: Dynamic Programming Algorithms

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

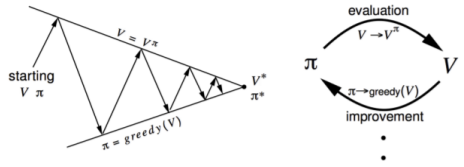


图 4: rl

一些存疑的地方：在 RL 中总是会存在各种分类，这对初学者不太友好。比如上面的总结，以及讲义中给出的两种策略优化的方式，在第二种方式中为何不直接迭代 $Q(s,a)$ 然后更新策略，以此迭代。另外 policy evaluation 是对价值函数 V 的计算，和策略是隐含关系，为何不是值迭代？看上面的表格，就有一点错乱的感觉。中间 2,3 行是期望方差，后列对应的却是评估和迭代，而末列 3,4 行为迭代，对应的中间却一个期望方程，一个最优方程。

简化：值迭代 + 最后求最值，值迭代 + 同时策略迭代。

理论补充说明 (证明见 Sutton)：策略优化是一个保序的过程。一轮优化得到的 $\pi'(s) = \arg\max_a q^\pi(s,a)$ 有 $v_{\pi'}(s) \geq v^\pi(s)$，且

$$\begin{aligned} v^*(s) &= \max_a q^*(s,a) \\ q^*(s,a) &= R(s,a) + \gamma \sum_{s' \in S} P\left(s'|s,a\right) v^*\left(s'\right) \end{aligned} \tag{8}$$

于是得到：

$$\begin{aligned} v^*(s) &= \max_a R(s,a) + \gamma \sum_{s' \in S} P\left(s'|s,a\right) v^*\left(s'\right) \\ q^*(s,a) &= R(s,a) + \gamma \sum_{s' \in S} P\left(s'|s,a\right) \max_{a'} q^*(s',a') \end{aligned} \tag{9}$$

取名为**最优 Bellman 方程**。

以上给出了1的 MDP 建模过程，并给出了在 MDP 已知的情况下，如何做策略评估，策略迭代，价值迭代。但现实世界中 MDP 模型不存在，或很难给出的情况很常见。这种情况，可用图5来表示。学术界称其为 Model-free。

问题：如何做 Model-free 的预测 (值估计) 和控制 (策略优化)？

图 5: RL Free

第四节 Markov Models Free

Markov Models Free 表示没有状态转移和奖励函数的 RL 模型，如图5所示。此时采用让 Agent 与环境交互，搜集数据，以频率统计的方式来模拟 MDP。于是定义 trajectory/episode $\{S_1, A_1, R_1, S_2, \ldots, S_T, A_T, R_T\}$，表示 Agent 从状态 S_1 开始做一个完整的 (直到结束,terminal) 状态、动作、及时获得的采集链。

那么在策略 π 下，如何从利用这些 trajectories 来评估价值函数？

4.1 评估 (预测)

对价值函数 V 的估计。

4.1.1 MC

$N(s)$ 表示状态 s 在 trajectories 出现过的次数 (有一次 tracjectory 表示一次的理解误差)，根据 G_t 定义(3.1)，容易从 trajectories 算出状态 s 下的 G_t，因 V_t 表示 G_t 的期望，在 MC 过程，可用平均值替代，于是结合(1)，在一个 episode/tracjectory 中，容易得到

$$
\begin{aligned}
&N\left(S_t\right) \leftarrow N\left(S_t\right)+1 \\
&v\left(S_t\right) \leftarrow v\left(S_t\right)+\frac{1}{N\left(S_t\right)}\left(G_t-v\left(S_t\right)\right)
\end{aligned}
$$

在不严格要求统计关系的模型中，也可以将其简化为：$v\left(S_t\right) \leftarrow v\left(S_t\right)+\alpha\left(G_t-v\left(S_t\right)\right)$

MC 方式的特点：

- 必须等到一个 episode 结束才能更新，从完整的 episode 中学习
- 只能在 episodic(terminating，得结束) 的环境中学习
- 不需要模型具有 Markov 特性，在非 Markov 环境中效果更好

4.1.2 TD

Temporal-Difference Learning，克服了 MC 的必须实验到状态终结的问题 (方差大)，将 G_t 替换为 $R_{t+1}+\gamma v(S_{t+1})$ 即为 $TD(0)$，于是有

$$v\left(S_t\right) \leftarrow v\left(S_t\right)+\alpha\left(R_{t+1}+\gamma v\left(S_{t+1}\right)-v\left(S_t\right)\right) \tag{10}$$

