强化学习笔记

Why Reinforcement Learning

背景

多学科交叉，共同解决 decision making 问题

计算机科学：机器学习算法

工程：决定序列行为

神经科学：反馈系统

经济学：博弈论

原理

- 序列决策问题，选择行为，得到最大收益
- 无标签，尝试并反馈
- 通过反馈调节

与其它机器学习区别

- 在 exploration 与 exploitation 间权衡
- 从目标为导向的智能体和不确定的环境之间进行相互作用的整个问题

和 supervised learing 区别

- There is no supervised, only a reward signal
- Feedback is delayed, not instantaneous
- Agent's action affect the subsequent data it receives

Date / /

- Time really matters

和 ~~unsu~~ unsupervised learning 的区别

- UL 挖掘无标签数据之间的内部关联
  RL 最大化 reward

奖励 Rewards

核心问题

- R_t 反馈信号
- 时刻 t 行为好坏
- Agent 目的 最大化累积回报
- RL 奖励假设

  All goals can be described by the maximisation of expected cumulative reward

示例

无人机 / 无人车

+ : 按轨迹行行

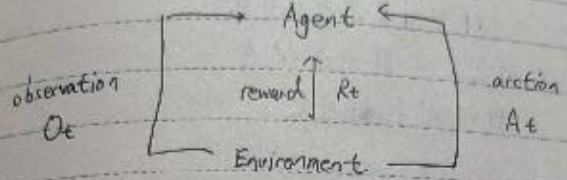
− : 翻车

序列决策 Sequential Decision Making

- 目的: 最大化 累积回报
- 动作可能有长期后果
- 奖励有延迟
- 牺牲立即回报获得长期回报

Agent and Environment

- Agent 需优化部分
- Environment 无法控制
- 区分



对 Agent

1. 执行动作 A_t
2. 接收观察 O_t
3. 接收常量回报 R_t

对 Environment

1. 接收行为 A_t 并反应

History and State

- History is 一串 O、A、R

  $H_t = O_1 R_1 A_1, \cdots, O_t R_t A_t$

- 下一步发生什么取决于历史

  Agent 所选 Action

  Environment 选 observations / rewards

- State 是历史的一种表述
- 本质上 $S_t = f(H_t)$

Date / /

环境状态

1. S_t^e 是环境的内部表达
   包括用来决定下一个观测或收益的所有数据，通常对个体并不完全可见
2. 可包含一些无关信息
   有时对个体可见

智能体状态

1. S_t^a 是智能体的内部表达
2. 它是强化学习可以利用的信息
   可是历史的函数 $S_t^a = f(H_t)$

示例

机器人控制：

环境状态：所有零件参数状态

智能体状态：传感器所获数据

信息状态 Information State

· 包括历史上所有有用信息

Markov Decision Processes

马尔可夫性

- 时刻 t 的状态 S_t 满足

$$P[S_{t+1} | S_t] = P[S_{t+1} | S_1, \cdots S_t]$$

- 状态 S_t 包含所有历史相关信息

示例

下棋只关心当前局面

状态转移矩阵

$$P_{ss'} = P[S_{t+1} = s' | S_t = s]$$

$$P = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

其中：n 为状态个数

每行元素相加等于 1

片段 episode

从初始状态 S_1 到终止状态 S_T 的序列过程

如任务总从终止状态结束，为 episodic task

无终止状态，continuing task

马尔可夫过程 Markov Process，MP

- 满足马尔可夫性的随机过程
- 无记忆

Date / /

马尔可夫链 Markov Chain

状态空间为可数集的马尔可夫过程

生成模式

确定性模式

非确定性模式

n阶马尔可夫模型

· 状态间的转移仅依赖于前n个状态的过程

隐藏模式

马尔可夫决策过程 MDP

马尔可夫奖励过程 MRP

· 带有 values 的 Markov Chain

一个数组 $\{S, P, R, \gamma\}$

S是有限的状态集

P是状态转移矩阵

R是奖励函数

γ是衰减系数

G_t 是从时间序列 t 开始的所有折扣回报

连续性任务 $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

片段性任务 $G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$

状态价值函数 $v(s)$ 是从状态 s 开始的期望回报

$v(s) = E[G_t | S_t = s]$

贝尔曼方程

$$v(s) = E[R_{t+1} + \gamma G_{t+1} | S_t = s]$$

矩阵形式和求解

