Making sequence of good decisions given a model of the world

Introduction to Reinforcement Learning

Bogdan Ivanyuk-Skulskyi, Dmytro Kuzmenko

Department of Mathematics,
National University of Kyiv-Mohyla Academy

February 13, 2023

Models, Policies, Values

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

Model of the world

- Markov Process
- Markov Reward Process (MRPs)
- Markov Decision Process (MDPs)
- Evaluation and Control in MDPs

Markov Property

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

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

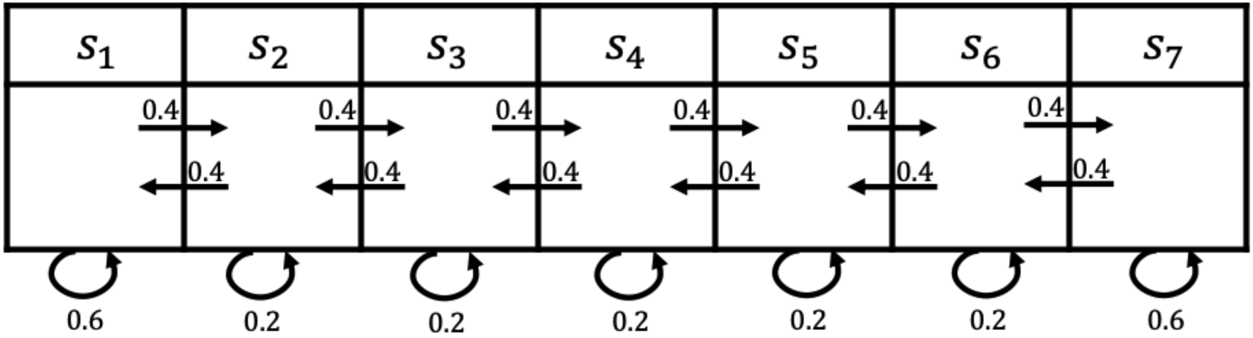
- Future is independent of past given present

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 a dynamics/transition model that specifies $p(s_{t+1} = s' | s_t = s)$
- If finite number (N) of states, can express P as a matrix

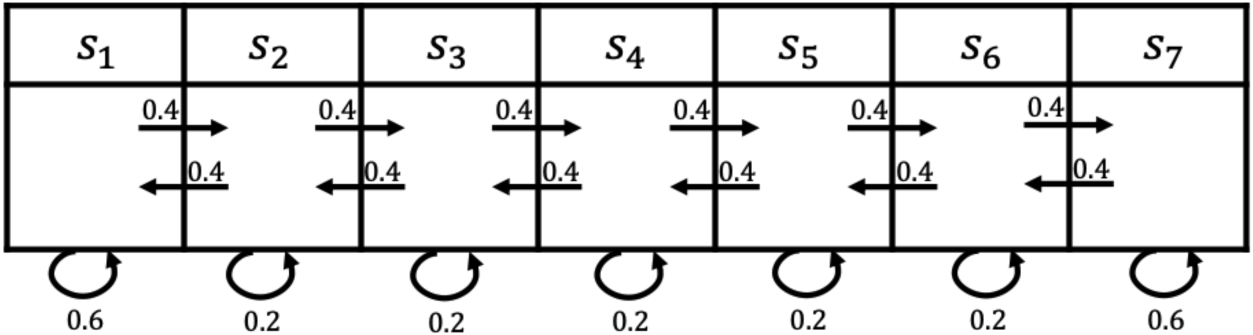
$$P = \begin{pmatrix} P(s_1|s_1) & P(s_2|s_1) & \dots & P(s_N|s_1) \\ P(s_1|s_2) & P(s_2|s_2) & \dots & P(s_N|s_2) \\ \dots & \dots & \dots & \dots \\ P(s_1|s_N) & P(s_2|s_N) & \dots & P(s_N|s_N) \end{pmatrix}$$

Example: Mars Rover Markov Chain Transition Matrix



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

Example: Mars Rover Markov Chain Episode



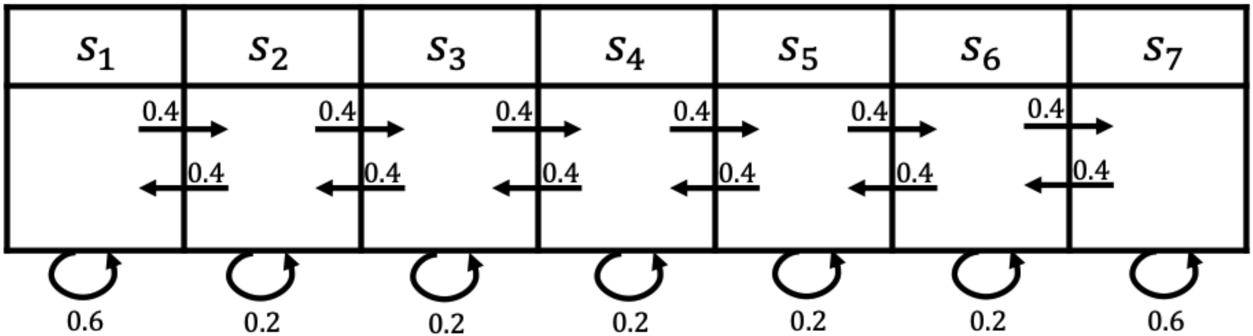
Example: Sample episodes starting from S4

