David Silver Lecture on Dynamic Programming

- Agenda:
  1. Policy Evaluation
  2. Policy Iteration
  3. Value Iteration
  4. Extension to DP

- Introduction: DP = Optimization for Sequential problems
  - ↳ Breaks problems into subproblems and reassembles subproblems.
  - ↳ DP requires two properties:
    1. Optimal Substructure: Optimal Solution to problem constructed from optimal Solution to subproblems
       ∴ You can divide problem into subproblems
    2. Overlapping Subproblems
       ∴ You can cache Solutions to subproblems to Save time.

- Bellman Equation gives decomposition into subproblems
  - ↳ Recursive decomposition
- Caching done by Value function: Maximum reward from any state onward.

- Planning: We are given Structure of MDP and we want to Solve it.
  - ↳ Prediction: Input: MDP $\langle S, A, P, R, \gamma \rangle$ and policy π
    Output: Value Function

↳ POLICY EVALUATION

↳ Control: Input : MDP $\langle S, A, P, R, \gamma \rangle$

Output : What's the best policy π^* and optimal value function v^*

- **Policy Evaluation :** Given MDP and policy, evaluate value function

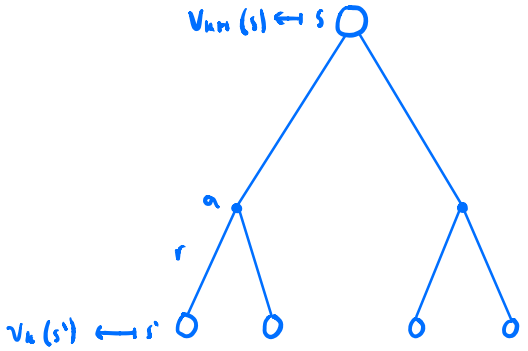
  ↳ Use Bellman Expectation Equation to do policy evaluation

  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_\pi$

  ↑ Value function (Start off with arbitrary value, typically zero)

  ↳ Use synchronous backup

  ↳ Sweep over all states in every iteration



$$v_{k+1}(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_k(s') \right)$$

↳ Turn this into iterative update

- Value function helps us determine better policies (more to come)

- **Iterative Policy Evaluation :**

  Given a policy :

  ① Initialize all scores to some value

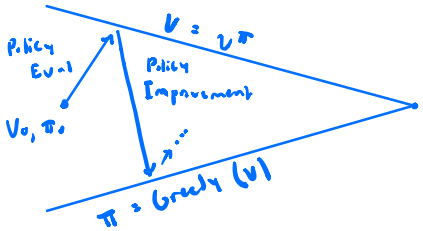
② Iteratively estimate the value function at each state s as a probability sum of value functions from all states s' which can transition to s, plus the reward associated with the transition to s.

↳ Iterate Bellman Equation

③ Repeat step 2 to convergence

- Policy Iteration:
  - Given a policy π, how can you guarantee improvement to π?
    - ↳ Solve this in 2 steps:
      - ① Evaluate Policy (as in previous section)
      - ② Improve policy by acting greedily w.r.t. V_π

        π' : greedy (V_π)
    - ↳ Iterate Evaluate and Improvement steps
    - ↳ Returns a deterministic optimal policy
- After initial policy, all other policies are deterministic



Starting point does not affect outcome

- Example Problem: Car rentals and returns at two locations
  - ↳ Redistribute cars each night

↳ What policy maximizes income

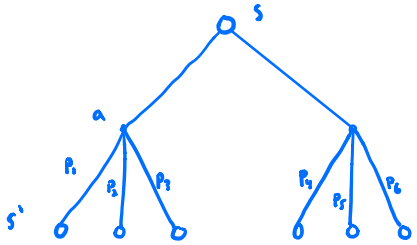
↳ State = # of cars in each lot

↳ Action = # of cars to move from 1 location to the other

Process:

- Begin with some policy: Trivial policy is move 0 cars
- Evaluate policy to build up value function
- Take the action that yields the highest value function and use this as your policy in the next iteration
- Reiterate

- **Take a Step back:** Policy Evaluation



$$V_{k+1}(s) = \sum_{a \in A} \pi(a|s) \left(R^{a}_{ss'} + V_k(s') \right)$$

Take the probability weighted sum of rewards associated with all possible transitions.

- **Precise Example:** Consider a deterministic policy $a = \pi(s)$

- How do we improve policy? Act Greedily!

