Lecture 01: Introduction to Reinforcement Learning

Paul Swoboda

Table of Contents

General Info

Contact

- Email: paul.swoboda@uni-mannheim.de, always prepend [RL 2023] in the header of your email.
- Office: Room B2.05, Building B 6, 26
- Individual appointments on request (remote or personally)

Teaching Materials and Q&A

- Website: paulswoboda.net/rl2023
- Discord server: https://discord.gg/W9QKxsTz48

Acknowledgments

- Slides are courtesy of Wilhelm Kirchgässner, Maximilian Schenke, Oliver Wallscheid and Daniel Weber from Paderborn University

Regulations

Registration

- Write me an email with your matrikulation ID and your full name.

Final exam

- Oral examination or written, depending on number of participants
- 30 minutes (oral) / 90-120 minutes (written)

Homework assignments

- Homework sheets with 1-2 week deadlines
- Coding (python) and classical question-answer queries
- Equally distributed over lecture time frame
- Regulation:
  - Admission to exam if 50+ % points obtained from exercise sheets
  - Final grade: 50 % exercise, 50 % exam
  - Exercise groups of up to 3 people

Course Outline

The course will cover the following content (not necessarily in that order):

- Conceptual basics and historical overview
- Markov decision processes
- Dynamic programming
- Monte Carlo learning
- Temporal difference learning
- Bootstrapping
- On- and Off-policy strategies
- Function approximation and deep learning
- Policy gradient methods

We will try to learn fun agents!

Recommended Textbooks

- Reinforcement learning: an introduction,
  - R. Sutton and G. Barto
  - MIT Press, 2nd edition, 2018
  - Available online

- Reinforcement learning (lecture script)
  - D. Silver
  - Entire slide set available here
  - YouTube lecture series (click here)

Table of Contents

The Basic Reinforcement Learning Structure

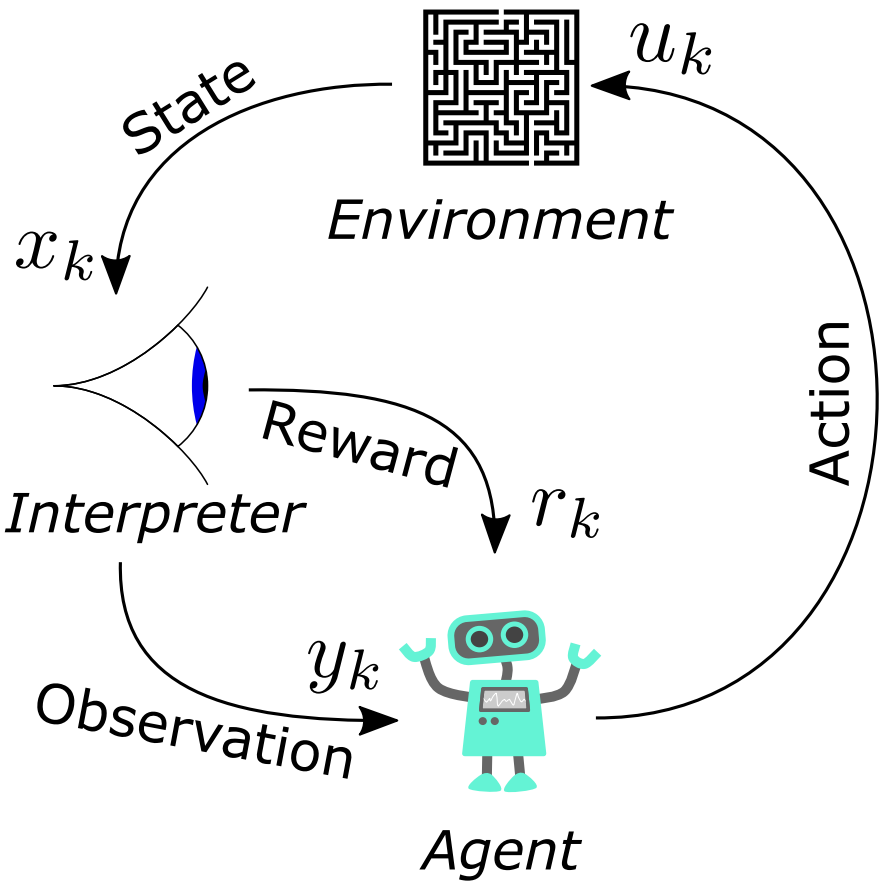
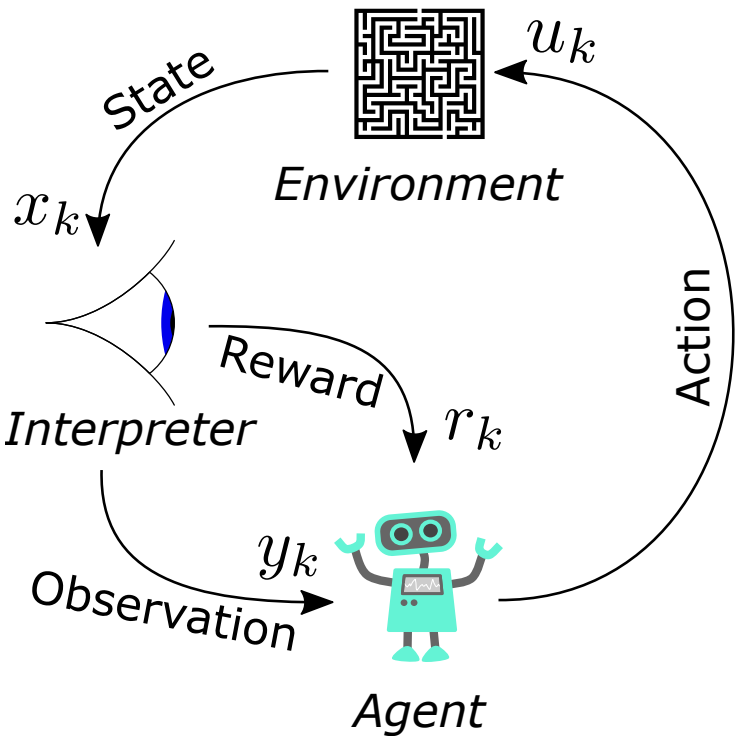