其中 $R_{t+1}+\gamma v(S_{t+1})$ 叫 TD target，$\sigma_t = R_{t+1}+\gamma v(S_{t+1})-v(S_t)$ 叫 TD error。需要注意的是，$v(S_t)$ 利用了 $v(S_{t+1})$，bootstrapping 方式，属于自举。

容易想出 $TD(\infty)$ 即为 MC。他们的关系可用下图刻画：

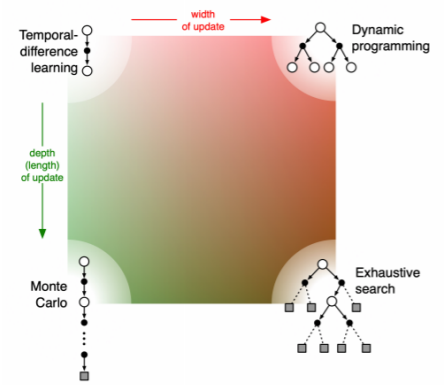


图 6: mcdptd

4.2 优化 (控制)

策略优化：价值函数迭代 +Arg Max 或策略迭代 (策略估计 $Q(s,a)$+策略改进)。

回想上节的策略迭代，是知道 $R(s,a), P(s'|s,a)$ 的，但这里未知，如何在采样的过程中进行策略改进？已有答案是 $\epsilon-Greedy$ 探索法。

$\epsilon-Greedy$ exploration 是指以 $1-\epsilon$ 的概率选择使当前动作价值函数最大的动作，以 ϵ 的概率选择其余动作，得到新策略：

$$\pi(a|s)=\begin{cases}\epsilon/|\mathcal{A}|+1-\epsilon & \text{if } a^*=\arg\max_{a\in\mathcal{A}} Q(s,a)\\ \epsilon/|\mathcal{A}| & \text{otherwise}\end{cases}$$

以这种方式改进策略，有如下定理保证：

定理 4.1 (Policy improvement theorem) *For any $\epsilon-greedy$ policy π, the $\epsilon-greedy$ policy π' with respect q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$.*

证明见 Sutton。

于是容易写出 $\epsilon - greedy$ 版本的 MC Exploration 算法流程：

Algorithm 1

Initialize $Q(S,A) = 0, N(S,A) = 0, \epsilon = 1, k = 1$
$\pi_k = \epsilon$-greedy(Q)
loop
  Sample k-th episode $(S_1, A_1, R_2, ..., S_T) \sim \pi_k$
  for each state S_t and action A_t in the episode **do**
    $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$
  end for
  $k \leftarrow k + 1, \epsilon \leftarrow 1/k$
  $\pi_k = \epsilon$-greedy(Q)
end loop

图 7: mcepsion

4.2.1 Sarsa

在 $\epsilon - greedy$ 策略改进中使用 TD 而不是 MC 即为 Sarsa 算法。这和在价值更新中将 MC 改进为 TD 是同样的道理，且在一定程度上，能减少函数值的方差。于是容易从(10)中写出 $Q(s,a)$ 函数版本 (值估计转化为策略优化)。$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$

综合以上，容易写出 one-step 的 Sarsa 算法流程：

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

图 8: sarsa

脑补一下 n-step 版本。

4.2.2 Q-Learning

Sarsa 明显是自举的 (bootstrapping)，其在下一步状态所做的动作仍由当前 $Q(s,a)$ 函数选出，在策略控制上其被分类为 On-Policy control，而本小节的 Q-Learning 为 Off-Policy control，其在下一个状态的动作选择不是由当前 Q 选出。

$$Q\left(S_t,A_t\right)\leftarrow Q\left(S_t,A_t\right)+\alpha\left[R_{t+1}+\gamma\max_a Q\left(S_{t+1},a\right)-Q\left(S_t,A_t\right)\right] \quad (11)$$

4.3 重要采样的应用

策略 π 在优化过程中，其形式可能会变得复杂，不好采样，于是根据(2.3)，可选择辅助策略 μ 来生成 episodes：$S_1,A_1,R_2,\ldots,S_T\sim\mu$，计算其 G_t。

