Machine Learning Lecture 10

Reinforcement Learning

Qian Ma

Sun Yat-sen University

Spring Semester, 2020

Acknowledgement

- A large part of slides in this lecture are originally from
  - Prof. Andrew Ng (Stanford University)
  - Prof. Weinan Zhang (Shanghai Jiao Tong University)

Prof. Andrew Ng
Stanford University

Prof. Weinan Zhang
Shanghai Jiao Tong University

Content

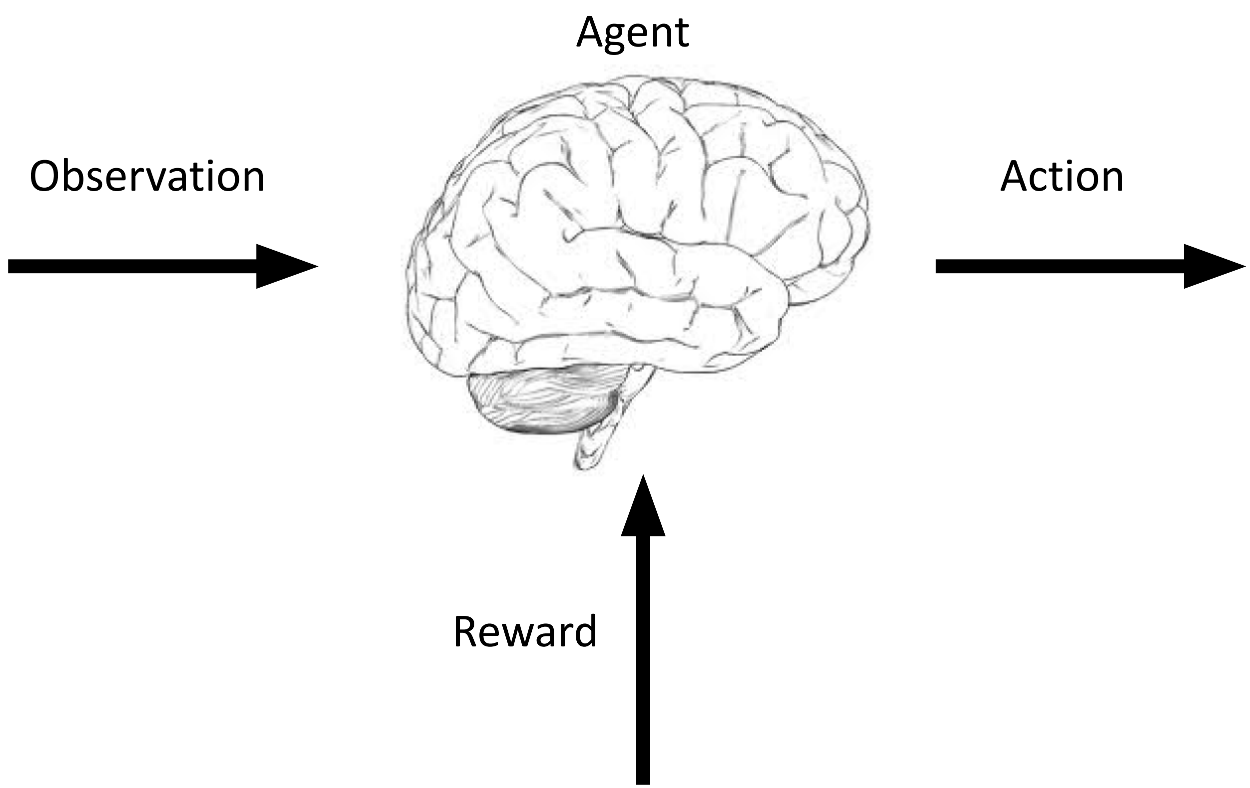
- Introduction to Reinforcement Learning
- Model-based Reinforcement Learning
  - Markov Decision Process
  - Planning by Dynamic Programming
- Model-free Reinforcement Learning
  - Monte-Carlo
  - Temporal Difference

Content

- Introduction to Reinforcement Learning
- Model-based Reinforcement Learning
  - Markov Decision Process
  - Planning by Dynamic Programming
- Model-free Reinforcement Learning
  - Monte-Carlo
  - Temporal Difference