Fig. 1.1: The basic RL operation principle (derivative of www.wikipedia.org, CC0 1.0)

Key characteristics:

- No supervisor
- Data-driven
- Discrete time steps
- Sequential data stream (not i.i.d. data)
- Agent actions affect subsequent data (sequential decision making)

Agent and Environment



At each step k the agent:

- Picks an action a_k.
- Receives an observation o_k.
- Receives a reward r_k.

At each step k the environment:

- Receives an action a_k.
- Emits an observation o_{k+1}.
- Emits a reward r_{k+1}.

The time increments $k \leftarrow k + 1$.

Remark on time

Classically, a one step time delay is assumed between executing the action and receiving the corresponding observation and reward. In many applications the resulting time interval $\Delta T = t_k - t_{k+1}$ is a constant.

Some Basic Definitions from the Literature

What is reinforcement?

- "*Reinforcement is a consequence applied that will strengthen an organism's future behavior whenever that behavior is preceded by a specific antecedent stimulus.[...]There are two types of reinforcement, known as positive reinforcement and negative reinforcement; positive is where by a reward is offered on expression of the wanted behaviour and negative is taking away an undesirable element in the persons environment whenever the desired behaviour is achieved.*", wikipedia.org (obtained 2020-02-07)

What is learning?

- "*Acquiring knowledge and skills and having them readily available from memory so you can make sense of future problems and opportunities.*", From Make It Stick: The Science of Successful Learning, Brown et al., Harvard Press, 2014