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}}\ q_\pi(s, a)$$

Use the value function based on current policy

Iterate over all actions

↳ Pick actions in a way that maximize action value q

q is immediate reward with action a + reward at next state.

↳ Prove that this improves action value in one step

$$q_\pi(s, \pi'(s)) = \max_{a \in A} q_\pi(s, a) \geq q_\pi(s, \pi(s)) = v_\pi(s)$$

- $q_\pi(s, \pi'(s))$: Follow policy π' for 1 step, then follow π afterwards
- $\max_{a \in A}$: Definition of argmax that defined π'
- $\geq$: Definition of max

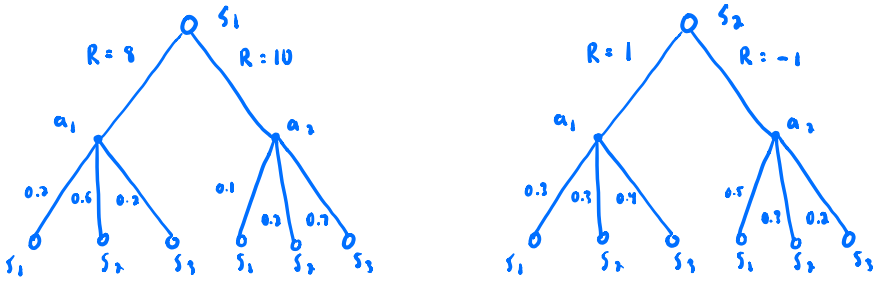
$\therefore$ Value function improves over 1 step.

$\therefore v_{\pi'}(s) \geq v_\pi(s)$

Show greedy policy selection improves value for one step

Continue at 50:00

- Value Iteration in Homework Problem: Assignment 4, Problem 1



- Initialize scores at maximum over actions

	s_1	s_2	s_3
$v_0(s)$	10	1	0

- At each step $i+1$, take the Bellman operator of the value function of each state s at i. Estimate value function of each state s as a probability weighted sum of transition rewards and rewards at s'

$$q(s, a) = \hat{R}_{ss'} + \sum_{s' \in S} P^{a}_{ss'} \cdot v_{i-1}(s')$$

$$q(s_1, a_1) = 8 + (0.2 \cdot 10 + 0.6 \cdot 1 + 0.2 \cdot 0) = 10.6$$

$$q(s_1, a_2) = 10 + (0.1 \cdot 10 + 0.2 \cdot 1 + 0.7 \cdot 0) = \underline{11.2} \leftarrow v_1(s_1)$$

$$q(s_2, a_1) = 1 + (0.3 \cdot 10 + 0.3 \cdot 1 + 0.4 \cdot 0) = 4.3$$

$$q(s_2, a_2) = -1 + (0.5 \cdot 10 + 0.3 \cdot 1 + 0.2 \cdot 0) = 4.3$$

$\} v_1(s_2)$

$$q(s_1, a_1) = 8 + (0.2 \cdot 11.2 + 0.6 \cdot 4.3 + 0.2 \cdot 0) = \underline{12.82}$$

$$q(s_1, a_2) = 10 + (0.1 \cdot 11.2 + 0.2 \cdot 4.3 + 0.7 \cdot 0) = 11.98$$

$$q(s_2, a_1) = 1 + (0.3 \cdot 11.2 + 0.3 \cdot 4.3 + 0.4 \cdot 0) = 5.65$$

$$q(s_2, a_2) = -1 + (0.5 \cdot 11.2 + 0.3 \cdot 4.3 + 0.2 \cdot 0) = \underline{5.89}$$

$$\left|(V_{k+1} - V^*)(s)\right| \leq \left|V_k(s) - V^*(s)\right|$$

Policy Iteration

π + MDP $\rightarrow$ MRP

$\forall$ MRP $\exists$ $V(s)$ s.t. $V = R + \gamma PV$

until convergence

$$V(s) = R + \gamma \sum_{s'} V(s') P(s,s')$$

Greedily Improve Policy

$$\pi_{k+1}(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} \left(R^a_{ss'} + V(s')\right) P^a_{ss'}$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} \left(R^a_{ss'} + V^*(s')\gamma\right) P^a_{ss'}$$

MRP = Expected discounted reward from s.

given π

Policy Evaluation

π_k + MDP $\rightarrow$ MRP

Use $V = (I - \gamma P)^{-1} R$

as inner loop in

proving contraction

Concern BL Contraction

Mapping