若 Off-Policy 为 Monte Carlo，由(2.3)可得 G_{told} 和 G_{tnew} 的关系：

$$G_t^{\pi/\mu}=\frac{\pi\left(A_t|S_t\right)}{\mu\left(A_t|S_t\right)}\frac{\pi\left(A_{t+1}|S_{t+1}\right)}{\mu\left(A_{t+1}|S_{t+1}\right)}\cdots\frac{\pi\left(A_T|S_T\right)}{\mu\left(A_T|S_T\right)}G_t$$

于是在采样策略 μ 下的其价值更新变为：

$$V\left(S_t\right)\leftarrow V\left(S_t\right)+\alpha\left(G_t^{\pi/\mu}-V\left(S_t\right)\right)$$

若 Off-Policy 为 TD，容易得到其价值更新为：

$$V\left(S_t\right)\leftarrow V\left(S_t\right)+\alpha\left(\frac{\pi\left(A_t|S_t\right)}{\mu\left(A_t|S_t\right)}\left(R_{t+1}+\lambda V\left(S_{t+1}\right)\right)-V\left(S_t\right)\right) \quad (12)$$

问题：离线策略 Q-Learning 为何不能使用重要采样？

因为 Q-Learning 不需要在策略分布上做价值的期望估计。完整答案请看 这里。

4.4 DP 和 TD 的差异

Expected Update (DP)	Sample Update (TD)
Iterative Policy Evaluation $V(s) \leftarrow \mathbb{E}\left[R+\gamma V\left(S'\right)|s\right]$	TD Learning $V(S) \leftarrow^{\alpha} R+\gamma V\left(S'\right)$
Q-Policy Iteration $Q(S,A) \leftarrow \mathbb{E}\left[R+\gamma Q\left(S',A'\right)|s,a\right]$	Sarsa $Q(S,A) \leftarrow^{\alpha} R+\gamma Q\left(S',A'\right)$
Q-Value Iteration $Q(S,A) \leftarrow \mathbb{E}\left[R+\gamma \max_{a'\in\mathcal{A}} Q\left(S',A'\right)|s,a\right]$	Q-Learning $Q(S,A) \leftarrow^{\alpha} R+\gamma \max_{a'\in\mathcal{A}} Q\left(S',a'\right)$
where $x \leftarrow^{\alpha} y$ is defined as $x \leftarrow x+\alpha(y-x)$	

第五节　函数参数化

前面提出的一些值估计，策略改进的方法，但都是以统计为基础，在和环境交互的同时，搜集状态序列，计算统计量，进行价值，状态动作函数的更新 (表格式计算)。经典的例子 Cliff walk：4×16 个状态；Mountain car：1600 个状态；Tic-Tac-Toe：1000 个状态。但当面对西洋棋 (10^{47})，围棋 (10^{170})，器械臂、直升机 (连续状态) 等情况，就显得肌无力。使用带参数的函数，优化参数是可行的。数学上可表达如下：

$$\begin{aligned}\hat{v}(s,\mathbf{w}) &\approx v^{\pi}(s)\\ \hat{q}(s,a,\mathbf{w}) &\approx q^{\pi}(s,a)\\ \hat{\pi}(a,s,\mathbf{w}) &\approx \pi(a|s)\end{aligned} \tag{13}$$

这样做还有另一个好处，有了关于状态的具体函数，可计算所有状态的价值。

问题：怎么具体设计函数？参数如何更新？策略如何优化？本小节回答前两问，第三问见节6.1(当采用可微函数时)。

函数逼近的可选方案：

- 特征的线性组合
- 神经网络
- 决策树

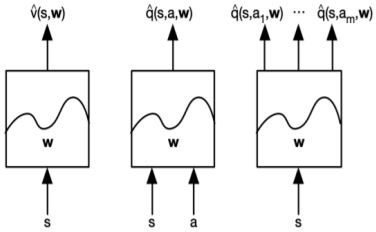


图 9: func design

- 最邻距离

这里只关心可微方式，输入输出参考9。

5.1　线性组合

对于特征的线性组合，若用 $\mathbf{x}(s) = (x_1(s), \ldots, x_n(s))^T$ 表示状态特征向量，则价值函数可表示为：$\hat{v}(s, \mathbf{w}) = \mathbf{x}(s)^T\mathbf{w} = \sum_{j=1}^{n} x_j(s)w_j$，若目标函数采用平方差，则优化目标：$J(\mathbf{w}) = \mathbb{E}_\pi \left[\left(v^\pi(s) - \mathbf{x}(s)^T\mathbf{w} \right)^2 \right]$ 其梯度更新：$\Delta\mathbf{w} = \alpha \left(v^\pi(s) - \hat{v}(s, \mathbf{w}) \right) \mathbf{x}(s)$ ，若把参数更新方式写成文字形式，有：

