COMP6247: Reinforcement and Online Learning

Mahesan Niranjan

School of Electronics and Computer Science
University of Southampton

Markov Decision Processes, Dynamic Programming
Chapters 3 and 4, Sutton and Barto

Spring Semester 2020/21

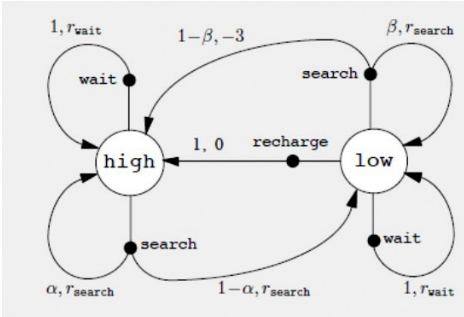
Foundations

- $S_0,\ A_0,\ R_1,\ S_1,\ A_1,\ R_2,\ S_2,\ A_2,\ R_3 \ldots$
- Dynamics of the Markov Decision Process
  $p\,(s', r \mid s, a) = Pr\,(S_t = s', R_t = r \,|S_{t-1} = s, A_{t-1} = a)$
- p is a function of four spaces: $\mathcal{S} \times \mathcal{R} \times \mathcal{A} \times \mathcal{S}$, mapping to $[0, 1]$
- $\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p\,(s', r \mid s, a) \; = \; 1 \;\; \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ (Eqn 3.3 in S & B)
- This being a joint probability, we can extract other probabilities:
  - State transition: $p(s'|s, a) \; = \; \sum_{r \in \mathcal{R}} p(s', r|s, a)$
  - Expected Reward: $r(s, a) \; = \; \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a)$
  - Expected reward for state - action - next state
    $p(r, a, s') \; = \; \sum_{r \in \mathcal{R}} \frac{p(s', r|s, a)}{p(s'|s, a)}$

Example: Recycling Robot

Page 52, Sutton and Barto

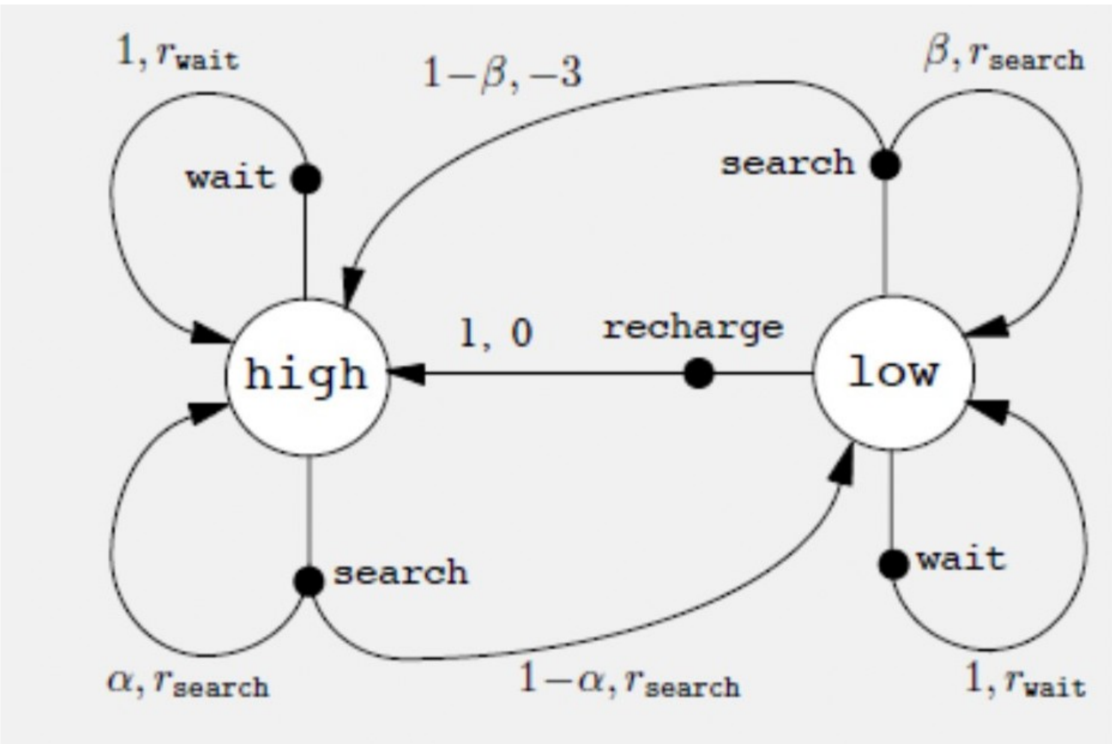
s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1-\alpha$	$r_{\texttt{search}}$
low	search	high	$1-\beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-



