Subject: Westlake University, Reinforce Learning, Lecture 1, Basic Concepts

Date: from December 19, 2024 to December 20, 2024

Contents

Lecture 1, Basic Concepts

Bilibili:Lecture 1, Basic Concepts

Grid world example

Grid world example

本课程和 textbook 自始至终以 grid world 举例，因为这非常容易理解也很直观。在网格世界中，可以想象有一个机器人（称为 agent）每次只能移动一格，它的目的是在初始位置（图1中绿色）以“最好”的方式到达终点（图1中黄色），中途会遇到一些障碍物（图1中红色）而不能进入.

事实上这个问题很直观地就可以理解，但是实际上仍然可以十分复杂，例如：

1. 目标是只需找到一个确定位置到达 Target 的最佳方式还是任意给定位置都可以找到最佳方式?
2. 目标中的“最好”如何理解？最短的路径？不要撞墙？还是不要撞到障碍物?
3. 障碍物是绝对不能进入还是可以但是进入了会有惩罚？惩罚多少？如何施加?
4. 当 agent 清晰周围环境时，规划路径实际上很容易，但是当不知晓时如何解决?

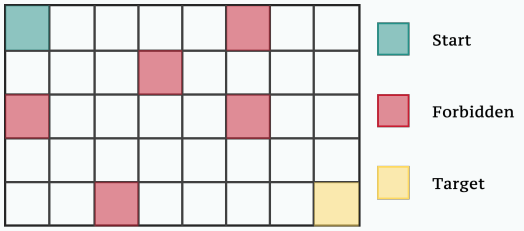

Figure 1: A grid world example

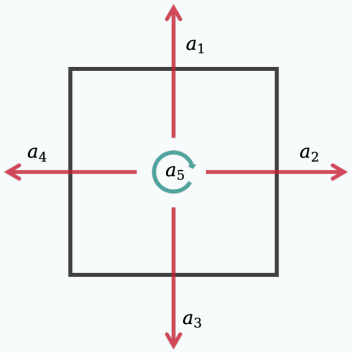

Figure 2: Action example

Basic Concepts – under a grid world example

State and action

State: 描述 agent 在环境中的状态，在 grid world 中指的是位置(location)，记为 s_i.

State Space: state 的集合，记为 $\mathcal{S} = \{s_i\}_{i=1}^{n}$.

Action: 表示 agent 在某状态下采取的行动，记为a_i(例如图2).

Action space: action 的集合，其依赖于所处状态，因为不同状态下能做出的行动可能不同.

Action Space of state: 某状态下对应的 action space，记为$\mathcal{A}(s_j) = \{a_i\}_{i=1}^{k}$.

State transition

Station Transition: 某一个状态采取了行动转变成另一个状态，例如记为$s_1 \xrightarrow{a_2} s_2$.其定义了 agent 与 environment 交互的行为，可以用表格(tabular)的形式表现出来，但只能表现确定性的情况(deterministic case).

	a_1(upward)	a_2(rightward)	a_3(downward)	a_4(leftward)	a_5(still)
s_1	s_1	s_2	s_4	s_1	s_1
⋮	⋮	⋮	⋮	⋮	⋮
s_9	s_6	s_9	s_9	s_8	s_9

Table 1: A tabular representation of the state transition process

State transition probability: 使用概率表示 state transition，如$\mathbb{P}(s_2|s_1, a_2) = 0.9$表示在 state s_1时采取 action a_2 变成 state s_2 的概率.这样就可以将 deterministic case 变为 stochastic case.

Policy and reward

Policy: 告诉 agent 在某状态时应该采取什么行动，记为π，在数学上实际上就是条件概率.例如在 state s_1时可以采用3种 action，记为$\pi(a_1|s_1) = 0, \pi(a_2|s_1) = 1, \pi(a_3|s_1) = 0$，那么说明一定会采取 action a_2. 同一个 state 下不同 action 的 π 的和为1. Policy 也可以使用表格(tabular)形式表达（如表），称为 tabular representation.

	a_1(upward)	a_2(rightward)	a_3(downward)	a_4(leftward)	a_5(still)
s_1	0.2	0.1	0	0.5	0.2
⋮	⋮	⋮	⋮	⋮	⋮
s_9	1	0	0	0	0

Table 2: A tabular representation of a policy

Reward: 一个实数标量，一般当为正时表示对模型的奖励(encouragement)，为负时表示对模型的惩罚(punishment)[a].

1. 其实际上是人与机器交互的手段(human-machine interface)，用以引导使 agent 的表现符合预期.
2. 对于较复杂的任务，reward 的设计也需要很好的前置专家知识，但是较于其他需要更多专业知识的方法已经较为简单
3. Reward 也可以使用表格(tabular)形式表达（如表3所示）.同时，reward 也可以用条件概率以随机化，例如$\mathbb{P}(r = -1|s_1, a_1) = 0.9$表示在 state s_1 时采取 action a_1 产生的 reward 为 $r = -1$ 的概率为0.9.
4. 一个好的 policy 并不是每一步都选择当前最大的 reward，因为 immediate reward 最大并不意味着最后的 total reward 最大.
5. Reward 取决于当前的状态和动作，并不取决于下一步的状态，如原地不动和"撞墙"下一步的状态相同，但是 reward 一般不同.

	a_1(upward)	a_2(rightward)	a_3(downward)	a_4(leftward)	a_5(still)
s_1	$r_{bounary}$	0	0	$r_{bounary}$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_9	$r_{forbidden}$	$r_{bounary}$	$r_{bounary}$	0	r_{target}

Table 3: A tabular representation of a policy

[a]不设置/设置 reward=0 就是不惩罚，也意味着鼓励；鼓励也可以是负，惩罚相应变成正，本质上数学是一样的.

