Basic Concepts about Reinforcement Learning

Cao Ying

Outlines

Basic Concept and Terminology

1. What is reinforcement learning
2. Differences among RL, SL and UL
3. Elements of RL

RL as Markov Decision Process

1. Markov Decision Process
2. Value functions V and Q
3. Bellman equation and its optimality
4. Three approaches

What is Reinforcement Learning

Learning from interaction to achieve a goal.

✓Reward signal defines the goal.

✓The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying out

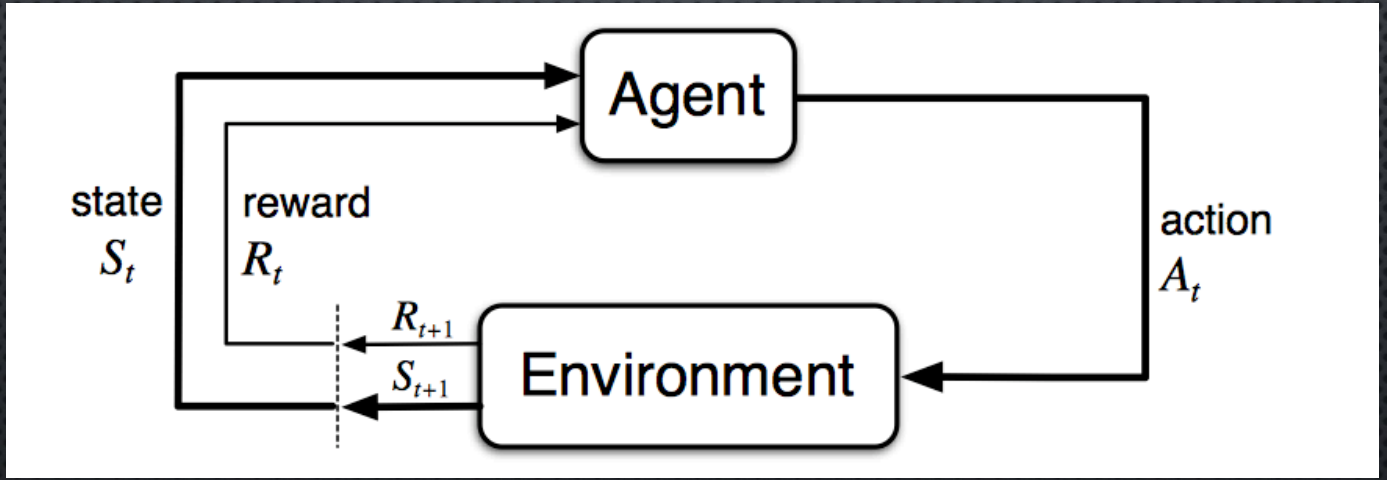
✓exploration – exploitation dilemma：neither exploration nor exploitation can be pursued exclusively without failing at the task

Differences among SL, USL and RL

	Description of the task	Objective	Methodology	characteristics
reinforcement learning	• map situations to actions to maximize a numerical reward.	learn to act	decision making	learn from the agent's *own experiences*
supervised learning	• learn from labeled examples to generalize 1. each example is a situation 2. label is the right action to take	predict and generalize	parameterized function approximator	• training samples represents the desired behaviors which are *both correct and representative* of all the situations. • not adequate for learning from interaction.
unsupervised learning	• find structure hidden in unlabeled data	find hidden structure		

Elements of RL: The Agent–Environment Interface

- ✓ A dynamic system : the leaner's actions influence its later inputs.



Elements of RL : Policy

- A policy is the agent's behavior, a mapping from state to action

$$\pi(a|s) = \mathbb{P}[a|s]$$

- a simple function or lookup table

Elements of RL: Reward

- ✓ reward is a scalar which defines the goal in reinforcement problem
- ✓ a reward signal indicates what is good in an immediate sense.