Context Around Reinforcement Learning

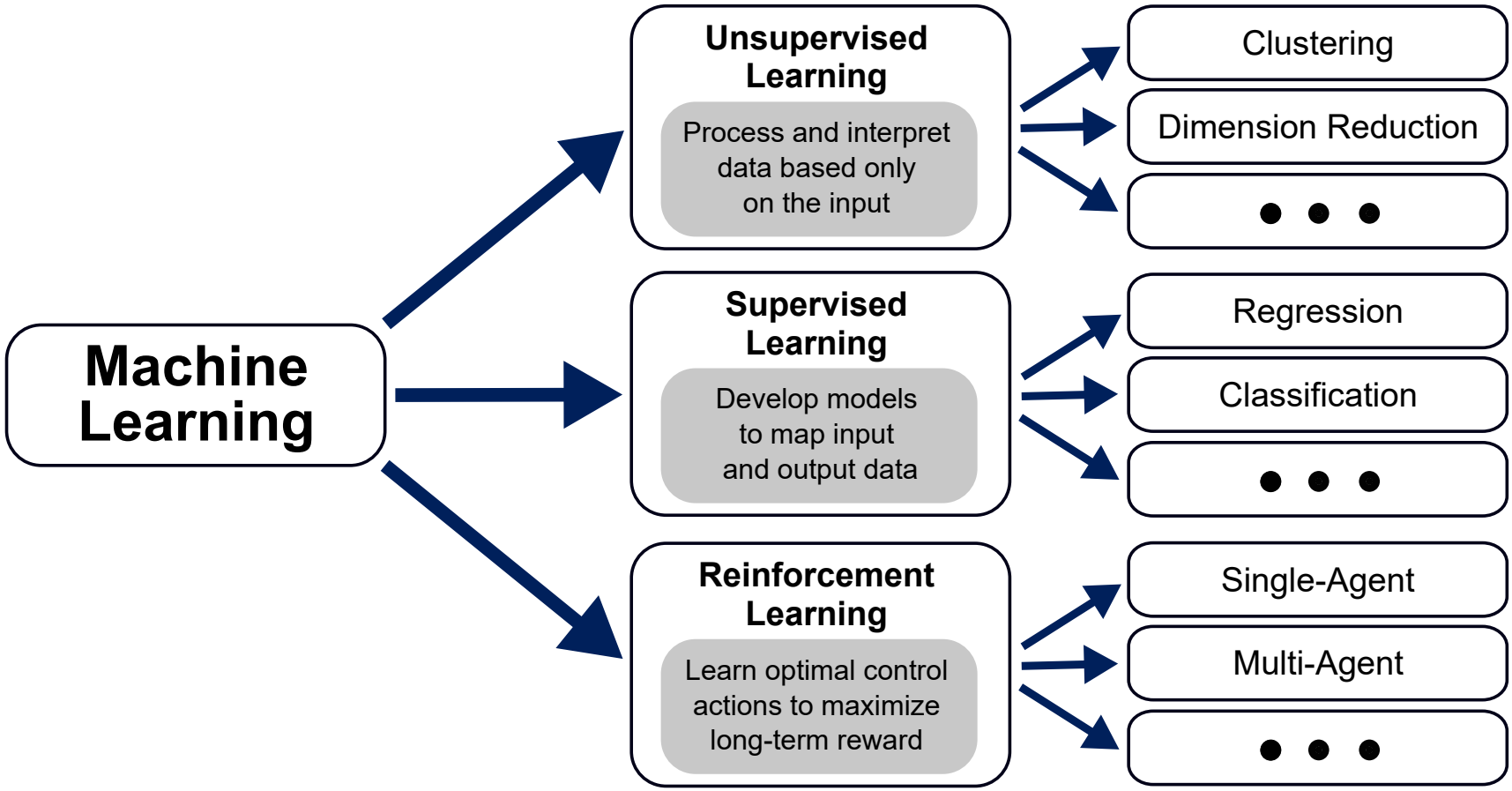


Fig. 1.2: Disciplines of machine learning

Context Around Machine Learning

Deep Learning (DL)

A class of ML which uses huge, layered models (e.g. large artifical neural networks) to progressively extract more information from the data.

Machine Learning (ML)

A subset of AI involved with the creation of algorithms which can modify itself without human intervention to produce desired output by feeding itself through structured data.

Artifical Intelligence (AI)

Any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. AI is often used to describe machines that mimic "cognitive" functions that humans associate with the human mind.