参数变化量 = 步长 × 预测误差 × 特征向量

数学抽象做完了，回到实际环境中，需要把理想的 $v^\pi(s)$ 替换回实际中的值。结合上一节的 MC,TD 更新方式，容易得到各自对应的更新版本。

对 MC: $\Delta\mathbf{w} = \alpha \left(G_t - \hat{v}(s_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{v} \left(s_t, \mathbf{w} \right)$

对 TD(0)：$\Delta\mathbf{w} = \alpha \left(R_{t+1} + \gamma\hat{v}(s_{t+1}, \mathbf{w}) - \hat{v}(s_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{v} \left(s_t, \mathbf{w} \right)$

在线性组合的方式下，

MC：$\Delta\mathbf{w} = \alpha \left(G_t - \hat{v}(s_t, \mathbf{w}) \right) \mathbf{x}(s_t)$

TD(0)：$\Delta\mathbf{w} = \alpha \left(R_{t+1} + \gamma\hat{v}(s_{t+1}, \mathbf{w}) - \hat{v}(s_t, \mathbf{w}) \right) \mathbf{x}(s_t)$

需要注意的是，上述梯度下降为 semi-gradient，因为其目标值中它自己。

同理可得，控制算法的更新方式：

MC：

$$\Delta\mathbf{w} = \alpha \left(G_t - \hat{q} \left(s_t, a_t, \mathbf{w} \right) \right) \nabla_{\mathbf{w}} \hat{q} \left(s_t, a_t, \mathbf{w} \right) \tag{14}$$

Sarsa：

$$\Delta\mathbf{w} = \alpha \left(R_{t+1} + \gamma\hat{q} \left(s_{t+1}, a_{t+1}, \mathbf{w} \right) - \hat{q} \left(s_t, a_t, \mathbf{w} \right) \right) \nabla_{\mathbf{w}} \hat{q} \left(s_t, a_t, \mathbf{w} \right) \tag{15}$$

Q-Learning：

$$\Delta \mathbf{w} = \alpha \left(R_{t+1} + \gamma \max_{a} \hat{q}\left(s_{t+1}, a, \mathbf{w}\right) - \hat{q}\left(s_t, a_t, \mathbf{w}\right) \right) \nabla_{\mathbf{w}} \hat{q}\left(s_t, a_t, \mathbf{w}\right) \quad (16)$$

Sarsa 的 VFA(Value Function Approximation) 控制算法流程：

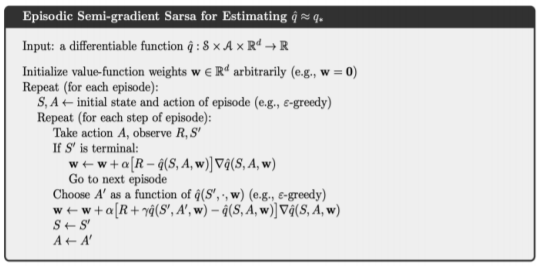


图 10: sarsa vfa

问题：参数逼近的控制算法收敛性如何？见下表：

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	√	(√)	X
Sarsa	√	(√)	X
Q-Learning	√	X	X

(√) moves around the near-optimal value function

图 11: converge vfa

5.2　DQN

特征线性组合的难点和以前 CV 手工设计特征类似，如何做出好的特征表示，如何对复杂问题进行有效的特征选取等问题。类似于 CNN 替换 SIFT，ORB 等特征提取方式，2015 年，DeepMind 发表了一篇名为“Human-level control through deep reinforcement learning”的文章，将 DL 引入了 RL，给出了一种有效的价值函数的非线性表达方法，同时不需要手工设计特征，并在 Breakout, Pong, Montezuma’s Revenge, Private Eye 四款游戏上达到了人类专家水平。随之而来的是 DQN 的各种升级，可参考 DQN 综述。

