CS159 Lecture 1: Markov Decision Processes

Ugo Rosolia

Caltech

Spring 2021

Table of Contents

Table of Contents

Markov Decision Process

A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, T_s, c)$, where

- $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$ is a set of states;
- $\mathcal{A} = \{1, \ldots, |\mathcal{A}|\}$ is a set of actions;
- The function $T_s : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes the probability of transitioning to a state s' given the action a and the system's state s,

$$T_s(s, a, s') := \mathbb{P}(s_{k+1} = s' | s_k = s, a_k = a) = p(s'|s, a);$$

- The cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns an instantaneous cost to each state-action pairs;

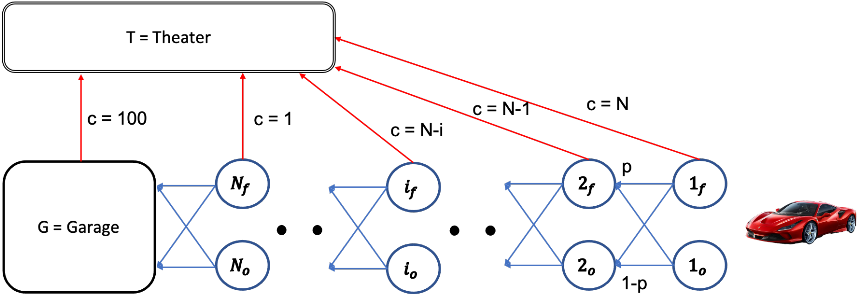
Markov Decision Process

Actions

- Park
- Move Forward

States

- Parking spot #n free (n_f)
- Parking spot #n occupied (n_0)
- Theater
- Garage
- Start



Deterministic And Random Policies

Deterministic Policies

Define the set of deterministic policies Π^d. A deterministic policy $\pi^d \in \Pi^d$ maps states to actions, i.e.,

$$a_k = \pi^d(s_k).$$

Random Policies

Define the set of random policies Π^r. A random policy $\pi^r \in \Pi^r$ maps states to probability distributions, i.e.,

$$a_k \sim \pi^r(s_k).$$

Deterministic And Random Policies

Deterministic Policies

Define the set of deterministic policies Π^d. A deterministic policy $\pi^d \in \Pi^d$ maps states to actions, i.e.,

$$a_k = \pi^d(s_k).$$

Random Policies

Define the set of random policies Π^r. A random policy $\pi^r \in \Pi^r$ maps states to probability distributions, i.e.,

$$a_k \sim \pi^r(s_k).$$

Markov Decision Process

A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, T_s, R)$, where

- $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$ is a set of states;
- $\mathcal{A} = \{1, \ldots, |\mathcal{A}|\}$ is a set of actions;
- The function $T_s : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes the probability of transitioning to a state s' given the action a and the system's state s,

$$T_s(s, a, s') := \mathbb{P}(s_{k+1} = s' | s_k = s, a_k = a) = p(s'|s, a);$$

- The cost function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns an instantaneous cost to each state-actions pairs;

Goal

Find a policy $\boldsymbol{\pi}^* = [\pi_0^*, \pi_1^*, \ldots]$ defined as

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t) | \boldsymbol{\pi}\right]$$

Markov Decision Process

Goal

Find a policy $\boldsymbol{\pi}^* = [\pi_0^*, \pi_1^*, \ldots]$ defined as

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t) | \boldsymbol{\pi}\right]$$

- The discount factor $\lambda \in (0, 1)$.
- The action $a_t = \pi_t(s_t)$ or $a_t \sim \pi_t(s_t)$.
- $\mathbb{E}[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t) | \boldsymbol{\pi}]$ denotes the expectation under the policy $\boldsymbol{\pi}$.

Markov Decision Process – Assumptions

Assumption 1. (Stationary costs and transition probabilities) The cost function $c(s, a)$ and the transition probabilities $\mathbb{P}(s'|s, a)$ do not vary.

Assumption 2. (Bounded costs) The cost function $|c(s, a)| \leq M < \infty$ for all $a \in \mathcal{A}$ and $s \in \mathcal{S}$.

Assumption 3. (Discrete State and Action Spaces) The state space $\mathcal{S}$ and the action space $\mathcal{A}$ are finite and discrete.

Assumption 4. (Discounting) The future costs are discounted by a factor λ and $0 \leq \lambda < 1$.

Table of Contents

Deterministic And Random Policies

Deterministic Policies

Define the set of deterministic policies Π^d. A deterministic policy $\pi^d \in \Pi^d$ maps states to actions, i.e.,

$$a_k = \pi^d(s_k).$$

Random Policies

Define the set of random policies Π^r. A random policy $\pi^r \in \Pi^r$ maps states to probability distributions, i.e.,

$$a_k \sim \pi^r(s_k).$$

Deterministic Vs Random Policies

- For **unconstrained problems** we have that

$$\min_{\boldsymbol{\pi}\in\Pi^d} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t)|\boldsymbol{\pi}\right] = \min_{\boldsymbol{\pi}\in\Pi^r} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t)|\boldsymbol{\pi}\right]$$

  There is no performance gain in optimizing over the larger set of random policies.

- For **constrained problems** we have that

$$\begin{aligned} \min_{\boldsymbol{\pi}\in\Pi^d} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t)|\boldsymbol{\pi}\right] \\ \text{s.t.} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} g(s_t, a_t)|\boldsymbol{\pi}\right] \le \epsilon. \end{aligned} \ge \begin{aligned} \min_{\boldsymbol{\pi}\in\Pi^r} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t)|\boldsymbol{\pi}\right] \\ \text{s.t.} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} g(s_t, a_t)|\boldsymbol{\pi}\right] \le \epsilon. \end{aligned}$$

  A randomized policy perform better for constrained problems.

Deterministic Vs Random Policies

- For **unconstrained problems** we have that

$$\min_{\boldsymbol{\pi}\in\Pi^d} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t)|\boldsymbol{\pi}\right] = \min_{\boldsymbol{\pi}\in\Pi^r} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t)|\boldsymbol{\pi}\right]$$

  There is no performance gain in optimizing over the larger set of random policies.