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

Markov Reward Process

- Markov Reward Process is a Markov Chain + rewards
- Definition of Markov Reward Process
  - S is a (finite) set of states ($s \in S$)
  - P is a dynamics/transition model that specifies $P(s_{t+1} = s' | s_t = s)$
  - R is a reward function $R(s_t = s) = \mathbb{E}[r_t | s_t = s]$
  - Discount factor $\gamma \in [0, 1]$
- 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 and Value Function

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

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

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

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

Computing the Value of a Markov Reward Process

MRP value function satisfies

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

where

- $R(s)$ is an immidiate reward
- $\gamma \sum_{s' \in S} P(s'|s)V(s')$ is a 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

$$\begin{pmatrix} V(s_1) \\ \dots \\ V(s_N) \end{pmatrix} = \begin{pmatrix} R(s_1) \\ \dots \\ R(s_N) \end{pmatrix} + \gamma \begin{pmatrix} P(s_1|s_1) & \dots & P(s_N|s_1) \\ \dots & \dots & \dots \\ P(s_1|s_N) & \dots & P(s_N|s_N) \end{pmatrix} \begin{pmatrix} V(s_1) \\ \dots \\ V(s_N) \end{pmatrix} \tag{1}$$

$$V = R + \gamma P V$$

Matrix Form of Bellman Equation for MRP

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

$$\begin{pmatrix} V(s_1) \\ \dots \\ V(s_N) \end{pmatrix} = \begin{pmatrix} R(s_1) \\ \dots \\ R(s_N) \end{pmatrix} + \gamma \begin{pmatrix} P(s_1|s_1) & \dots & P(s_N|s_1) \\ \dots & \dots & \dots \\ P(s_1|s_N) & \dots & P(s_N|s_N) \end{pmatrix} \begin{pmatrix} V(s_1) \\ \dots \\ V(s_N) \end{pmatrix} \tag{2}$$

$$V = R + \gamma PV$$

$$V - \gamma PV = R$$

$$(I - \gamma P)V = R$$

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

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

ITERATIVE ALGORITHM FOR COMPUTING VALUE OF A MRP

- Dynamic programming
- Init $V_0(s) = 0$ for all s V = [0, 0, 0, ...]
- 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 states ($s \in S$)
  - A is a (finite) set of actions ($a \in A$)
  - P is a dynamics/transition model that specifies $P(s_{t+1} = s' | s_t = s)$
  - R is a reward function $R(s_t = s, a_t = a) = \mathbb{E}[r_t | s_t = s, a_t = a]$
  - Discount factor $\gamma \in [0, 1]$

Example: Mars Rover MDP

S1	S2	S3	S4	S5	S6	S7
						

$$P(s'|s,a_1) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} P(s'|s,a_2) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

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^π, P^π, γ), 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 ise 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^π and P^π

MDP Policy Evaluation, Iterative Algorithm

- Init $V_0(s) = 0$ 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

Classwork

- Dynamics: $p(s_6|s_6, a_1) = 0.5$, $p(s_7|s_6, a_1) = 0.5$, ...
- Rewards: 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 $V_{k+1}(s_6)$

Classwork: Answer

- Dynamics: $p(s_6|s_6, a_1) = 0.5, p(s_7|s_6, a_1) = 0.5$, ...
- Rewards: 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 $k = 1, \gamma = 0.5$

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

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

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

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

MDP Control

- Compute the optimal policy

$$\pi^*(s) = argmax_\pi V^\pi(s)$$

- 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)
  - Not unique, may have two policies with identical 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 $i = 0$
- Init $\pi_0(s)$ 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 π_i
  - $\pi_{i+1} \leftarrow$ Policy **improvement**
  - $i = i + 1$

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 π

Policy Improvement

x	s1	s2	s3
a1	0.1	0.2	0.7
a2	0.4	0.5	0.1
a3	0.5	0.3	0.2

- Compute state-action value of a policy π_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 π_{i+1}, for all $s \in S$

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

Deeper Into Policy Improvement Step

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

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

$$\pi_{i+1}(s) = argmax_a Q^{\pi}(s,a)$$

- Suppose we take $\pi_{i+1}(s)$ for one action, then follow π_i forever
  - Our expected sum of rewards is at least as good as if we had always followed π_i
- But new proposed policy is to always follow π_{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 π_i is suboptimal, where π_{i+1} is the new policy we get from policy improvement on π_i

Proof: Monotonic Improvement in Policy

$$V^{\pi_i}(s) \geq max_a Q^{\pi_i}(s,a)$$

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

$$= R(s, \pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi_{i+1}(s)) V^{\pi_i}(s')$$

$$\leq R(s, \pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi_{i+1}(s)) \left(max_{a'} Q^{\pi_i}(s', a') \right)$$

$$= R(s, \pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi_{i+1}(s)) \left(R(s', \pi_{i+1}(s')) + \gamma \sum_{s'' \in S} P(s''|s', \pi_{i+1}(s')) V^{\pi_i}(s'') \right)$$

$$\ldots$$

$$= V^{\pi_{i+1}}(s)$$

MDP: Computing Optimal Policy and Optimal Value

- Policy iteration computes infinite horizon value of a policy and then improves that 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