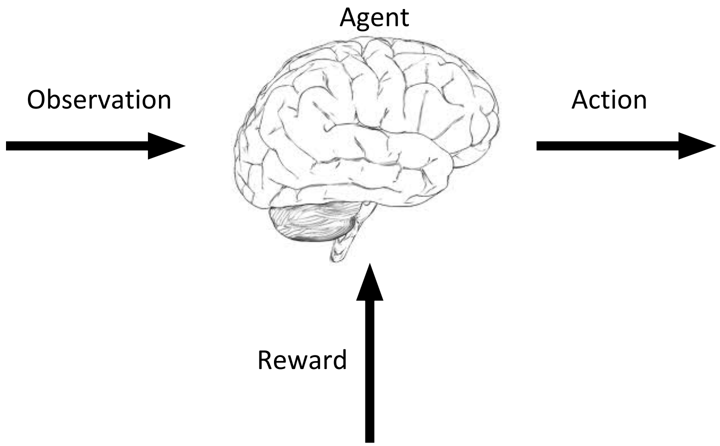
Reinforcement Learning

- Learning from interaction
  - Given the current situation, what to do next in order to maximize utility?



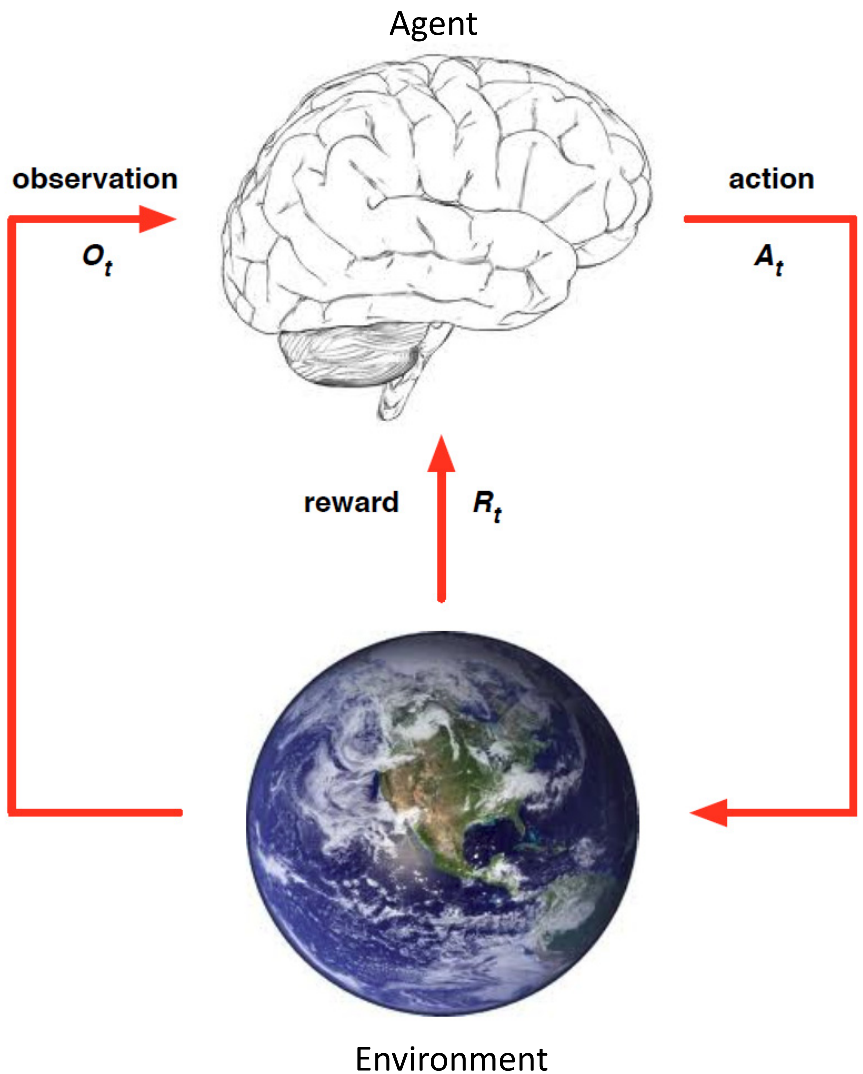
Reinforcement Learning Definition

- A computational approach by learning from interaction to achieve a goal



- Three aspects
  - Sensation: sense the state of the environment to some extent
  - Action: able to take actions that affect the state and achieve the goal
  - Goal: maximize the cumulative reward over time

Reinforcement Learning



- At each step t, the agent
  - Receives observation O_t
  - Receives scalar reward R_t
  - Executes action A_t
- The environment
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at environment step

