# Lecture 2: Making Sequences of Good Decisions Given a Model of the World

Emma Brunskill

CS234 Reinforcement Learning

Winter 2021

# Refresh Your Knowledge 1. Piazza Poll

In a Markov decision process, a large discount factor $\gamma$ means that short term rewards are much more influential than long term rewards. [Enter your answer in piazza]

- True
- False
- Don't know

# Refresh Your Knowledge 1. Piazza Poll

In a Markov decision process, a large discount factor $\gamma$ means that short term rewards are much more influential than long term rewards. [Enter your answer in piazza]

- True
- False
- Don't know

False. A large $\gamma$ implies we weigh delayed / long term rewards more. $\gamma = 0$ only values immediate rewards

# Today's Plan

- Last Time:
  - Introduction
  - Components of an agent: model, value, policy
- This Time:
  - Making good decisions given a Markov decision process
- Next Time:
  - Policy evaluation when don't have a model of how the world works

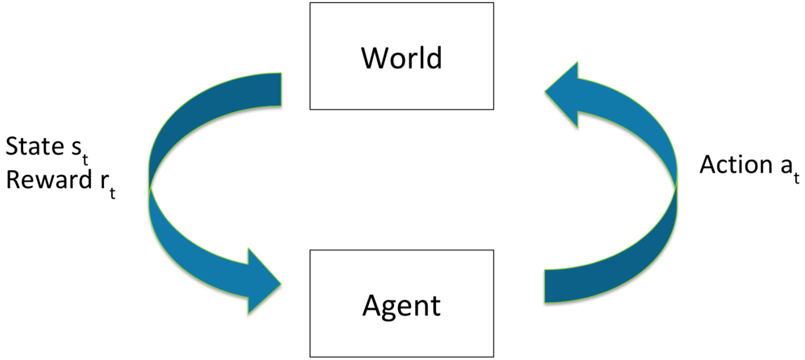
# Models, Policies, Values

- **Model**: Mathematical models of dynamics and reward
- **Policy**: Function mapping agent's states to actions
- **Value function**: future rewards from being in a state and/or action when following a particular policy

# Today: Given a model of the world

- Markov Processes
- Markov Reward Processes (MRPs) states (dynamics known), estimate rew
- Markov Decision Processes (MDPs) given states+actions, estimate policy
- Evaluation and Control in MDPs

# Full Observability: Markov Decision Process (MDP)




- MDPs can model a huge number of interesting problems and settings
  - Bandits: single state MDP
  - Optimal control mostly about continuous-state MDPs
  - Partially observable MDPs = MDP where state is history

# Recall: Markov Property

- Information state: sufficient statistic of history
- State $s_t$ is Markov if and only if:

$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|h_t, a_t)$$

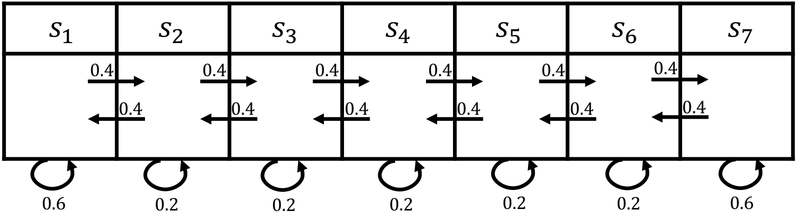
- Future is independent of past given present

# Markov Process or Markov Chain

- Memoryless random process
  - Sequence of random states with Markov property
- Definition of Markov Process
  - $S$ is a (finite) set of states ($s \in S$)
  - $P$ is dynamics/transition model that specifices $p(s_{t+1} = s' | s_t = s)$
- Note: no rewards, no actions
- If finite number ($N$) of states, can express $P$ as a matrix

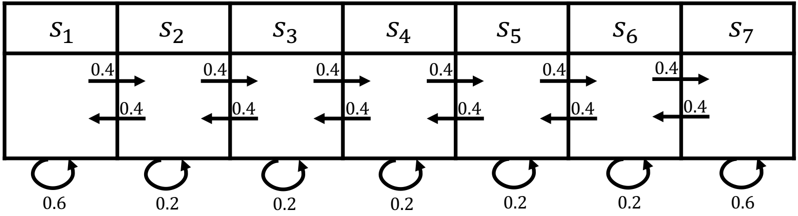
$$P = \begin{pmatrix} P(s_1|s_1) & P(s_2|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_1|s_N) & P(s_2|s_N) & \cdots & P(s_N|s_N) \end{pmatrix}$$