1. reward is not a place to impart to the agent about how to achieve the goal
2. reward is your way of communicating to the robot what you want not how you want it achieve

Elements of RL: Return

the cumulative reward an agent receives in the long run

- the simplest case for *episodic task*

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

- discounted return for *continuing task*

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$0 \leq \gamma \leq 1$ is the discounting rate,

shortsighted $0 \leftarrow \gamma \rightarrow 1$ farsighted

The goal of reinforcement learning is not maximization of immediate reward , but cumulative reward in long run.

Markov Decision Process

- ✓ Markov Decision Process is a tuple $< S, A, T, R >$
  - S : is the state space
  - A : possible actions that cant be executed by the agent.
  - $T : S \times A \times S \rightarrow [0, 1]$ is the transition function and encodes the probability $P(s'|s, a)$ of reaching state s' after executing action a in state s
  - $R: S \times A \rightarrow \mathcal{R}$: reward function
- ✓ ideally, a state should summarize past sensations so as to retain all "essential" information
- ✓ Markov Assumption

$$\Pr\{S_{t+1} = s', R_{t+1} = r \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\}$$

$$p(s', r|s, a) \doteq \Pr\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\}$$

RL as Markov Decision Process

- ✓ If a reinforcement learning has the Markov Property, it is basically a MDP
- ✓ If the state and action are finite, it is a finite MDP
- ✓ To define a MDP, you need to give
  - state and action sets
  - one-step dynamics of the environment
    1. state-transition probabilities

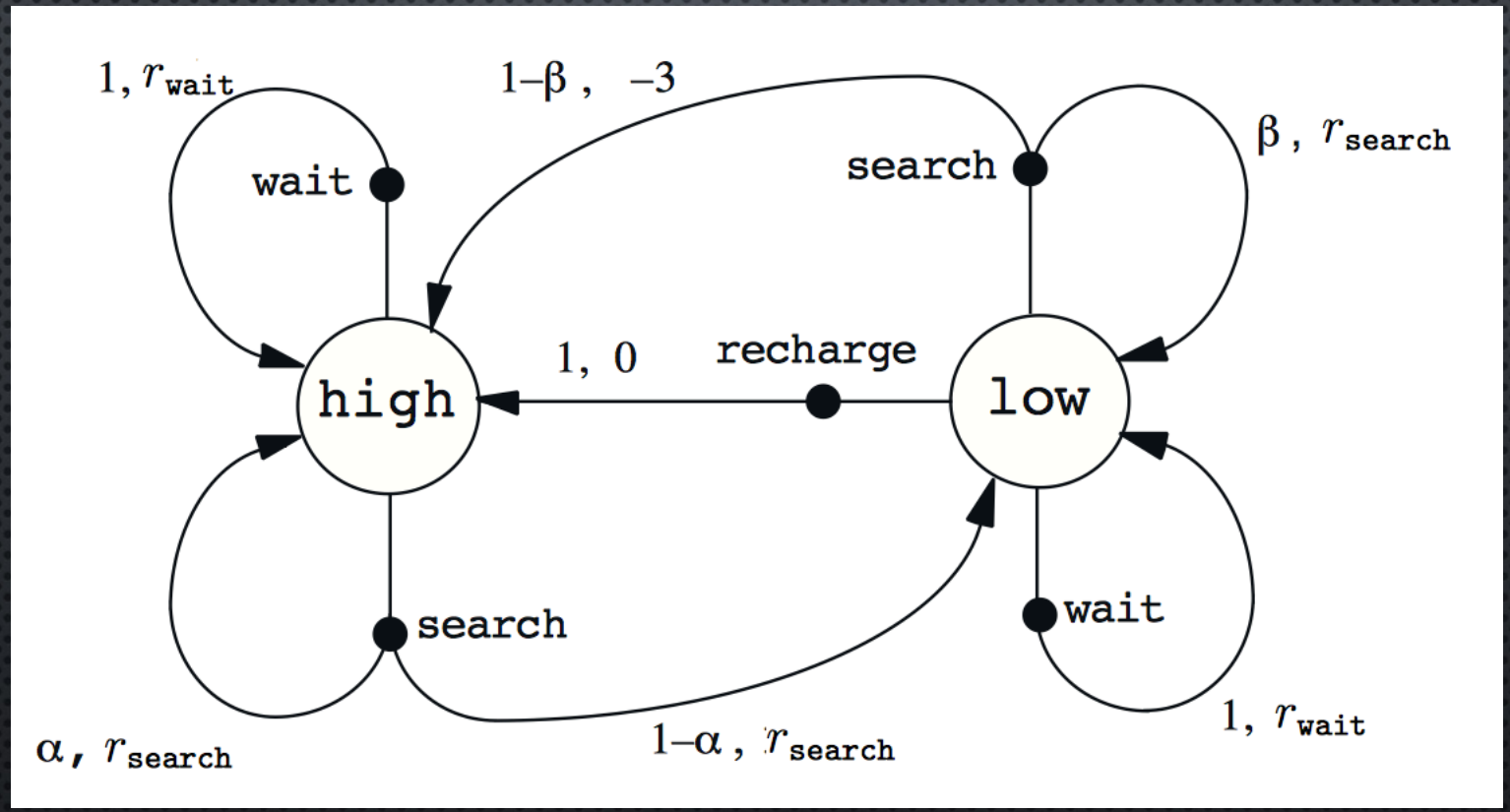
$$\mathbf{P}^{a}_{ss'} = \Pr\left\{s_{t+1} = s' \mid s_t = s, a_t = a\right\} \quad \text{for all } s, s' \in S, a \in A(s).$$