Elements of RL Systems

- History is the sequence of observations, action, rewards

$$H_t = O_1, R_1, A_1, O_2, R_2, A_2, \ldots, O_{t-1}, R_{t-1}, A_{t-1}, O_t, R_t$$

  - i.e. all observable variables up to time *t*
  - E.g., the sensorimotor stream of a robot or embodied agent
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- State is the information used to determine what happens next (actions, observations, rewards)
- Formally, state is a function of the history

$$S_t = f(H_t)$$

Elements of RL Systems

- Policy is the learning agent's way of behaving at a given time
  - It is a map from state to action
  - Deterministic policy

$$a = \pi(s)$$

  - Stochastic policy

$$\pi(a|s) = P(A_t = a | S_t = s)$$

Elements of RL Systems

- Reward
  - A scalar defining the goal in an RL problem
  - For immediate sense of what is good

- Value function
  - State value is a scalar specifying what is good in the long run
  - Value function is a prediction of the cumulative future reward
    - Used to evaluate the goodness/badness of states (given the current policy)

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

Elements of RL Systems

- Reward
  - A scalar defining the goal in an RL problem
  - For immediate sense of what is good

- Value function
  - State value is a scalar specifying what is a good state in the long run, i.e., the cumulative reward

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

  - Action value is a scalar specifying what is a good action at a specific state in the long run

$$Q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a]$$

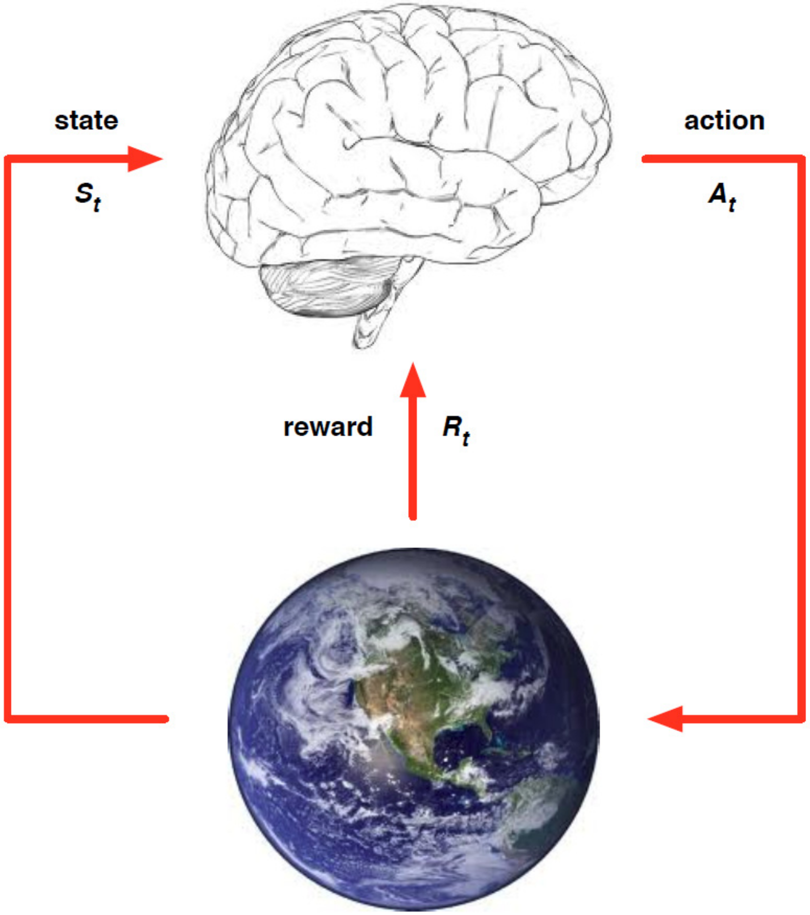
Elements of RL Systems

- A Model of the environment that mimics the behavior of the environment
  - Predict the next state

