Reinforcement Learning, Bellman Equations and Dynamic programming

Seminar in Statistics: Learning Blackjack - Talk on 04.04.16

Christoph Buck, Daniela Hertrich

Agent-Environment Interface (in discrete time steps)

- State: $S_t \in \mathcal{S}$
- Reward: $R_t \in \mathbb{R}$
- Action: $A_t \in \mathcal{A}(S_t)$

Policy

In each state, the agent can choose between different actions. The probability that the agent selects a possible action is called policy.

- $\pi_t(s|a)$: probability that $A_t = a$ if $S_t = s$

Return

The return is the sum of the rewards.

- Unified Notation of the return: $G_t = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$ where T is allowed to be ∞ and $0 < \gamma \leq 1$

The Markov Property

Decisions are assumed to be a function of the current state only.

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

The Markov Decision Processes

A task is a Markov Decision Process (MDP) if it satisfies the Markov Property.

- Given any state and action, s and a, the probability of each possible next state and reward, s', r, is:
$$p(s', r|s, a) = Pr\{S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\}$$
- Given any current state and action, s and a, together with any next state, s', the expected value of next reward is:
$$r(s, a, s') = E[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s']$$

Value functions

Value functions estimate how good it is for the agent to be in a given state (state-value function) or how good it is to perform a certain action in a given state (action-value function).

- State-value function: The value of a state s under a policy π is the expected return when starting in s and following π thereafter:

$$v_\pi(s) = E_\pi[G_t|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

- Action-value function: The value of the expected return taking action a in state s under policy π:

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

Bellman optimality equation

- Bellman optimality equation for v_*:

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$$

- Bellman optimality equation for q_*:

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]$$

Policy Improvement Theorem

Let π and π' be any pair of deterministic policies such that, for all $s \in \mathcal{S}$,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s). \tag{1}$$

Then the policy π' must be as good as, or better than, π. That is, it must obtain greater or equal expected return from all states $s \in \mathcal{S}$:

$$v_{\pi'}(s) \geq v_\pi(s). \tag{2}$$

Moreover, if there is strict inequality of (1) at any state, then there must be strict inequality of (2) at at least one state.

Value Iteration

Algorithm 1 Value iteration: Pseudocode

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in S^+$)

repeat
 $\Delta \leftarrow 0$
 for each $s \in S$: **do**
  $v \leftarrow V(s)$
  $V(s) \leftarrow max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')$
  $\Delta \leftarrow max(\Delta, |v - V(s)|)$
 end for
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that $\pi(s) = argmax_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

References

R. Sutton/A. Barto: *Reinforcement Learning: An Introduction*, 2^{nd} edition, in progress, 2014/2015