$$v = R + \gamma P v$$

线性方程

$$v = (I - \gamma P)^{-1} R$$

MDPs

$\{S, A, P, R, \gamma\}$

- A 是有限的动作集

策略 Policy

策略 π 是概率的集合或分布

$$\pi(a|s) = P[A_t = a | S_t = s]$$

- 定义了个体的行为方式
- 是 RL 问题的终极目标
- 仅与当前状态有关
- 每一个确定的 Policy 是静态
- 个体可以随着时间更新策略

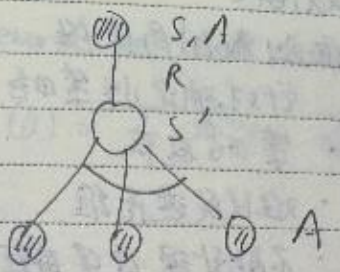
给定一个 MDP $M = \{S, A, P, R, \gamma\}$ 和策略 π

那么状态序列 $S_1, S_2 \cdots$ 是一个马尔科夫过程 $\{S, P^\pi\}$

状态和奖励序列 $S_1, R_2, S_2 \cdots$ 是一个 MRP

$\{S, P^\pi, R^\pi, \gamma\}$

拓扑图



算法：

初始化 $Q(S,a), \forall s \in S, a \in A(S)$，且

Q(终止状态，·) $=0$

repeat (对于每个片片)

初始化状态 S

repeat (对于片段中的每一步)

根据 Q 选择一个在 S 状态的动作 A

执行动作 A，观测 R, S'

$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma \max_a Q(S',a) - Q(S,A))$

$S \leftarrow S'$

until S 是终止状态

until 收敛

Policy Gradient

基于值函数的局限性

- 针对确定性策略
- 策略退化
- 难以处理高维
- 不能处理连续空间

策略梯度定理

策略梯度目标函数

用参数 θ 建模策略 $\pi_\theta(s,a)$,

如何寻找最优 θ

Start value

$$J_1(\theta) = V^{\pi_\theta}(s_1) = E_{\pi_\theta}[v_1]$$

Average Value

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

↑

基于当前策略下马尔科夫链关于状态的静态分布

Average reward per time-step

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) R_s^a$$

数值法求梯度

· 对于 θ 的每一个维度 $k \in [0,1]$

通过给 θ 的第 k 维加入一点

扰动 ε,

然后估计对第 k 维的偏导数

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \varepsilon u_k) - J(\theta)}{\varepsilon}$$

1. 每次求 θ 的梯度要算 n 次
2. 简单噪声大 效率低

策略梯度算法

出发点

1. 找到一种合适的目标函数 J，满足
   · 最大化 J 相当于最大化期望回报
   · 能建立 $\nabla_\theta J$ 与 $\nabla_\theta \pi_\theta$ 的关系

策略梯度的推导

轨迹：一个样件 τ

轨迹的回报：$R(\tau) = \sum_{t=1}^{T} r_t R(s_t, a_t)$

$p_\theta(\tau)$ 表示 轨迹 τ 出现的概率

$R(\tau)$ 的期望

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

目的 让 $R(\tau)$ 越大越好

所以强化学习的目标函数

$$U(\theta) = \sum_\tau R(\tau) p_\theta(\tau)$$

求解 $\nabla_\theta U(\theta)$

两种角度：

从似然的角度

$$\nabla_\theta U(\theta) = \nabla_\theta \sum_\tau p_\theta(\tau) R(\tau)$$

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

利用当前策略 π_θ 采样 m 条轨迹

$$\nabla_\theta U(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log p_\theta(\tau) R(\tau)$$

从重要性采样的角度：

$$\nabla_\theta U(\theta) = E_{\tau \sim p_{old}}\left[\frac{\nabla_\theta p_\theta(\tau)}{p_{old}(\tau)} R(\tau)\right]$$

减小方差

1. 引入基线
2. 修改回报函数
3. Actor-Critic 方法
4. 优势函数

Actor-Critic

实际更新算法

考虑单条轨迹

$$\hat{g} = \sum_{t=0}^{T} \nabla_\theta \log p_\theta(a_t|s_t)\left(\sum_{k=t}^{T} \gamma^{k-t} R(s_k, a_k)\right)$$