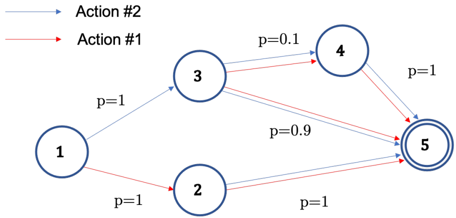
- For **constrained problems** we have that

$$\begin{aligned} \min_{\boldsymbol{\pi}\in\Pi^d} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t)|\boldsymbol{\pi}\right] \\ \text{s.t.} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} g(s_t, a_t)|\boldsymbol{\pi}\right] \le \epsilon. \end{aligned} \quad \ge \quad \begin{aligned} \min_{\boldsymbol{\pi}\in\Pi^r} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t)|\boldsymbol{\pi}\right] \\ \text{s.t.} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} g(s_t, a_t)|\boldsymbol{\pi}\right] \le \epsilon. \end{aligned}$$

  A randomized policy performs better for constrained problems.

Deterministic Vs Random Policies

- The action space $\mathcal{A} = \{\text{Action 1}, \text{Action 2}\}$, state space $\mathcal{S} = \{1, 2, 3, 4, 5\}$ and the state $s = 5$ is a sink state.
- The cost function $c(s, a) = 0$ for all $s \in \mathcal{S} \setminus \{3\}, a \in \mathcal{A}$ and $c(3, a) = -1$ for all $a \in \mathcal{A}$.
- The constraint function $g(s, a) = 0$ for all $s \in \mathcal{S} \setminus \{4\}, a \in \mathcal{A}$ and $g(4, a) = 1$ for all $a \in \mathcal{A}$.
- Pick $\epsilon < 0.1$, then a deterministic policy must choose Action 1 from $s = 1$ to meet the constraint $\mathbb{E}\left[\sum_{t=0}^{H} g(s_t, a_t) | \boldsymbol{\pi}\right] \leq \epsilon$.



Value Functions

Value Function

The value function $v_{\boldsymbol{\pi}}$ is a vector in $\mathbb{R}^{|\mathcal{S}|}$ where each entry $v_{\pi}(s)$ represents the cumulative cost of applying the policy $\pi \in \Pi^d$ from the state $s \in \mathcal{S}$, i.e.,

$$v_{\boldsymbol{\pi}}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t) | \boldsymbol{\pi}, s\right].$$

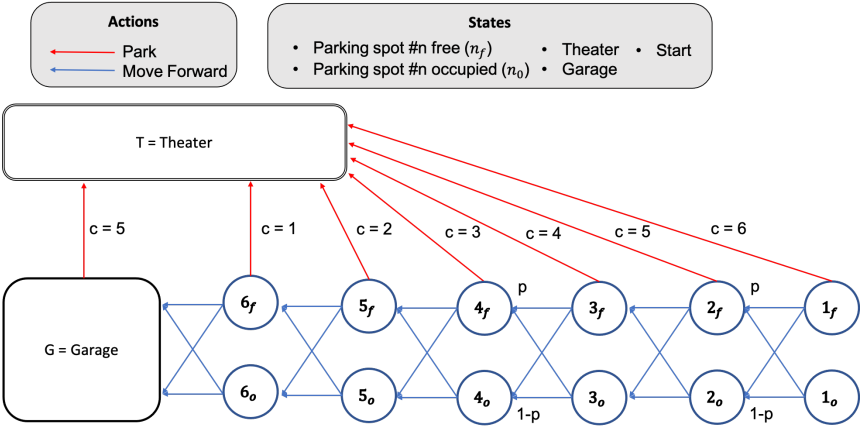
Consider a stationary policy $\boldsymbol{\pi} = [\pi, \pi, \ldots]$ with $\pi \in \Pi^d$. Then $v_{\boldsymbol{\pi}}$ is the unique solution of

$$v = r_\pi + \lambda P_\pi v$$

where

- the vector $r_\pi \in \mathbb{R}^{|\mathcal{S}|}$ where $r_\pi(s) = c(s, \pi(s))$
- the matrix $P_\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ where $P_\pi(s, s') = p(s'|s, \pi(s))$
- the value function $v = (I - \lambda P_\pi)^{-1} r_\pi = \sum_{t=0}^{\infty} \lambda^t P_\pi^t r_\pi$

Value Function – The Parking Example



The set of states

$\mathcal{S} = \{1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T\}$.

Value Function – The Parking Example

Two actions are available: $\{\texttt{move forward}, \texttt{park}\}$.

Let π_m be a deterministic policy that selects the action `move forward`, then P_{π_m} is defined by the following table:

1f	1o	2f	2o	…	G	T	
		p	1-p				1f
		p	1-p				1o
				$\ddots$			$\vdots$
					1		6f
					1		6o
						1	G
						1	T

where each entry $P_\pi(s, s') = p(s'|s, \pi(s))$ for $s \in \mathcal{S}$, $s' \in \mathcal{S}$ and $\mathcal{S} = \{1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T\}$.

Value Function – The Parking Example

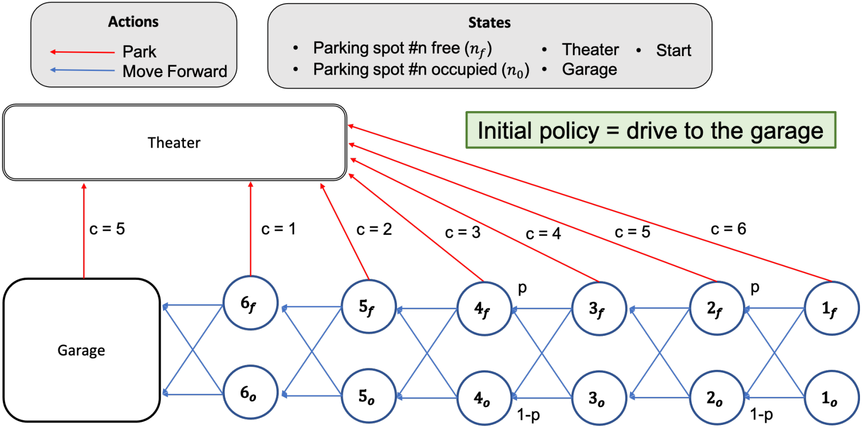
Two actions are available: $\{\texttt{move forward}, \texttt{park}\}$.