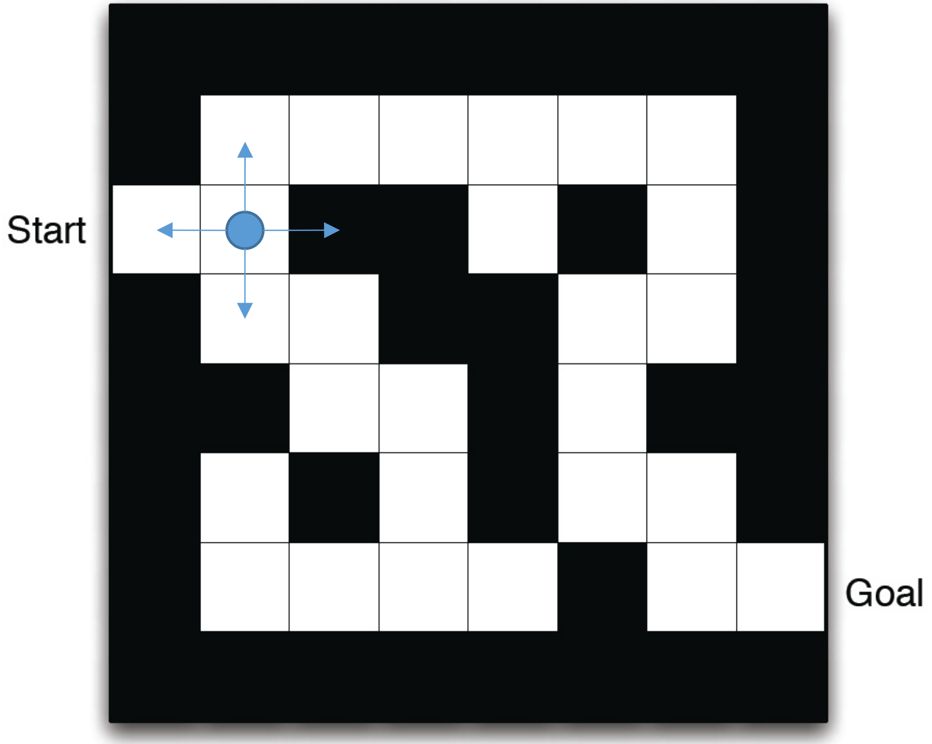
$$\mathcal{P}_{sa}(s') = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