考虑单步更新

$$\hat{g}_t = \nabla_\theta \log p_\theta(a_t|s_t)\left(\sum_{k=t}^{T} \gamma^{k-t} R(s_k, a_k)\right)$$

MC 策略梯度

初始化 θ

for 每条轨迹 $\sim \pi_\theta$ do

for $t=0$ to T do

$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) g_t$

end do

end for

Date / /

Actor - Critic 算法 维持两个参数

Critic 更新 Q 函数的参数 w

Actor 使用 Critic 的方向更新 θ

$$\Delta\theta = \alpha \nabla_\theta \log p_\theta(a|s) Q_w(s,a)$$

$$Q_w(s_t, a_t) \approx \sum_{k=t}^{T} (\gamma^{k-t} R(s_k, a_k))$$

带资格迹的策略梯度

前向视角 TD(λ)

用 λ 回报值去估计优势函数

$$\Delta\theta = \alpha (G_t^\lambda - V_v(s_t)) \nabla_\theta \log p_\theta(a_t|s_t)$$

后向视角 TD(λ)

$$\delta = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)$$

$$e_t = \lambda e_{t-1} + \nabla_\theta \log p_\theta(a_t|s_t)$$

$$\Delta\theta = \alpha \delta e_t$$

PPO:

Promimal Policy Optimization (KL Penalty)

PPO 原理为在目标函数后加一个约束 KL 散度 $\beta KL(\theta, \theta')$，来保证 θ 和 θ' 的相似性

Date: / /

奖励函数

$$R_s^\pi = \sum_{a \in A} \pi(a|s) R_s^a$$

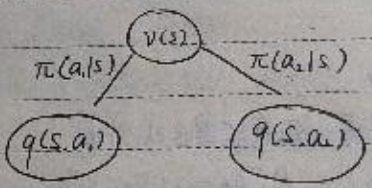
基于策略 π 的价值函数

$$v_\pi(s) = E_\pi[G_t | S_t = s]$$

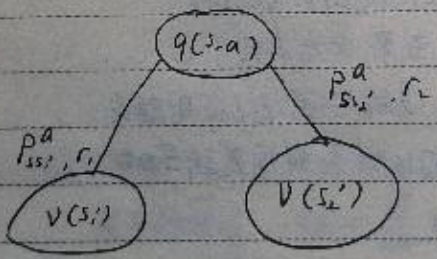
行为价值函数

$$q_\pi(s,a) = E_\pi[G_t | S_t = s, A_t = a]$$

Bellman 期望方程



$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$$



$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

最优价值函数 $v_*(s)$ $q_*(s,a)$

指在从所有策略产生的状态价值函数中选取使状态 s 价值最大的

$$v_*(s) = \max_\pi v_\pi(s)$$

最优行为价值函数 $q_*(s,a)$

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

最优策略

定理

存在一个最优策略 π_* 比其他任何策略更好或相等

有相同的最优价值函数

有相同的行为价值函数

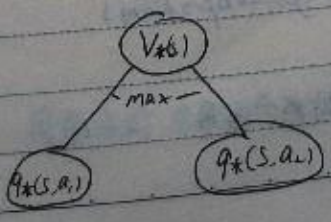
寻找最优策略

最大化 $q_*(s,a)$

$$\pi_*(a|s) = \begin{cases} 1, & \text{if } a = \underset{a \in A}{\operatorname{argmax}}\, q_*(s,a) \\ 0, & \text{otherwise} \end{cases}$$