# Example: Mars Rover Markov Chain Transition Matrix, $P$




$$P = \begin{pmatrix} 0.6 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0.2 & 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.2 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0.2 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 0.2 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0.6 \end{pmatrix}$$

# Example: Mars Rover Markov Chain Episodes



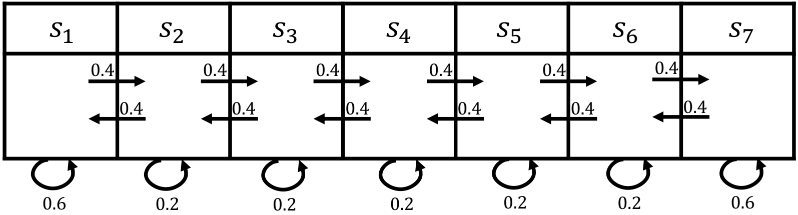

Example: Sample episodes starting from S4

- $s_4, s_5, s_6, s_7, s_7, s_7, \ldots$
- $s_4, s_4, s_5, s_4, s_5, s_6, \ldots$
- $s_4, s_3, s_2, s_1, \ldots$

# Markov Reward Process (MRP)

- Markov Reward Process is a Markov Chain + rewards
- Definition of Markov Reward Process (MRP)
  - $S$ is a (finite) set of states ($s \in S$)
  - $P$ is dynamics/transition model that specifices $P(s_{t+1} = s' | s_t = s)$
  - $R$ is a reward function $R(s_t = s) = \mathbb{E}[r_t | s_t = s]$
  - Discount factor $\gamma \in [0, 1]$
- Note: no actions
- If finite number ($N$) of states, can express $R$ as a vector

# Example: Mars Rover MRP




- Reward: $+1$ in $s_1$, $+10$ in $s_7$, 0 in all other states

# Return & Value Function

- Definition of Horizon ($H$)
  - Number of time steps in each episode
  - Can be infinite
  - Otherwise called **finite** Markov reward process
- Definition of Return, $G_t$ (for a MRP)
  - Discounted sum of rewards from time step $t$ to horizon $H$

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{H-1} r_{t+H-1}$$

- Definition of State Value Function, $V(s)$ (for a MRP)
  - Expected return from starting in state $s$

$$V(s) = \mathbb{E}[G_t | s_t = s] = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{H-1} r_{t+H-1} | s_t = s]$$

# Discount Factor

- Mathematically convenient (avoid infinite returns and values)
- Humans often act as if there's a discount factor $< 1$
- $\gamma = 0$: Only care about immediate reward
- $\gamma = 1$: Future reward is as beneficial as immediate reward
- If episode lengths are always finite ($H < \infty$), can use $\gamma = 1$

# Computing the Value of a Markov Reward Process

- Markov property provides structure
- MRP value function satisfies

$$V(s) = \underbrace{R(s)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s' \in S} P(s'|s)V(s')}_{\text{Discounted sum of future rewards}}$$

# Matrix Form of Bellman Equation for MRP

- For finite state MRP, we can express $V(s)$ using a matrix equation

$$\begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix} = \begin{pmatrix} R(s_1) \\ \vdots \\ R(s_N) \end{pmatrix} + \gamma \begin{pmatrix} P(s_1|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \ddots & \vdots \\ P(s_1|s_N) & \cdots & P(s_N|s_N) \end{pmatrix} \begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix}$$

$$V = R + \gamma PV$$

# Analytic Solution for Value of MRP

- For finite state MRP, we can express $V(s)$ using a matrix equation

$$\begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix} = \begin{pmatrix} R(s_1) \\ \vdots \\ R(s_N) \end{pmatrix} + \gamma \begin{pmatrix} P(s_1|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \ddots & \vdots \\ P(s_1|s_N) & \cdots & P(s_N|s_N) \end{pmatrix} \begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix}$$

$$\begin{aligned} V &= R + \gamma P V \\ V - \gamma P V &= R \\ (I - \gamma P) V &= R \\ V &= (I - \gamma P)^{-1} R \end{aligned}$$

