MDPs and the RL Problem

CMSC 471 – Fall 2012

Class #25 – Tuesday, November 26

Russell & Norvig Chapter 21.1-21.3

Thanks to Rich Sutton and Andy Barto for the use of their slides (modified with additional in-class exercises)

Today's Class

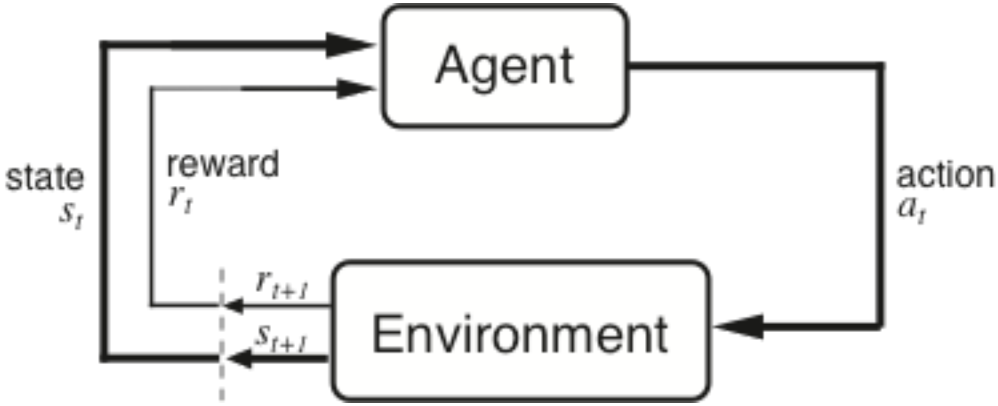
- Extra Credit
- HW6
- Project Deadlines/Milestones
- Reinforcement Learning
- Dry Run #1

The Reinforcement Learning Problem

Objectives:

- reinforce/expand concepts of value and policy iteration, including discounting of future rewards;
- present idealized form of the RL problem for which we have precise theoretical results;
- introduce key components of the mathematics: value functions and Bellman equations;
- describe trade-offs between applicability and mathematical tractability;

The Agent-Environment Interface



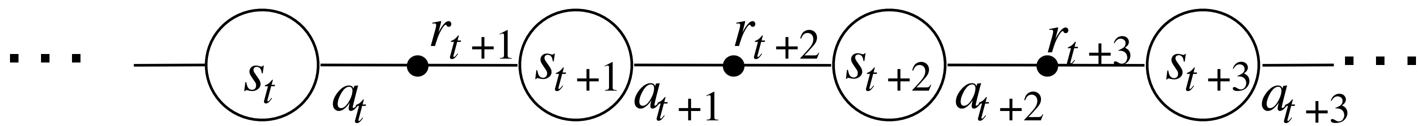
Agent and environment interact at discrete time steps : $t = 0, 1, 2, \ldots$

Agent observes state at step t: $s_t \in S$

produces action at step t: $a_t \in A(s_t)$

gets resulting reward : $r_{t+1} \in \Re$

and resulting next state : s_{t+1}



The Agent Learns a Policy

Policy at step t, π_t :

a mapping from states to action probabilities

$\pi_t(s,a) =$ probability that $a_t = a$ when $s_t = s$

- Reinforcement learning methods specify how the agent changes its policy as a result of experience.
- Roughly, the agent's goal is to get as much reward as it can over the long run.

Returns

Suppose the sequence of rewards after step t is :

$$r_{t+1}, r_{t+2}, r_{t+3}, \ldots$$

What do we want to maximize?

In general,

we want to maximize the **expected return**, $E\{R_t\}$, for each step t.

Note: R&N use R for one-step reward instead of r

Episodic tasks: interaction breaks naturally into episodes, e.g., plays of a game, trips through a maze.

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T,$$

where T is a final time step at which a **terminal state** is reached, ending an episode.

Returns for Continuing Tasks

Continuing tasks: interaction does not have natural episodes.