Bellman 最优方程

$$v_*(s) = \max_a q_*(s,a)$$



Date / /

$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_*(s')$$

Bellman 最优方程 与 Bellman 方程 的差异

- 利用 π_* 的特点，将求期望的等式转化为 max
- π 已知 π_* 未知
- 对应 优化 对应 评价

求解 Bellman 最优方程

- 非线性
- 迭代方法：价值迭代
  策略迭代
  Q学习
  Sarsa 等

动态规划 Dynamic Programming DP

动态：

指的是该问题的时间维度部分

规划：

指去优化一个策略

动态规划方法需包含两个性质

- Optimal Structure
- Overlapping subproblems

MDP 满足上面两个性质

贝尔曼方程是递归的形式，把问题分成子问题
值函数保存和重用问题的解

预测

input：

MDP $<S, A, P, R, \gamma>$ 和策略 π or MRP $<S, P^{\pi}, R^{\pi}, \gamma>$

output：

价值函数 V_{π}

控制

input： MDP $<S, A, P, R, \gamma>$

output： 最优价值函数 V_* 和最优策略 π_*

迭代法策略评估

给定一个策略 π，求对应的值函数 $V_{\pi}(s)$

解决方案

直接解：$V_{\pi} = (I - \gamma P^{\pi})^{-1} R^{\pi}$，可以求闭合解，时间复杂度 $O(n^3)$

迭代解：应用 Bellman 期望方程

$$v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

具体方法：同步反向迭代（synchronous backups）

Date

对于第 k+1 次迭代，

所有状态 s 的价值用 $V_k(s')$ 计算，

并更新该状态第 k+1 次迭代中使用的价值 $V_k(s)$

for k=1, 2, ... do

for 所有的状态 $s \in S$ do

使用迭代求更新值函数 $V_{k+1}(s)$

end for

end do

$$V_\pi(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \cdot V_\pi(s') \right)$$

$$V_{k+1}(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \cdot V_k(s') \right)$$

$$V^{k+1} = R^\pi + \gamma P^\pi V^k$$

如何改善策略

给定一个策略下的迭代更新价值函数

贪婪的选取动作，

使得后继状态价值增加最多

$$\pi' = greedy(V_\pi) \Leftrightarrow a' = \arg\max_a q_\pi(s, a)$$

策略改善

策略评价：求 V_π。使用方法：迭代式策略提升

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

伪代码：

for $i \in \{1, \dots, N\}$ do

Run policy π_θ for T timesteps, collecting $\{s_t, a_t, r_t\}$

Estimate advantages

$\pi_{old} \leftarrow \pi_\theta$

for $j \in \{1, \dots, M\}$ do

$$J_{PPO}(\theta) = \sum_{t=1}^{T} \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}\hat{A}_t - \lambda KL[\pi_{old}|\pi_\theta]$$

Update θ by a gradient method wrt $J_{PPO}(\theta)$

end for

for $j \in \{1, \dots, B\}$ do

$$L_{BL}(\phi) = -\sum_{t=1}^{T}\left(\sum_{t'>t} r^{t'-t} r_{t'} - V_\phi(s_t)\right)^2$$

Update ϕ by a gradient method wrt $L_{BL}(\phi)$

end for

if $KL[\pi_{old}|\pi_\theta] > \beta_{high} KL_{target}$ then

$\lambda \leftarrow \alpha\lambda$

else if $KL[\pi_{old}|\pi_\theta] < \beta_{low} KL_{target}$ then

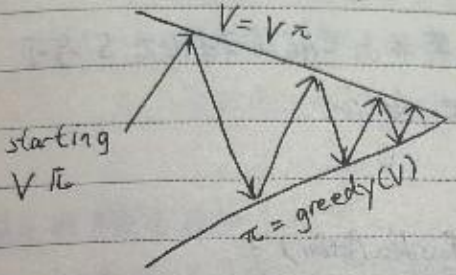
$\lambda \leftarrow \lambda/\alpha$

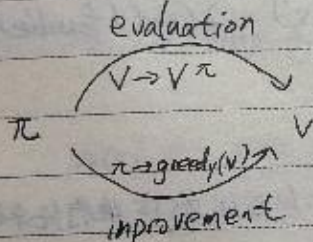
end if

end for

策略提升：

提升策略 $\pi' \geq \pi$ 。使用方法：多步策略提升





本质上是使用当前策略产生新样本，
然后使用新的样本更好的估计策略的价值
利用策略的价值更新策略

Policy - Evaluation + Policy - Improvement

Policy - Improvement 将已有的动作选择策略 $\pi(S,A)$ 和 V 矩阵带入与最优值做比较，从而将 $\pi(S,A)$ 更新为最优

π

Policy - Improvement

1. 初始化Q矩阵，将 V矩阵与 π(S,A) 带入状态循环

↓

2. 带入动作集合计算出可能的下一状态 S' 与可执行的 Possible Action

↓

3. 用式子 $Q(S, PossibleAction) = R_1(S') + R_2(S') - 2(cost(PossibleAction) + \gamma V(s'))$