  - Predicts the next (immediate) reward

$$\mathcal{R}_s(a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$



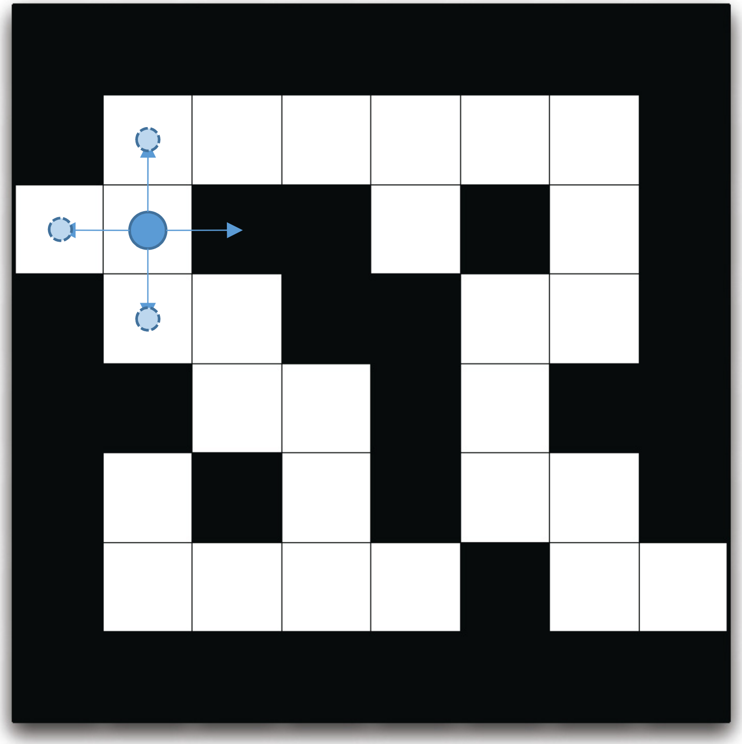
Maze Example



- State: agent's location
- Action: N,E,S,W

Maze Example

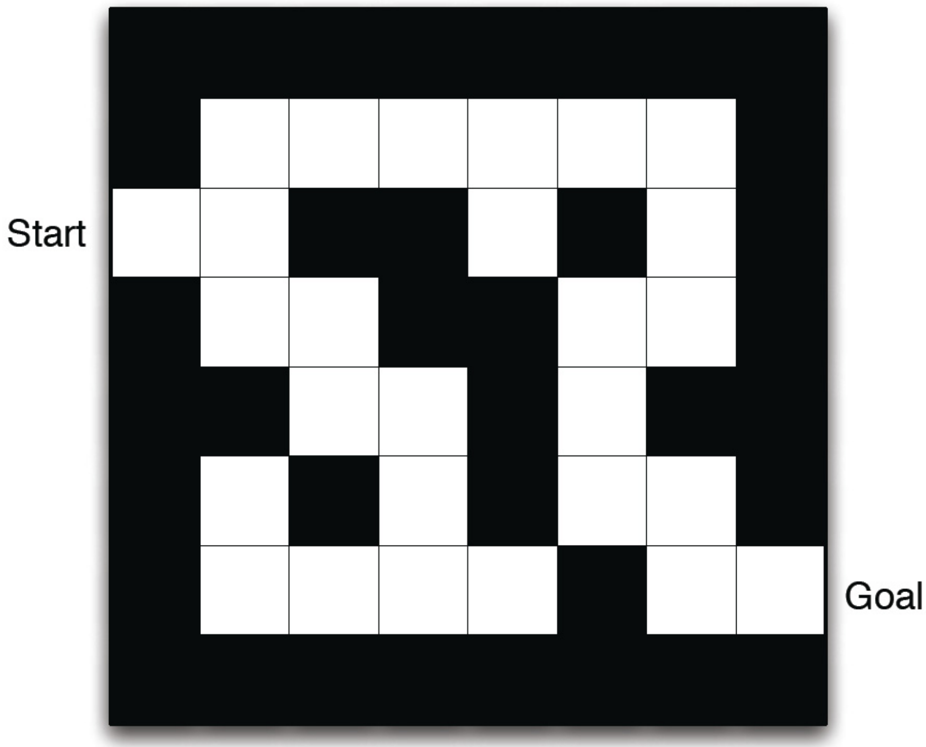
Start



Goal

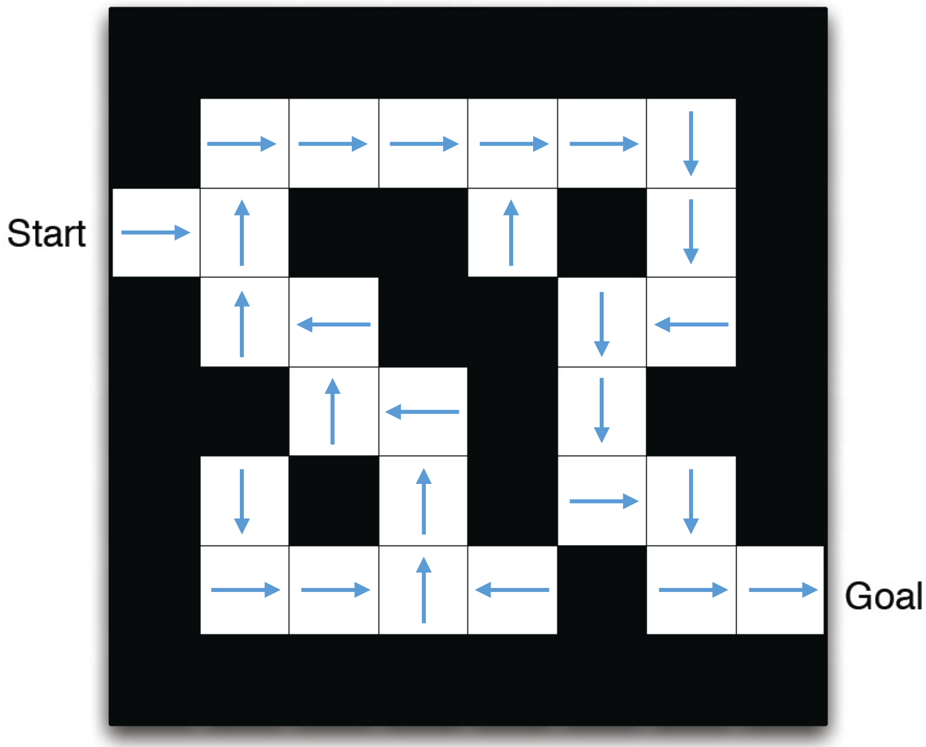
- State: agent's location
- Action: N,E,S,W
- State transition: move to the next grid according to the action
  - No move if the action is to the wall

Maze Example



- State: agent's location
- Action: N,E,S,W
- State transition: move to the next grid according to the action
- Reward: -1 per time step