- Solving directly requires taking a matrix inverse $\sim O(N^3)$
- Note that $(I - \gamma P)$ is invertible

# Iterative Algorithm for Computing Value of a MRP

- Dynamic programming
- Initialize $V_0(s) = 0$ for all $s$
- For $k = 1$ until convergence
  - For all $s$ in $S$

$$V_k(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s) V_{k-1}(s')$$

- Computational complexity: $O(|S|^2)$ for each iteration ($|S| = N$)

# Markov Decision Process (MDP)

- Markov Decision Process is Markov Reward Process + actions
- Definition of MDP
  - $S$ is a (finite) set of Markov states $s \in S$
  - $A$ is a (finite) set of actions $a \in A$
  - $P$ is dynamics/transition model for **each action**, that specifies $P(s_{t+1} = s' | s_t = s, a_t = a)$
  - $R$ is a reward function[1]

$$R(s_t = s, a_t = a) = \mathbb{E}[r_t | s_t = s, a_t = a]$$

  - Discount factor $\gamma \in [0, 1]$
- MDP is a tuple: $(S, A, P, R, \gamma)$

[1] Reward is sometimes defined as a function of the current state, or as a function of the (state, action, next state) tuple. Most frequently in this class, we will assume reward is a function of state and action

# Example: Mars Rover MDP

| $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ |
|---|---|---|---|---|---|---|
| | | |  | | | |

$$P(s'|s,a_1) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad P(s'|s,a_2) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- 2 deterministic actions

# MDP Policies

- Policy specifies what action to take in each state
  - Can be deterministic or stochastic
- For generality, consider as a conditional distribution
  - Given a state, specifies a distribution over actions
- Policy: $\pi(a|s) = P(a_t = a|s_t = s)$

# MDP + Policy

- MDP + $\pi(a|s)$ = Markov Reward Process
- Precisely, it is the MRP $(S, R^\pi, P^\pi, \gamma)$, where

$$R^\pi(s) = \sum_{a \in A} \pi(a|s) R(s,a)$$
$$P^\pi(s'|s) = \sum_{a \in A} \pi(a|s) P(s'|s,a)$$

- Implies we can use same techniques to evaluate the value of a policy for a MDP as we could to compute the value of a MRP, by defining a MRP with $R^\pi$ and $P^\pi$

# MDP Policy Evaluation, Iterative Algorithm

- Initialize $V_0(s) = 0$ for all $s$
- For $k = 1$ until convergence
  - For all $s$ in $S$

$$V_k^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_{k-1}^\pi(s')$$

- This is a **Bellman backup** for a particular policy

# Check Your Understanding: MDP 1 Iteration of Policy Evaluation, Mars Rover Example

- Dynamics: $p(s_6|s_6, a_1) = 0.5$, $p(s_7|s_6, a_1) = 0.5$, $\ldots$
- Reward: for all actions, $+1$ in state $s_1$, $+10$ in state $s_7$, 0 otherwise
- Let $\pi(s) = a_1 \; \forall s$, assume $V_k =$[1 0 0 0 0 0 10] and $k = 1$, $\gamma = 0.5$
- Compute $V_{k+1}(s_6)$

Select answer on piazza poll:

- 0
- 2.5
- 10
- I don't know

# Check Your Understanding

| $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ |
|---|---|---|---|---|---|---|
| | | |  | | | |

- We will shortly be interested in not just evaluating the value of a single policy, but finding an optimal policy. Given this it is informative to think about properties of the potential policy space.
- First for the Mars rover example [ 7 discrete states (location of rover); 2 actions: Left or Right]
- How many deterministic policies are there?
- Select answer on piazza poll: 2 / 14 / $7^2$ / $2^7$ / Not sure
- Is the optimal policy (one with highest value) for a MDP unique?
- Select answer on piazza poll: Yes / No / Not sure