计算Q矩阵

↓

4. 用策略 π(S,A) 与 Qmax 所在的动作进行比较，若是不符合则令一个 flag：Policy-stable-False

Policy-Iteration Evaluation + Policy-Improvement

1. 计算奖励矩阵，为初始化Q矩阵与V矩阵

↓

2. 判断 Policy-Stable 是否为 False，如为 True 则输出结果 π(S,A)，如为 ~~True~~ False 则进入迭代循环过程

↓

3. 令 Policy_Stable = True

4. 执行 Policy-Evaluation 算法

5. 执行 Policy_Improvement 算法，得到 Policy_stable 的结果返回第2步

Date : :

最优化原则 Principle of Optimality

可分解为两部分

1. 从状态 s 到后继状态 s' 并采取了最优的初始动作 A_*.
2. 在后继状态 s' 开始为最优策略进行

确定性的价值迭代

知道子问题 $V_k(s')$ 的解.

于是 one-step lookhead 就可得到 $V_{k+1}(s)$ 的解

$$V_{k+1}(s) \leftarrow \max_{a \in A} \left[R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_k(s') \right]$$

例子 - 最短路径 Shortest Path

问题:

g	A	B	C
D	E	F	G
H	I	J	K
L	M	N	O

g点的 reward = 0, 其他的状态 reward = -1

解题过程

- 首先初始化每个状态 (即 A 到 O) 的 V 都为 0
- 接着进行第一次迭代 (即 k=1, 此时根据公式)

k=1 时的迭代为:

$$V_2(s) = \max_a \left[R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_1(s') \right]$$

显然，因为 $\forall s \in S$: $V_1(s) = 0$ 即

所以 max 都是一样的，导致 $\forall s \in S$: $V_2(s) = -1$

接着进行 $k=2$ 时的迭代

值迭代 Value Iteration

问题：寻找最优策略元

解决方案：从初始状态价值开始同步迭代计算

最终收敛，最终过程中没有遗漏

任何策略元

同步备份下的值迭代算法

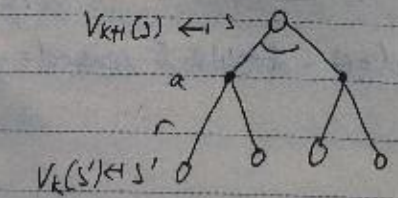
for $k = 1, 2, \cdots$ do

for 所有的状态 $s \in S$ do

通过 $V_k(s')$ 更新 $V_{k+1}(s)$

end for

end for



$$V_{k+1}(s) = \max_{a \in A} \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_k(s') \right)$$