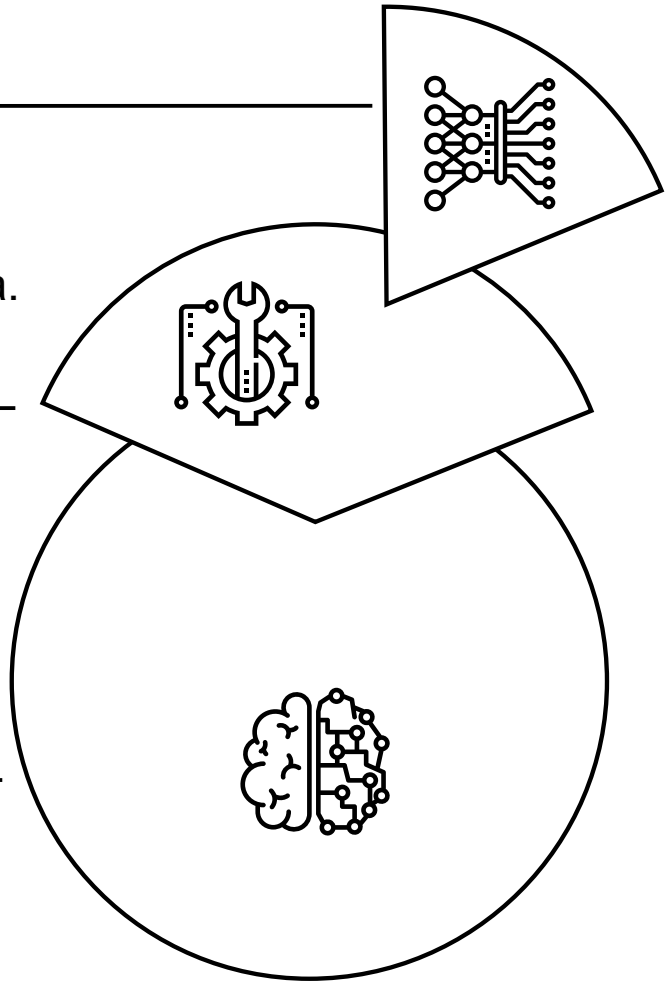


Fig. 1.3: The broader scope around machine learning

Many Faces of Reinforcement Learning

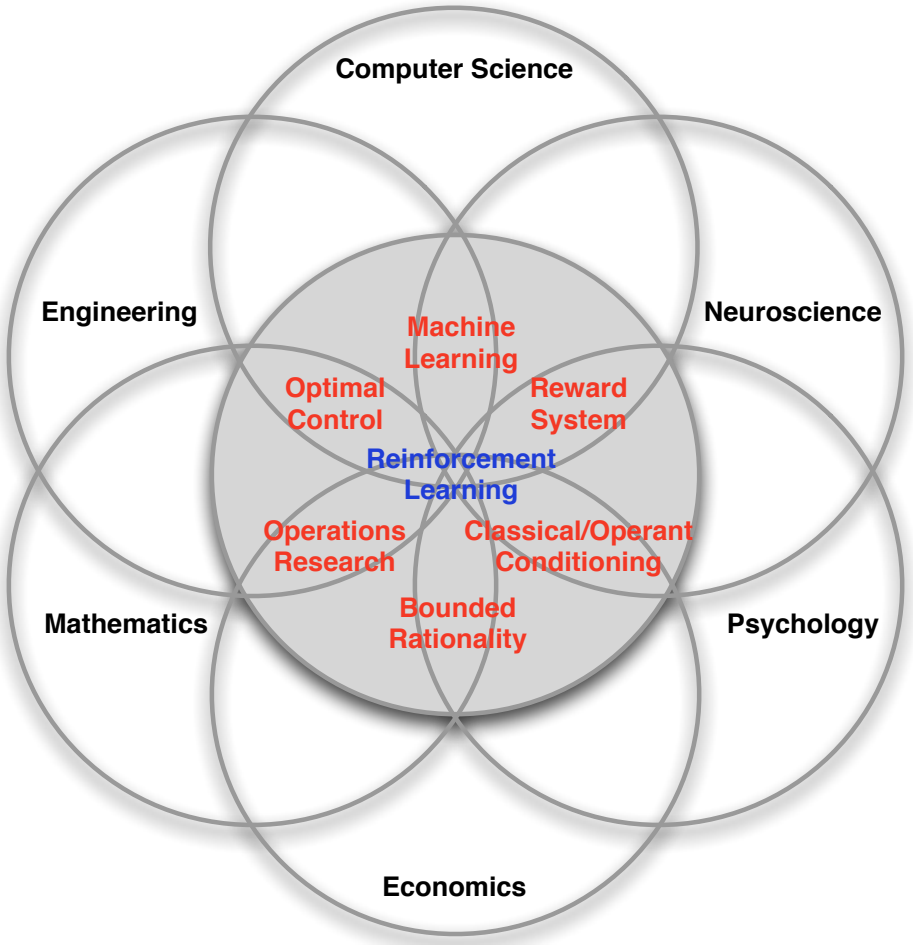


Fig. 1.4: RL and its neighboring domains
(source: D. Silver, Reinforcement learning, 2016. CC BY-NC 4.0)

Table of Contents

Methodical Origins

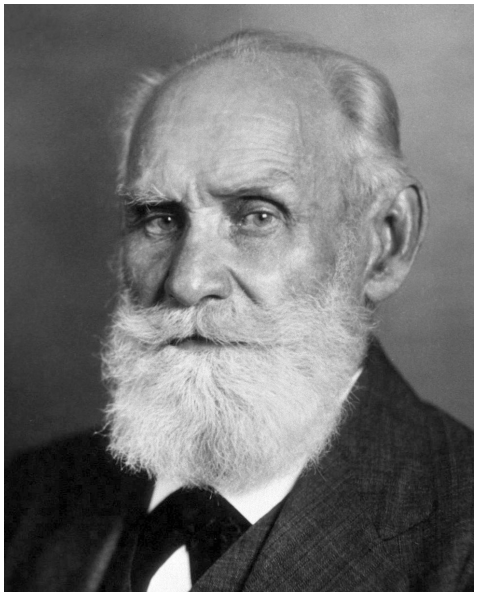


Fig. 1.5: Ivan Pavlov (1849-1936)

- Classical conditioning

Fig. 1.6: Andrei Markov (1856-1922)

- Stochastic process formalism

Fig. 1.7: Richard Bellman (1920-1984)

- Optimal sequential decision making

History of Reinforcement Learning

Huge field with many interconnections to different fields. One could give a lecture only on the historic development. Hence, interested readers are referred to:

- Chapter 1.7 of Barto/Sutton, *Reinforcement learning: an introduction*, 2nd edition, MIT Press, 2018
- 30 minutes talk of A. Barto (YouTube link)
- Paper on more recent developments: Arulkumaran et al., *A Brief Survey of Deep Reinforcement Learning*, arXiv:1708.05866, 2017

Contemporary Application Examples

Limited selection from a broad field:

- Swinging-up and balance a cart-pole / an inverted pendulum
- Flipping pancakes with a roboter arm
- Controlling electric drive systems
- Drifting with a RC-car
- Driving an autonomous car
- Playing Atari Breakout
- Play strategy board game Go at super-human performance
- Making money with stock trading
- ...

Table of Contents

Vocabulary Preview

In this section, some of the most important RL terms are shortly explained and summarized:

- Reward and return
- State
- Action
- Policy
- Value function
- Model
- Exploration and exploitation

Reward

- A reward is a scalar random variable R_k with realizations r_k.
- Often it is considered a real-number $r_k \in \mathbb{R}$ or an integer $r_k \in \mathbb{Z}$.
- The reward function may be naturally given or is a design degree of freedom (i.e., can be manipulated).
- It fully indicates how well an RL agent is doing at step k.
- Hence, the agent's task is to maximize its reward over time.

Theorem 1.1: Reward hypothesis

All goals can be described by the maximization of the expected cumulative reward:

$$\max \mathbb{E}\left[\sum_{i=0}^{\infty} R_{k+i+1}\right]. \tag{1.1}$$

Question: Which situations may conflict with the reward hypothesis?

