A Peek into RL

Introduction to Reinforcement Learning

01

Terminology

Basic concepts in RL
The major part of this workshop

02

Markov Process

Markov Decision Process,
Markov Reward Process,
Bellman Equations…

03

Classic Algorithms

MC, TD, DP…
A simple introduction

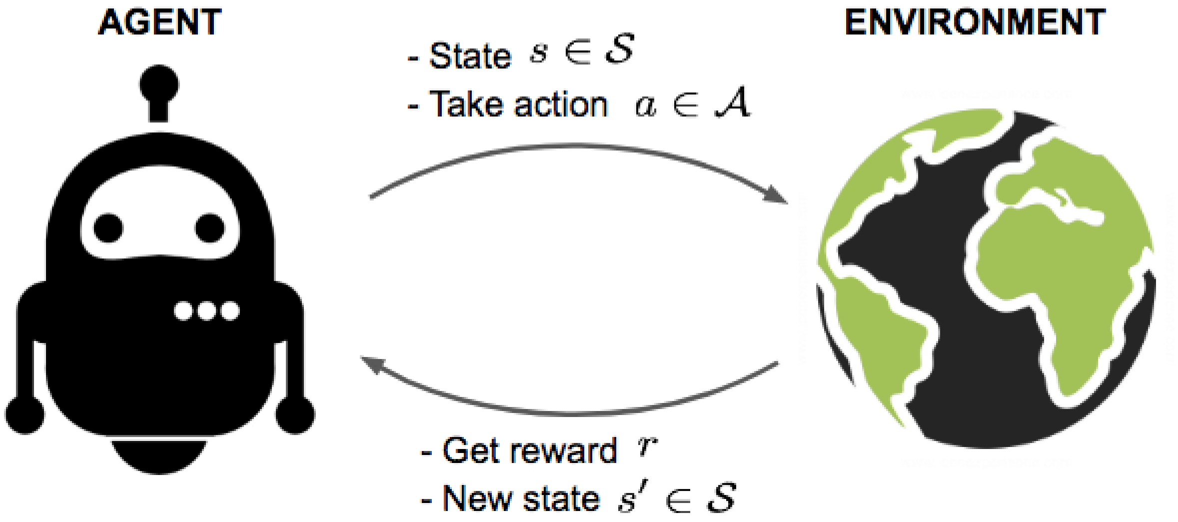
04

Extensions and Recommendation

Real-life applications; Courses and materials for our audience to study independently

Introduction

What’s Reinforcement Learning?


AGENT
ENVIRONMENT
- State $s \in \mathcal{S}$
- Take action $a \in \mathcal{A}$
- Get reward r
- New state $s' \in \mathcal{S}$

Teminologies: Rewards

- Reward Rt: A scalar feedback signal
- Indicates how well agent is doing at step t
- The agent’s job is to maximize cumulative reward (Reward Hypothesis)
- E.g., +/−ve reward for winning/losing a game

Find an optimal way to make decisions

- Sequential Decision Making
- With delayed rewards / penalties
- No future / long term feedbacks

- Policy: Agent’s behavior function
- Value function: How good is each state and/or action
- Model: Agent’s representation of the environment

Teminologies: Policy

- Policy: Agent’s behavior, a map from state to action
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a \mid s) = P[A_t = a \mid S_t = s]$

Teminologies: Value Function

- Value Function: A prediction of future reward
- Used to evaluate the goodness/badness of states (to select between actions)

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

- Future reward / Return: a total sum of discounted rewards going forward

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- State Value: $V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$
- Action Value: $Q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$ → $V_\pi(s) = \sum_{a \in \mathcal{A}} Q_\pi(s, a)\pi(a|s)$

Optimal Value and Policy

- The optimal value function produces the maximum return:

$$V_*(s) = \max_\pi V_\pi(s), Q_*(s,a) = \max_\pi Q_\pi(s,a)$$

- The optimal policy achieves optimal value functions:

$$\pi_* = \arg\max_\pi V_\pi(s), \pi_* = \arg\max_\pi Q_\pi(s,a)$$

- Relationship: $V_{\pi_*}(s) = V_*(s)$, $Q_{\pi_*}(s,a) = Q_*(s,a)$

Teminologies: Model

- A model predicts what the environment will do next
- $\mathcal{P}$ predicts the next state