异步动态规划 Asynchronous Dynamic Programming

减少计算量，不迭如果想收敛

则需要满足一条件：

所有状态都需要能被持续选择

异步动态规划主要有三个 idea

1. In-place dynamic programming 就地动态规划
2. Prioritised sweeping 优先清理
3. Real-time dynamic programming

实时动态规划

全宽备份 Full-Width Backup

对每次 Backup，

需所有的后继状态和动作要被考虑在内。

维数灾难

↓

样本备份 Sample Backup

优点：

- Model-free 无模型，无需知道 P 和 R
- 避免维数灾难
- backup 的时间复杂度固定与状态数无关

1. 强化学习中主要使用样本备份
2. 直接通过采样得到转移记录
3. 通过采样代替矩阵

Monte-Carlo Reinforcement Learning

· 不基于模型 model free

在不清楚MDP的状态转移概率和即时奖励的情况下，直接从经历完整的episode来学习状态价值

完整的episode里包含的信息有：

状态的转移

使用的行为序列

中间状态获得的即时奖励

到达终止状态时的即时奖励

All episodes must terminate

在有限时间内到达终点并获得回报

value = mean return

将状态s在每一个样本中获得的回报的值

分为：

first visit

every visit

Date / /

Monte-Carlo Policy evaluation

目标：

在无下，

从一系列完整的 episode 经历中，

学习得该策略下的状态价值函 V_π

First-Visit Monte-Carlo Policy Evaluation

对于每一个 episode，仅当该状态 S 首次出现的时间七列入计算

状态出现的次数加1：$N(s) \leftarrow N(s)+1$

总的收获值更新：$S(s) \leftarrow S(s)+G_t$

状态 S 的价值：$V(s) = S(s)/N(s)$

当 $N(s) \to \infty$ 时，$V(s) \to V_\pi(s)$

Every-visit Monte-Carlo Policy Evaluation

对于每一个 Episode，~~每当~~ 状态 S 每次出现都要用于计算 S 对应的值

Incremental Mean

每得到一次收获，就计算其平均收获

每次收获为 x_j

k 次的平均收获为 $\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j$

k-1 次的平均收获为 $\mu_{k-1} = \frac{1}{k-1}\sum_{j=1}^{k-1} x_j$

Date / /

Incremental Monte-Carlo Updates

对于 episode 里的每个状态 S_t，有一个收获 G_t，每次要计到一个 S_t，计算状态的平均价值 $V(S_t)$

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

静态与非静态问题

非静态问题，

将 MC 方法变为增量式，

去掉记计数值 $N(S_t)$

引入 α 来更新状态价值

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

时序差分学习

Temporal-Difference Learning

- 不需了解模型参数
- 可以学习不完整的 episode
- 通过合理的 bootstrapping，先估计某状态在该状态序列完整后可能得到的收益，并在此上利用前文方法

TD学习中，

算法在估计某一状态的收获时，

用离开该状态的即时奖励 R_{t+1}

与下一时刻状态每 S_{t+1} 的预估 $V(S_{t+1})$

乘以衰减系数 γ 组成

$$V(S_t) \leftarrow V(S_t) + \alpha\underbrace{(\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{TD target}} - V(S_t))}_{\text{TD error } \delta_t}$$

基于表格的 TD(0) 策略评价算法

repeat (对于每个片段)

初始化状态 S

repeat (对于片段中每一步)

通过 $\pi(\cdot|s)$ 采样 A

执行动作 A，观测 R，S'

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

until S 是终止状态

until 收敛

对于 TD，agent 每走一步它都可以更新一次

Date / /

对比分析 DP 和 TD

DP 利用了贝尔曼方程

$V(s) \leftarrow E[R + \gamma V(S') | s]$

TD 同样，但有改动

- 全宽备份 → 样本备份
- 增加学习率

MC 与 TD 优缺点

1. MC 对比 TD 之一

TD 可在知道结果前学习，MC 必须等到结果

TD 可在持续进行的环境中学习

TD 有多个驱动力

2. MC 对于 TD 之二

MC：零偏差；

高方差；

收敛性较好；

对初始值不敏感

随样本数量增加，方差近→0

TD：有偏差；

低方差；

对模初始值更敏感

通常比 MC 有效，并未随着增加 偏差→0

Maximum entropy objective function

$$\pi^* = \arg\max_{\pi} \sum_{s\in S, a\in A} E[r + \alpha H(\pi(a|s))]$$

Soft Actor-Critic

提出了 entropy regularization 的思想

$$H(P) = E_{x\sim P}[-\log P(x)]$$

让 expected return 与 entropy 之间做一个权衡

第一部分为 expected return

第二部分为 entropy

$$\pi^* = \arg\max E_{\tau\sim\pi}[\sum \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot|s_t)))]$$

$$V^\pi(s) = E_{\tau\sim\pi}[\sum \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot|s_t)))| s_0 = s]$$

$$Q^\pi(s,a) = E_{s'\sim P, a'\sim\pi}[R(s,a,s') + \gamma(Q^\pi(s',a') - \alpha \log \pi(a'|s'))]$$

Date / /

MC对TD之三

TD 在 Markov 环境更有效

MC 在非 Markov 环境下更有效

n-步预测

从当前状态 S_t 开始往后到终止为止

观察当状态 S_{t+n-1}，使用这n个

状态产生的即时奖励以及 S_{t+n}

的预估价值来计算当前 S_t 的值

n步收获

$n=1$: $G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$ TD(0)

$n=2$: $G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$

...

