Model Predictive Control and Reinforcement Learning
– MDPs, Policy and Value Iteration –

Joschka Boedecker and Moritz Diehl

University Freiburg

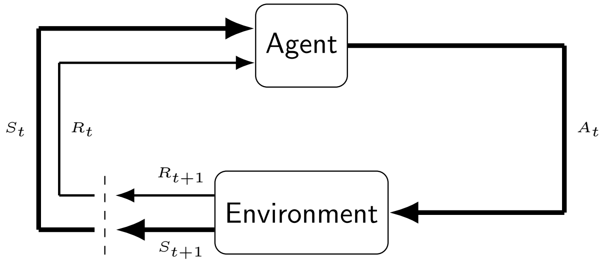
July 27, 2021

Lecture Overview

Acknowledgement

Slide contents are partially based on *Reinforcement Learning: An Introduction* by Sutton and Barto and the Reinforcement Learning lecture by David Silver.

Agent and Environment



Time steps t: $0, 1, 2, \dots$
States: $S_0, S_1, S_2, \dots$
Actions: $A_0, A_1, A_2, \dots$
Rewards: $R_1, R_2, R_3, \dots$

Markov Decision Processes

A finite Markov Decision Process (MDP) is a 4-tuple $\langle \mathcal{S}, \mathcal{A}, p, \mathcal{R} \rangle$, where

- $\mathcal{S}$ is a finite number of states,
- $\mathcal{A}$ is a finite number of actions,
- p is the transition probability function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$,
- and $\mathcal{R}$ is a finite set of scalar rewards. We can then define expected reward $r(s, a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$ and $r(s, a, s') = \mathbb{E}[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s']$.

Markov Property

A state-reward pair (S_{t+1}, R_{t+1}) has the Markov property iff:

$$\mathsf{Pr}\{S_{t+1}, R_{t+1} | S_t, A_t\} = \mathsf{Pr}\{S_{t+1}, R_{t+1} | S_t, A_t, \dots, S_0, A_0\}.$$

The future is independent of the past given the present.

Markov Decision Processes

A finite Markov Decision Process (MDP) is a 4-tuple $\langle \mathcal{S}, \mathcal{A}, p, \mathcal{R} \rangle$, where

- $\mathcal{S}$ is a finite number of states,
- $\mathcal{A}$ is a finite number of actions,
- p is the transition probability function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$,
- and $\mathcal{R}$ is a finite set of scalar rewards. We can then define expected reward $r(s, a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$ and $r(s, a, s') = \mathbb{E}[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s']$.

A deterministic system is a special case of an MDP:

$$p(s_{t+1} | s_t, u_t) = \begin{cases} 1 & s_{t+1} = f(s_t, a_t) \\ 0 & \text{otherwise} \end{cases}$$

Rewards

- A reward R_t in time step t is a **scalar** feedback signal.
- R_t indicates how well an agent is performing **at single time step** t.

Reward Hypothesis

All of what we mean by goals and purposes can be well thought of as the maximization/minimization of the expected value of the cumulative sum of a received scalar signal (called reward/cost).

Examples:

- Chess: $+1$ for winning, -1 for losing
- Walking: $+1$ for every time step not falling over
- Investment Portfolio: difference in value between two time steps

Return

- The agent aims at maximizing the expected **cumulative reward**
- Non-discounted: $G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$
- Discounted: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
- Discounting with $\gamma \in [0, 1]$ to prevent from infinite returns (e.g. in infinite horizon control problems)
- Returns at successive time steps are related to each other:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

MDP: Example

Description

Imagine a house cleaning robot. It can have three charge levels: *high*, *low* and *none*. At every point in time, the robot can decide to *recharge* or to *explore* unless it has no battery. When exploring, the charge level can reduce with probability ρ. Exploring is preferable to recharging, however it has to avoid running out of battery.

Formalize the above problem as an MDP.

MDP: Example

Solution

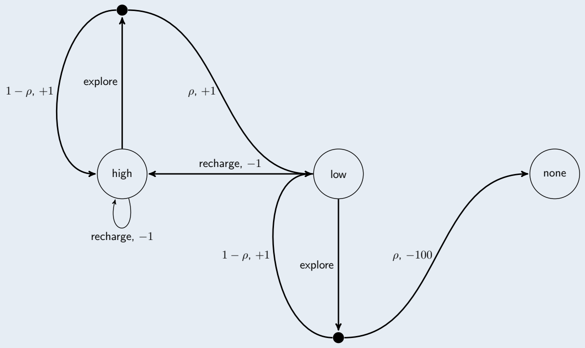
For the given problem, we set:

- $\mathcal{S} = \{\text{high}, \text{low}, \text{none}\}$
- $\mathcal{A} = \{\text{explore}, \text{recharge}\}$
- $\mathcal{R} = \{+1, -1, -100\}$ for exploring, recharging, and transitions leading to none, respectively.
- p has entries with value 1 for transitions $(\text{high}, -1, \text{high}, \text{recharge})$, $(\text{low}, -1, \text{high}, \text{recharge})$ and $(\text{none}, 0, \text{none}, \cdot)$. It further has entries with value ρ for transitions $(\text{high}, +1, \text{low}, \text{explore})$ and $(\text{low}, -100, \text{none}, \text{explore})$ and entries with value $1 - \rho$ for transitions $(\text{high}, +1, \text{high}, \text{explore})$ and $(\text{low}, +1, \text{low}, \text{explore})$.