DQN 对 Atari Games 的建模12，从中可以看到基本思想就是：输入像素图，输出状态动作函数值。该建模方式有一些固有的坑，论文中对采样关

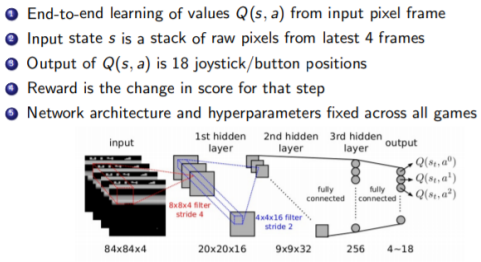


图 12: DQN Atari

联，目标值非平稳两个问题给出了解决方案，分别是经验池采样，固定目标函数。

经验池是指设定一个缓存区 $\mathcal{D}$，存放历史样本 (s,a,r,s')，并从中采样。固定目标函数是指，使用另外一组参数 $\mathbf{w}^-$ 来计算目标值 $r+\gamma\max_{a'}\hat{Q}(s',a',\mathbf{w}^-)$，这里样本来自 $\mathcal{D}$。最终参数更新量为：

$$\Delta\mathbf{w}=\alpha\left(r+\gamma\max_{a'}\hat{Q}\left(s',a',\mathbf{w}\right)-Q(s,a,\mathbf{w})\right)\nabla_{\mathbf{w}}\hat{Q}(s,a,\mathbf{w}) \tag{17}$$

第六节　策略参数化

6.1　基础建模

上一节提出了函数参数化，并给出了值函数的实现例子，这一节给出策略函数的实现方式。虽然 arg max 值函数可得到策略，但这种方式基本不能给出随机策略，实际情况，会有大量非完全确定的策略需求，比如在石头剪刀布游戏中，其最优策略 (纳什均衡点) 为随机策略，在一些带有闭环的探索游戏中，也需要随机策略来避免无限循环。而若将策略函数设置为动作的概率分布函数，则可实现这一点，同时有也可省去值函数。

设 $\pi_\theta(s,a)$ 是以 θ 为参数的策略函数，如何定义策略的质量？

在 episodic 环境中，可定义为：$J_1(\theta)=V^{\pi_\theta}(s_1)=\mathbb{E}_{\pi_\theta}[v_1]$，始状态的期望价值。在持续 (continuing) 环境中，可以有两种定义方式：利用平均价值有，$J_{avV}(\theta)=\sum_s d^{\pi_\theta}(s)V^{\pi_\theta}(s)$，利用一步时间段的平均回报值有，$J_{avR}(\theta)=\sum_s d^{\pi_\theta}(s)\sum_a\pi_\theta(s,a)R(s,a)$，其中 d^{π_θ} 是 π_θ 的马尔科夫平稳分布。直观来讲，最后一种更为方便，其对应优化目标可表示为：

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_t r\left(s_t, a_t^\tau\right)\right] \tag{18}$$

其中 τ 是策略函数 π_θ 下的一个采样 trajectory。

对于 $J(\theta)$ 的优化：若 $J(\theta)$ 可微，可用梯度下降，共轭梯度，拟-牛顿等方法，若 $J(\theta)$ 不可微，或倒数难算，可用采用 Cross-entropy method (CEM)，Hill climbing，Evolution algorithm 等。

在多步 MDP 模型中，状态-动作 trajectory 服从以下关系：

$$\tau = (s_0, a_0, r_1, \ldots, s_{T-1}, r_{T-1}, s_T) \sim (\pi_\theta, P(s_{t+1}|s_t, a_t)),$$

用 $R_\tau = \sum_{t=0}^T R(s_t, a_t)$ 表示一 trajector 的回报和。于是

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^T R\left(s_t, a_t\right)\right] = \sum_\tau P(\tau;\theta)R(\tau), \tag{19}$$

其中 $P(\tau;\theta) = \mu\left(s_0\right)\prod_{t=0}^{T-1}\pi_\theta\left(a_t|s_t\right)p\left(s_{t+1}|s_t, a_t\right)$ 表示策略 π_θ 下，该 trajectory 出现的概率。此时优化目标为：

$$\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta \sum_\tau P(\tau;\theta)R(\tau) \tag{20}$$

结合2.3的对数技巧，容易得出式(19)的梯度为：

$$\nabla_\theta J(\theta) = \sum_\tau P(\tau;\theta)R(\tau)\nabla_\theta \log P(\tau;\theta) \tag{21}$$

