Reinforcement Learning

Jack Lanchantin

December 1, 2015

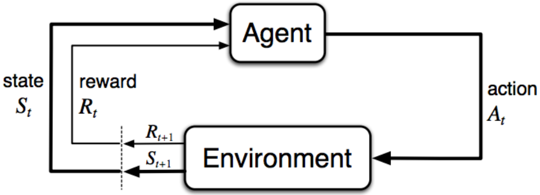
Outline

Outline

Reinforcement Learning

- **Learning from interactions to achieve a goal**
- **Agent**: learner and decision maker
- **Environment**: what the learner interacts with (everything outside the agent)

Reinforcement Learning

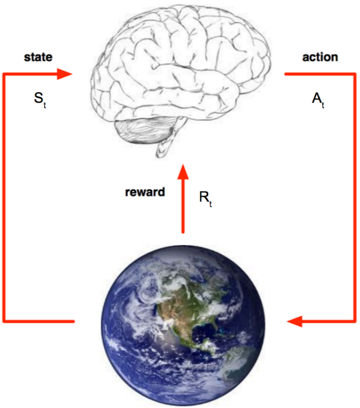


- At each time step t, the agent receives the environment **state** $S_t \in \mathcal{S}$, and the agent then selects an **action** $A_t \in \mathcal{A}(S_t)$
    - $\mathcal{S}$ is the set of possible states
    - $\mathcal{A}(S_t)$ is set of actions available in state S_t
- One time step later, the 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



Reinforcement Learning Objective

- 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
- **Learn a mapping of S $\rightarrow$ A which maximises future reward**

Outline

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:

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

Outline

Policies

- At each time step, the agent implements a mapping π states to actions, where π 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

Value Functions

- Almost all RL algorithms involve estimating value functions
- **Value Functions:** functions of states (or state–action pairs) that estimate how good it is for the agent to be in a certain state (or to perform an action in a given state).
  - "How good" is defined in terms of future rewards that can be expected (i.e. expected return)
- Two types of value functions: State-Value and Action-Value

State-Value Function for policy π

- 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**:

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

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

Action-Value Function for policy π

- The value of taking action a in state s under a policy π, denoted $q_\pi(s, a)$, is the **expected return starting from s, taking the action a, and following policy π thereafter**:

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

Optimal Value Functions

- Key idea of reinforcement learning is to use value functions to search for a good policy π
- A policy π is better than π' if its expected return is $\geq$ to that of π' for all states
  - I.e. $\pi \geq \pi'$ iff $v_{\pi}(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$

Optimal Value Functions

- The *optimal* policies are denoted π_*
- The *optimal state-value functions* are denoted v_* and defined:

$$v_*(s) = \max_{\pi} v_\pi(s) \text{ for all } s \in \mathcal{S}$$

- The *optimal action-value functions* are denoted q_* and defined:

$$\begin{aligned} q_*(s,a) &= \max_{\pi} q_\pi(s,a) \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A} \\ &= \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = a, A_t = a] \end{aligned}$$

Computing Value Functions

- Expand out State-Value function

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma v_\pi(s')\big]
\end{aligned}
$$

- Can use dynamic programming techniques to compute optimal value functions, and thus find optimal policies

Value Iteration

- Method for approximating v_*

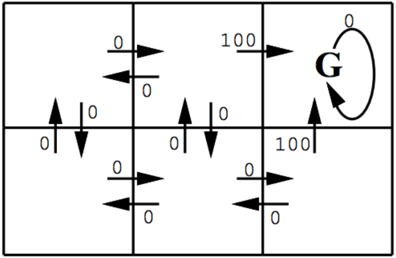
$$v_{k+1}(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = a, A_t = a]$$
$$= \max_a \sum_{s',r} p(s', r | s, a) \big[r + \gamma v_k(s') \big]$$

for all $s \in \mathcal{S}$

- For abitrary v_0, the sequence $\{v_k\}$ will converge to v_* after many iterations

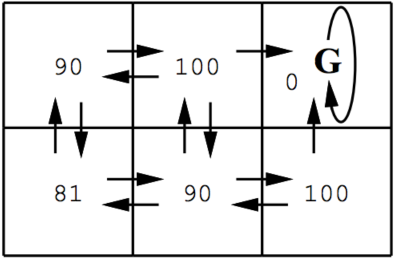
Outline

Board Example



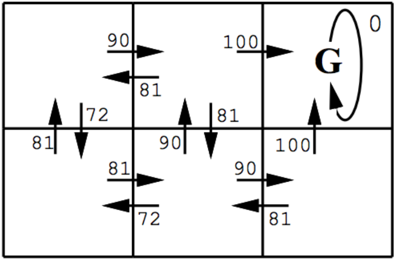
$r(s, a)$ (immediate reward) values

Board Example



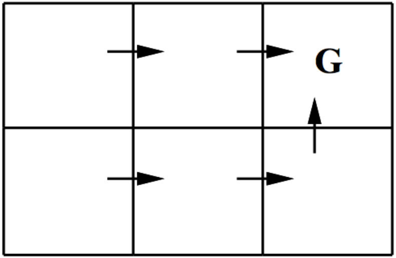
State-Value function $v_*(s)$ values

Board Example



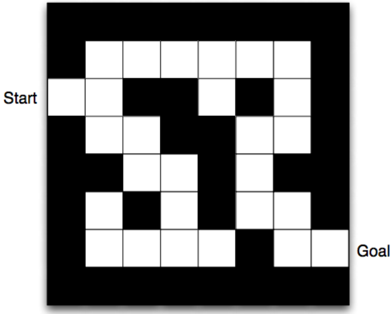
Action-Value function $q_*(s, a)$ values

Board Example



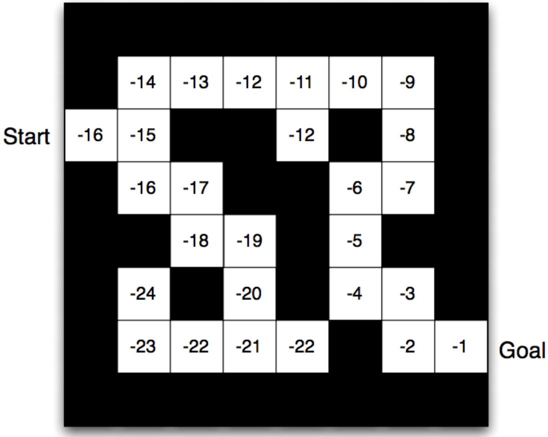
One possible π_*

Maze Example



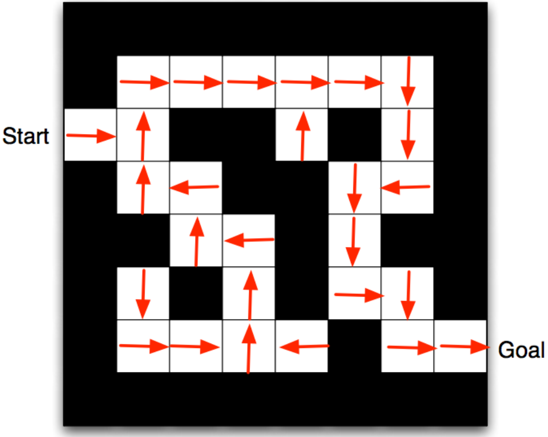
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze Example



- Numbers represent value $v_\pi(s)$ of each state s

Maze Example



- Arrows represent policy $\pi(s)$ for each state s