$n=\infty$ $G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$ MC

n步TD学习状态价值函数的更新公式为

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

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

n步 TD 策略评价

repeat (对于每一个片段)

reapeat 对于片段中的每一步

根据 $\pi(\cdot|S_t)$ 选择动作 A_t

执行动作 A_t,

观察到 R_{t+1}, S_{t+1} 将其存储

if $\tau = t-n+1 \geq 0$ then

$$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$$

if $\tau+n < T$,

then $G \leftarrow G + \gamma^n V(S_{\tau+n})$

$V(S_\tau) \leftarrow V(S_\tau) + \alpha[G - V(S_\tau)]$

end if

untill 直到终止状态

untill 收敛

将n步回报值平均

不同的n下的n步回报值效果不同。

做加权平均，构成一个有效回报值。

引入参数：λ

λ 收获

G_t^λ 表示从1到∞时所有步收获

对任意n步，加权重 $(1-\lambda)\lambda^{n-1}$

Date / /

公式：

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$$

入预测

写成 TD(λ)：$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{(n)} - V(S_t))$

n-step TD：$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{(n)} - V(S_t))$

前向*认识 TD(λ)

要更新一个状态的价值 $V(S_t)$

必须完整走完整个 episode

有与 MC 方法一样的劣势

两个概念

1. 频率启发
2. 就近启发

引入：资格迹 Eligibility Traces，ES

定义：

$E_0(S) = 0$

$E_t(S) = \gamma\lambda E_{t-1}(S) + 1(S_t = S)$

基于信任分配问题

Date / /

向后视角认识 TD(λ)

设当前状态为 S_t，TD 偏差为 δ_t，

S_{t-1} 处的值函数更新应乘

以一个衰减因子 $\gamma\lambda$，S_{t-2} 处

应乘以 $(\gamma\lambda)^2$。

体现在公式中：

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$

前向视角与后向视角的 TD(λ) 等价

On-policy Learning

即采样策略 π 和目标策略 π 一致

- 直接使用样本估计期望去估计总和
- 简单，收敛性好
- 数据利用性更差
- 限定随机性策略

Off-policy Learning

即采样策略 μ 和目标策略 π 不一致

- 一般采样策略 μ 选用随机性策略
- 目标策略 π 选用确定性策略

- 需要结合重要性采样
- 方差更大，收敛性更差
- 数据利用效好
- 采样策略需要比目标策略更有探索性

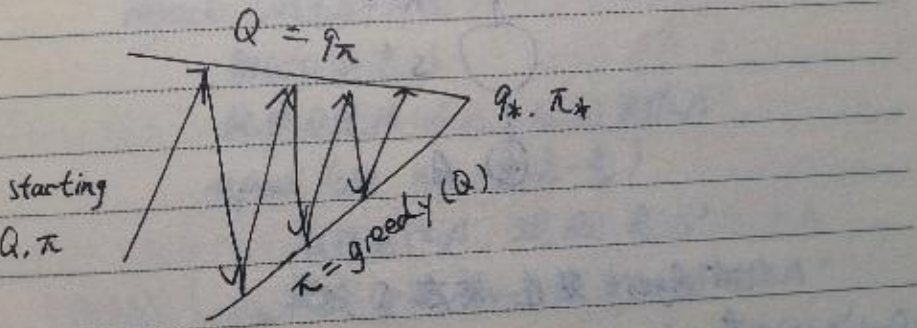
On policy Monte-Carlo Control

Generalised Policy Iteration

Model-free Policy Iteration Using Action-Value Funtion

用 $Q(s,a)$ 代替状态价值

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}}\, Q(s,a)$$



ϵ-greedy 探索

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon & \text{if } a^* = \underset{a \in A}{\operatorname{argmax}}\, Q(s,a) \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases}$$

ε- greedy 策略提升定理

对于任意 ε- greedy 策略 π，

使用相应的 q_π 得到的 ε-greedy 策略 π'

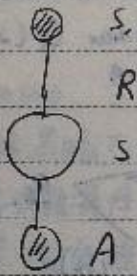
是在 π 上的一次策略提升

$$V_{\pi'}(s) \geq V_\pi(s)$$

Monte - Carlo Control

策略改进使用 ε-greedy 方法

SARSA



一个 Agent 处在状态 S，

基于ε- greedy 法选择一个行为 A，

与环境交互 给出即时收益 R，

进入下一个状态 S'，

基于 ε-greedy 法产生一个行为 A'，

不执行，得到 Q(S',A')