Let π_p be a deterministic policy that selects the action `park`, then P_{π_p} is defined by the following table:

1f	1o	2f	2o	...	G	T	
						1	1f
		p	1-p				1o
				$\ddots$			$\vdots$
					1		6f
					1		6o
						1	G
						1	T

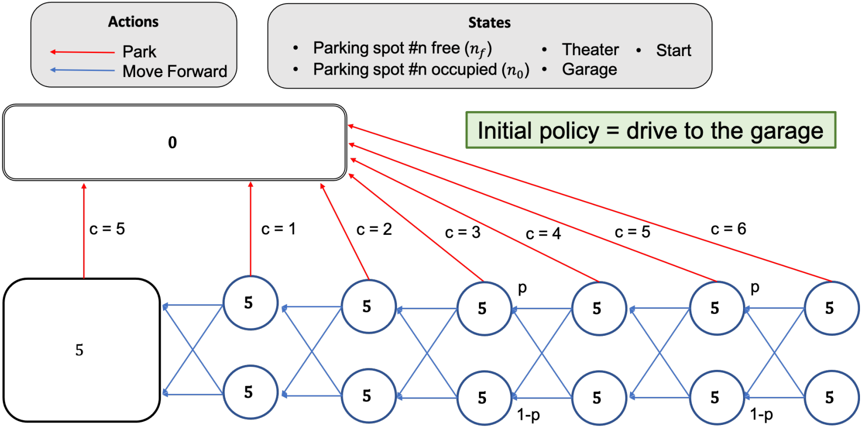
where each entry $P_\pi(s, s') = p(s'|s, \pi(s))$ for $s \in \mathcal{S}$, $s' \in \mathcal{S}$ and $\mathcal{S} = \{1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T\}$.

Value Function – The Parking Example



State Vector $= [1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T]$.
Value Function $= [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 0]$.

Value Function – The Parking Example



State Vector $= [1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T]$.
Value Function $= [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 0]$.

Markov Decision Process

Goal

Find a stationary policy $\boldsymbol{\pi}^* = [\pi^*, \pi^*, \ldots]$ defined as

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t) | \boldsymbol{\pi}\right]$$

Optimality Conditions

Given a value function which satisfies:

$$v^*(s) = \arg\min_{a \in \mathcal{A}} c(s, a) + \sum_{s' \in \mathcal{S}} \lambda v^*(s') p(s'|s, a).$$

Then, the optimal policy is:

$$\pi(s) = \min_{a \in \mathcal{A}} c(s, a) + \sum_{s' \in \mathcal{S}} \lambda v^*(s') p(s'|s, a)$$

Markov Decision Process

Goal

Find a stationary policy $\boldsymbol{\pi}^* = [\pi^*, \pi^*, \ldots]$ defined as

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t c(s_t, a_t) | \boldsymbol{\pi}\right]$$

Optimality Conditions

Given the optimal value function v^* that satisfies the Bellman recursion $v^* = Bv^*$ defined as follows:

$$v^*(s) = \min_{a \in \mathcal{A}} [c(s, a) + \sum_{s' \in \mathcal{S}} \lambda v^*(s') p(s'|s, a)], \ \forall s \in \mathcal{S}.$$

Then, the optimal policy is:

$$\pi^*(s) = \arg\min_{a \in \mathcal{A}} [c(s, a) + \sum_{s' \in \mathcal{S}} \lambda v^*(s') p(s'|s, a)]$$

Table of Contents

Value Iteration

Algorithm Steps:

1. Select $v^0 \in \mathbb{R}^{|\mathcal{S}|}$, set $k = 0$ and pick a tolerance $\epsilon \geq 0$

Value Iteration

Algorithm Steps:

1. Select $v^0 \in \mathbb{R}^{|\mathcal{S}|}$, set $k = 0$ and pick a tolerance $\epsilon \geq 0$
2. For each $s \in \mathcal{S}$ compute $v^{k+1} \in \mathbb{R}^{|\mathcal{S}|}$ where

$$v^{k+1}(s) = \min_{a \in \mathcal{A}} [c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a) v^k(s')]$$

Value Iteration

Algorithm Steps:

1. Select $v^0 \in \mathbb{R}^{|\mathcal{S}|}$, set $k = 0$ and pick a tolerance $\epsilon \geq 0$
2. For each $s \in \mathcal{S}$ compute $v^{k+1} \in \mathbb{R}^{|\mathcal{S}|}$ where

$$v^{k+1}(s) = \min_{a \in \mathcal{A}} [c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a) v^k(s')]$$

3. If

$$||v^{k+1} - v^k|| \geq \epsilon \frac{(1-\lambda)}{2\lambda}$$

   set $k = k + 1$ and go to step 2.