Trajectories, returns, and episodes

Trajectory: 是一个 state - action - reward chain，例如

$$s_1 \xrightarrow[r=0]{a_2} s_2 \xrightarrow[r=0]{a_3} s_5 \xrightarrow[r=0]{a_3} s_8 \xrightarrow[r=1]{a_2} s_9$$

Return: 针对一个 Trajectory，将所有的 reward 累和，因此又称为 total rewards 或 cumulative rewards。

1. 可以用来评估一个 policy 的好坏(和比较两 policy).
2. Return 包含 immediate reward 和 future rewards.其中 immediate reward 是在初始 state (initial state)采取 action 后立即获得的 reward，future rewards 指离开 initial state 后获得的 rewards 总和.
3. 为避免短视(short-sighted)的决定，应该关注 return 而不是 immediate reward.

Discounted return: 一个 trajectory 有时是无穷的，例如当达到目的地后可能"原地踏步"，如果此时仍然有施加 reward 就会导致 return $\to \infty$，此时可以给每一步的 reward 施加一个"折扣"(discount) $\gamma \in [0, 1)$，第 k 个 reward r_k 变成$\gamma^k r_k$.

1. 由于$\gamma < 1$，所以最后的 return 不会发散.
2. 通过控制 γ 的大小，可以控制模型更关注前面的(γ 小，短视) action 还是未来的(γ 大，远视) action.

Episode/Trial: 通常当一个任务会终止时（存在 terminal states），即有限长度的 Trajectory 被称为 Episode/Trial.这样的任务被称为 episodic task. 当不存在 terminal states 时，这种任务被称为 continuing tasks.一般来说没有绝对的一直持续的任务，但是根据解决问题的时间尺度可以这样设置. 事实上有统一的方法描述 episodic 和 continuing task，例如

1. 将 terminal states 看作特殊的 state，到达 target state 后通过设计 action space / 设计某条件概率为 1 强制"原地踏步"，并且同时设置之后的 reward=0，这样的 state 称为 absorbing states.
2. 就把 terminal state 当成一个普通的 state，达到 terminal state 的 reward 设计为正. 这样做更具有一般性，且有可能跳出局部最优解. 本课程使用2.

Markov decision process(MDP)

Markov decision process (MDP)

马尔可夫决策过程(Markov decision processes, MDPs)是描述随机动力学系统的通用框架，由于在 agent 与环境交互过程中具有随机性，因此强化学习事实上也可以形式化入 MDP 框架，其中关键组成部分如下（同时也作为对前面内容的总结回顾）：

1. 集合(Sets)
   - State space：state 的集合，记为 $\mathcal{S}$
   - Action space：action 的集合，与 state 有关，记为 $\mathcal{A}(s)$
   - Reward set：reward 的集合，与 state 和 action 都有关，记为 $\mathcal{R}(s,a)$
2. 模型(Model)
   - State transition probability：在某 state 采取某 action 后，state 转变为 s' 的概率记为 $p(s'|s,a)$，其需要满足 $\sum_{s'\in\mathcal{S}} p(s' \mid s,a) = 1, \forall (s,a)$
   - Reward probability：在某 state 采取某 action 后获得 reward r 的概率记为 $p(r|s,a)$，其需要满足 $\sum_{r\in\mathcal{R}(s,a)} p(r \mid s,a) = 1, \forall (s,a)$
3. 马尔可夫性(Markov property)：指随机过程的一种无记忆属性(memoryless property)，这里指下一步的 state 和 reward 都有前面的 state 和 action 无关

$$p(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = p(s_{t+1} \mid s_t, a_t), \tag{1a}$$

$$p(r_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = p(r_{t+1} \mid s_t, a_t) \tag{1b}$$

Note 1. 对于所有的 (s,a)，$p(s'|s,a)$ 和 $p(r|s,a)$ 被称为 *model / dynamics*，其可以是时变/非稳定的*(time-variant / nonstationary)* 也可以是非时变/稳定的*(time-invariant/stationary)*。环境可以随时间变化的就是时变的，例如在 *grid world* 中某些时候某些区域会变成 *forbidden area.*

Note 2. 当 *Markov decision process* 中的 *decision/policy* 确定了，那么其就等同马尔可夫过程*(Markov process, MP).*因为在 *MDP* 中包含了具有概率性的 *model*，但是 *MP* 只是一个确定性的过程.

Summary

Markov decision process (MDP)

强化学习是 agent 与 environment 不断交互的过程。Agent 是一个可以感知当前状态（state）、维护改进策略（policy）和执行动作（action）的决策者（decision-maker）。

Agent 执行动作会改变其 state，同时会获得 reward，再执行动作依此循环，形成了强化学习的整个流程.

agent take action $\rightarrow$ execute $\rightarrow$ state changed + reward obtained $\rightarrow$ take action $\rightarrow \cdots$

Q & A

Questions and answers

Q1: 可以将所有的 reward 设置为负吗?

A1: 可以。事实上，对一个 action 鼓励/ 不鼓励是由 reward 的相对值而不是绝对值决定。因此将所有的 reward 设置为负但不同的 action 有差别仍会找到最优 policy.

Q2: reward 是下一 state s' 的函数吗?

A2: 不是。虽然直觉上来说下一个状态 s' 的好坏决定这一 reward 的相对值，但是实际上 reward 仅是当前 state s 和 action a 的函数。原因在于 s' 也决定于 s, a，并且数学上有[a]

$$p(r \mid s, a) = \sum_{s'} p(r \mid s, a, s')p(s' \mid s, a) \tag{2}$$

因此我们使用 $p(r|s, a)$ 而不是 $p(r|s, a.s')$.

[a] 公式2也是推导第二节 Bellman equation 的重要工具

First update: March 9, 2025

Last update: March 10, 2025

References