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

  - BV yields a value function over all states s

Value Iteration (VI)

- Set $k = 1$
- Init $V_0(s) = 0$ for all states s
- Loop until convergence: $(||V_{k+1} - V_k||_\infty \leq \epsilon$
  - For each state S

$$V_{k+1}(s) = max_a \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V_k(s') \right]$$

  - View as Bellman backup on value function

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

$$\pi_{k+1} = argmax_a \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V_k(s') \right]$$

Policy Iteration as Bellman Operator

- Bellman backup operator B^π 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^π
- To do policy evaluation, repeatedly apply operator until V stops changing

$$V^\pi = B^\pi B^\pi \dots B^\pi V$$

Policy Iteration as Bellman Operator

- Bellman backup operator B^π 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

$$\pi_{k+1}(s) = argmax_a \left[R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V^{\pi_k}(s') \right]$$

Contraction Operator

V = [1, 2]
V' = [5, 9]

labs(V- V'_)l_inf = l1 - 5, 2 - 9l_inf =
l4, 7l_inf = 7

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

O = 1/2
lOV - OV'l = l[1/2, 1] - [2.5, 4.5]l_inf =
l-2, -3.5l_inf = 3.5

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

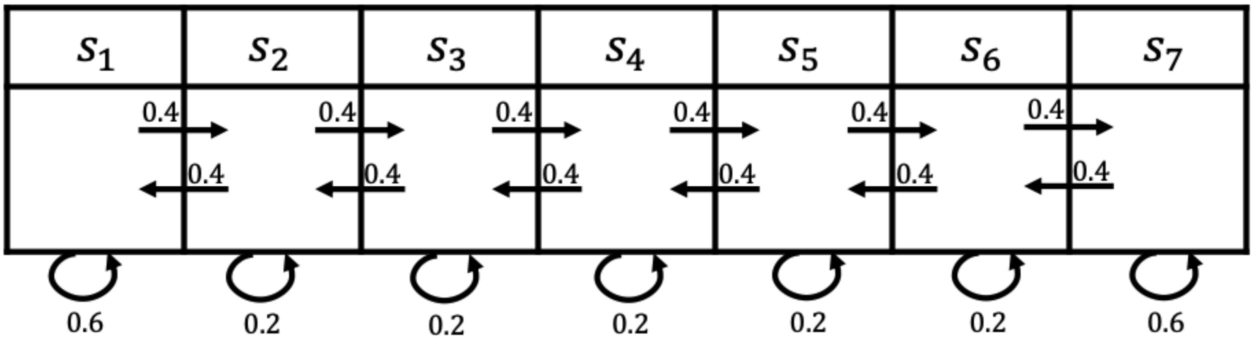
$$
\begin{aligned}
\left\|BV_k - BV_j\right\| &= \left\|max_a \left(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')\right) - max_{a'} \left(R(s,a') + \gamma \sum_{s' \in S} P(s'|s,a') V_j(s')\right)\right\| \\
&\leq max_a \left\|\left(R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s') - R(s,a) + \gamma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

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

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

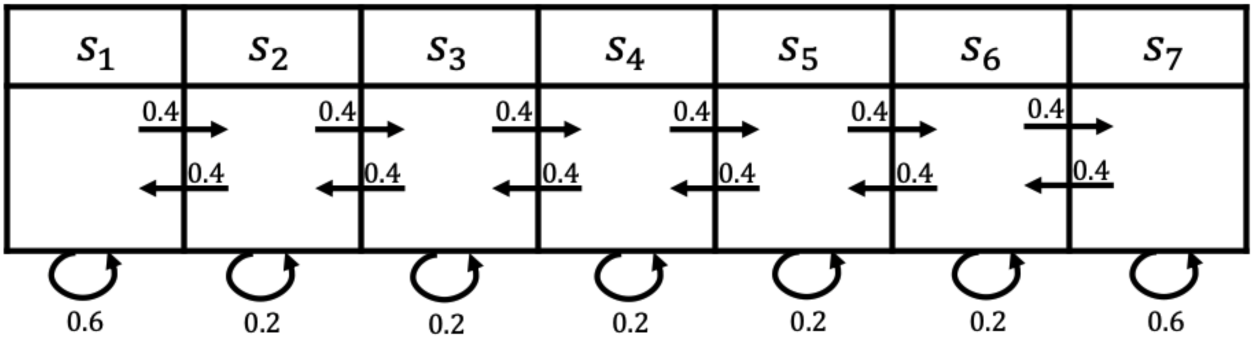
Example: Mars Rover



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: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 10 = 1.25$

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: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 10 = 1.25$
- 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: $0 + \frac{1}{2} \times 0 + \frac{1}{4} \times 0 + \frac{1}{8} \times 1 = 0.125$

Value vs Policy Iteration

- Velue 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