- Robot collects garbage cans
- Finite battery life, needs to recharge
- Search to collect or wait for can
- Reward when can collected

Example: Recycling Robot

State Transition Diagram



Example: Recycling Robot

States, Actions and Rewards

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1 - \alpha$	$r_{\texttt{search}}$
low	search	high	$1 - \beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-

- Homework: Read Example 3.3 in detail.

Returns and Episodes

- Of interest: expected return (not immediate reward)

$$G_t = R_{t+1} + R_{t+2} + \ldots + R_T$$

- Assumed *episodes*: start to finish subsequences
- Each episode ends in *terminal state*
- Continuous tasks

$$G_t = R_{t+1} + \gamma R_{t+2}, + \gamma^2 R_{t+3}, + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- Discounting: $0 \leq \gamma \leq 1$
- Recursive Structure:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \ldots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma^{R}_{t+3} + \gamma^2 R_{t+4} + \ldots \right) \\ &= R_{t+1} + \gamma\, G_{t+1} \qquad \text{Eqn.3.9, S\&B} \end{aligned}$$

Value Functions

- Value function of a state ($v_\pi(s)$), under a policy π:

$$\begin{aligned} v_\pi(s) &= E_\pi\left[G_t|S_t = s\right] \\ &= E_\pi\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right] \end{aligned}$$

- Note: *total*, *expected*
- If you start from this state (s at time t) and execute policy π, what is your expected total reward?
- Similarly, value of action (at a state):

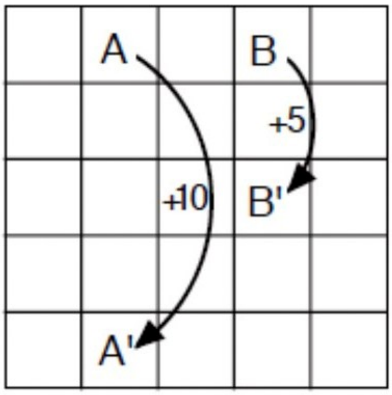
$$\begin{aligned} q_\pi(s,a) &= E_\pi\left[G_t|S_t = s,\ A_t = a\right] \\ &= E_\pi\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s,\ A_t = a\right] \end{aligned}$$

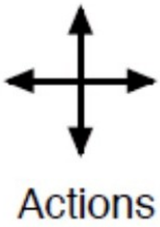
Recursive Structure and Bellman Equations

$$\begin{aligned} v(s) &= E_\pi\left[G_t|S_t = s\right] \\ &= E_\pi\left[R_{t+1} + \gamma G_{t+1}|S_t = s\right] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma E_\pi\left[G_{t+1}|S_{t+1} = s'\right]\right] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma\, v_\pi(s')\right] \qquad \forall s \end{aligned}$$

- $v(s)$ written in terms of expectation over all actions and $v(s')$, the values of resulting states.
- Simultaneous equations with $v(s)$ as unknowns.

Gridworld Example





3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

- Rewards: -1 for going off grid; +10, +5 at A and B; 0 for all others
- Homework: Exercise 3.15, Page 61, S & B: "Adding a constant to all rewards does not change the relative values of states".