    2. expected rewards for state–action-next-state

$$\mathbf{R}^{a}_{ss'} = E\left\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\right\} \quad \text{for all } s, s' \in S, a \in A(s).$$

- ✓ deterministic

way to summary dynamics of finite MDP



Value Functions V and Q

- value function estimate how good it is to perform a given action in a given state
- the notation of "how good" is defined in terms of expected return
- the *state-value function for policy* π is define as follows:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

V: the value of state *s* under a policy π is the expected return when starting in s and following π thereafter.

Value Functions V and Q

- the *action –value function for policy* π is define as follows:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t=s, A_t=a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \,\middle|\, S_t=s, A_t=a\right]$$

Q: value of take action a in state s , following π thereafter.

- The value functions can be estimated from experience

Monte Carlo methods

Bellman equation

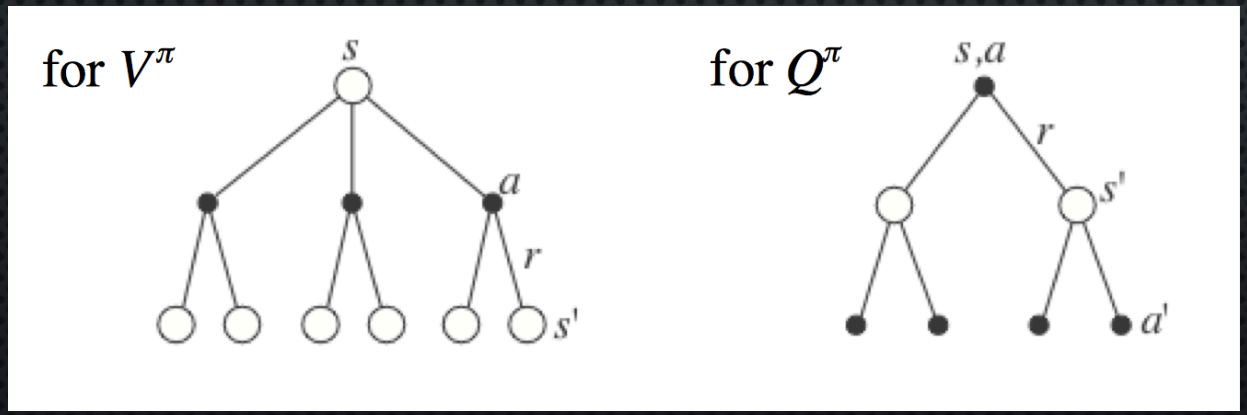
- A fundamental property of value functions used throughout reinforcement learning and dynamic programming is that they satisfy particular recursive relationships.