Value Iteration

Algorithm Steps:

1. Select $v^0 \in \mathbb{R}^{|\mathcal{S}|}$, set $k = 0$ and pick a tolerance $\epsilon \geq 0$
2. For each $s \in \mathcal{S}$ compute $v^{k+1} \in \mathbb{R}^{|\mathcal{S}|}$ where

$$v^{k+1}(s) = \min_{a \in \mathcal{A}} [c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a) v^k(s')]$$

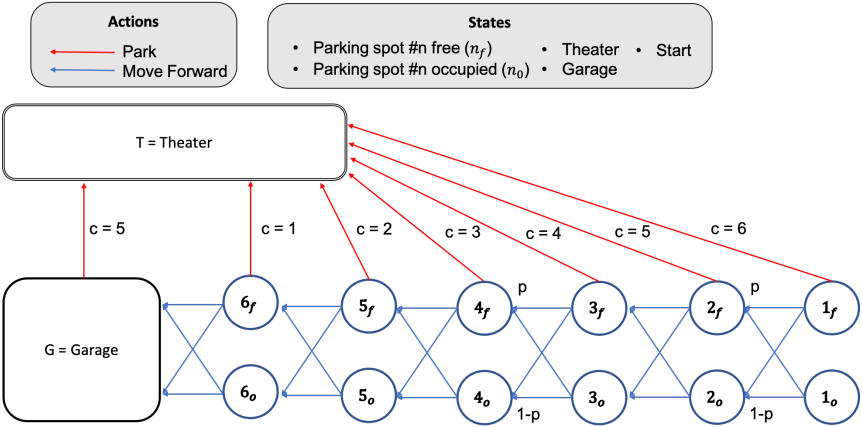
3. If

$$||v^{k+1} - v^k|| \geq \epsilon \frac{(1-\lambda)}{2\lambda}$$

   set $k = k + 1$ and go to step 2.
4. Define the control policy

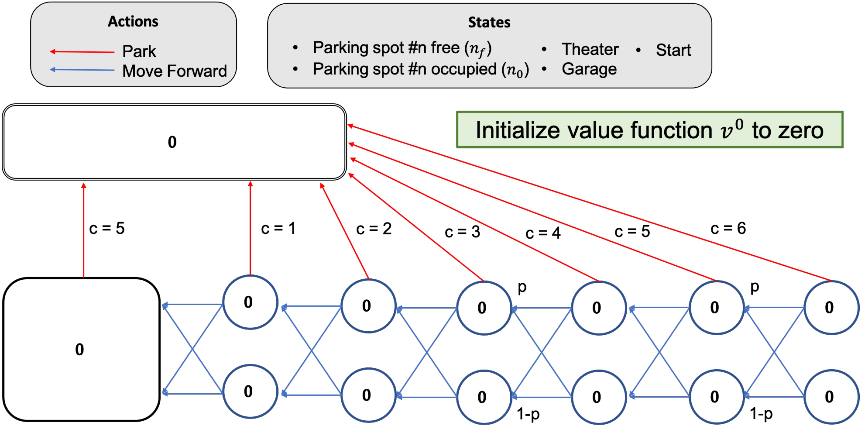
$$\pi^{\mathrm{vi}}(s) = \arg\min_{a \in \mathcal{A}} [c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a) v^{k+1}(s')]$$

Value Iteration



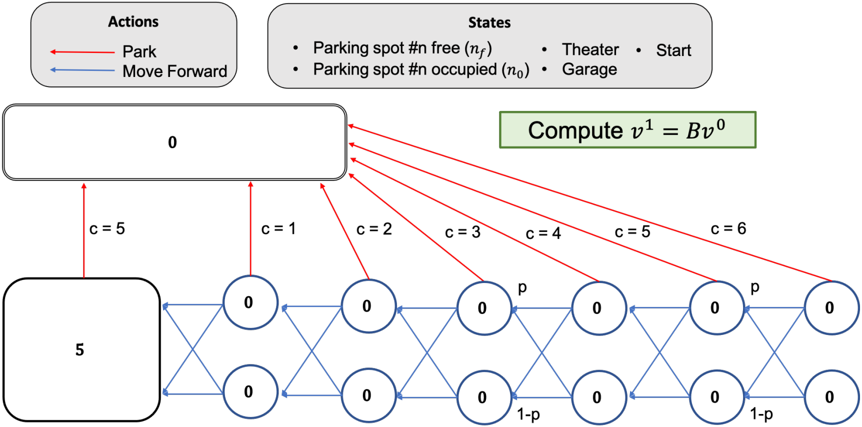
State Vector $= [1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T]$.

Value Iteration



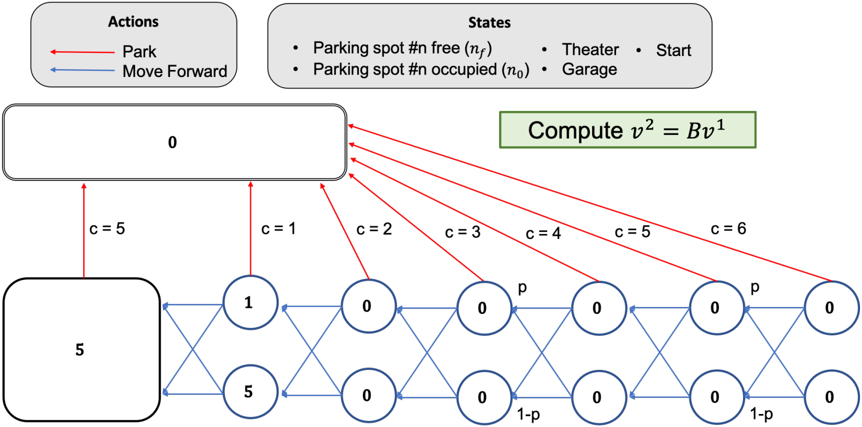
State Vector $= [1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T].$

Value Iteration



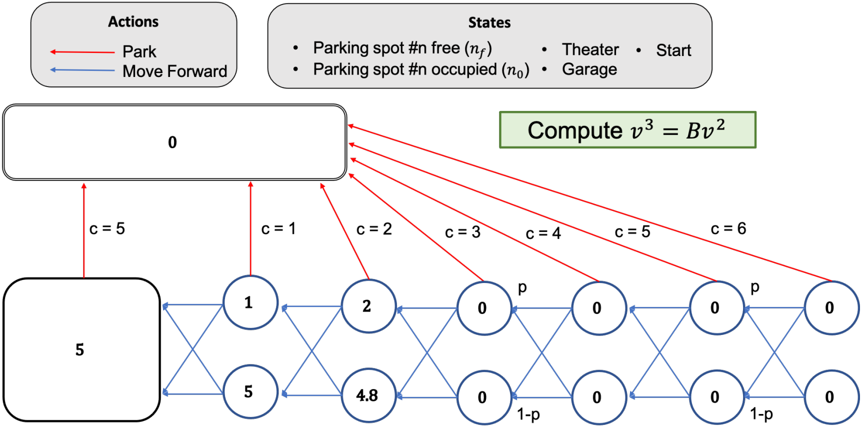
$$\text{State Vector} = [1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T].$$

Value Iteration



State Vector $= [1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T]$.

