LECTURE 3 - DYNAMIC PROGRAMMING.

- GOAL: Given full dynamics of the MDP, find an optimal policy π^*.

POLICY EVALUATION.

- Given a policy π and full knowledge of MDP dynamics, we want to compute $V_\pi(s)$.

- From Lecture 2, we have the Bellman Expectation Equation $V_\pi(s) = \mathbb{E}[R_{t+1} + \gamma V_\pi(S_{t+1})] =$

$$\sum_a \pi(a|s) \left[R_s^a + \gamma \sum_{s'} p(s'|s,a) V_\pi(s') \right].$$

- These form a system of $|S|$ linear equations. Existence/uniqueness is guaranteed if either

$$\begin{cases} \gamma < 1 \text{ OR} \\ \text{termination is guaranteed from all states by following } \pi. \end{cases}$$

- the system is defined by $V^\pi = R^\pi + \gamma P^\pi V^\pi$ where $V^\pi \in \mathbb{R}^{|S|}$

$$R^\pi \in \mathbb{R}^{|S|}, \quad R_s^\pi = \mathbb{E}[R_{t+1} | S_t = s] = \sum_a \pi(a|s) R_s^a$$

$$P^\pi \in \mathbb{R}^{|S| \times |S|}, \quad P_{ss'}^\pi = p(S_{t+1} = s' | S_t = s) = \sum_a \pi(a|s) P_{ss'}^a$$

- the solution V^π is a fixed point of the function $T(V) = R^\pi + \gamma P^\pi V : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$.

- We can find it by iterative methods. We define the sequence $V_\pi^k = T(V_\pi^{k-1})$

$$V_\pi^0 = 0$$

Under the assumptions we mentioned,

$$V_\pi^k \xrightarrow[k]{} V_\pi.$$

$V_\pi^k \xrightarrow[k]{} V_\pi$.

$$V_\pi^k(s) = \sum_a \pi(a|s)\left[R_s^a + \gamma \sum_{s'} P_{ss'}^a \cdot V_\pi^{k-1}(s')\right]$$

- WHEN IMPLEMENTING THE METHOD, WE CAN USE 2 ARRAYS: ONE FOR OLD VALUE FUNCTIONS AND ONE FOR NEW VALUES, OR WE CAN DO IT IN PLACE USING ONLY ONE ARRAY. THIS STILL CONVERGES.
- COMMON STOPPING CRITERIA: $\max_s |V_k(s) - V_{k-1}(s)| < \varepsilon$.

POLICY IMPROVEMENT.

GOAL: GIVEN A POLICY π W/ VALUE V_π, CAN WE DEFINE A NEW POLICY π' ST $V_{\pi'} \geq V_\pi$?

POLICY IMPROVEMENT THEOREM: GIVEN 2 POLICIES π, π' IF $Q_\pi(s, \pi'(s)) \geq V_\pi(s) \quad \forall s \in S \Rightarrow$

$V_{\pi'}(s) \geq V_\pi(s) \quad \forall s \in S$.

COROLLARY: GIVEN π, CONSIDER DERMINISNC π' DEFINED BY $\pi'(s) = \arg\max_a Q_\pi(s,a) = \arg\max_a \mathbb{E}[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s, A_t = a]$.

POLICY π' SATISFIES THM BECAUSE GIVEN ANY STATE $s \in S$,

$Q_\pi(s, \pi'(s)) = Q_\pi(s, \arg\max_a Q_\pi(s,a)) =$

$\max_a Q_\pi(s,a) \geq Q_\pi(s, \pi(s)) = V_\pi(s)$.

$\therefore \pi'(s) = \text{greedy}(\pi) \geq \pi$.

PROOF THM. GIVEN ANY STATE $s, \ldots \Rightarrow V_\pi(s) \leq V_{\pi'}(s)$

$V_\pi(s) = Q_\pi(s, \pi(s)) \leq Q_\pi(s, \pi'(s)) =$

$\mathbb{E}_{\pi'}[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s] \leq$

$\mathbb{E}_{\pi'}[R_{t+1} + \gamma Q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] =$

$$\mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma V_\pi(S_{t+2}) \mid S_t = s]] =$$

$$\mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 V_\pi(S_{t+2}) \mid S_t = s]$$

$$= \dots = \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s]$$

$$= V_{\pi'}(s) \quad \blacksquare$$

- Stopping condition? What if $\pi' = \pi$?

Thm: If $\pi' = \text{greedy}(\pi) = \pi \Rightarrow \pi' = \pi = \pi^*$

proof: $\pi' = \pi^* \Leftrightarrow \pi'$ satisfies Bellman optimality Eq.

