PAI Review Session

Reinforcement Learning

Johannes Kirschner

January 24, 2019

Markov Decision Processes

MDP:

S (finite) set of states

A (finite) set of actions

$P(s'|s,a)$ probability of going from s to s' with action a.

$R(s)$ or $R(s,a)$ or $R(s,a,s')$: Reward function

Markov Decision Processes

MDP:

S (finite) set of states

A (finite) set of actions

$P(s'|s,a)$ probability of going from s to s' with action a.

$R(s)$ or $R(s,a)$ or $R(s,a,s')$: Reward function

Policy:

$\pi(a|s)$ probability of selecting action a in state s

Value functions - or when do we get the reward?

If we run a policy π starting from a state s_0, we get a sequence:

$s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots$

Value functions - or when do we get the reward?

If we run a policy π starting from a state s_0, we get a sequence:

$s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots$

$a_i \sim \pi(\cdot|s_i)$

$s_{i+1} \sim p(\cdot|s_i, a_i)$

$r_i = r(s_i, a_i, s_{i+1})$

Value functions - or when do we get the reward?

If we run a policy π starting from a state s_0, we get a sequence:

$s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots$

$a_i \sim \pi(\cdot|s_i)$

$s_{i+1} \sim p(\cdot|s_i, a_i)$

$r_i = r(s_i, a_i, s_{i+1})$

A tuple (s_i, a_i, r_i, s_{i+1}) is called a *transition*.

Value functions - or when do we get the reward?

If we run a policy π starting from a state s_0, we get a sequence:

$s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots$

$a_i \sim \pi(\cdot|s_i)$

$s_{i+1} \sim p(\cdot|s_i, a_i)$

$r_i = r(s_i, a_i, s_{i+1})$

A tuple (s_i, a_i, r_i, s_{i+1}) is called a *transition*.

Value function: $V^\pi(s_0) = \mathbb{E}[\sum_{i=0}^{T} \gamma^i r_i] = \mathbb{E}[\sum_{i=0}^{T} \gamma^i r(s_i, a_i, s_{i+1})]$

Value functions - or when do we get the reward?

If we run a policy π starting from a state s_0, we get a sequence:

$s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots$

$a_i \sim \pi(\cdot|s_i)$

$s_{i+1} \sim p(\cdot|s_i, a_i)$

$r_i = r(s_i, a_i, s_{i+1})$

A tuple (s_i, a_i, r_i, s_{i+1}) is called a *transition*.

Value function: $V^\pi(s_0) = \mathbb{E}[\sum_{i=0}^{T} \gamma^i r_i] = \mathbb{E}[\sum_{i=0}^{T} \gamma^i r(s_i, a_i, s_{i+1})]$

After picking action a_i, we *observe* (r_i, s_{i+1}).

Episodes

Value function: $V^{\pi}(s_0) = \mathbb{E}[\sum_{i=0}^{T} \gamma^i r_i]$

Episodes

Value function: $V^{\pi}(s_0) = \mathbb{E}[\sum_{i=0}^{T} \gamma^i r_i]$

- $T = \infty$: Continuous setting (agent interacts forever with env.)

Episodes

Value function: $V^{\pi}(s_0) = \mathbb{E}[\sum_{i=0}^{T} \gamma^i r_i]$

- $T = \infty$: Continuous setting (agent interacts forever with env.)
- $T =$ finite: Episodic setting (game ends after T steps)
  - *can use multiple episodes for learning*

Episodes

Value function: $V^\pi(s_0) = \mathbb{E}[\sum_{i=0}^{T} \gamma^i r_i]$

- $T = \infty$: Continuous setting (agent interacts forever with env.)
- $T =$ finite: Episodic setting (game ends after T steps)
  - *can use multiple episodes for learning*

Terminal states: Special states where game 'ends'.

- Can replace by additional, 'looping' state with no reward

Reinforcement Learning

Optimal Policy: π^* is optimal if $V^{\pi^*} = \max_\pi V^\pi$

Reinforcement Learning

Optimal Policy: π^* is optimal if $V^{\pi^*} = \max_\pi V^\pi$

Planning

$\rightarrow$ find π^* given the MDP

- $\triangleright$ Value iteration
- $\triangleright$ Policy iteration

Learning

$\rightarrow$ find π^* with unknown MDP

- $\triangleright$ *Model based* RL (R-max)
- $\triangleright$ *Model free* RL (Q-learning, Policy Search)

Model-Based RL

- Learn MDP, then use it to find optimal policy
- Need an exploration policy (random, Rmax) to gather data

Rmax

- For simplicity, assume we know the transitions.
  - For general version refer to lecture slides.

Rmax

- For simplicity, assume we know the transitions.
  - For general version refer to lecture slides.

Algorithm (Rmax) (episodic setting)

1: $\hat{R}_0(s, a) = R_{max}$
2: For episodes i=1,2,...
3: Compute optimal policy π_i in MDP with $\hat{R}_{i-1}$
4: Use policy π_i for episode i
5: Use data to update $\hat{R}_i$

Rmax

- For simplicity, assume we know the transitions.
  - For general version refer to lecture slides.

Algorithm (Rmax) (episodic setting)

1: $\hat{R}_0(s, a) = R_{max}$
2: For episodes i=1,2,...
3: Compute optimal policy π_i in MDP with $\hat{R}_{i-1}$
4: Use policy π_i for episode i
5: Use data to update $\hat{R}_i$

- *Systematically rules out suboptimal policies*

Q-learning

- Q-function $Q^\pi(s, a) = \mathbb{E}[r(s, a, s') + \gamma V^\pi(s')]$
- Q*-function: Q^{π^*}

Q-learning

- Q-function $Q^{\pi}(s,a) = \mathbb{E}[r(s,a,s') + \gamma V^{\pi}(s')]$
- Q*-function: Q^{π^*}
- Bellman's Theorem: Policy is optimal $\iff$ it is greedy on Q
  - $\pi^*(a|s) = \arg\max_a Q^{\pi}(a,s)$
  - $V^{\pi^*}(s) = \max_a Q^*(s,a)$

Q-learning

- Q-function $Q^\pi(s, a) = \mathbb{E}[r(s, a, s') + \gamma V^\pi(s')]$
- Q*-function: Q^{π^*}

- Bellman's Theorem: Policy is optimal $\iff$ it is greedy on Q
  - $\pi^*(a|s) = \arg\max_a Q^\pi(a, s)$
  - $V^{\pi^*}(s) = \max_a Q^*(s, a)$
  - $\rightarrow$ If we know the Q^* function, we know the optimal policy

Q-learning

- Q-function $Q^\pi(s,a) = \mathbb{E}[r(s,a,s') + \gamma V^\pi(s')]$
- Q*-function: Q^{π^*}
- Bellman's Theorem: Policy is optimal $\iff$ it is greedy on Q
  - $\pi^*(a|s) = \arg\max_a Q^\pi(a,s)$
  - $V^{\pi^*}(s) = \max_a Q^*(s,a)$
  - $\rightarrow$ If we know the Q^* function, we know the optimal policy

Q-learning $=$ estimating Q^* from transitions $\{(s,a,r,s')\}$
Update-rule: $Q_{i+1}(s,a) = (1-\alpha)Q_i(s,a) + \alpha(r + \max_{a'} \gamma Q_i(s',a'))$