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

Joschka Boedecker and Moritz Diehl

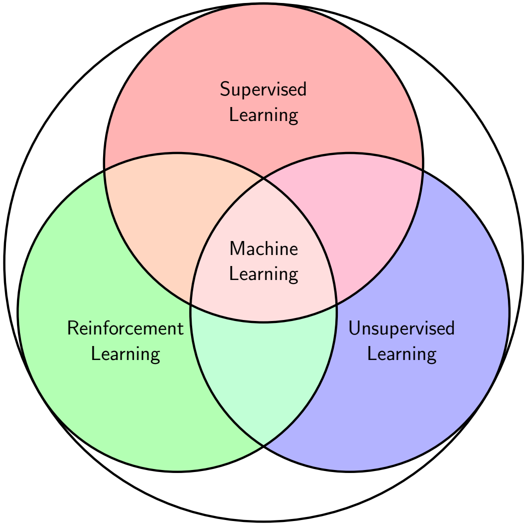
University of Freiburg

Fall School on Model Predictive Control and Reinforcement Learning
Freiburg, 6-10 October 2025

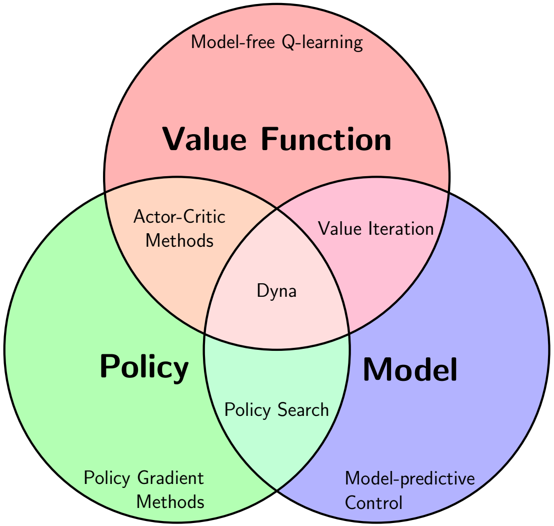
Acknowledgement

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

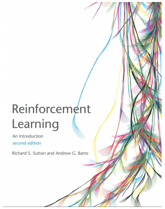
Reinforcement Learning in Machine Learning



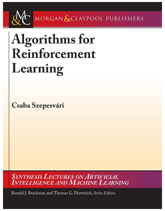
RL Overview


Model-free Q-learning
Value Function
Actor-Critic Methods
Value Iteration
Dyna
Policy
Model
Policy Search
Policy Gradient Methods
Model-predictive Control

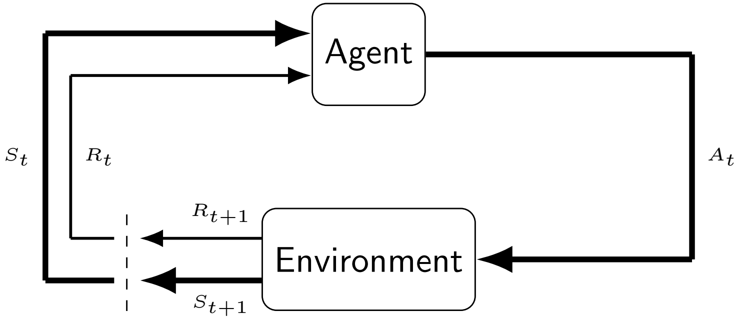
Literature


Reinforcement Learning: An Introduction (Sutton and Barto, 2018) `http://incompleteideas.net/book/the-book.html`

Algorithms for Reinforcement Learning (Szepesvári, 2010) `https://sites.ualberta.ca/~szepesva/RLBook.html`



Agent and Environment



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

Markov Decision Processes

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

- $\mathcal{S}$ is a finite number of states,
- $\mathcal{A}$ is a finite number of actions,
- P is the transition probability function $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$,
- and $r : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{R}$, where $\mathcal{R}$ is a finite set of scalar rewards.