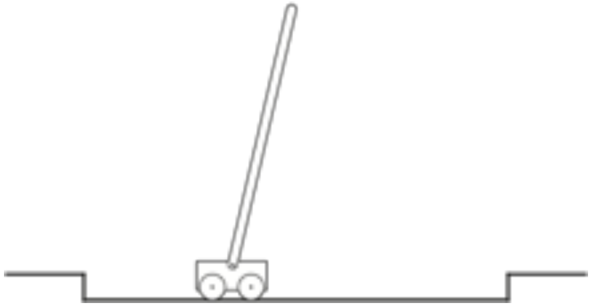
Discounted return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

where $\gamma, 0 \leq \gamma \leq 1$, is the **discount rate**.

shortsighted $0 \leftarrow \gamma \rightarrow 1$ farsighted

An Example



Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track.

As an **episodic task** where episode ends upon failure:

reward = +1 for each step before failure

$\Rightarrow$ return = number of steps before failure

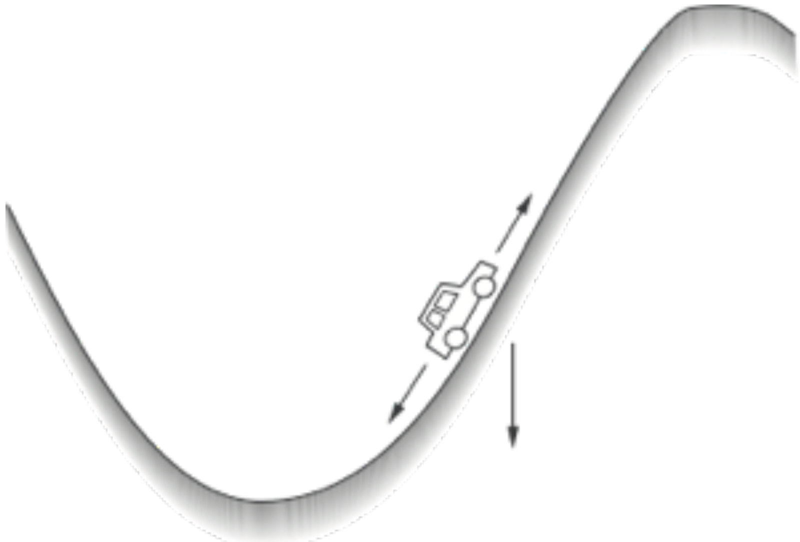
As a **continuing task** with discounted return:

reward = –1 upon failure; 0 otherwise

$\Rightarrow$ return = $-\gamma^k$, for k steps before failure

In either case, return is maximized by avoiding failure for as long as possible.

Another Example



Get to the top of the hill as quickly as possible.

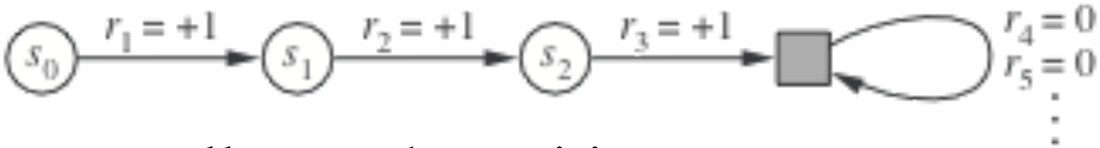
reward $= -1$ for each step where **not** at top of hill

$\Rightarrow$ return $=$ $-$ number of steps before reaching top of hill

Return is maximized by minimizing number of steps to reach the top of the hill.

A Unified Notation

- In episodic tasks, we number the time steps of each episode starting from zero.
- We usually do not have to distinguish between episodes, so we write s_t instead of $s_{t,j}$ for the state at step *t* of episode *j*.
- Think of each episode as ending in an absorbing state that always produces a reward of zero:



- We can cover all cases by writing

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

where γ can be 1 only if a zero-reward absorbing state is always reached.