MDP: Example

Solution

The transition graph therefore is:



Policies

- The policy defines the behaviour of the agent:
  - can be stochastic: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$
  - or deterministic: $\pi(s) = a$
- Due to the Markov property, knowledge of the current state s is sufficient to make an informed decision.

Value Functions

- Value Function $v_\pi(s)$ is the expected return when starting in s and following π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right]$$

- Action-Value Function q_π is the expected return when starting in s, taking action a and following π thereafter:

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right]$$

- Simple connection:

$$v_\pi(s) = \mathbb{E}_\pi[q_\pi(s, \pi(s))] \tag{1}$$

Bellman Equation

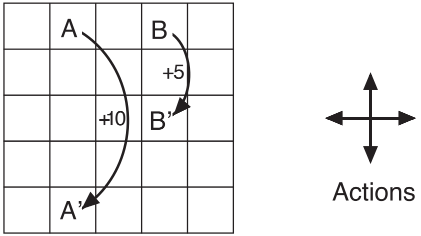
- The Bellman Equation expresses a relationship between the value of a state and the values of its successor states
- The value function v_π is the unique solution to its Bellman Equation

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi[R_t + \gamma G_{t+1} | S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']\right] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right]
\end{aligned}
$$

Bellman Equation for v_π

$$
v_\pi(s) = \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right].
$$

Bellman Equation: Example

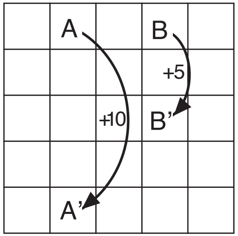


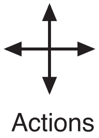
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Description

Actions move the agent deterministically. Actions that would move the agent off the grid cost -1 with no state change. All other actions are free. However, every action performed by the agent in A moves it to A' with a reward of $+10$, each action in B moves it to B' with a reward of $+5$. Assume a uniform policy. v_π with a discounting factor of $\gamma = 0.9$ is to the right. Show exemplary for state $s_{0,0}$ with $v_\pi(s_{0,0}) = 3.3$ that the Bellman equation is satisfied.

Bellman Equation: Example





3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Solution

$$\begin{aligned} v_\pi(s_{0,0}) &= 0.25 \cdot (-1 + \gamma \cdot 3.3) + 0.25 \cdot (+0 + \gamma \cdot 8.8) + \\ &\quad 0.25 \cdot (+0 + \gamma \cdot 1.5) + 0.25 \cdot (-1 + \gamma \cdot 3.3) \\ &= 3.3025 \approx 3.3 \end{aligned}$$

Bellman Equation

For a deterministic system and a deterministic policy, the Bellman Equation simplifies to:

Bellman equation for value-function v_π for a deterministic system and policy

$$v_\pi(s) = r + \gamma \max_{a'} v_*(f(s,a)).$$

We equivalently obtain a corresponding system of equations for the Q-function:

Bellman Equation for action-value function q_π

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a') \right].$$

Optimality of Policies

We consider a policy as optimal if the value (i.e. its expected return under the policy) in every state is at least as high as for any other policy:

Optimality of a policy π_*

A policy π_* is called *optimal* $:\Leftrightarrow$
For all $s \in S$:

$$v_{\pi_*}(s) \geq v_\pi(s) \text{ for all } \pi \tag{2}$$

The corresponding *optimal value function* is denoted by v_*.

- This requires a search among all, possibly infinitely many, policies. This seems to be rather impractical.
- Is there an easier way to check if a policy π and corresponding value function v_π is actually optimal?

Bellman Optimality Equation

Intuitively, the Bellman Optimality Equation expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state:

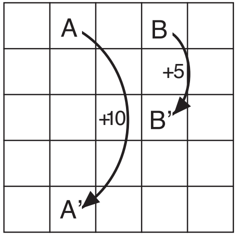
$$\begin{aligned} v_*(s) &= \max_a q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \end{aligned}$$

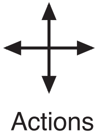
Bellman Optimality Equation for v_*

The Bellman Equation for the optimal value function v_* is defined as:

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')].$$

Bellman Optimality Equation: Example





3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Non-optimality of the uniform random policy

$$\begin{aligned} v_\pi(s_{0,0}) = 3.3025 \quad & \neq \quad \max\{-1+\gamma\cdot 3.3, 0+\gamma\cdot 8.8, \\ & \qquad 0+\gamma\cdot 1.5, -1+\gamma\cdot 3.3\} \\ & = \quad 7.92 \end{aligned}$$

Bellman Optimality Equation

For a deterministic system and a deterministic policy, the Bellman Optimality Equation simplifies to:

Bellman equation for the optimal value-function v_* for a deterministic system and policy

$$v_*(s) = r + \gamma \max_{a'} v_*(f(s,a)).$$

Equivalently, there exists a Bellman optimality equation for Q-functions:

Bellman equation for the optimal action-value function q_*

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_*(s',a')].$$

How can we turn these equations into practical algorithms to find optimal policies π_*?