将 $\nabla_\theta \log P(\tau;\theta)$ 展开：

$$\begin{aligned}
\nabla_\theta \log P(\tau;\theta) &= \nabla_\theta \log\left[\mu\left(s_0\right)\prod_{t=0}^{T-1}\pi_\theta\left(a_t|s_t\right)p\left(s_{t+1}|s_t, a_t\right)\right] \\
&= \nabla_\theta\left[\log\mu\left(s_0\right) + \sum_{t=0}^{T-1}\log\pi_\theta\left(a_t|s_t\right) + \log p\left(s_{t+1}|s_t, a_t\right)\right] \\
&= \sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta\left(a_t|s_t\right)
\end{aligned} \tag{22}$$

于是多步 MDP 的策略梯度最终表示为：

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{i=1}^m R\left(\tau_i\right)\sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta\left(a_t^i|s_t^i\right) \tag{23}$$

最终的梯度公式可以做一个直观认识：对于权重函数的期望 (积分)，其梯度方向为原始概率的对数的导数乘以对应权重 (注意这里是一个 trajectory)。有了这一直观认识，我们可以猜一猜其他各种形式的策略优化函数的梯度。

当权重值为 G_t(在一个 trajectory，从时间 t 开始获得的奖励) 时：

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} G_t \cdot \nabla_\theta \log \pi_\theta\left(a_t|s_t\right)\right] \tag{24}$$

当权重值为 $Q_w(s_t, a_t)$ 时：

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} Q_w(s_t, a_t) \cdot \nabla_\theta \log \pi_\theta\left(a_t|s_t\right)\right] \tag{25}$$

验证：见周博磊讲义第 5 章。也就是说以上猜测确为事实。

不过在一个 trajectory 中，G_t 往往方差较大，如何做到减小其方差，但保持其期望不变？答案是减去回报值的期望。

基准值 $b(s_t) = \mathbb{E}[r_t + r_{t+1} + \ldots + r_{T-1}]$，有 $\mathbb{E}_\tau[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t))] = 0$，并且

$$\begin{aligned} E_\tau\left[\nabla_\theta \log \pi_\theta\left(a_t|s_t\right)\left(G_t - b\left(s_t\right)\right)\right] &= E_\tau\left[\nabla_\theta \log \pi_\theta\left(a_t|s_t\right) G_t\right] \\ \operatorname{Var}_\tau\left[\nabla_\theta \log \pi_\theta\left(a_t|s_t\right)\left(G_t - b\left(s_t\right)\right)\right] &< \operatorname{Var}_\tau\left[\nabla_\theta \log \pi_\theta\left(a_t|s_t\right) G_t\right] \end{aligned} \tag{26}$$

于是优化函数变为：

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \left(G_t - b_w(s_t)\right) \cdot \nabla_\theta \log \pi_\theta\left(a_t|s_t\right)\right] \tag{27}$$

如法炮制，(25)也可做减基准值的改动。不过这里还是回到 RL 原始的发展顺序来。在教材中，(25)被称为 Actor-Critic Policy Gradient，原因在于 $Q_w(s, a)$ 担任了 Actor 的角色，$\pi_\theta(a|s)$ 扮演了 Critic 角色，他们各自有各自的参数，这和生成模型 GAN 异曲同工。

6.1.1　演员评论家

当用线性价值函数来逼近 Actor 时：$Q_{\mathbf{w}}(s, a) = \phi(s, a)^T \mathbf{w}$。此时 Critic 由线性的 $TD(0)$ 更新，Actor 由策略梯度更新。一个简单版本见13。

上面提到的 Actor-Critic 算法的减基改动，怎么做？回想一下策略 π 下的 Q, V 的定义是什么。$Q^{\pi,\gamma}(s, a) = \mathbb{E}_\pi\left[r_1 + \gamma r_2 + \ldots | s_1 = s, a_1 = a\right]$

Algorithm 2 Simple QAC

1: **for** each step **do**
2: generate sample s, a, r, s', a' following π_θ
3: $\delta = r + \gamma Q_{\mathbf{w}}(s', a') - Q_{\mathbf{w}}(s, a)$ #TD error
4: $\mathbf{w} \leftarrow \mathbf{w} + \beta\delta\psi(s, a)$
5: $\theta \leftarrow \theta + \alpha\nabla_\theta \log \pi_\theta(s, a) Q_{\mathbf{w}}(s, a)$
6: **end for**