Value Functions

- The **value of a state** is the expected return starting from that state; depends on the agent's policy:

State - value function for policy π :

$$V^{\pi}(s) = E_{\pi}\left\{R_t \mid s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}$$

- The **value of taking an action in a state under policy** π is the expected return starting from that state, taking that action, and thereafter following π :

Action - value function for policy π :

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t \mid s_t = s, a_t = a\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}$$

Bellman Equation for a Policy π

The basic idea:

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \cdots \\ &= r_{t+1} + \gamma\left(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} \cdots\right) \\ &= r_{t+1} + \gamma R_{t+1} \end{aligned}$$

So:

$$\begin{aligned} V^\pi(s) &= E_\pi\left\{R_t \middle| s_t = s\right\} \\ &= E_\pi\left\{r_{t+1} + \gamma V\left(s_{t+1}\right) \middle| s_t = s\right\} \end{aligned}$$

Or, without the expectation operator:

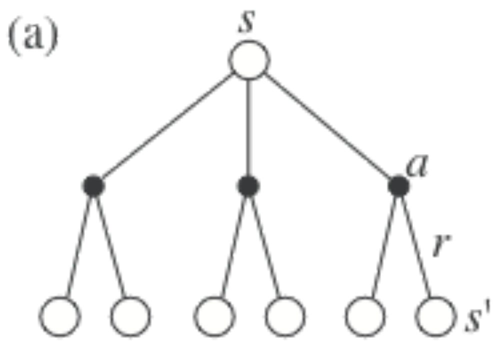
$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[r_{ss'}^a + \gamma V^\pi(s') \right]$$

More on the Bellman Equation

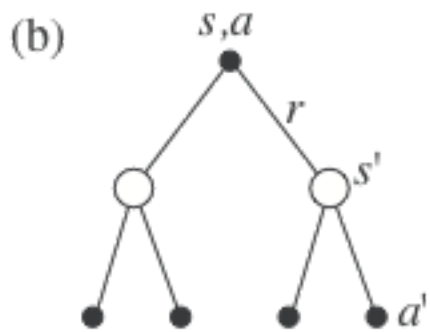
$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s') \right]$$

This is a set of equations (in fact, linear), one for each state.
The value function for π is its unique solution.

Backup diagrams:



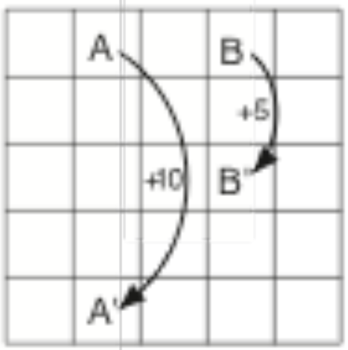
for V^π



for Q^π

Gridworld

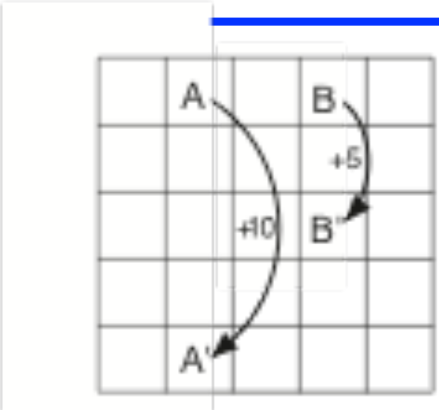
- Actions: `north`, `south`, `east`, `west`; deterministic.
- In special states A and B, all actions move to A' and B', with reward +10 and +5, respectively.
- If would take agent off the grid: no move but reward = –1
- All other actions have the expected effect and produce reward = 0, except actions that move agent out of special states A and B as shown.



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

State-value function for equiprobable random policy;
$\gamma = 0.9$

Verifying the Value Function



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

State-value function for equiprobable random policy; $\gamma = 0.9$

- Recall that:

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^\pi(s') \right]$$

- In state A, all actions take the agent to state A' and have reward 10.

