Chapter 4: Dynamic Programming

- DP problems solve finite MDP's of known envs.
- use value function to organize and structure search of good policies
- turn Bellman equations into assignments
+ provides update rules for improving approximation of desired value function

Policy Evaluation

- what is the state-value function v_π for an arbitrary π?
- recall

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

- if env dynamics are known, then you have $|S|$ linear equations in $|S|$ unknowns
+ solution doable, but tedious
* start with incremental approach
* $\{v_0, v_1, v_2, \dots\}$ mapping $S^+ \to \mathbb{R}$ (state state to real number)
+ pick v_0 at random, and incrementally update

$$v_{k+1}(s) \doteq \mathbb{E}_\pi\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s\right]$$
$$= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_k(s')\right]$$

- $v_k = v_\pi$ for a given point-in-time guess
+ $\{v_k\} \to v_\pi$ as $k \to \infty$
* $\boxed{\text{iterative policy evaluation}}$

+ update v_{k+1} from v_k by replacing current state value with individually of all states + immediate reward
* called $\boxed{\text{expected update}}$

* called expected because it uses expectation of other states, not a sample of the next one.
* can use equations or backup diagram to illustrate expected update
- 2 ways to code this
  1. use 2 arrays, one for old values of $v_k(s)$, one for new $v_{k+1}(s)$
  2. use 1 array, update "in-place". Changes based on state-update order
+ ② converges faster
+ $\boxed{\text{sweep}}$ through state space

EXAMPLE 4.1: GRIDWORLD 4x4

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

actions

$R_t = -1$

so $r(s,a,s') = -1 \ \forall s, s', a$

Refer to figure 4.1 for use of random policy for v_k and greedy policy based off v_k

EXERCISE 4.1 $q_\pi(11,d)$? $q_\pi(7,d)$

$q_\pi(11,d) = -1$

$q_\pi(7,d) = -3.4$ (assuming 3rd iteration) ✓

EXERCISE 4.2 15 under 13.

$v_\pi(15) = ?$

without modifying original transitions

$$v_\pi(15) = \frac{1}{4}\left[-1 + \gamma v_\pi(12) - 1 + \gamma v_\pi(13) - 1 + \gamma v_\pi(14) - 1 + \gamma v_\pi(15)\right]$$

if 13 dynamics change

$$v_\pi(13) = \frac{1}{4}\left[-1 + \gamma v_\pi(12) - 1 + \gamma v_\pi(9) - 1 + \gamma v_\pi(14) - 1 + \gamma v_\pi(15)\right]$$

2 equations for $v_\pi(13)$ and $v_\pi(15)$

EXERCISE 4.3: $q_\pi(s,a) = ?$ for 4.3, 4.4, 4.5

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$
$$q_\pi(s,a) = \mathbb{E}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]$$
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \cdot \left[r + \gamma v_\pi(s')\right]$$
$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s')\, q_\pi(s',a')\right]$$

now iterative policy evaluation for action-values

$$q_{k+1}(s,a) = \sum_{s',r} p(s',r|s,a) \cdot \left[r + \gamma \sum_{a'} \pi(a'|s')\, q_k(s',a')\right]$$ ✓

Policy Improvement

- how do we decide to try a different policy than the current v_π for a state s?
+ pick new a, then continue normally
* use $q_\pi(s,a)$ to see diff.

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

* adjust one, then see what is diff for $v_\pi(s)$
* if better once, we might assume better for all
- $\boxed{\text{policy improvement theorem}}$
+ π, π' are deterministic policies $\forall s \in S \Rightarrow q_\pi(s, \pi'(s)) \geq v_\pi(s)$
+ then π' must be as good if not better than π: $v_{\pi'}(s) \geq v_\pi(s)$

proof:

$$v_\pi(s) \leq q_\pi(s, \pi'(s))$$
$$= \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)\right]$$
$$= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$
$$\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s\right]$$
$$\mathbb{E}\left[R_{t+2} + \gamma v_\pi(S_{t+2}) \mid S_{t+1}, A_{t+1} = \pi'(S_{t+1})\right]$$
$$= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots$$
$$= v_{\pi'}(s)$$

- $\boxed{\text{policy improvement}}$: improving an original policy with new policy greedy w.r.t. value function
- works just as well for stochastic policies

Policy Iteration

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

E: eval
I: improve

Pseudocode for Policy Iteration for $v \approx v_*$

1) Initialization

$V(s) \in \mathbb{R}$ & $\pi(s) \in A(s)$ arbitrarily for $s \in S$, $V(\text{terminal}) = 0$