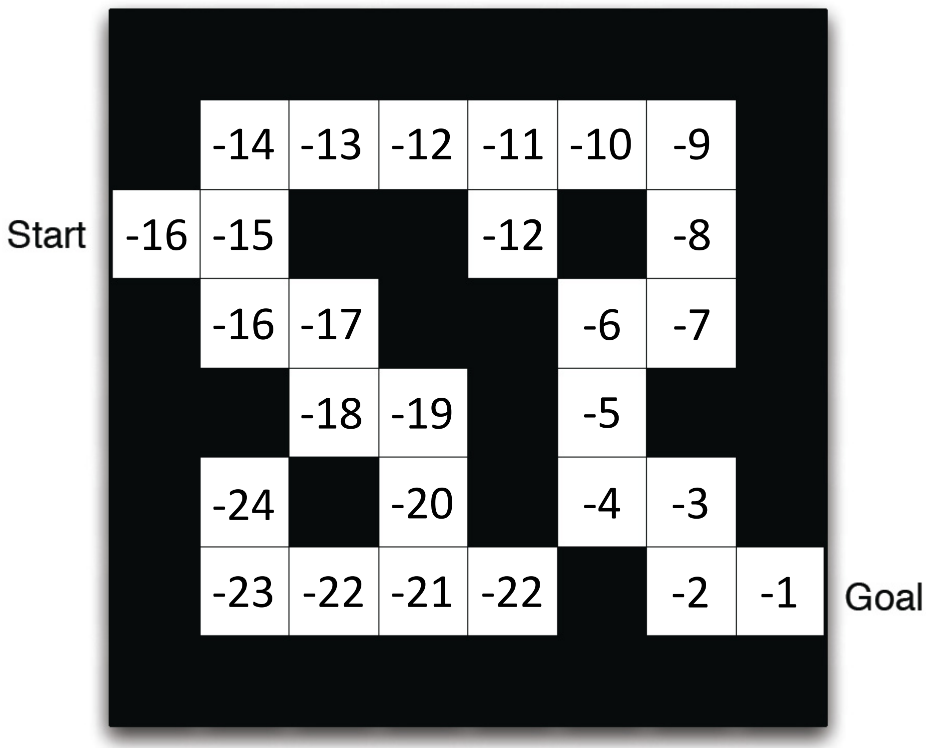
Maze Example



- State: agent's location
- Action: N,E,S,W
- State transition: move to the next grid according to the action
- Reward: -1 per time step

- Given a policy as shown above
  - Arrows represent policy $\pi(s)$ for each state s

Maze Example



- State: agent's location
- Action: N,E,S,W
- State transition: move to the next grid according to the action
- Reward: -1 per time step

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

Categorizing RL Agents

- Model based RL
  - Policy and/or value function
  - Model of the environment
  - E.g., the maze game above, game of Go

- Model-free RL
  - Policy and/or value function
  - No model of the environment
  - E.g., general playing Atari games

Content

- Introduction to Reinforcement Learning
- Model-based Reinforcement Learning
  - Markov Decision Process
  - Planning by Dynamic Programming
- Model-free Reinforcement Learning
  - Monte-Carlo
  - Temporal Difference

Markov Property

"The future is independent of the past given the present"

- Definition
  - A state S_t is Markov if and only if

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

- Properties
  - The state captures all relevant information from the history
  - Once the state is known, the history may be thrown away
  - i.e. the state is sufficient statistic of the future

Markov Decision Process

- A Markov decision process is a tuple (S, A, $\{P_{sa}\}$, γ, R)
- S is the set of states
  - E.g., location in a maze, or current screen in an Atari game
- A is the set of actions
  - E.g., move N, E, S, W, or the direction of the joystick and the buttons
- P_{sa} are the state transition probabilities
  - For each state $s \in S$ and action $a \in A$, P_{sa} is a distribution over the next state in S
- $\gamma \in [0,1]$ is the discount factor for the future reward
- $R : S \times A \mapsto \mathbb{R}$ is the reward function
  - Sometimes the reward is only assigned to state

Markov Decision Process

The dynamics of an MDP proceeds as

- Start in a state s_0
- The agent chooses some action $a_0 \in A$
- The agent gets the reward $R(s_0,a_0)$
- MDP randomly transits to some successor state $s_1 \sim P_{s_0 a_0}$
- This proceeds iteratively

$$s_0 \xrightarrow[R(s_0,a_0)]{a_0} s_1 \xrightarrow[R(s_1,a_1)]{a_1} s_2 \xrightarrow[R(s_2,a_2)]{a_2} s_3 \cdots$$

- Until a terminal state s_T or proceeds with no end
- The total payoff of the agent is