图 13: QAC

$V^{\pi,\gamma}(s) = \mathbb{E}_\pi [r_1 + \gamma r_2 + \ldots | s_1 = s] = \mathbb{E}_{a\sim\pi}[Q^{\pi,\gamma}(s, a)]$ 因状态价值函数是状态-动作价值函数的无偏估计，因此，只需做 $Q \leftarrow Q - V$(按算法更新写法) 即可。于是我们得到一个重要的函数：优势函数。

定义 6.1 (Advantage function)

$$A^{\pi,\gamma}(s, a) = Q^{\pi,\gamma}(s, a) - V^{\pi,\gamma}(s)$$

其对对应的策略梯度为：

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1}(Q_w(s_t, a_t) - V(s_t)) \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1}(A_w(s_t, a_t) \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)\right] \end{aligned} \tag{28}$$

以上就是基本的策略梯度算法。那么策略梯度有什么问题？我想应该需要更详细的推导，分析，以及实验了。关于策略梯度能克服不可微操作，可参考PG overcome the non-differentiable computation。

6.2　策略梯度的改进

不同算法的优缺点，除了理论推导，实际实验情况也非常重要。

6.2.1 Policy Gradient→TRPO→ACKTR→PPO

在High-Dimensional Continuous Control Using Generalized Advantage Estimation一文中，可以看到，策略梯度可以有多种：

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) G_t\right] - \text{ REINFORCE} \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) Q^{\mathrm{w}}(s,a)\right] - \text{Q Actor-Critic} \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) A^{\mathrm{w}}(s,a)\right] - \text{ Advantage Actor-Critic} \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) \delta\right] - \text{ TD Actor-Critic}\end{aligned} \tag{29}$$

前三种已经见过了，对于第四种，其和优势函数的关系，可能并不能一眼看出。其实我们有如下结论：

推论 6.1 设价值函数,TD 误差分别为 $V^{\pi_\theta(s)}$,$\delta^{\pi_\theta} = r(s,a)+\gamma V^{\pi_\theta}(s')-V^{\pi_\theta}(s)$，则 $\mathbb{E}_{\pi_\theta}[\delta^{\pi_\theta}|s,a] = A^{\pi_\theta}(s,a)$。

根据(4)，即可证明。

策略梯度的问题：

- 因为在线学习的原因，采样效率不高
- 策略更新过大或者步长不合理会导致训练崩溃
  - 和强监督不同这里更新的数据不是独立的
  - 步长太大，导致策略不好，不好的策略搜集了更差的数据，恶性循环严重
  - 很难从坏策略中逃出来，大概率致使模型崩塌

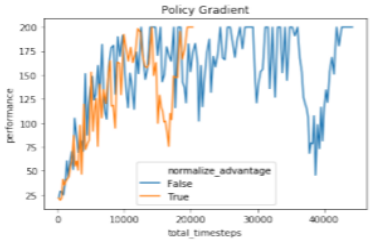


图 14: traning unstatble

为了解决训练不稳定，在线更新问题，John Schulman 等人在 TRPO 中提出了用置信区间和自然梯度下降来克服训练不稳定问题，并顺势而为，用重要采样将在线改为离线。

核心思想就是限制策略更新前后的差异。因为策略是概率函数，于是可用 KL 散度来限制。

$$d^* = \arg\max J(\theta + d),\ \text{s.t.}\ KL\left[\pi_\theta \| \pi_{\theta+d}\right] = c$$

将重要采样用上则为：

$$J_{\theta_{\text{old}}}(\theta) = \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} R_t\right]$$
$$\text{subject to}\ KL\left(\pi_{\theta_{\text{old}}}(.|s_t)\,||\,\pi_\theta(.|s_t)\right) \le \delta$$

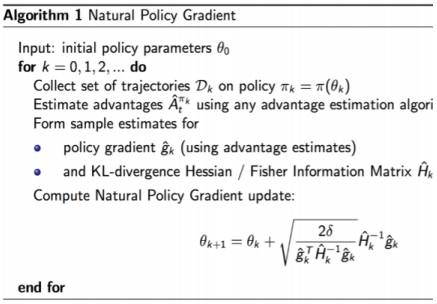
经过一些计算，得到更新方式：

$$\theta_{t+1} = \theta_t + \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g$$