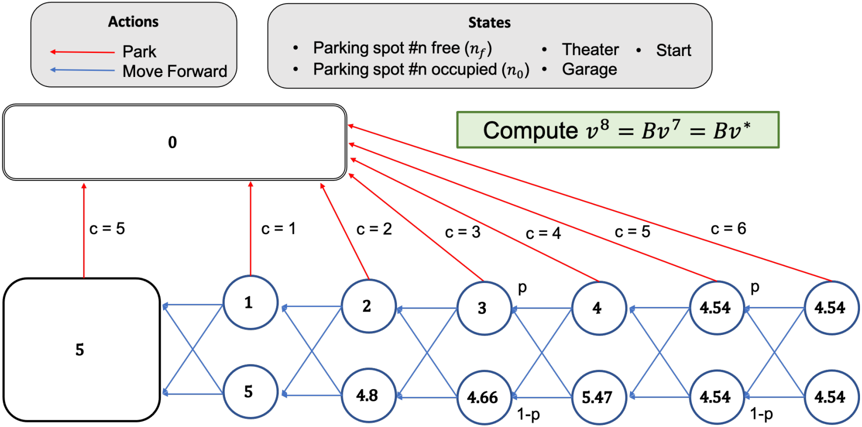
Value Iteration



Compute $v^3 = Bv^2$

State Vector $= [1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T]$.

Value Iteration



State Vector $= [1_f, 1_o, 2_f, 2_o, 3_f, 3_o, 4_f, 4_o, 5_f, 5_o, 6_f, 6_o, G, T]$.

Value Iteration: Properties

Theorem

Let $\{v^k\}$ be a sequence defined by the Bellman recursion and consider the stopping rule

$$||v^{k+1} - v^k||_\infty < \epsilon \frac{(1-\lambda)}{2\lambda} \quad (1)$$

Then we have that

- v^k converges in norm to v^* and the convergence is linear with rate λ.
- If (1) holds for a finite N, then (1) holds for $k \geq N$.
- If (1) holds for a finite N, then $||v^{N+1} - v^*||_\infty < \epsilon/2$ and π^{vi} is ϵ-optimal.

Variants to the Value Iteration with better convergence rate in Chapter 6 of "Markov decision processes: discrete stochastic dynamic programming" by M. Puterman. John Wiley & Sons, 2014.

Value Iteration: Convergence Proof

Define the Bellman backup operator $B : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$

$$Bv(s) = \min_{a \in \mathcal{A}} [c(s, a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, a) v(s')]$$

which is a contraction as

$$\begin{aligned}
|Bv_0(s) - Bv_1(s)| = |\min_{a \in \mathcal{A}} [c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a) v_0(s')] \\
- \min_{a \in \mathcal{A}} [c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a) v_1(s')]| \\
&\leq \max_{a \in \mathcal{A}} \lambda |\sum_{s' \in \mathcal{S}} p(s'|s,a) v_0(s') - \sum_{s' \in \mathcal{S}} p(s'|s,a) v_1(s')| \\
&= \max_{a \in \mathcal{A}} \lambda \sum_{s' \in \mathcal{S}} p(s'|s,a) |v_0(s') - v_1(s')| \\
&\leq \lambda \max_{s' \in \mathcal{S}} |v_0(s') - v_1(s')|.
\end{aligned}$$

Then, by the fixed-point theorem, we have that $Bv^* = v^*$ and the sequence $v^{k+1} = Bv^k = B^{k+1} v^0$ converges to v^*.

Value Iteration: Suboptimality Proof

Now define the Bellman backup for a policy π as $B_\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$

$$B_\pi v(s) = c(s, \pi(a)) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, \pi(a)) v(s').$$

We notice that

$$\begin{aligned}
||v^* - v^{k+1}||_\infty &= ||Bv^* - v^{k+1}||_\infty \\
&\leq ||Bv^* - Bv^{k+1}||_\infty + ||Bv^{k+1} - v^{k+1}||_\infty \\
&= ||Bv^* - Bv^{k+1}||_\infty + ||Bv^{k+1} - Bv^k||_\infty \\
&\leq \lambda ||v^* - v^{k+1}||_\infty + \lambda ||v^{k+1} - v^k||_\infty.
\end{aligned}$$

Rearranging terms and leveraging the stopping rule yields to

$$||v^{k+1} - v^*||_\infty \leq \frac{\lambda}{1-\lambda} ||v^{k+1} - v^k||_\infty \leq \frac{\epsilon}{2}$$

Table of Contents

Policy Iteration

Algorithm Steps:

1. Set $k = 0$ and select a policy $\pi^k \in \Pi^d$.

Policy Iteration

Algorithm Steps:

1. Set $k = 0$ and select a policy $\pi^k \in \Pi^d$.
2. **(Policy Evaluation).** Compute the value function $v^k_{\pi^k} \in \mathbb{R}^{|\mathcal{S}|}$ that is the solution to the following equation:

$$v^k_{\pi^k}(s) = c(s, \pi^k(s)) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, \pi^k(s)) v^k_{\pi^k}(s')].$$

Recall that $v^k_{\pi^k} = (I - P_{\pi^k})^{-1} r_{\pi^k}$.