Optimal Policies and Value Functions

- $v_*(s) = \max_\pi v_\pi(s)$
- Which policy gives us largest value at every state?
- Similarly: $q_*(s, a) = \max_\pi q_\pi(s, a)$
- $q_*(s, a) = E\left[R_{t+1} + \gamma v_*(S_{t+1}) \,|\, S_t = s, A_t = a\right]$

Bellman Equations for Optimal Value Functions

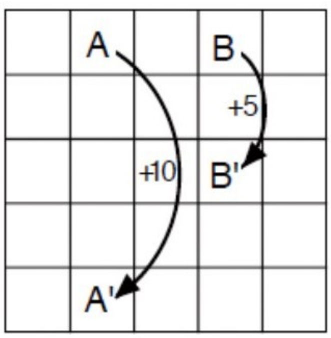
- For optimal state values

$$\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a E_{\pi_*}[G_t \mid S_t = s,\ A_t = a] \\
&= \max_a E_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s,\ A_t = a] \\
&= \max_a E\,[R_{t+1} + \gamma\, v_*(S_{t+1}) \mid S_t = s,\ A_t = a] \\
&= \max_a \sum_{s',r} p(s',r|s,a)\ \left[r + \gamma\, v_*(s')\right]
\end{aligned}$$

- Similarly for optimal action values

$$\begin{aligned}
q_*(s,a) &= E\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a') \mid S_t = s,\ A_t = a\right] \\
&= \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} q_*(s',a')\right]
\end{aligned}$$

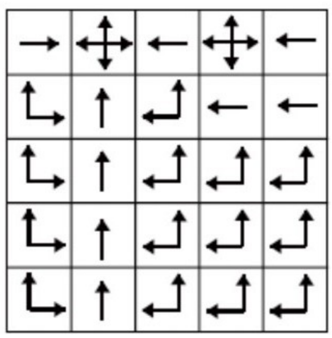
Gridworld Example



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

Recycling Robot: Bellman Equations

Example 3.9, S & B

$$\begin{aligned}
v_*(\mathtt{h}) &= \max\left\{\begin{array}{l} p(\mathtt{h}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{l}) + \gamma v_*(\mathtt{l})], \\ p(\mathtt{h}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{l}) + \gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} \alpha[r_\mathtt{s} + \gamma v_*(\mathtt{h})] + (1-\alpha)[r_\mathtt{s} + \gamma v_*(\mathtt{l})], \\ 1[r_\mathtt{w} + \gamma v_*(\mathtt{h})] + 0[r_\mathtt{w} + \gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} r_\mathtt{s} + \gamma[\alpha v_*(\mathtt{h}) + (1-\alpha) v_*(\mathtt{l})], \\ r_\mathtt{w} + \gamma v_*(\mathtt{h}) \end{array}\right\}.
\end{aligned}$$

$$v_*(\mathtt{l}) = \max\left\{\begin{array}{l} \beta r_\mathtt{s} - 3(1-\beta) + \gamma[(1-\beta)v_*(\mathtt{h}) + \beta v_*(\mathtt{l})], \\ r_\mathtt{w} + \gamma v_*(\mathtt{l}), \\ \gamma v_*(\mathtt{h}) \end{array}\right\}.$$

$$v_*(s) \;=\; \max_a \sum_{s',r} p(s',r|s,a)\;\left[r \;+\; \gamma\, v_*(s')\right]$$

Dynamic Programming

- Family of algorithms
- Policy Evaluation – State value function $v(s)$ for given policy π
- Bellman equations:

$$v_\pi(s) \;=\; \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\;\left[r \;+\; \gamma v_\pi(s')\right]$$

- Turn the above into an iterative assignment *iterative policy evaluation*:

$$v_{k+1}(s) \;=\; \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\;\left[r \;+\; \gamma v_k(s')\right]$$

- Variations in implementing updates
  - *expected* updates (after seeing all future states)
  - *sweeps* (in place) overwrite at every state visited

Iterative Policy Evaluation Algorithm

S & B, Page 75

- Evaluating a policy $\longrightarrow$ What is $v(s)$ under this policy?

