Alice Gao Lecture Notes | Self-Study

Shikhar Saxena

January 20, 2023

Contents

Alice Gao Lecture Notes [1]

MDP

- Revenue function $R(s, a, s')$
  - ⋆ Here Revenue function might not depend on all of the parameters.

Rewards

Assume $R(s)$: the reward of entering state s.

- Total Reward
  - ⋆ Sum of Rewards at each time step.
  - ⋆ If sum is infinite can't compare policies.
- Average Reward
  - ⋆ Divide *total reward* by the number of time steps n, such that $\lim_{n\to\infty}$.
  - ⋆ For finite total reward, this'll always be zero.
- Discounted Reward
  - ⋆ Discount factor $0 \leq \gamma < 1$
  - ⋆ Discount helps us set a preference over our rewards, in past and in future per se.

Expected Utility of a Policy

$R(s)$ same as assumed before.

$V^{\pi}(s)$: Expected utility of **entering** state s, following policy π.

[1]Reference: Lec18 and Lec19

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)V^*(s') \tag{1}$$

Q is when we are already **in** a state and take an action. Essentially, it tells us the expected utility of **leaving** this state (by taking action a).

$\pi^*(s) = arg\max_a Q^*(s,a)$

Relationship between V and Q

$$V^*(s) = R(s) + \gamma \max_a Q^*(s,a) \tag{2}$$

From (1) and (2) we get the Bellman equation:

$$V^*(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)V^*(s') \tag{3}$$

Goal states usually have a value already assigned to them. Also, unreachable states have a value as well (zero, mostly). For non-goal states, we'll have to solve the Bellman equation to get the optimal value function.

Computing Bellman Equations Efficiently

We can't compute them efficiently in-general since max function is non-linear. So this is a non-linear problem.

Instead, we use an approximation approach, *Value Iteration* Approach.

Value Iteration Algorithm

$V_i(s)$: values for i-th iteration

1. Start with arbitrary initial values $V_0(s)$
2. At the i-th iteration, compute $V_{i+1}(s)$ (sort of like gradient-descent update rule). Old values are plugged in to get the new values for next iteration.
3. Terminate when $\max_s |V_i(s) - V_{i+1}(s)|$ is small enough

If the updates are applied infinitely often, this algorithm is guaranteed to converge to optimal values.