Bellman equation for V and Q

$$Q^{\pi}(s,a) = \sum_{s' \in S} P^{a}_{ss'} \cdot [R^{a}_{ss'} + \gamma \cdot \sum_{a' \in A(s')} \pi(s',a') \cdot Q^{\pi}(s',a')]$$

$$V^{\pi}(s) = \sum_{a \in A(s)} \pi(s,a) \sum_{s' \in S} P^{a}_{ss'} \cdot [R^{a}_{ss'} + \gamma \cdot V^{\pi}(s')]$$

backup diagram



Optimal Value Functions

- A policy π is defined to be better than or equal to a policy π', if its expected return is greater than or equal to that of for all states

$$\pi \geq \pi' \quad \text{if and only if} \quad V^{\pi}(s) \geq V^{\pi'}(s) \quad \forall\, s \in S$$

- There is always at least one (and possibly many) policies that is better than or equal to all the others. This is an optimal policy π^*

Optimal Value Functions

- Optimal polices share the same optimal state value function V^*

$$V^*(s) := V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s) \quad \forall \quad s \in S$$

- Optimal polices also share the same optimal action-value function

$$Q^*(s,a) := Q^{\pi^*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \quad \forall \quad s \in S,\ a \in A(s)$$

Relationship between V^* and Q^*

- Q^* the expected return for
    1. start with state s
    2. take action a
    3. thereafter following an optimal policy π^*

thus, we can write

$$Q^*(s,a) = E\left\{r_{t+1} + \gamma V^*(s_{t+1}) \middle| s_t = s, a_t = a\right\}$$

Bellman Optimality Equation for V^*

value of a state under an optimal policy π^*

=

expected return for the best action from that

$$
\begin{aligned}
V^*(s) &= \max_{a \in A(s)} Q^{\pi^*}(s,a) \\
&= \max_{a \in A(s)} E\left\{ r_{t+1} + \gamma V^*(s_{t+1}) \,\middle|\, s_t = s, a_t = a \right\} \\
&= \max_{a \in A(s)} \sum_{s'} \mathbf{P}^a_{ss'} \left[\mathbf{R}^a_{ss'} + \gamma V^*(s') \right]
\end{aligned}
$$

For finite MDPs, the Bellman optimality equation for V^* has a unique solution independent of the policy.

Bellman Optimality Equation for V^*

- The Bellman optimality equation is a system of equations, one for each state
- Once one has V^*, there will be one or more actions at which the maximum is obtained in the Bellman optimality equation.
- Any policy that assigns nonzero probability only to these actions is an optimal policy.
- if one uses V^* to evaluate the one-step consequences, then a greedy policy is actually optimal in the long-term sense.
- By means of V^*, the optimal expected long-term return is turned into a quantity that is locally and immediately a[illegible] for each state.

Bellman Optimality Equation for Q^*

$$Q^*(s,a) = E\left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1},a') \,\middle|\, s_t = s, a_t = a \right\}$$

$$= \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a'} Q^*(s',a') \right]$$

- for any state s, it can simply find any action that maximizes Q^*(s,a).
- The action–value function effectively caches the results of all one–step–ahead searches.
- The optimal action–value function allows optimal actions to be selected with knowing nothing about the environment's dynamics.

Let's recap: Why V^* and Q^* are Useful

V^*

- Any policy that is greedy with respect to V^* is an optimal policy. Given V^*, one-step-ahead search produces the long-term optimal actions.

Q^*

- Given Q^*, the agent does not even have to do a one-step-ahead search

Solving Bellman Optimality Equation

- Finding an optimal policy by solving the Bellman Optimality Equation requires
  1. accurate knowledge of environment dynamics
  2. enough space and time to do the computation
  3. the Markov Property
- A solution requires an exhaustive search
- BUT, number of states is often huge
- We usually have to settle for approximations

Many RL methods can be understood as approximately solving the Bellman Optimality Equation

Three approach to RL

- Value-based RL
  - estimate the optimal action-value function
- Policy-based RL
  - search directly for the optimal policy
- Model-based RL
  - build a model of the environment