Reinforcement Learning

Jack Lanchantin

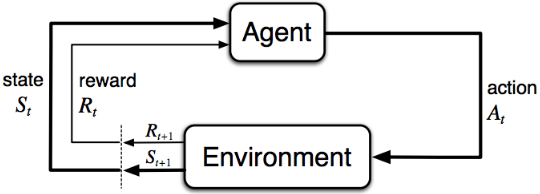
November 30, 2015

Outline

Outline

- **Framing of the problem of learning from interaction to achieve a goal.**
- **Agent**: learner and decision maker
- **Environment**: what the learner interacts with (everything outside the agent)
- Agent selects actions and the environment responds to those actions and presents new situations

Reinforcement Learning

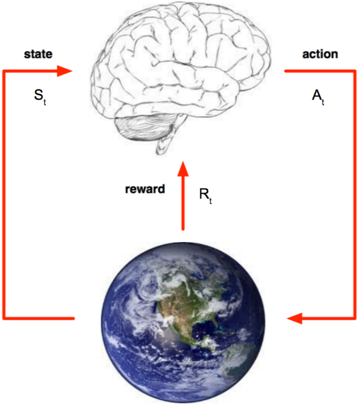


- At each time step t, the agent receives the environment **state** $S_t \in \mathcal{S}$, and the agent then selects an **action** $A_t \in \mathcal{A}(S_t)$
  - $\mathcal{S}$ is the set of possible states (whatever information is available to the agent).
  - $\mathcal{A}(S_t)$ is set of actions available in state S_t
- One time step later, the agent receives a **reward**, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and ends up in a new state S_{t+1}

Reinforcement Learning

At each step t,

- The agent:
  - Receives state S_t
  - Receives scalar reward R_t
  - Executes action A_t
- The environment:
  - Receives action A_t
  - Emits state S_t
  - Emits scalar reward R_t



Policies

- At each time step, the agent implements a mapping π_t from states to probabilities of selecting each possible action, where π_t is called a **policy**
  - $\pi_t(a|s) =$ probability that $A_t = a$ if $S_t = s$

Reinforcement Learning Objective

The agent's goal is to maximize the total amount of reward it receives over the long run by changing its policy as a result of its experience

- If the sequence of rewards after time step t is $R_{t+1}, R_{t+2}, R_{t+3}, \ldots,$ then we want to maximize the return G_t
- The agent chooses A_t to maximize the discounted return:

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

  where γ is the discount rate and $0 \leq \gamma \leq 1$
- The closer γ is to 1, the more the agent accounts for future rewards

Outline

Markov Property

- Probability of transitioning to new state s' with reward r:

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

- Probability with Markov assumption:

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

- Markov assumption allows us to predict the next state and expected rewards from knowledge of only the current state
    - Assume that the current state tells us everything we need to know for future (e.g. current state of checker board)

Markov Decision Processes

- A reinforcement learning task that satisfies the Markov property is called a Markov Decision Process (MDP)
- Given any state s and action a, the probability of each possible pair of next state and reward $(s',\ r)$, is denoted

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

- From this probability representation, we can compute anything else we might need to know about the environment...

Markov Decision Processes

- State-transition probabilities:

$$p(s'|s,a) = Pr\{S_{t+1} = s' | S_t = s, A_t = a\} = \sum_{r \in \mathcal{R}} p(s', r | s, a)$$

- Expected rewards for state–action pairs:

$$r(s,a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r | s, a)$$

- Expected rewards for state–action–next-state triples:

$$r(s,a,s') = \mathbb{E}[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s'] = \frac{\sum_{r \in \mathcal{R}} r * p(s', r | s, a)}{p(s'|s,a)}$$

MDP Example

- At each tilmestep t, a can-collecting robot decides whether it should (1) actively search for a can, (2) remain in place and wait for someone to bring it a can, or (3) go back to charging station to recharge its battery

Outline

Value Functions

- Value Functions: functions of state–action pairs that estimate how good it is for the agent to perform a given action in a given state.
  - "How good" is defined in terms of future rewards that can be expected (i.e. expected return)
- Recall that a policy π maps a state s and action a to probability $\pi(a|s)$
- The value of a state s under policy π, denoted $v_\pi(s)$, is the expected return when starting in state s and following π thereafter

Value Functions

- **State-value** function for policy π:

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

  $\mathbb{E}_\pi[\cdot]$ is the expected value of a r.v. given the agent follows policy π

- **Action-value** function for policy π:

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

- Both v_π and q_π can be estimated from experience using Monte Carlo methods

Computing Value Functions

- Key idea of reinforcement learning is to use value functions to search for a good policy π
- Can use dynamic programming techniques to compute optimal value functions, and thus find optimal policies

References

Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. MIT Press 2015.