  Exercise: Verify the state-value function shown for A

- *Exercise: Verify the state-value function for the state at the lower left ($V^\pi = -1.9$)*

Optimal Value Functions

- For finite MDPs, policies can be **partially ordered**:

$$\pi \geq \pi' \quad \text{if and only if} \quad V^{\pi}(s) \geq V^{\pi'}(s) \;\; \text{for all } s \in S$$

- There is always at least one (and possibly many) policies that is better than or equal to all the others. This is an **optimal policy**. We denote them all π *.
- Optimal policies share the same **optimal state-value function**:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad \text{for all } s \in S$$

- Optimal policies also share the same **optimal action-value function**:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \;\; \text{for all } s \in S \text{ and } a \in A(s)$$

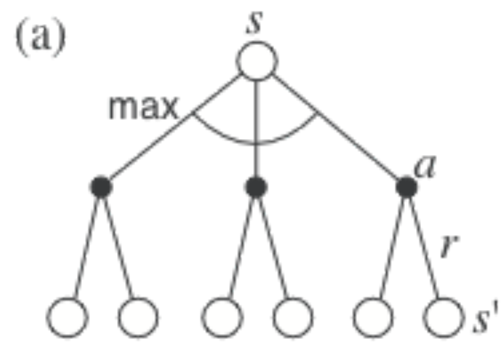
This is the expected return for taking action *a* in state *s* and thereafter following an optimal policy.

Bellman Optimality Equation for *V**

The value of a state under an optimal policy must equal the expected return for the best action from that state:

$$
\begin{aligned}
V^*(s) &= \max_{a \in A(s)} Q^{\pi^*}(s,a) \\
&= \max_{a \in A(s)} E\left\{ r_{t+1} + \gamma V^*(s_{t+1}) \,\middle|\, s_t = s, a_t = a \right\} \\
&= \max_{a \in A(s)} \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^*(s') \right]
\end{aligned}
$$

The relevant backup diagram:



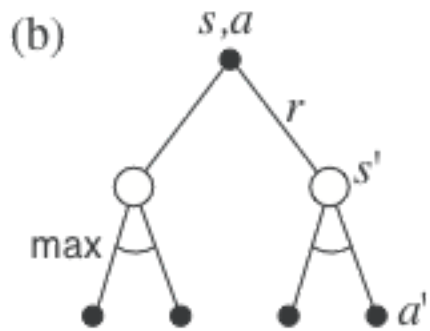
V^* is the unique solution of this system of nonlinear equations.

Bellman Optimality Equation for Q^*

$$Q^*(s,a) = E\left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \,\middle|\, s_t = s, a_t = a \right\}$$

$$= \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a'} Q^*(s', a') \right]$$

The relevant backup diagram:



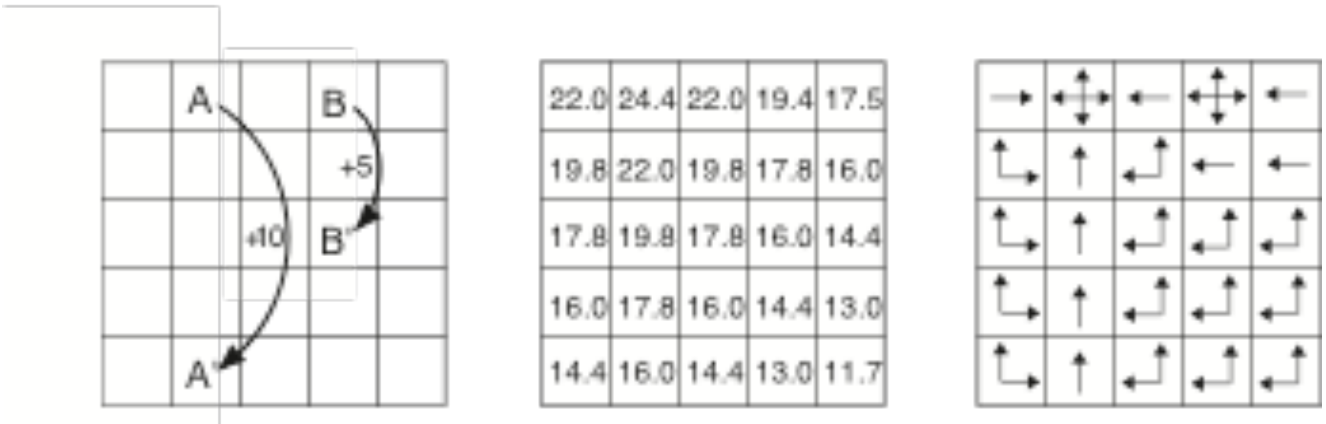
Q^* is the unique solution of this system of nonlinear equations.