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

- $\mathcal{R}$ predicts the next (immediate) reward

$$\mathcal{R}_{s}^{a} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

- **Know the model / Model-based RL**: Planning with perfect information

  Find the optimal solution by Dynamic Programming (DP)
  E.g., Longest increasing subsequence

- **Does not know the model**: learning with incomplete information

  Model-free RL or learn the model in the algorithm

Categories of RL Algorithms

- **Model-based**: Rely on the model of the environment; Either the model is known or the algorithm learns it explicitly.
- **Model-free**: No dependency on the model during learning.
- **On-policy**: Use the deterministic outcomes or samples from the target policy to train the algorithm.
- **Off-policy**: Training on a distribution of transitions or episodes produced by a different behavior policy rather than that produced by the target policy.

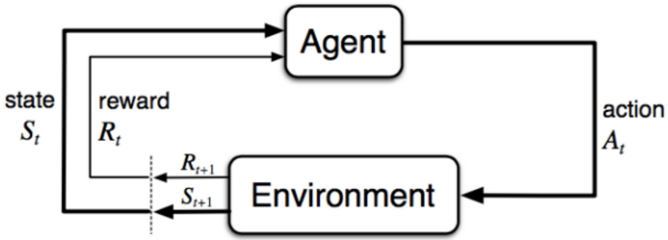
Markov Process

MRP, MDP, Bellman Equations

MDPs

- Markov decision processes (MDPs) formally describe an environment for reinforcement learning where the environment is fully observable
- All states in MDP has the Markov property: the future only depends on the current state, not the history. A state St is Markov if and only if:

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



MDPs

- Recap: State transition probability $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$

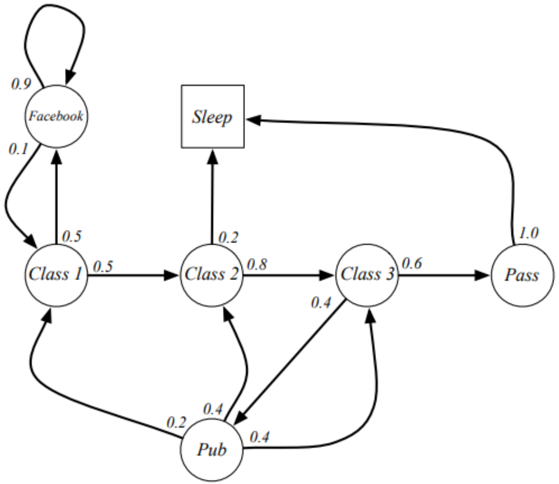
  State transition matrix $\mathcal{P}$ defines transition probabilities from all states s to all successor states s',

$$\mathcal{P} \;=\; \textit{from} \overset{\textit{to}}{\begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix}}$$

  where each row of the matrix sums to 1.

- A Markov process is a memoryless random process, i.e. a sequence of random states S1, S2, ... with the Markov property. Represented as a tuple <$\mathcal{S}$, $\mathcal{P}$>

Example: Markov Chain Transition Matrix



$\mathcal{P} =$

	C1	C2	C3	Pass	Pub	FB	Sleep
C1		0.5				0.5	
C2			0.8				0.2
C3				0.6	0.4		
Pass							1.0
Pub	0.2	0.4	0.4				
FB	0.1					0.9	
Sleep							1

MDPs

- Markov Reward Process (MRP): A Markov reward process is a Markov chain with values.
- Represented as a tuple $<\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma>$
- A Markov deicison process (MDP) consists of five elements $\mathcal{M}=\langle\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma\rangle$

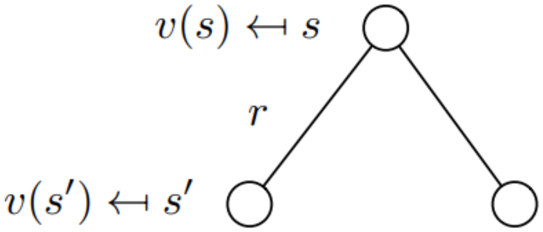
$\mathcal{S}$ - a set of states;
$\mathcal{A}$ - a set of actions;
P - transition probability function;
R - reward function;
γ - discounting factor for future rewards. In an unknown environment, we do not have perfect knowledge about P and R.