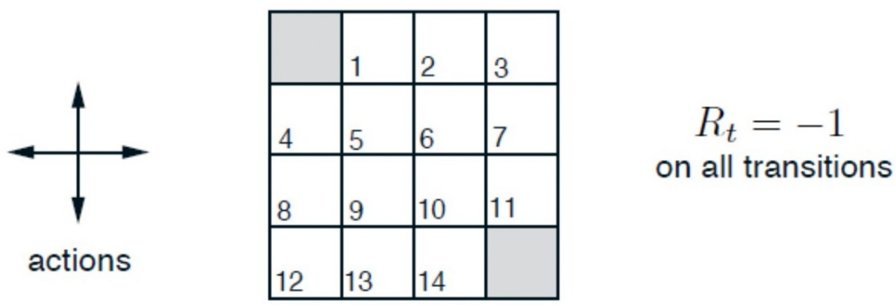
Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

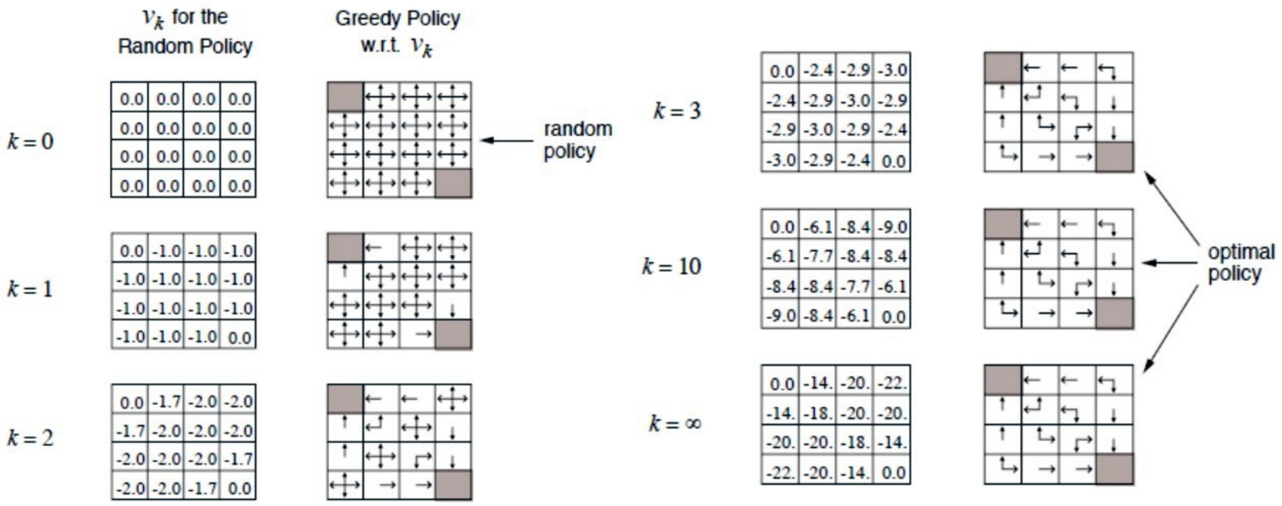
Gridworld Example

S & B, Page 76



- $p(6, -1|5, \text{right}) = 1$, $p(7, -1|7, \text{right}) = 1$, $p(10, r|5, \text{right}) = 0$
- Undiscounted episodic tasks, terminal states at (shaded) corners.
- Iterate to improve policy

Gridworld Example



$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + v_k(s') \right]$$

Policy Iteration Algorithm

Page 80, S & B

Evaluate, improve...

$$\pi_0 \to v_{\pi_0} \to \pi_1 \to v_{\pi_1} \to \ldots \pi_* \to v_*$$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   - $\Delta \leftarrow 0$
   - For each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - $a \leftarrow \pi(s)$
   - $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
   - If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, then stop and return V and π; else go to 2

Value Iteration Algorithm

Page 83, S & B

$$
\begin{aligned}
v_{k+1}(s) &= \max_a E\left[R_{t+1} + \gamma\, v_k(S_{t+1})\,|S_t = s,\, A_t = a\right] \\
&= \max_a \sum_{s',r} p(s', r|s, a)\ \left[r + \gamma v_k(s')\right]
\end{aligned}
$$

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$

Summary

- Markov Decision Processes (MDP) in which we have knowledge of the environment
- Task is to evaluate a given policy and to find an optimal policy
- Solved by dynamic programming
- **Next: What if the environment is not known?**