Why Optimal State-Value Functions are Useful

Any policy that is greedy with respect to V^* is an optimal policy.

Therefore, given V^*, one-step-ahead search produces the long-term optimal actions.

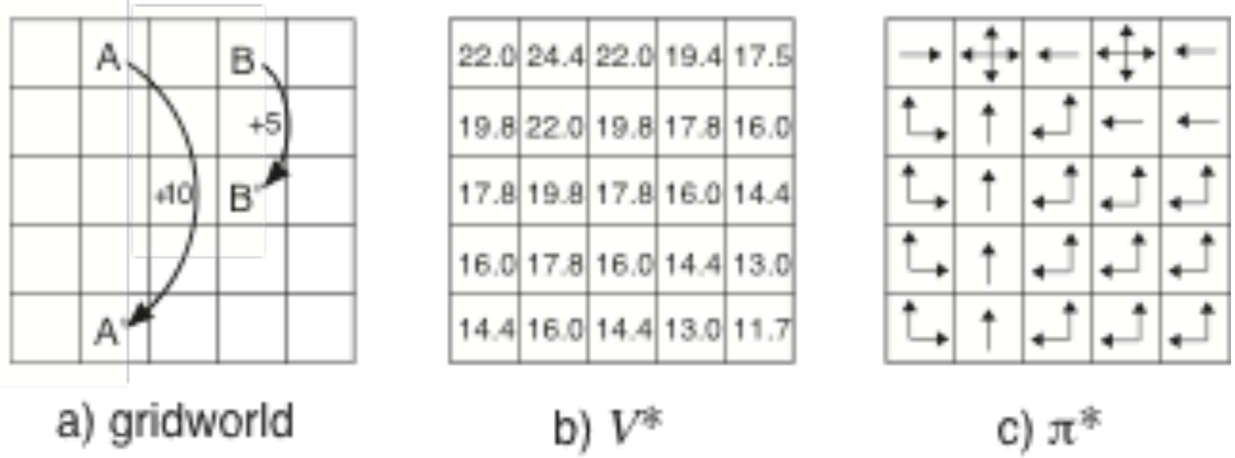
E.g., back to the gridworld:



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

a) gridworld b) V^* c) π^*

Verifying V*



- Recall that: $$V^*(s) = \max_{a \in A(s)} \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^*(s') \right]$$
- *Exercise: Verify that V*(A) = 24.4*
  - *All actions have the same effect & are therefore equally good...*
- *Exercise: Verify that V*([1,1]) = 14.4*
  - *What would V* be (given other V* values) for each possible optimal action? And therefore, what is the best action(s)?*
- **Note that V* is easy to verify but not easy to find! (That's why we need RL...)**

What About Optimal Action-Value Functions?

Given Q^*, the agent does not even have to do a one-step-ahead search:

$$\pi^*(s) = \arg\max_{a \in A(s)} Q^*(s,a)$$

Solving the Bellman Optimality Equation

- Finding an optimal policy by solving the Bellman Optimality Equation requires the following:
  - accurate knowledge of environment dynamics;
  - enough space and time to do the computation;
  - the Markov Property.
- How much space and time do we need?
  - polynomial in the number of states (via dynamic programming methods; Chapter 4),
  - ***But:*** the number of states is often huge (e.g., backgammon has about 10^{20} states).
- We usually have to settle for approximations.
- Many RL methods can be understood as approximately solving the Bellman Optimality Equation.