# Lecture Break

# Check Your Understanding Results

# Check Your Understanding: MDP 1 Iteration of Policy Evaluation, Mars Rover Example

- Dynamics: $p(s_6|s_6,a_1) = 0.5$, $p(s_7|s_6,a_1) = 0.5$, $\ldots$
- Reward: for all actions, $+1$ in state $s_1$, $+10$ in state $s_7$, 0 otherwise
- Let $\pi(s) = a_1 \ \forall s$, assume $V_k$ =[1 0 0 0 0 0 10] and $k = 1$, $\gamma = 0.5$
- $$V_k^\pi(s) = r(s,\pi(s)) + \gamma \sum_{s' \in S} p(s'|s,\pi(s)) V_{k-1}^\pi(s')$$

$$V_{k+1}(s_6) = r(s_6,a_1) + \gamma * 0.5 * V_k(s_6) + \gamma * 0.5 * V_k(s_7)$$

$$V_{k+1}(s_6) = 0 + 0.5 * 0.5 * 0 + .5 * 0.5 * 10$$

$$V_{k+1}(s_6) = 2.5$$

# Check Your Understanding

| $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ |
|---|---|---|---|---|---|---|
| | | |  | | | |

- 7 discrete states (location of rover)
- 2 actions: Left or Right
- How many deterministic policies are there?
  $2^7$
- Is the optimal policy for a MDP always unique?
  No, there may be two actions that have the same optimal value function

# MDP Control

- Compute the optimal policy

$$\pi^*(s) = \arg\max_{\pi} V^{\pi}(s)$$

- There **exists a unique optimal value function**
- Optimal policy for a MDP in an infinite horizon problem is deterministic

# MDP Control

- Compute the optimal policy

$$\pi^*(s) = \arg\max_{\pi} V^{\pi}(s)$$

- There exists a unique optimal value function
- Optimal policy for a MDP in an infinite horizon problem (agent acts forever is
  - Deterministic
  - Stationary (does not depend on time step)
  - Unique? Not necessarily, may have state-actions with identical optimal values

# Policy Search

- One option is searching to compute best policy
- Number of deterministic policies is $|A|^{|S|}$
- Policy iteration is generally more efficient than enumeration

# MDP Policy Iteration (PI)

- Set $i = 0$
- Initialize $\pi_0(s)$ randomly for all states $s$
- While $i == 0$ or $\|\pi_i - \pi_{i-1}\|_1 > 0$ (L1-norm, measures if the policy changed for any state):
  - $V^{\pi_i} \leftarrow$ MDP V function policy **evaluation** of $\pi_i$
  - $\pi_{i+1} \leftarrow$ Policy **improvement**
  - $i = i + 1$

# New Definition: State-Action Value Q

- State-action value of a policy

$$Q^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^{\pi}(s')$$

- Take action $a$, then follow the policy $\pi$

# Policy Improvement

- Compute state-action value of a policy $\pi_i$
  - For $s$ in $S$ and $a$ in $A$:

$$Q^{\pi_i}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s')$$

- Compute new policy $\pi_{i+1}$, for all $s \in S$

$$\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s,a) \; \forall s \in S$$

# MDP Policy Iteration (PI)

- Set $i = 0$
- Initialize $\pi_0(s)$ randomly for all states $s$
- While $i == 0$ or $\|\pi_i - \pi_{i-1}\|_1 > 0$ (L1-norm, measures if the policy changed for any state):
  - $V^{\pi_i} \leftarrow$ MDP V function policy **evaluation** of $\pi_i$
  - $\pi_{i+1} \leftarrow$ Policy **improvement**
  - $i = i + 1$

# Delving Deeper Into Policy Improvement Step

$$Q^{\pi_i}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s')$$

# Delving Deeper Into Policy Improvement Step

$$Q^{\pi_i}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s')$$

$$\max_a Q^{\pi_i}(s,a) \geq R(s,\pi_i(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_i(s)) V^{\pi_i}(s') = V^{\pi_i}(s)$$

$$\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s,a)$$

- Suppose we take $\pi_{i+1}(s)$ for one action, then follow $\pi_i$ forever
  - Our expected sum of rewards is at least as good as if we had always followed $\pi_i$
- But new proposed policy is to always follow $\pi_{i+1}$ ...