Policy Iteration

Algorithm Steps:

1. Set $k = 0$ and select a policy $\pi^k \in \Pi^d$.
2. **(Policy Evaluation).** Compute the value function $v^k_{\pi^k} \in \mathbb{R}^{|\mathcal{S}|}$ that is the solution to the following equation:

$$v^k_{\pi^k}(s) = c(s, \pi^k(s)) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, \pi^k(s)) v^k_{\pi^k}(s')].$$

   Recall that $v^k_{\pi^k} = (I - P_{\pi^k})^{-1} r_{\pi^k}$.
3. **(Policy Improvement).** Set

$$\pi^{k+1}(s) = \min_{a \in \mathcal{S}} \left[c(s, a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, a) v^k_{\pi^k}(s')]. \right]$$

Policy Iteration

Algorithm Steps:

1. Set $k = 0$ and select a policy $\pi^k \in \Pi^d$.
2. **(Policy Evaluation).** Compute the value function $v^k_{\pi^k} \in \mathbb{R}^{|\mathcal{S}|}$ that is the solution to the following equation:

$$v^k_{\pi^k}(s) = c(s, \pi^k(s)) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, \pi^k(s)) v^k_{\pi^k}(s')].$$

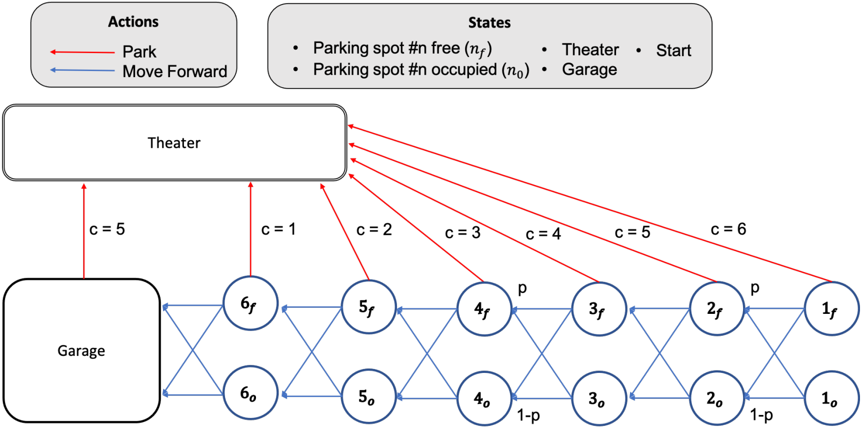
   Recall that $v^k_{\pi^k} = (I - P_{\pi^k})^{-1} r_{\pi^k}$.

3. **(Policy Improvement).** Set

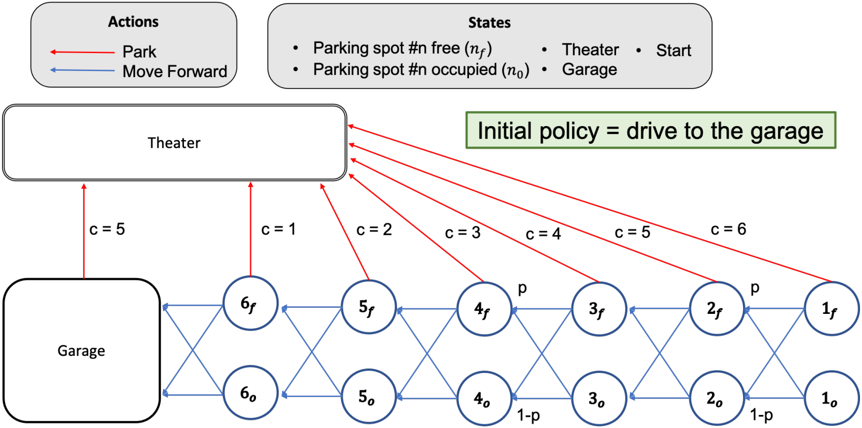
$$\pi^{k+1}(s) = \min_{a \in \mathcal{S}} \left[c(s, a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, a) v^k_{\pi^k}(s')]. \right]$$