DYNAMIC PROGRAMMING

Policy Evaluation

Policy Evaluation: for a given policy π, compute the state-value function V^π

Recall: **State - value function for policy** π **:**

$$V^\pi(s) = E_\pi\left\{R_t \mid s_t = s\right\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}$$

Bellman equation for V^π :

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V^\pi(s')\right]$$

— a system of $|S|$ simultaneous linear equations

Iterative Methods

$$V_0 \rightarrow V_1 \rightarrow \cdots \rightarrow V_k \rightarrow V_{k+1} \rightarrow \cdots \rightarrow V^\pi$$

a "sweep"

A sweep consists of applying a **backup operation** to each state.

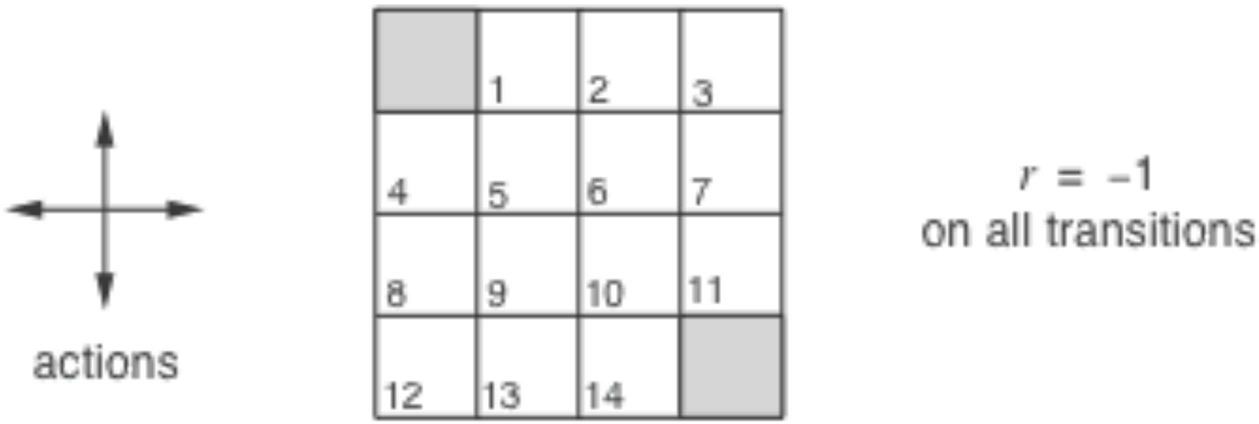
A **full policy evaluation backup**:

$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V_k(s') \right]$$

Iterative Policy Evaluation

Input π, the policy to be evaluated
Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx V^\pi$