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots$$

Reward on State Only

- For a large part of cases, reward is only assigned to the state
  - E.g., in maze game, the reward is on the location
  - In game of Go, the reward is only based on the final territory
- The reward function $R(s) : S \mapsto \mathbb{R}$
- MDPs proceed

$$s_0 \xrightarrow[R(s_0)]{a_0} s_1 \xrightarrow[R(s_1)]{a_1} s_2 \xrightarrow[R(s_2)]{a_2} s_3 \cdots$$

- cumulative reward (total payoff)

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$

MDP Goal and Policy

- The goal is to choose actions over time to maximize the expected cumulative reward

$$\mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots]$$

- $\gamma \in [0,1]$ is the discount factor for the future reward, which makes the agent prefer immediate reward to future reward
  - In finance case, today's $1 is more valuable than $1 in tomorrow
- Given a particular policy $\pi(s) : S \mapsto A$
  - i.e. take the action $a = \pi(s)$ at state *s*
- Define the value function for π

$$V^\pi(s) = \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi]$$

  - i.e. expected cumulative reward given the start state and taking actions according to π

Bellman Equation for Value Function

- Define the value function for π

$$V^\pi(s) = \mathbb{E}[R(s_0) + \underbrace{\gamma R(s_1) + \gamma^2 R(s_2) + \cdots}_{\gamma V^\pi(s_1)} | s_0 = s, \pi]$$

$$= \underbrace{R(s)}_{\text{Immediate Reward}} + \underbrace{\gamma}_{\text{Time decay}} \sum_{s' \in S} \underbrace{P_{s\pi(s)}(s')}_{\text{State transition}} \underbrace{V^\pi(s')}_{\text{Value of the next state}}$$

Bellman Equation

Optimal Value Function

- The optimal value function for each state *s* is best possible sum of discounted rewards that can be attained by any policy

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

- The Bellman's equation for optimal value function

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s')$$

- The optimal policy

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^*(s')$$

- For every state *s* and every policy π

$$V^*(s) = V^{\pi^*}(s) \geq V^{\pi}(s)$$

Value Iteration & Policy Iteration

- Note that the value function and policy are correlated

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s')$$

$$\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^\pi(s')$$

- It is feasible to perform iterative update towards the optimal value function and optimal policy
  - Value iteration
  - Policy iteration

Value Iteration

- For an MDP with finite state and action spaces

$$|S| < \infty, |A| < \infty$$

- Value iteration is performed as

  1. For each state *s*, initialize *V*(*s*) = 0.
  2. Repeat until convergence {

     For each state, update

$$V(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s')V(s')$$

  }

- Note that there is no explicit policy in above calculation

Policy Iteration

- For an MDP with finite state and action spaces

$$|S| < \infty, |A| < \infty$$

- Policy iteration is performed as

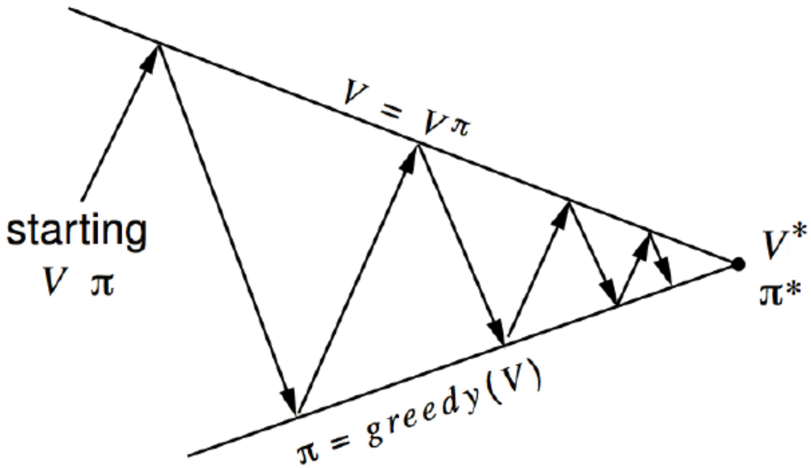
1. Initialize π randomly
2. Repeat until convergence {
   a) Let $V := V^{\pi}$
   b) For each state, update