Markov Property

The transition function has the Markov property iff:

$$\text{Pr}\{S_{t+1} \mid S_t, A_t\} = \text{Pr}\{S_{t+1} \mid S_t, A_t, \ldots, S_0, A_0\}.$$

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 $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$,
- and $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$, where $\mathcal{R}$ is a finite set of scalar rewards.

A deterministic system is a special case of an MDP:

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

Rewards in the spotlight

- We denote $R_{t+1} := r(S_t, A_t)$.
- A reward R_t in time step t is a **scalar** feedback signal.
- R_t indicates how well an agent is performing **at the single time step** t.

Agent goal: maximize the **expected discounted cumulative reward**
$G_t = R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-(t+1)} R_T$.

Which property should γ have?

- $\gamma \in (0, 1)$

Why do we need discounting?

- T can be infinite. By introducing discounting, we avoid infinite cumulative reward for infinite horizon MDPs.

Rewards in the spotlight (cont'd)

Reward Hypothesis

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

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

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

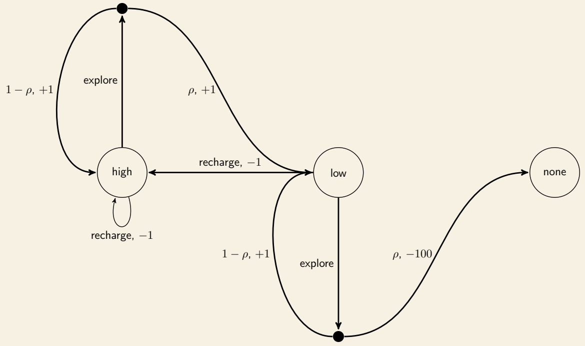
For the given problem, we set:

- $\mathcal{S} = \{\text{high}, \text{low}, \text{none}\}$
- $\mathcal{A} = \{\text{explore}, \text{recharge}\}$
- $\mathcal{R} = \{+1, -1, -100\}$ 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:



Components of RL Systems – Models

- The model defines the transitions between states in an environment
  - Given a current state and action, the model P yields the next state.
  - $P(s^+ \mid s, a) = \mathsf{Pr}\{S_{t+1} = s^+ \mid S_t = s, A_t = a\}$

Components of RL Systems – Policies

- The policy defines the behavior of the agent:
  - is a mapping from a state to an action
  - can be stochastic: $\pi(a \mid s) = \mathbb{P}[A = a \mid S = s]$
  - or deterministic: $\pi(s) = a$
- Due to the Markov property, knowledge of the current state s is sufficient to make an informed decision.

Components of RL Systems – Value Functions

- Let $S_t \sim P(\cdot \mid S_{t-1}, A_{t-1})$ and $A_t \sim \pi(\cdot \mid S_t)$.
- Value Function $V^\pi(s)$ is the expected return when starting in s and following π:

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

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

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

- Simple connection:

$$V^\pi(s) = \mathbb{E}_\pi[Q^\pi(s,a)]$$

Exploration and Exploitation

- Fundamental problem in Reinforcement Learning
- The agent has to exploit what it knows in order to obtain high reward (**Exploitation**)...
- ...but it has to explore to possibly do better in the future (**Exploration**).

Example: You want to go out for dinner. Do you...

- go to your favourite restaurant
- or try a new one?

Discussion

Where would you apply Reinforcement Learning?