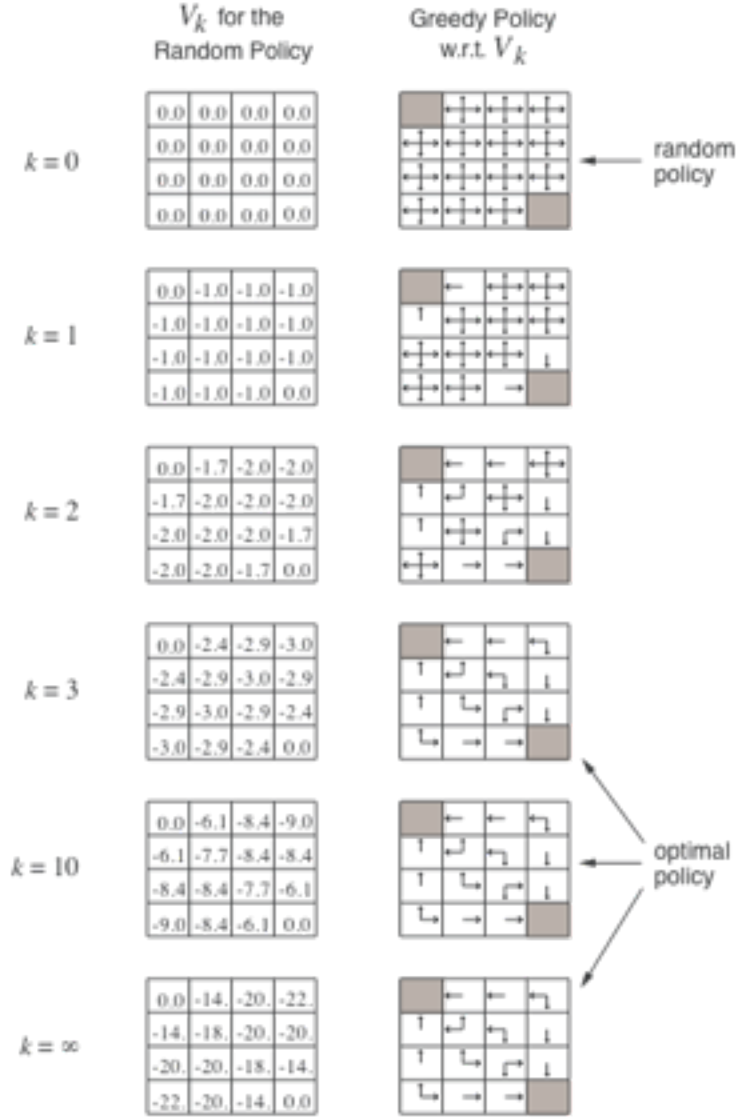
A Small Gridworld



- An undiscounted episodic task
- Nonterminal states: 1, 2, . . ., 14;
- One terminal state (shown twice as shaded squares)
- Actions that would take agent off the grid leave state unchanged
- Reward is –1 until the terminal state is reached

Iterative Policy Eval for the Small Gridworld

$\pi =$ random (uniform) action choices



Policy Improvement

Suppose we have computed V^{π} for a deterministic policy π.

For a given state s,
would it be better to do an action $a \neq \pi(s)$?

The value of doing a in state s is :

$$Q^{\pi}(s,a) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s, a_t = a\right\}$$
$$= \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

It is better to switch to action a for state s if and only if

$$Q^{\pi}(s,a) > V^{\pi}(s)$$

Policy Improvement Cont.

Do this for all states to get a new policy π' that is **greedy** with respect to V^π :

$$\pi'(s) = \arg\max_a Q^\pi(s,a)$$
$$= \arg\max_a \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma V^\pi(s')\right]$$

Then $V^{\pi'} \geq V^\pi$

Policy Improvement Cont.

What if $V^{\pi'} = V^{\pi}$?

i.e., for all $s \in S$, $V^{\pi'}(s) = \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^{\pi}(s') \right]$?

But this is the Bellman Optimality Equation.

So $V^{\pi'} = V^*$ and both π and π' are optimal policies.

Policy Iteration

$$\pi_0 \rightarrow V^{\pi_0} \rightarrow \pi_1 \rightarrow V^{\pi_1} \rightarrow \mathrm{L}\ \pi^* \rightarrow V^* \rightarrow \pi^*$$

policy evaluation

policy improvement
"greedification"

Policy Iteration

1. Initialization

   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

   Repeat

   $\quad\Delta \leftarrow 0$

   $\quad$For each $s \in \mathcal{S}$:

   $\quad\quad v \leftarrow V(s)$

   $\quad\quad V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$

   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$

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

3. Policy Improvement

   policy-stable $\leftarrow$ true

   For each $s \in \mathcal{S}$:

   $\quad b \leftarrow \pi(s)$

   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$

   $\quad$If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ false

   If *policy-stable*, then stop; else go to 2

Value Iteration

Recall the **full policy evaluation backup**:

$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V_k(s') \right]$$

Here is the **full value iteration backup**:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P_{ss'}^a \left[R_{ss'}^a + \gamma V_k(s') \right]$$

Value Iteration Cont.