# Monotonic Improvement in Policy

- Definition
$$V^{\pi_1} \geq V^{\pi_2} : V^{\pi_1}(s) \geq V^{\pi_2}(s), \forall s \in S$$
- Proposition: $V^{\pi_{i+1}} \geq V^{\pi_i}$ with strict inequality if $\pi_i$ is suboptimal, where $\pi_{i+1}$ is the new policy we get from policy improvement on $\pi_i$

# Proof: Monotonic Improvement in Policy

$$
\begin{aligned}
V^{\pi_i}(s) &\leq \max_a Q^{\pi_i}(s, a) \\
&= \max_a R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^{\pi_i}(s')
\end{aligned}
$$

# Proof: Monotonic Improvement in Policy

$$
\begin{aligned}
V^{\pi_i}(s) \leq & \max_a Q^{\pi_i}(s,a) \\
= & \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s') \\
= & R(s,\pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_{i+1}(s)) V^{\pi_i}(s') \text{ //by the definition of } \pi_{i+1} \\
\leq & R(s,\pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_{i+1}(s)) \left( \max_{a'} Q^{\pi_i}(s',a') \right) \\
= & R(s,\pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_{i+1}(s)) \\
& \left( R(s',\pi_{i+1}(s')) + \gamma \sum_{s'' \in S} P(s''|s',\pi_{i+1}(s')) V^{\pi_i}(s'') \right) \\
& \vdots \\
= & V^{\pi_{i+1}}(s)
\end{aligned}
$$

# Check Your Understanding: Policy Iteration (PI)

- Note: all the below is for finite state-action spaces
- Set $i = 0$
- Initialize $\pi_0(s)$ randomly for all states $s$
- While $i == 0$ or $\|\pi_i - \pi_{i-1}\|_1 > 0$ (L1-norm, measures if the policy changed for any state):
  - $V^{\pi_i} \leftarrow$ MDP V function policy **evaluation** of $\pi_i$
  - $\pi_{i+1} \leftarrow$ Policy **improvement**
  - $i = i + 1$
- **If policy doesn't change, can it ever change again?**
- Select on piazza poll: Yes / No / Not sure
- **Is there a maximum number of iterations of policy iteration?**
- Select on piazza poll: Yes / No / Not sure

# Lecture Break after Policy Iteration

# Results for Check Your Understanding Policy Iteration

- Note: all the below is for finite state-action spaces
- Set $i = 0$
- Initialize $\pi_0(s)$ randomly for all states $s$
- While $i == 0$ or $\|\pi_i - \pi_{i-1}\|_1 > 0$ (L1-norm, measures if the policy changed for any state):
  - $V^{\pi_i} \leftarrow$ MDP V function policy **evaluation** of $\pi_i$
  - $\pi_{i+1} \leftarrow$ Policy **improvement**
  - $i = i + 1$
- **If policy doesn't change, can it ever change again?**
  No
- **Is there a maximum number of iterations of policy iteration?**
  $|A|^{|S|}$ since that is the maximum number of policies, and as the policy improvement step is monotonically improving, each policy can only appear in one round of policy iteration unless it is an optimal policy.

# Check Your Understanding Explanation of Policy Not Changing

- Suppose for all $s \in S$, $\pi_{i+1}(s) = \pi_i(s)$
- Then for all $s \in S$, $Q^{\pi_{i+1}}(s, a) = Q^{\pi_i}(s, a)$
- Recall policy improvement step

$$Q^{\pi_i}(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^{\pi_i}(s')$$

$$\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s, a)$$

$$\pi_{i+2}(s) = \arg\max_a Q^{\pi_{i+1}}(s, a) = \arg\max_a Q^{\pi_i}(s, a)$$

Therefore policy cannot ever change again

# MDP: Computing Optimal Policy and Optimal Value

- Policy iteration computes optimal value and policy
- Value iteration is another technique
    - Idea: Maintain optimal value of starting in a state $s$ if have a finite number of steps $k$ left in the episode
    - Iterate to consider longer and longer episodes