Bellman Equations

- Bellman equations refer to a set of equations that decompose the value function into the immediate reward plus the discounted future values.

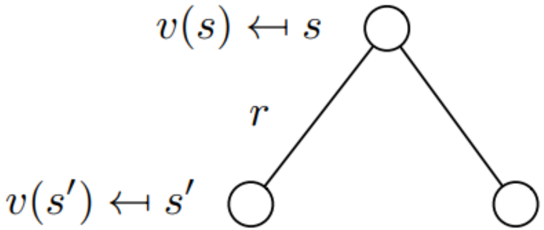
$$\begin{aligned} V(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) | S_t = s] \end{aligned}$$

$$\begin{aligned} Q(s, a) &= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{a \sim \pi} Q(S_{t+1}, a) \mid S_t = s, A_t = a] \end{aligned}$$



$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Bellman Equations

- Bellman equations refer to a set of equations that decompose the value function into the immediate reward plus the discounted future values.
- 

Matrix Form:

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in S} \mathcal{P}_{ss'} v(s')$$

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Bellman Equations

- Linear Equations → Could be solved directly
- Computational complexity is O(n^3) for n states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs,

  e.g. Dynamic programming (DP)
  Monte-Carlo evaluation (MC)
  Temporal-Difference learning (TD)

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

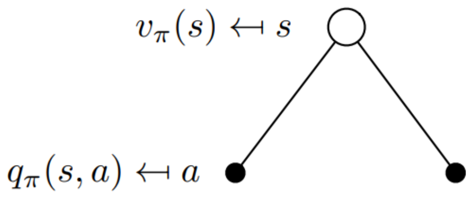
$$(I - \gamma \mathcal{P})\, v = \mathcal{R}$$

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

Bellman Expectation Equation (For MDPs)

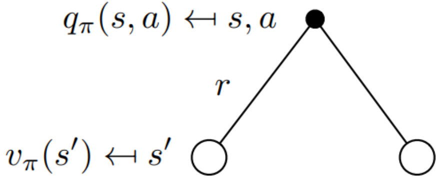
$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$

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

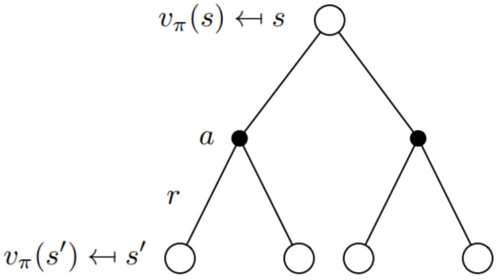


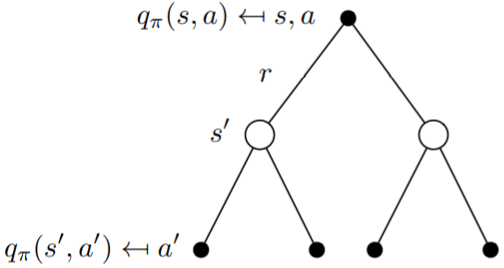
$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]$$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$



Bellman Expectation Equations





$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

Bellman Expectation Equations

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s,a)$$

$$Q_\pi(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_\pi(s')$$

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \Big(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_\pi(s') \Big)$$

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

Matrix Form:

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

$$v_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

Bellman Optimality Equations

$$V_*(s) = \max_{a \in \mathcal{A}} Q_*(s,a)$$

$$Q_*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_*(s')$$

$$V_*(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_*(s')\right)$$

$$Q_*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} \max_{a' \in \mathcal{A}} Q_*(s',a')$$

- Non-linear
- No closed form solutions in general
- Many iterative solution methods