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

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

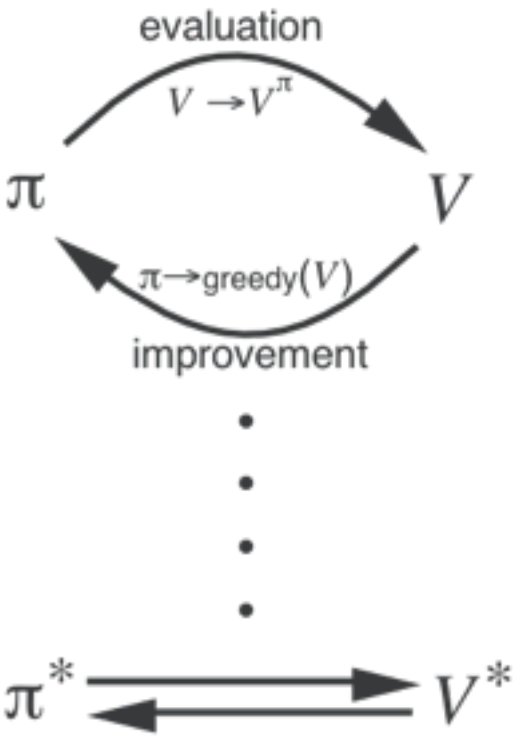
Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Asynchronous DP

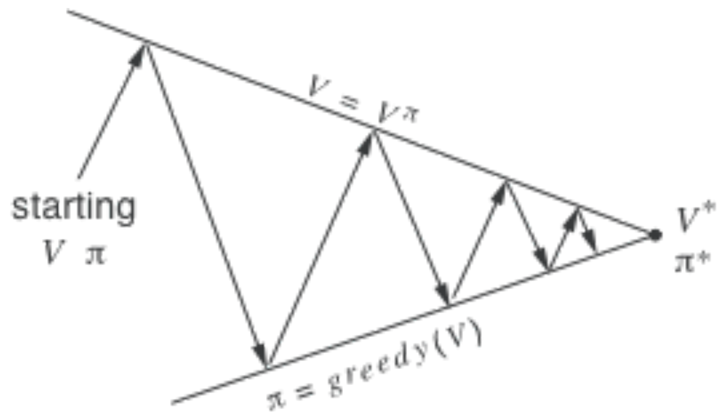
- All the DP methods described so far require exhaustive sweeps of the entire state set.
- Asynchronous DP does not use sweeps. Instead it works like this:
  - Repeat until convergence criterion is met:
    - Pick a state at random and apply the appropriate backup
- Still need lots of computation, but does not get locked into hopelessly long sweeps
- Can you select states to backup intelligently? YES: an agent's experience can act as a guide.

Generalized Policy Iteration

Generalized Policy Iteration (GPI):
any interaction of policy evaluation and policy improvement, independent of their granularity.



A geometric metaphor for convergence of GPI:



Efficiency of DP

- To find an optimal policy is polynomial in the number of states…
- BUT, the number of states is often astronomical, e.g., often growing exponentially with the number of state variables (what Bellman called "the curse of dimensionality").
- In practice, classical DP can be applied to problems with a few millions of states.
- Asynchronous DP can be applied to larger problems, and appropriate for parallel computation.
- It is surprisingly easy to come up with MDPs for which DP methods are not practical.

Summary

- Policy evaluation: backups without a max
- Policy improvement: form a greedy policy, if only locally
- Policy iteration: alternate the above two processes
- Value iteration: backups with a max
- Full backups (to be contrasted later with sample backups)
- Generalized Policy Iteration (GPI)
- Asynchronous DP: a way to avoid exhaustive sweeps
- **Bootstrapping**: updating estimates based on other estimates