其中

$$H = \nabla_\theta^2 KL\left(\pi_{\theta_t} \| \pi_\theta\right) = E_{a,s\sim\pi_{\theta_t}}\left[\nabla_\theta \log \pi_\theta(a,s) \nabla_\theta \log \pi_\theta(a,s)^T\right]$$

综合以上，TRPO 算法的自然梯度下降算法流程：

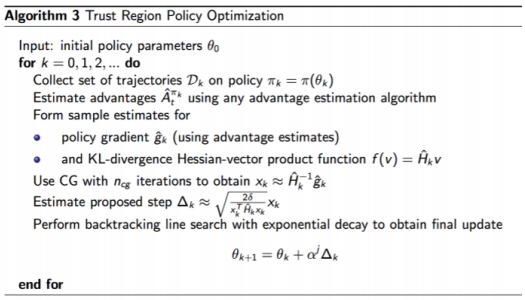

Algorithm 1 Natural Policy Gradient

Input: initial policy parameters θ_0
for $k = 0, 1, 2, \ldots$ **do**
Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algori
Form sample estimates for
- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian / Fisher Information Matrix $\hat{H}_k$

Compute Natural Policy Gradient update:

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{H}_k^{-1} \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$$

end for


Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0
for $k = 0, 1, 2, \ldots$ **do**
Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Form sample estimates for
- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$
Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$
Perform backtracking line search with exponential decay to obtain final update

$$\theta_{k+1} = \theta_k + \alpha^j \Delta_k$$

end for

图 15: Nature TRPO　　图 16: Conjugate Nature TRPO

将 Fisher Information Matrix 的逆用共轭算法实现的算法流程见16。

而 ACKTR 则对 FIM 的逆的计算做了进一步改进 (使用了矩阵的分块计算)。

PPO 做了两点改进，第一将合并了 TRPO 的限制条件和函数主体：

$$\text{maximize}_\theta\ \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t\right] - \beta \mathbb{E}_t\left[KL\left[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right]\right]$$

合并的好处是省略了二阶导数的计算，完全转化为一阶导数，时间上更快。

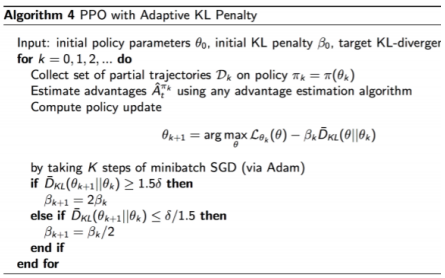
在前后策略差异的限制上，δ 也做了动态调整，见17。

第二对新旧策略和优势函数上做了一些简单粗暴的限制。具体如下：

$$L_t(\theta) = \min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)$$

直观上理解，当新旧策略的比率在 $(1-\epsilon, 1+\epsilon)$ 之外时，优势函数将按如上截取。也就是说要当新旧策略差距大时，对策略函数做惩罚。

算法流程如下：



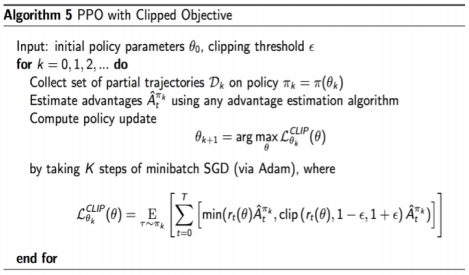
Algorithm 4 PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-diverger
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta||\theta_k)$$

  by taking K steps of minibatch SGD (via Adam)
  if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  else if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \leq \delta/1.5$ **then**
    $\beta_{k+1} = \beta_k/2$
  end if
end for

图 17: PPO KL Penalty



Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

  by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for

图 18: PPO Clip

6.2.2 Q-learning→DDPG→TD3→SAC

- SAC

第七节 Model Based

参考基于模型的策略优化。

第八节 模仿学习

参考模仿学习

第九节 分布式系统

后续也许着重了解。

第十节　多智能体

- 一
- 二

第十一节　Code

参考 code 目录，或者

- tianshou
- stable-baselines3
- Hands-on-RL版本老，目录 Code 部分是对其的更新
- raylib

第十二节　RLHF

- 包含 ChatGPT Reward Model 训练流程