# Bellman Equation and Bellman Backup Operators

- Value function of a policy must satisfy the Bellman equation

$$V^\pi(s) = R^\pi(s) + \gamma \sum_{s' \in S} P^\pi(s'|s) V^\pi(s')$$

- Bellman backup operator
  - Applied to a value function
  - Returns a new value function
  - Improves the value if possible

$$BV(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V(s')$$

  - $BV$ yields a value function over all states $s$

# Value Iteration (VI)

- Set $k = 1$
- Initialize $V_0(s) = 0$ for all states $s$
- Loop until [finite horizon, convergence]:
  - For each state $s$

$$V_{k+1}(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

  - View as Bellman backup on value function

$$V_{k+1} = BV_k$$

$$\pi_{k+1}(s) = \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

# Policy Iteration as Bellman Operations

- Bellman backup operator $B^\pi$ for a particular policy is defined as

$$B^\pi V(s) = R^\pi(s) + \gamma \sum_{s' \in S} P^\pi(s'|s)V(s)$$

- Policy evaluation amounts to computing the fixed point of $B^\pi$
- To do policy evaluation, repeatedly apply operator until $V$ stops changing

$$V^\pi = B^\pi B^\pi \cdots B^\pi V$$

# Policy Iteration as Bellman Operations

- Bellman backup operator $B^\pi$ for a particular policy is defined as

$$B^\pi V(s) = R^\pi(s) + \gamma \sum_{s' \in S} P^\pi(s'|s)V(s)$$

- To do policy improvement

$$\pi_{k+1}(s) = \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V^{\pi_k}(s')$$

# Going Back to Value Iteration (VI)

- Set $k = 1$
- Initialize $V_0(s) = 0$ for all states $s$
- Loop until [finite horizon, convergence]:
  - For each state $s$

$$V_{k+1}(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

  - Equivalently, in Bellman backup notation

$$V_{k+1} = BV_k$$

- To extract optimal policy if can act for $k+1$ more steps,

$$\pi(s) = \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_{k+1}(s')$$

# Contraction Operator

- Let $O$ be an operator,and $|x|$ denote (any) norm of $x$
- If $|OV - OV'| \leq |V - V'|$, then $O$ is a contraction operator

# Will Value Iteration Converge?

- Yes, if discount factor $\gamma < 1$, or end up in a terminal state with probability 1
- Bellman backup is a contraction if discount factor, $\gamma < 1$
- If apply it to two different value functions, distance between value functions shrinks after applying Bellman equation to each

# Proof: Bellman Backup is a Contraction on $V$ for $\gamma < 1$

- Let $\|V - V'\| = \max_s |V(s) - V'(s)|$ be the infinity norm

$$\|BV_k - BV_j\| = \left\| \max_a \left( R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s') \right) - \max_{a'} \left( R(s,a') + \gamma \sum_{s' \in S} P(s'|s,a') V_j(s') \right) \right\|$$

# Proof: Bellman Backup is a Contraction on $V$ for $\gamma < 1$

- Let $\|V - V'\| = \max_s |V(s) - V'(s)|$ be the infinity norm

$$
\begin{aligned}
\|BV_k - BV_j\| &= \left\| \max_a \left( R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s') \right) - \max_{a'} \left( R(s,a') + \gamma \sum_{s' \in S} P(s'|s,a') V_j(s') \right) \right\| \\
&\leq \max_a \left\| \left( R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s') - R(s,a) - \gamma \sum_{s' \in S} P(s'|s,a) V_j(s') \right) \right\| \\
&= \max_a \left\| \gamma \sum_{s' \in S} P(s'|s,a)(V_k(s') - V_j(s')) \right\| \\
&\leq \max_a \left\| \gamma \sum_{s' \in S} P(s'|s,a) \|V_k - V_j\|) \right\| \\
&= \max_a \left\| \gamma \|V_k - V_j\| \sum_{s' \in S} P(s'|s,a)) \right\| \\
&= \gamma \|V_k - V_j\|
\end{aligned}
$$

- Note: Even if all inequalities are equalities, this is still a contraction if $\gamma < 1$

# Opportunities for Out-of-Class Practice

- Homework question: Prove value iteration converges to a unique solution for discrete state and action spaces with $\gamma < 1$
- Does the initialization of values in value iteration impact anything?