Bellman Equation

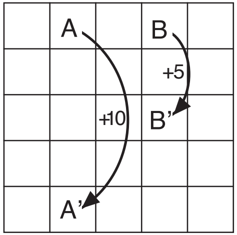
- The Bellman Equation expresses a relationship between the value of a state and the values of its successor states
- The value function V^π is the unique solution to its Bellman Equation

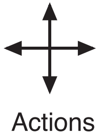
$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_t + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a \mid s) \sum_{s^+} P(s^+ \mid s, a) \left[r(s,a) + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s^+] \right] \\
&= \sum_a \pi(a \mid s) \sum_{s^+} P(s^+ \mid s, a) \left[r(s,a) + \gamma V^\pi(s^+) \right]
\end{aligned}
$$

Bellman Equation for V^π

$$
V^\pi(s) = \sum_a \pi(a \mid s) \sum_{s^+} P(s^+ \mid s, a) \left[r(s,a) + \gamma V^\pi(s^+) \right].
$$

Bellman Equation: Example



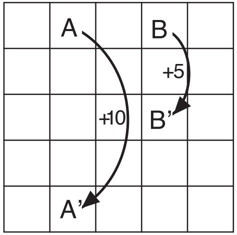


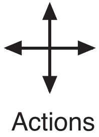
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

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 $\gamma = 0.9$ can be seen on 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

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

Optimality of Policies

We consider a policy 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* iff for all $s \in S$:

$$V^{\pi^*}(s) \geq V^{\pi}(s) \text{ for all } \pi. \tag{1}$$

The corresponding *optimal value function* is denoted by V^*.

- Finding π^* 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 the 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:

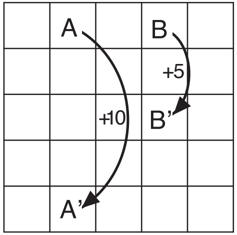
$$\begin{aligned} V^*(s) &= \max_a Q^{\pi^*}(s,a) \\ &= \max_a \mathbb{E}_{\pi^*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi^*}[R_{t+1} + \gamma V^*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s^+} P(s^+ \mid s,a)[r(s,a) + \gamma V^*(s^+)] \end{aligned}$$

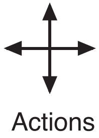
Bellman Optimality Equation for V^*

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

$$V^*(s) = \max_a \sum_{s^+} P(s^+ \mid s,a)[r(s,a) + \gamma V^*(s^+)].$$

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

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

$\Rightarrow$ random policy π is *not* optimal.

Bellman Optimality Equation

For a deterministic system with transition function f and deterministic policies, the Bellman Optimality Equation simplifies to:

Bellman equation for the optimal value-function V^* for a deterministic system and policies

$$V^*(s) = \max_a r(s,a) + \gamma V^*(f(s,a)).$$

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

Bellman equation for the optimal action-value function Q^*

$$Q^*(s,a) = \sum_{s^+} P(s^+ \mid s,a)[r(s,a) + \gamma \max_{a^+} Q^*(s^+,a^+)].$$

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.
$\forall s \in S$:

$$V^\pi(s) \doteq \sum_a \pi(a \mid s) \sum_{s^+} P(s^+ \mid s, a) \left[r(s,a) + \gamma V^\pi(s^+)\right]$$

If the environments dynamics are completely known, this is a system of $\mid \mathcal{S} \mid$ simultaneous linear equations in $\mid \mathcal{S} \mid$ unknowns. With the Bellman equation, we can iteratively update an initial approximation V^0:

$$\begin{aligned} V^{k+1}(s) &\doteq \mathbb{E}_\pi \left[R_{t+1} + \gamma V^k(S_{t+1}) \mid S_t = s\right] \\ &= \sum_a \pi(a \mid s) \sum_{s^+} P(s^+ \mid s, a) \left[r(s,a) + \gamma V^k(s^+)\right] \end{aligned}$$

Policy Evaluation

Algorithm 1 Iterative Policy Evaluation – for estimating $V \approx V^\pi$

```
1: Input: policy $\pi$ to be evaluated, threshold $\theta > 0$ determining accuracy of estimation
2: Initialize: $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$
3: while $\Delta > \theta$ do
4:     $\Delta \leftarrow 0$
5:     for each $s \in \mathcal{S}$ do
6:         $v \leftarrow V(s)$
7:         $V(s) \leftarrow \sum_a \pi(a \mid s) \sum_{s^+} P(s^+ \mid s, a)[r(s,a) + \gamma V(s^+)]$
8:         $\Delta \leftarrow \max(\Delta, \mid v - V(s) \mid)$
9:     end for
10: end while
```