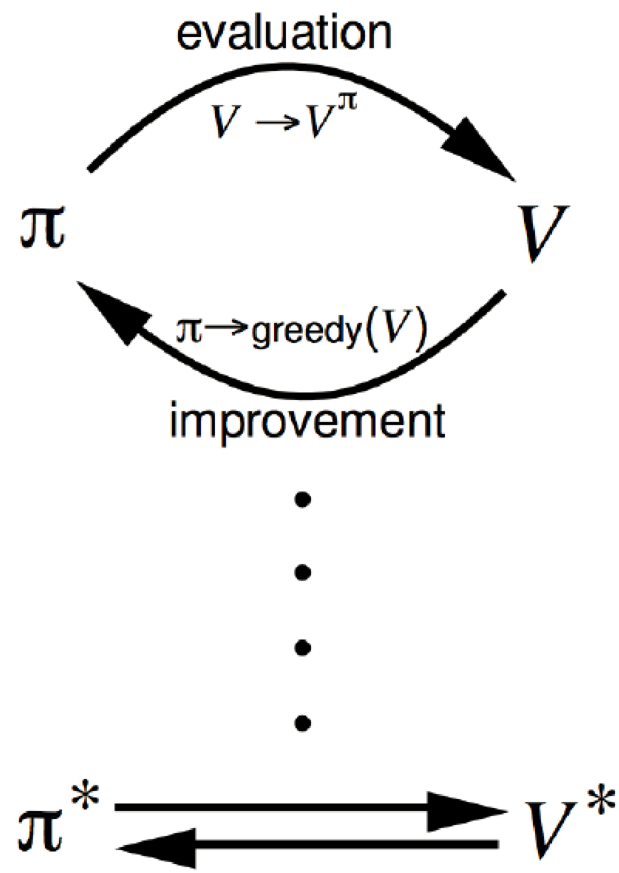
$$\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s')V(s')$$

   }

- The step of value function update could be time-consuming

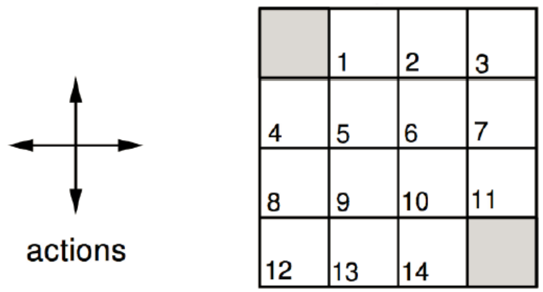
Policy Iteration





- Policy evaluation
  - Estimate V^π
  - Iterative policy evaluation
- Policy improvement
  - Generate $\pi' \geq \pi$
  - Greedy policy improvement

Evaluating a Random Policy in a Small Gridworld



r = -1
on all transitions

- Undiscounted episodic MDP (γ=1)
- Nonterminal states 1,…,14
- Two terminal states (shaded squares)
- Actions leading out of the grid leave state unchanged
- Reward is -1 until the terminal state is reached
- Agent follows a uniform random policy

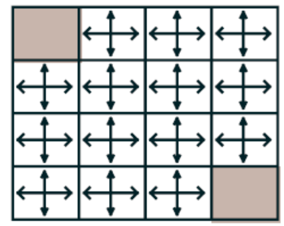
$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Evaluating a Random Policy in a Small Gridworld

V_k for the random policy | Greedy policy w.r.t. V_k

K=0

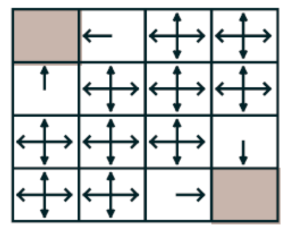
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



Random policy

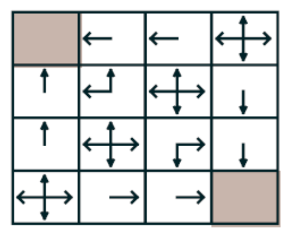
K=1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



K=2

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



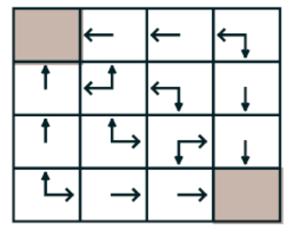
Evaluating a Random Policy in a Small Gridworld

V_k for the random policy

Greedy policy w.r.t. V_k

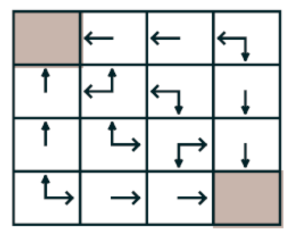
K=3

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0



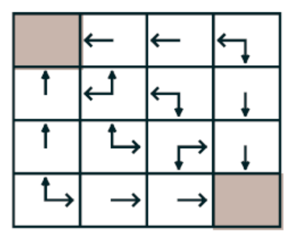
K=10

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



$K=\infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



$$V := V^{\pi}$$

Optimal policy