Policy Iteration: Overview

Idea: Alternate **evaluating** the value function v_π and **improving** the policy π to convergence.

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

Policy Evaluation

Compute the state-value function v_π for an arbitrary policy π.
$\forall s \in S$:

$$v_\pi(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_\pi(s')\right]$$

If the environments dynamics are completely known, this is a system of $|\mathcal{S}|$ simultaneous linear equations in $|\mathcal{S}|$ unknowns. With the Bellman equation, we can iteratively update an initial approximation v_0:

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi \left[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s\right] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_k(s')\right] \end{aligned}$$

Policy Evaluation

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

```
Input $\pi$, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
    $\Delta \leftarrow 0$
    Loop for each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a) \big[ r + \gamma V(s') \big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$
```

Policy Improvement

Once we have the value function for a policy, we consider which action a to select in a state s when we follow our old policy π afterwards. To decide this, we look at the Bellman equation of the state-action value function:

$$\begin{aligned} q_\pi(s,a) &\doteq \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a\right] \\ &= \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \end{aligned}$$

Policy improvement theorem

Let π and π' be any pair of deterministic policies. If, $\forall s \in S$,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s),$$

then the policy π' must be as good as, or better than, π. It follows that, $\forall s \in S$:

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

Policy Improvement

To implement this, we compute $q_\pi(s,a)$ for *all* states and *all* actions, and consider the greedy policy:

$$\begin{aligned}\pi'(s) &\doteq \arg\max_a q_\pi(s,a)\\ &= \arg\max_a \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t_1}) | S_t = s, A_t = a\right]\\ &= \arg\max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]\end{aligned}$$

Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

```
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
      $\Delta \leftarrow 0$
      Loop for each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r \,|\, s, \pi(s)) \big[r + \gamma V(s')\big]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   $\textit{policy-stable} \leftarrow \textit{true}$
   For each $s \in \mathcal{S}$:
      $\textit{old-action} \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \,|\, s, a) \big[r + \gamma V(s')\big]$
      If $\textit{old-action} \neq \pi(s)$, then $\textit{policy-stable} \leftarrow false$
   If $\textit{policy-stable}$, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2
```

Value Iteration

Performing policy evaluation to convergence *in every iteration* is costly and often not necessary.
A special case is to evaluate just once and combine it with the policy improvement step:

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathbb{E}\left[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p(s', r | s, a)\left[r + \gamma v_k(s')\right] \end{aligned}$$

Value Iteration

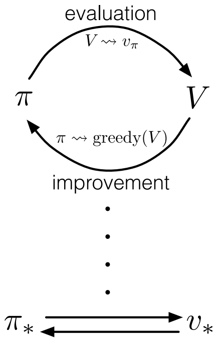
Value Iteration, for estimating $\pi \approx \pi_*$

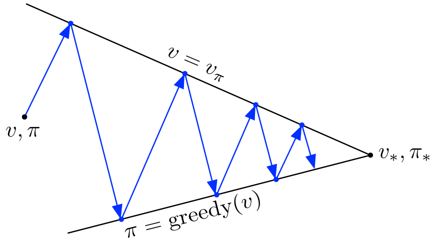
```
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\textit{terminal}) = 0$

Loop:
|   $\Delta \leftarrow 0$
|   Loop for each $s \in \mathcal{S}$:
|        $v \leftarrow V(s)$
|        $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
|        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
    $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
```

Generalized Policy Iteration


evaluation
$V \rightsquigarrow v_\pi$
π
V
$\pi \rightsquigarrow \text{greedy}(V)$
improvement
π_*
v_*



- Policy Evaluation: estimate v_π
- Policy Improvement: greedy

Summary

- MDPs allow us to formalize RL (and more generally, stochastic optimal control) problems, 4-tuple $\langle \mathcal{S}, \mathcal{A}, p, \mathcal{R} \rangle$, assume Markov Property holds
- Bellman Equations express a relationship between the value of a state and the values of its successor states, provide structure to search for an optimal policy *intelligently*
- Policy Iteration and Value iteration use the structure of the Bellman Equations and turn them into iterative algorithms for finding optimal policies given an MDP (with and without explicit representation of the policy)