4. If $\pi^k = \pi^{k+1}$ stop, $\pi^* = \pi^k$. Otherwise, set $k = k + 1$ and go to Step 2.

Policy Iteration

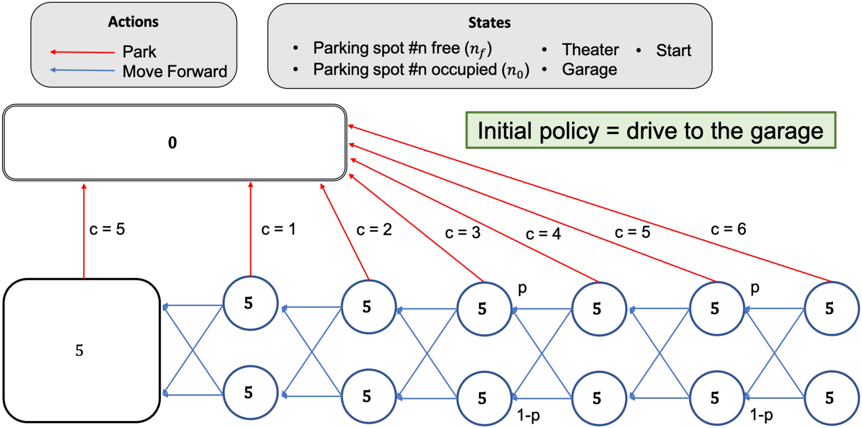


Policy Iteration

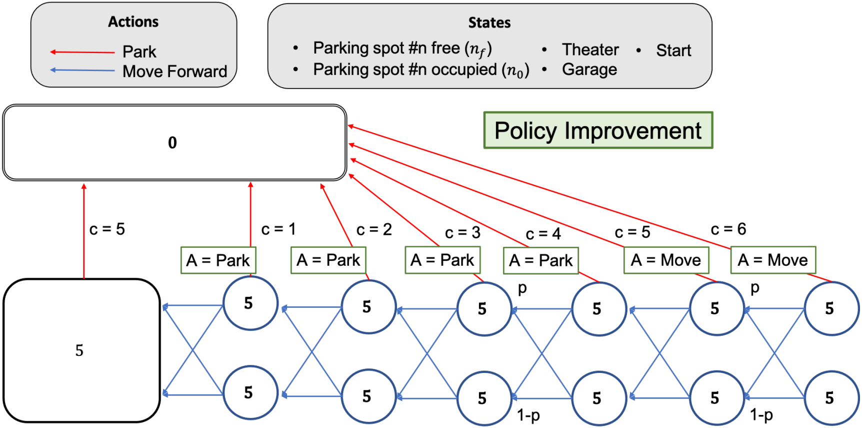


Initial policy = drive to the garage

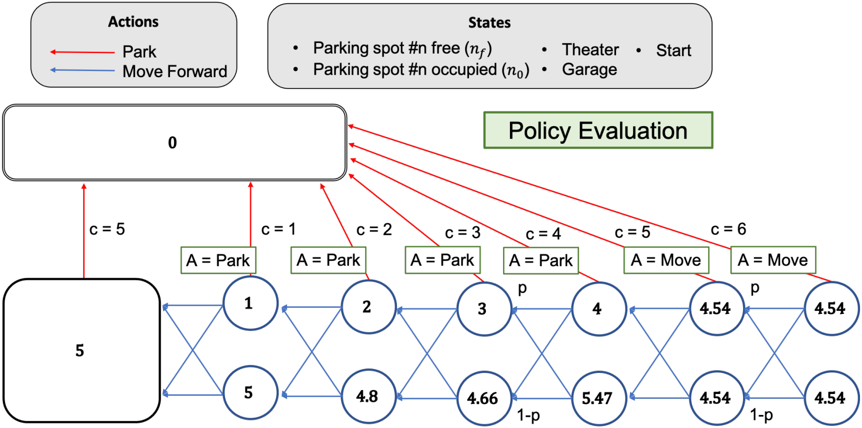
Policy Iteration



Policy Iteration



Policy Iteration



Policy Evaluation Step

Direct Strategy

Solve the linear system of equations

$$v^k_{\pi^k} = (I - \lambda P_{\pi^k})^{-1} r_{\pi^k}$$

Iterative Strategy

Set $v^{k,0}_{\pi^k}(s) = 0$

Iterate $v^{k,i+1}_{\pi^k}(s) = c(s, \pi^k(s)) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, \pi^k(s)) v^{k,i}_{\pi^k}(s')]$

Stop when $v^{k,i+1}_{\pi^k}(s) = v^{k,i}_{\pi^k}(s)$ for all $s \in \mathcal{S}$ and set $v^{k,i}_{\pi^k} = v^k_{\pi^k}$

Policy Evaluation Step

Direct Strategy

Solve the linear system of equations

$$v^{k}_{\pi^k} = (I - \lambda P_{\pi^k})^{-1} r_{\pi^k}$$

Iterative Strategy

Set $v^{k,0}_{\pi^k}(s) = 0$

Iterate $v^{k,i+1}_{\pi^k}(s) = c(s, \pi^k(s)) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, \pi^k(s)) v^{k,i}_{\pi^k}(s')]$

Stop when $v^{k,i+1}_{\pi^k}(s) = v^{k,i}_{\pi^k}(s)$ for all $s \in \mathcal{S}$ and set $v^{k,i}_{\pi^k} = v^{k}_{\pi^k}$

Policy Evaluation: Properties

Theorem

For the policy iteration algorithm we have that

- The value function is non-increasing, i.e., $v_{\pi^{k+1}}^{k+1} \leq v_{\pi^k}^{k}$
- The algorithm converges in a finite number of iterations
- Let π^∞ be the policy at convergence, then $\pi^\infty = \pi^*$

Policy Evaluation: Properties

Proof sketch:

- The value function is non-increasing and there is a finite number of policies (as the number of action is finite). Therefore, the policy iteration algorithm converges in a finite number of iterations
- At convergence we have that $\pi^{k+1} = \pi^k$ and therefore

$$v^{k+1}(s) = \min_{a \in \mathcal{A}} \left[c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a) v^{k+1}(s') \right], \forall s \in \mathcal{S}.$$

  Hence, v^{k+1} satisfies the Bellman equation and $\pi^{k+1} = \pi^*$.

Table of Contents

Linear Programming

Linear Programming

Let $\alpha(s) > 0$ for all $s \in \mathcal{S}$ and

$$\bar{v} = \arg\max_{v \in \mathbb{R}^{|\mathcal{S}|}} \quad \sum_{s \in \mathcal{S}} \alpha(s) v(s)$$

$$\text{subject to} \quad v(s) \leq c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a) v(s'), \qquad \forall a \in \mathcal{A},\ \forall s \in \mathcal{S}.$$

then, we have that $\bar{v} = v^*$.

Linear Programming

Proof Sketch. By feasibility of $\bar{v}$ we have

$$\bar{v}(s) \leq c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a)\bar{v}(s'),\ \forall a \in \mathcal{A},\ \forall s \in \mathcal{S}.$$

which is equivalent to

$$\bar{v}(s) \leq \min_{a \in \mathcal{A}} \left[c(s,a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s,a)\bar{v}(s') \right] = Bv(\bar{s}),\ \forall s \in \mathcal{S}.$$

Now recall that B is monotone and therefore $v(s) \leq Bv(s) \leq B^2 v(s) \leq \ldots \leq B^{\infty} v(s) = v^*(s),\ \forall s \in \mathcal{S}$..Hence, any feasible solution $v(s) \leq Bv(s) \leq v^*(s) = Bv^*(s)$. Concluding as $\alpha(s) > 0$, the feasible solution $v^*(s)$ is optimal.