# Value Iteration for Finite Horizon $H$

$V_k$ = optimal value if making $k$ more decisions
$\pi_k$ = optimal policy if making $k$ more decisions

- Initialize $V_0(s) = 0$ for all states $s$
- For $k = 1 : H$
  - For each state $s$

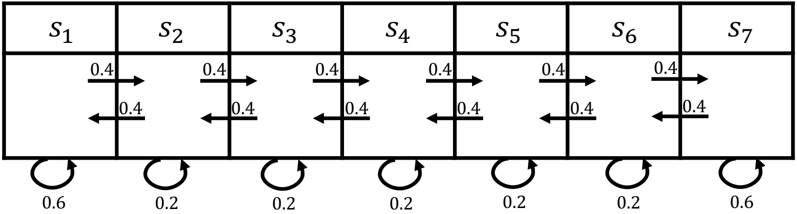
$$V_{k+1}(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

$$\pi_{k+1}(s) = \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

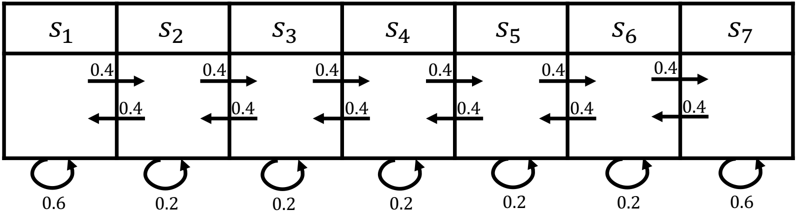
# Computing the Value of a Policy in a Finite Horizon

- Alternatively can estimate by simulation
  - Generate a large number of episodes
  - Average returns
  - Concentration inequalities bound how quickly average concentrates to expected value
  - Requires **no assumption** of Markov structure

# Example: Mars Rover




- Reward: $+1$ in $s_1$, $+10$ in $s_7$, 0 in all other states
- Sample returns for sample 4-step ($H$=4) episodes, $\gamma = 1/2$
  - $s_4, s_5, s_6, s_7$: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 10 = 1.25$




- Reward: $+1$ in $s_1$, $+10$ in $s_7$, 0 in all other states
- Sample returns for sample 4-step ($H$=4) episodes, start state $s_4$ , $\gamma = 1/2$
  - $s_4, s_5, s_6, s_7$: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 10 = 1.25$
  - $s_4, s_4, s_5, s_4$: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 0 = 0$
  - $s_4, s_3, s_2, s_1$: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 1 = 0.125$

# Check Your Understanding: Finite Horizon Policies

- Set $k = 1$
- Initialize $V_0(s) = 0$ for all states $s$
- Loop until $k == H$:
  - For each state $s$

$$V_{k+1}(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

$$\pi_{k+1}(s) = \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

Is optimal policy stationary (independent of time step) in finite horizon tasks? ? Piazza poll: Yes / No / Not sure

# Check Your Understanding: Finite Horizon Policies

- Set $k = 1$
- Initialize $V_0(s) = 0$ for all states $s$
- Loop until $k == H$:
  - For each state $s$

$$V_{k+1}(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

$$\pi_{k+1}(s) = \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

Is optimal policy stationary (independent of time step) in finite horizon tasks? In general no.

# Value vs Policy Iteration

- Value iteration:
  - Compute optimal value for horizon $= k$
    - Note this can be used to compute optimal policy if horizon $= k$
  - Increment $k$
- Policy iteration
  - Compute infinite horizon value of a policy
  - Use to select another (better) policy
  - Closely related to a very popular method in RL: policy gradient

# What You Should Know

- Define MP, MRP, MDP, Bellman operator, contraction, model, Q-value, policy
- Be able to implement
  - Value Iteration
  - Policy Iteration
- Give pros and cons of different policy evaluation approaches
- Be able to prove contraction properties
- Limitations of presented approaches and Markov assumptions
  - Which policy evaluation methods require the Markov assumption?

# Where We Are

- Last Time:
  - Introduction
  - Components of an agent: model, value, policy
- This Time:
  - Making good decisions given a Markov decision process
- Next Time:
  - Policy evaluation when don't have a model of how the world works