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:

$$\begin{aligned} Q^\pi(s,a) &\doteq \mathbb{E}\left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \sum_{s^+} P(s^+ \mid s,a)\left[r(s,a) + \gamma V^\pi(s^+)\right] \end{aligned}$$

Policy improvement theorem

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

$$Q^\pi(s, \pi(s)) \geq V^\pi(s),$$

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:

$$\begin{aligned}
\pi'(s) &\doteq \arg\max_a Q^\pi(s,a) \\
&= \arg\max_a \mathbb{E}\left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s, A_t = a\right] \\
&= \arg\max_a \sum_{s^+} P(s^+ \mid s,a)\left[r(s,a) + \gamma V^\pi(s^+)\right]
\end{aligned}$$

Policy Iteration – Complete algorithm

Algorithm 2 Policy iteration using iterative policy evaluation – for estimating $\pi \approx \pi^*$

```
1:  Initialize: $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$
2:  Policy Evaluation: follow the pseudo-code above, obtain $V \approx V^\pi$
3:  Execute Policy Improvement:
4:  while True do
5:      policy-stable $\leftarrow$ True
6:      for each $s \in \mathcal{S}$ do
7:          old-action $\leftarrow \pi(s)$
8:          $\pi(s) \leftarrow \arg\max_a \sum_{s^+} P(s^+ \mid s, a)[r(s,a) + \gamma V(s^+)]$
9:          if old-action $\neq \pi(s)$ then
10:             policy-stable $\leftarrow$ False
11:         end if
12:     end for
13:     if policy-stable then
14:         return $V \approx V^*$, $\pi \approx \pi^*$
15:     else
16:         Go to Policy Evaluation
17:     end if
18: end while
```

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:

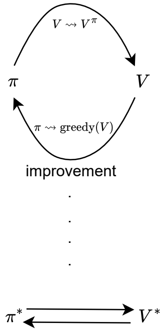
$$\begin{aligned} V^{k+1}(s) &\doteq \max_a \mathbb{E}\left[R_{t+1} + \gamma V^k(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \max_a \sum_{s^+} P(s^+ \mid s, a)\left[r(s,a) + \gamma V^k(s^+)\right] \end{aligned}$$

Value Iteration

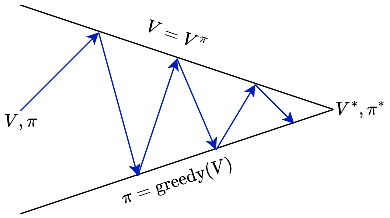
Algorithm 3 Value Iteration – for estimating $\pi \approx \pi^*$

```
1: Input: Threshold $\theta > 0$ determining accuracy of estimation
2: Initialize: $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$
3: while $\Delta > \theta$ do
4:     $\Delta \leftarrow 0$
5:     for each $s \in \mathcal{S}$ do
6:         $v \leftarrow V(s)$
7:         $V(s) \leftarrow \max_a \sum_{s^+} P(s^+ \mid s, a)[r(s,a) + \gamma V(s^+)]$
8:         $\Delta \leftarrow \max(\Delta, \mid v - V(s) \mid)$
9:     end for
10: end while
11: return deterministic policy $\pi \approx \pi^*$ such that $\pi(s) = \arg\max_a \sum_{s^+} P(s^+ \mid s, a)[r(s,a) + \gamma V(s^+)]$
```

Generalized Policy Iteration



$V \rightsquigarrow V^\pi$
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 $\langle \mathcal{S}, \mathcal{A}, P, r \rangle$, assuming the 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)