We want to prove that:

$$V_{\pi'}(s) = \max_a \mathbb{E}[R_{t+1} + \gamma V_{\pi'}(S_{t+1}) \mid S_t = s, A_t = a)$$

- $V_{\pi'}(s) = Q_{\pi'}(s, \pi'(s)) = \mathbb{E}[R_{t+1} + \gamma V_{\pi'}(S_{t+1}) \mid S_t = s, A_t = \pi'(s)]$

$= \mathbb{E}[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] = Q_\pi(s, \pi'(s)) =$

($\pi = \pi' \Rightarrow V_\pi = V_{\pi'}$)

$$\max_a Q_\pi(s, a) = \max_a Q_{\pi'}(s, a) =$$

($\pi = \pi'$)

$$\max_a \mathbb{E}[R_{t+1} + \gamma V_{\pi'}(S_{t+1}) \mid S_t = s, A_t = a] \quad \blacksquare$$

POLICY ITERATION.

$$\pi_0 \xrightarrow{E} V_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V_{\pi_1} \xrightarrow{I} \pi_2 \to \dots \to \pi_* \xrightarrow{E} V_*$$

VALUE ITERATION.

- ONE DRAWBACK OF POLICY ITERATION IS the AMOUNT OF COMPUTATION DONE EVERY TIME WE EVALUATE A POLICY.
- CONVERGENCE CAN STILL HOLD EVEN IF POL. EVAL. RUNS FOR ONLY k ITERATIONS
- $k = 1$ IS CALLED VALUE ITERATION

- When we perform policy evaluation for the k-th time, $V_k(s) = \mathbb{E}[R_{t+1} + \gamma V_{k-1}(S_{t+1}) \mid S_t = s]$ using the Bellman Ex. Eq, V_{k-1} will correspond to the value function s.t. $\pi_k = \text{greedy}(V_{k-1})$.

$$\pi_0 \xrightarrow{\mathbb{E}} V_0 \xrightarrow{=} \pi_2 \xrightarrow{\mathbb{E}} V_1$$

greedy (V_0).

$$\Rightarrow V_k(s) = \mathbb{E}[R_{t+2} + \gamma V_{k-1}(S_{t+1}) \mid S_t = s, A_t = \pi_k(s)]$$

$$= \max_a \mathbb{E}[R_{t+1} + \gamma V_{k-1}(S_{t+1}) \mid S_t = s, A_t = a]$$

$$\pi_k(s) = \arg\max_a Q_{k-1}(s,a) = \arg\max_a \mathbb{E}[R_{t+1} + \gamma V_{k-1}(S_{t+1}) \mid S_t = s, A_t = a]$$

$$\text{So } V_k(s) = \max_a \mathbb{E}[R_{t+1} + \gamma V_{k-1}(S_{t+1}) \mid S_t = s, A_t = a]$$

- Another way to view this update, is as iterating over the Bellman optimality equation in order to find the fixed point.

Example 4.3) The Gamblers Problem.

- Gambler moves a stake and coin is flipped.
  - $H \Rightarrow$ He wins as many $ he bet on that flip
  - $T \Rightarrow$ Looses everything " " "
- { He Reaches $100 / Runs out of money } $\Rightarrow$ game ends.

Formulate this problem as an MDP

- S = Gamblers capital $\in \{1, \dots, 99\}$ $S' = \{0, 100\}$
- Amount he bets on flip $= \{0, 1, \dots, \min\{s, 100-s\}\}$

- S = Gambler's capital $\in \{1, \ldots, 99\}$
- A = Amnt of money he bets on $A(s) = \{0, 1, \ldots, \min\{s, 100-s\}\}$
- $p = p(s', r \mid s, a)$ OR $p(s' \mid s, a)$ OR $\mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$.
- Rewards: 0 FOR ALL TRANSITIONS, EXCEPT FOR 1 when we reach the terminal state $s = 100$.
- $V_\pi(s) = \mathbb{E}[\underbrace{R_{t+1} + \ldots + R_T}_{\text{Ber}(p),\ p = \text{prob of winning}} \mid S_t = s]$ = probability of winning from state s
- π_*: policy that maximizes the probability of win from every state
- if p_H = prob heads is known, we can solve the MDP w/ dynamic programming.

Asynchronous Dynamic Programming.

- One drawback to DP methods comes when $|S|$ is huge. For ex, before 1 step of value iteration, we need to perform a sweep over the entire state space S.

One idea is to implement in-place algorithms. We update the value of a state as soon as it is ready. We can also choose which states to update the most according to their importance.