更新 Q(S,A)

SARSA 的价值函数更新公式

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$$

与在策略MC控制相比，二处修改

1. 策略估计不能变成了 SARSA
2. 每个 episode 更新一次变为
   每个 timestep 更新一次

程序结构图

初始化 ~~Q(S,A)~~ $Q(s,a), \forall s \in S, a \in A(s)$,
且 $Q(终止状态,\cdot)=0$

repeat (对于每个片段)

初始化状态 S

根据 Q 选择一个在 S 处的动作 A

repeat (对片段中的每一步)

执行动作 A，观测 R, S'

根据 Q 选择一个在 S' 处的动作 A'

$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$

$S \leftarrow S'$; $A \leftarrow A'$

until S 是终止状态

until 收敛

注：

1. 步长 α 一般随迭代的进行逐渐变小
   证明 Q 收敛
2. $Q(S,a)$ 可以一张大表格存储
   不适用规模太大的问题

期望 SARSA

$$Q(S_t,A_t) \leftarrow$$

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

$$\leftarrow Q(S_t,A_t) + \alpha[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1},a) - Q(S_t,A_t)]$$

1. 减少了由于 A' 的选择带来的方差
2. 更新相同步数时通用性更好
3. 可在 在/离策略中切换

n 步 SARSA

$n=1$ $q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1})$ (SARSA)

$n=2$ $q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}) \cdots$

$n=\infty$ $q_t^{(\infty)} = MC$

Date / /

前向认识 Sarsa(λ)

$$q_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}q_t^{(n)}$$

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha(q_t^{(\lambda)} - Q(S_t,A_t))$$

使用它更新Q价值需要遍历完整的episode

后向认识 Sarsa(λ)

$$E_0(s,a) = 0$$

$$E_t(s,a) = \gamma\lambda E_{t-1}(s,a) + \mathbb{I}(S_t = s, A_t = a)$$

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

$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta_t E_t(s,a)$$

Sarsa(λ) 算法

对于所有的 $s\in S$, $a\in A$ 初始化 $Q(s,a)$

repeat (对于每一个片段)

初始化 $E(s,a)=0$, $\forall s\in S$, $a\in A(s)$

选择 初始状态和初始动作 S, A

repeat 对于片段中的每一步

~~执行 动作~~ 执行动作 A, 观察到 R, S'

根据 Q 选择一个在 S' 处的 A'

$\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$

$E(S, A) \leftarrow E(S, A) + 1$

for 对于 所有 $s \in S$, $a \in A(s)$ do

$Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$

$E(s, a) \leftarrow \gamma \lambda E(s, a)$

end for

$S \leftarrow S'$, $A \leftarrow A'$

untill 直到终止状态

untill 直到收敛

重要性采样

Q-learning 进行离策略控制

主要表现：

个体都遵循着基于当前 $Q(s, a)$ 的一个 ε-greedy 策略；

目标策略是基于 $Q(s, a)$ 不包括 ε 的完全贪婪策略

$$\pi(s_{t+1}) = \arg\max_{a'} Q(s_{t+1}, a')$$

Q-learning 控制算法

定理：消极致至最优状态行为价值函数

$$Q(s, a) \to q_*(s, a)$$

SAC伪代码

for ~~each~~ each iteration do

for each environment step do

$a_t \sim \pi_\phi(a_t|s_t)$

$s_{t+1} \sim p(s_{t+1}|s_t, a_t)$

$D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$

end for

for each gradient step do

$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$

$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$

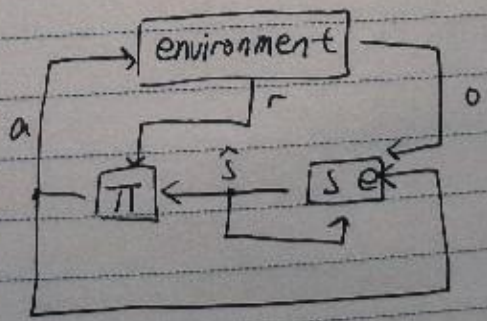
$\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$

$\bar{\theta}_i \leftarrow \tau \theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$

end for

end for

Partially Observable Markov Decision Process



Date / /

High-Low Hierarchies with Intrinsic Rewards