2) Policy Evaluation

Loop until $\Delta < \theta$:
  $\Delta = 0$
  Loop $s \in S$:
    $v = V(s)$
    $V(s) = \sum_{s',r} p(s',r|s,\pi(s))\left[r + \gamma V(s')\right]$
    $\Delta = \max(\Delta, |v - V(s)|)$

3) Policy Improvement

policy-stable = true
for each $s \in S$:
  old-action = $\pi(s)$
  $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma V(s')\right]$
  if old-action $\neq \pi(s)$, policy-stable = false
if policy-stable, return $V \approx v_*$, $\pi \approx \pi_*$; else go to 2

Refer to figure 4.2 for Jack's Car Rental

+ policy iteration can converge in very few iterations

EXERCISE 4.4: Modify pseudocode of policy iteration to remove subtle bug

if policy of V_π & $V_{\pi'}$ is within some small delta, stop. π and π' may differ, but if they are both optimal, V_π and $V_{\pi'}$ will converge on V_*

EXERCISE 4.5: Write pseudocode for action-value policy iteration

eval

$\Delta = \infty$
while $\Delta > \epsilon$ (some threshold)
  for $\forall s \in S$
    for $\forall a \in A(s)$
      $q = Q(s,a)$
      $Q(s,a) = r + \gamma\left[\sum_{s',a'} \pi(s',a')\, Q(s',a')\right]$
      $\Delta = \max|q - Q(s,a)|$

mistake: need $\sum_{s',r} p(s',r|s,a)$
just rusty

improve

policy_stable = true
for each $s \in S$
  a_old = $\pi(s)$
  $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s)\, Q(s,a)\right]$
  if a_old $\neq \pi(s)$
    policy_stable = false
  if policy_stable == false
    return $Q \approx q_*$, $\pi \approx \pi_*$

mistake: just $\arg\max_a Q(s,a)$
you have given the def of $Q(s,a)$. This is still right, but overcomplicated. You already have the function you want.

$V_*(s) = \arg\max_a q(s,a)$

EXERCISE 4.6 ϵ-soft $\epsilon/|A(s)|$ at least for action probability

1) initialization: $\pi(s) = \epsilon/|A(s)| \ \forall s$

2 & 3) $\pi(a|s)$ must be either $1 - \epsilon + \frac{\epsilon}{|A(s)|}$ for greedy or $\frac{\epsilon}{|A(s)|}$

Termination: since V_* for Δ may not converge, must converge around $\pi(a|s)$

Value Iteration

- policy iteration requires protracted iterative computation of state space
+ convergence to v_π occurs in $\lim \to \infty$
* how to stop sooner?
- special case: stop evaluation after single sweep of state space
+ $\boxed{\text{value iteration}}$ combine policy improvement with single eval loop.

$$v_{k+1}(s) \doteq \max_a \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a\right]$$
$$= \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_k(s')\right] \quad \forall s \in S$$

- naturally occurring from Bellman if viewed as an update rule
- algo:

init: $\theta > 0$ threshold
  $V(s) \ \forall s \in S^+$, $V(\text{terminal}) = 0$
Loop:
  $\Delta = 0$
  Loop $s \in S$:
    $v = V(s)$
    $V(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma V(s')\right]$
    $\Delta = \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

- faster convergence by interleaving multiple eval before improve sweep.

Asynchronous Dynamic Programming

- single sweep, in order, LONG TIME
- $\boxed{\text{asynchronous}}$, doesn't care for order.
+ must still update all states if it hasn't, even if done out of order
- allows for smarter, more fine-tuned state update feeding.
- can make real time updates without needing all other states
+ experience state informs current interpretation of reality

Generalized Policy Iteration

- finer and finer interleaving of eval and improve
- $\boxed{\text{generalized policy iteration}}$ ⇒ GPI
+ fundamental for RL
+ stabilize over time is the one full goal
+ competing and cooperating at the same time

$$\pi \xrightarrow[\text{eval}]{V \to v_\pi} V, \qquad V \xrightarrow[\text{improve}]{\pi \to \text{greedy}(V)} \pi, \quad \vdots \quad \pi_* \rightleftarrows V_*$$

Efficiency of DP

- worst case - polynomial in states + actions
+ guaranteed to find optimal policy in polynomial time even if total deterministic policies are k^n (n states k actions)
- curse of dimensionality affects the large state problem, not DP itself
+ DP handles quite well

Summary

- Final note: estimating based on estimates = $\boxed{\text{bootstrapping}}$
+ DP requires model of world + bootstraps
* Some RL use model & bootstrap. Some only model. Some only bootstrap.

EXERCISE 4.10: $q_{k+1}(s,a)$?

$$q_{k+1}(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} q_k(s')\right]$$

EXERCISE 4.8: There's no reward to overextending. Rather than risk more for no reward, bet small partial amounts till a full double of bet $= 100$