  Value Iteration

  Policy Iteration

  Q-learning

  Sarsa

Classical Algorithms

Dynamic Programming

- Policy Evaluation: $$V_{t+1}(s) = \mathbb{E}_\pi[r + \gamma V_t(s') | S_t = s] = \sum_a \pi(a|s) \sum_{s',r} P(s', r|s, a)(r + \gamma V_t(s'))$$
- Policy Improvement: $$Q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s, A_t = a] = \sum_{s',r} P(s', r|s, a)(r + \gamma V_\pi(s'))$$
- Policy Iteration: An iterative procedure to improve the policy when combining policy evaluation and improvement

$$\pi_0 \xrightarrow{\text{evaluation}} V_{\pi_0} \xrightarrow{\text{improve}} \pi_1 \xrightarrow{\text{evaluation}} V_{\pi_1} \xrightarrow{\text{improve}} \pi_2 \xrightarrow{\text{evaluation}} \dots \xrightarrow{\text{improve}} \pi_* \xrightarrow{\text{evaluation}} V_*$$

$$\begin{aligned} Q_\pi(s, \pi'(s)) &= Q_\pi(s, \arg\max_{a \in \mathcal{A}} Q_\pi(s, a)) \\ &= \max_{a \in \mathcal{A}} Q_\pi(s, a) \geq Q_\pi(s, \pi(s)) = V_\pi(s) \end{aligned}$$

About the Convergence

Someone raised a good question in the workshop: the convergence property of the policy iteration.
Typically, the proof in this link would be helpful.
Also, it should be noted that for the policy-based RL, only local optimum is guaranteed.
Here are the pros and cons of the policy-based RL:
Advantages:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies

Disadvantages:

- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

Monte-Carlo Learning

- MC methods need to learn from **complete** episodes

$$V(s) = \frac{\sum_{t=1}^{T} 1[S_t = s] G_t}{\sum_{t=1}^{T} 1[S_t = s]}$$

$$Q(s,a) = \frac{\sum_{t=1}^{T} 1[S_t = s, A_t = a] G_t}{\sum_{t=1}^{T} 1[S_t = s, A_t = a]}$$

Temporal Difference Learning

- TD learning can learn from **incomplete** episodes

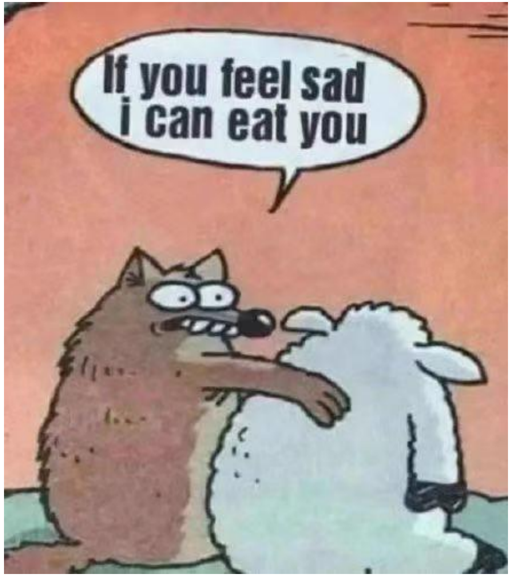
$$V(S_t) \leftarrow (1-\alpha)V(S_t) + \alpha G_t$$
$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$
$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

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

Extensions

- Exploration and Exploitation
- DeepMind AI: Link
- Games
- Different methods to solve the same problem
- More to be explored…


00:20
Hid
den
Who will win
180
OR
Bet
10
Random
OFF
Winning Streak
2
Start!


If you feel sad
i can eat you

References and Recommendations

Lectures:

- Stanford CS324
- 李宏毅《深度强化学习》
- RL by David Silver

Photos:

- A (Long) Peek into Reinforcement Learning
- Google

Textbooks:

- Reinforcement Learning: An introduction, by Sutton and Barto
- Algorithms for Reinforcement Learning by Csaba Szepesvari

THANKS!

Do you have any questions?

Contact：
Nickname, WISE@CUHK
Subscribe: Google Form
Telegram Group: Join