Table of Contents

Summary Policy and Value Iteration

Policy Iteration

`Policy Evaluation:` Find V_{π^k} by solving

$$V_{\pi^k}(s) = c(s, \pi^k(s)) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, \pi^k(a)) V_{\pi^k}(s'), \ \forall s \in \mathcal{S}.$$

`Policy Improvement:` Compute π^{k+1} as

$$\pi^{k+1}(s) = \min_{a \in \mathcal{A}} \left[c(s, a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, a) V_{\pi^k}(s') \right], \ \forall s \in \mathcal{S}.$$

Value Iteration

For any $V \in \mathbb{R}^{|\mathcal{S}|}$ compute

$$V^*(s) = \lim_{k \to \infty} B^k V(s), \ \forall s \in \mathcal{S}.$$

Summary Policy and Value Iteration

Policy Iteration

`Policy Evaluation:` Find V_{π^k} by solving

$$V_{\pi^k}(s) = c(s, \pi^k(s)) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, \pi^k(a)) V_{\pi^k}(s'), \ \forall s \in \mathcal{S}.$$

`Policy Improvement:` Compute π^{k+1} as

$$\pi^{k+1}(s) = \min_{a \in \mathcal{A}} \left[c(s, a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, a) V_{\pi^k}(s') \right], \ \forall s \in \mathcal{S}.$$

Value Iteration

For any $V \in \mathbb{R}^{|\mathcal{S}|}$ compute

$$V^*(s) = \lim_{k \to \infty} B^k V(s), \ \forall s \in \mathcal{S}.$$

Table of Contents

Approximate Policy Iteration

Policy Iteration

`Policy Evaluation:` Find V_{π^k} by solving

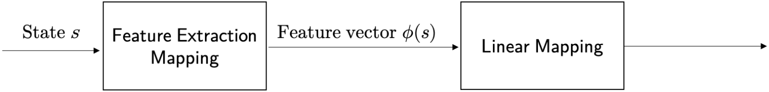
$$V_{\pi^k}(s) = c(s, \pi^k(s)) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, \pi^k(a)) V_{\pi^k}(s'), \ \forall s \in \mathcal{S}.$$

`Policy Improvement:` Compute π^{k+1} as

$$\pi^{k+1}(s) = \min_{a \in \mathcal{A}} \left[c(s, a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, a) V_{\pi^k}(s') \right], \ \forall s \in \mathcal{S}.$$

- Perform the policy evaluation step for all $s \in \bar{\mathcal{S}} \subset \mathcal{S}$
- Similar strategies for Value Iteration and Linear Programming

Approximation in the Value Space



Value function approximation

$$\hat{V}_\theta(s) = \sum_i \theta_i \phi_i(s) = \theta^\top \phi(s)$$

Chess example

- $\phi_1(s)$ = *material score* computed summing the points with the pieces on the board (pawn = 1, rook = 5, Knight and Bishops = 3, queen = 10)
- $\phi_2(s)$ = *mobility* given by the legal moves available,
- $\phi_3(s)$ = *center control* given by the number of pawns in the center
- $\phi_4(s)$ = *bishop's mobility* given by the amount of squared reachable by the bishop,

Policy Iteration w/ Value Function Approximation

We focus on a variant of approximate policy iteration based on Monte Carlo simulations and function approximation.

Approximate Policy Iteration

`Policy Evaluation:` For a set of representative states $\bar{\mathcal{S}} \subset \mathcal{S}$ run M simulations using the policy π^k. Then, compute the cost of each ith simulation from the state $s \in \bar{\mathcal{S}}$ denoted as $\bar{c}(i, s)$ and approximate the value function $\hat{V}_\theta(s) = \sum_{s \in \mathcal{S}} \theta^\top \phi(s)$ solving the following problem

$$\theta^k = \arg\min_\theta \sum_{s \in \bar{\mathcal{S}}} \sum_{i=1}^{M} ||\hat{V}_\theta(s) - \bar{c}(i, s)||.$$

`Policy Improvement:` Compute π^{k+1} as

$$\pi^{k+1} = \min_{a \in \mathcal{A}} \left[c(s, a) + \sum_{s' \in \mathcal{S}} \lambda p(s'|s, a) \hat{V}_{\theta^k}(s') \right].$$

Theoretical Basis for Approximate Policy Iteration

Theorem

If policies are approximately evaluated using an approximated value function such that

$$\max_s |V_{\theta^k}(s) - V_{\pi^k}(s)| \leq \delta, \quad \forall k = 0, 1, \dots$$

and the policy improvement is approximate

$$\max_s |B_{\pi^{k+1}} V_{\theta^k}(s) - B V_{\theta^k}(s)| \leq \epsilon, \quad \forall k = 0, 1, \dots.$$

Then, we have that

$$\limsup_{k \to \infty} \max_s |V_{\pi^k}(s) - V^*(s)| \leq \frac{\epsilon + 2\lambda\delta}{(1 - \lambda)^2}$$

Readings

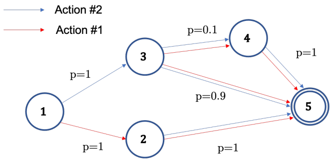
- Chapter 2 and Chapter 6.2 "Neuro-Dynamic Programming" Dimitri P. Bertsekas and John Tsitsiklis
- Chapter 6 "Markov decision processes: discrete stochastic dynamic programming." M. Puterman
- D. Bertsekas, "Feature-based aggregation and deep reinforcement learning: A survey and some new implementations." IEEE/CAA Journal of Automatica Sinica 6.1 (2018): 1-31.
- D. Bertsekas, "Biased aggregation, rollout, and enhanced policy improvement for reinforcement learning." arXiv preprint arXiv:1910.02426 (2019).
- D. P. De Farias, and B. Van Roy. "The linear programming approach to approximate dynamic programming." Operations research 51.6 (2003): 850-865.

Summary

- We discussed how to solve optimal control problem with discrete state and action spaces of the form

$$\pi^* = \arg\min_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \lambda^t r(s_t, a_t) | \pi\right].$$

- The solution can be computed exactly given a known model and state-action spaces of moderate size.
- Approximate dynamic programming can be used to reduce the computational complexity of syntehsis strategies.



What is next?

Optimal Control Problem with Continuous State Spaces:
In the next lectures we will

- Compute a control policy mapping continuous state to continuous control action

$$\pi : \mathbb{R}^n \to \mathbb{R}^d$$

- Leverage the same ideas to synthesize optimal policies, but computing/approximating the value function is harder for problem with constraints.
- Present learning-based strategies to approximate the value function in continuous state-action spaces.