Reward Examples

- Flipping a pancake:
  - Pos. reward: catching the $180°$ rotated pancake
  - Neg. reward: droping the pancake on the floor
- Stock trading:
  - Trading portfolio monetary value
- Playing Atari games:
  - Highscore value at the end of a game episode
- Driving an autonomous car:
  - Pos. reward: getting save from A to B without crashing
  - Neg. reward: hitting another car, pedestrian, bicycle,...
- Classical control task (e.g inverted pendulum, electric drive):
  - Pos. reward: following a given reference trajectory precisely
  - Neg. reward: violating system constraints and/or large control error

Reward Characteristics

Rewards can have many different flavors and are highly depending on the given problem:

- Actions may have short and/or long term consequences.
  - The reward for a certain action may be delayed.
  - Examples: Stock trading, strategic board games,...
- Rewards can be positive and negative real values.
  - Certain situations (e.g. car hits wall) might lead to a negative reward.
- Exogenous impacts might introduce stochastic reward components.
  - Example: A wind gust pushes the helicopter into a tree.

Remark on reward

The RL agent's learning process is heavily linked with the reward distribution over time. Designing expedient rewards functions is therefore crucially important for successfully applying RL. And often there is no predefined way on how to design the "best reward function".

The Reward Function Hassle

- "Be careful what you wish for - you might get it" (pro-verb)
- "...it grants what you ask for, not what you should have asked for or what you intend." (Norbert Wiener, American mathematician)

Fig. 1.8: Midas and Daughter (Good as Gold)
(source: www.flickr.com, by Robin Hutton CC BY-NC-ND 2.0)

Task-Dependent Return Definitions

Episodic tasks

- A problem which naturally breaks into subsequences (episodes).
- Examples: Most games, maze,...
- The return becomes a finite sum:

$$g_k = r_{k+1} + r_{k+2} + \cdots + r_N \,. \tag{1.2}$$

- Episodes end at their terminal step $k = N$.

Continuing tasks

- A problem which lacks a natural end (infinite horizon).
- Example: Process control tasks
- The return should be discounted to prevent infinite numbers:

$$g_k = r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \cdots = \sum_{i=0}^{\infty} \gamma^i r_{k+i+1} \,. \tag{1.3}$$

- Here, $\gamma \in \{\mathbb{R} | 0 \leq \gamma \leq 1\}$ is the discount rate.

Discounted Rewards

Numeric viewpoint

- If $\gamma = 1$ and $r_k >> 0$ for $k \to \infty$, g_k in (1.3) gets infinite.
- If $\gamma < 1$ and r_k is bounded for $k \to \infty$, g_k in (1.3) is bounded.

Strategic viewpoint

- If $\gamma \approx 1$: agent is farsighted.
- If $\gamma \approx 0$: agent is shortsighted (only interested in immediate reward).

Mathematical options

- The current return is the discounted future return:

$$\begin{aligned} g_k &= r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \cdots \\ &= r_{k+1} + \gamma \left(r_{k+2} + \gamma r_{k+3} + \cdots \right) \\ &= r_{k+1} + \gamma g_{k+1} \,. \end{aligned} \tag{1.4}$$

- If $r_k = r$ is a constant and $\gamma < 1$ one receives:

$$g_k = \sum_{i=0}^{\infty} \gamma^i r = r \sum_{i=0}^{\infty} \gamma^i = r \frac{1}{1-\gamma} \,. \tag{1.5}$$

State (1)

Environment state

- Random variable $\boldsymbol{S}_k^e$ with realizations $\boldsymbol{s}_k^e$
- Internal status representation of the environment, e.g
  - Physical states e.g. car velocity or motor current
  - Game states e.g. current chess board situation
  - Financial states e.g. stock market status
- Fully, limited or not at all visible by the agent
  - Sometimes even 'foggy' or uncertain
  - In general: $\boldsymbol{o}_k = \boldsymbol{f}(\boldsymbol{S}_k)$
- Continuous or discrete quantity

Bold symbols are non-scalar multidimensional quantities e.g. vectors and matrices.
Capital symbols denote random variables and small symbols their realizations.

State (2)

Agent state

- Random variable $\boldsymbol{S}_k^a$ with realizations $\boldsymbol{s}_k^a$
- Internal status representation of the agent
- In general: $\boldsymbol{s}_k^a \neq \boldsymbol{s}_k^e$ (e.g. due to measurement noise)
- Agent's condensed information relevant for next action
- Dependent on internal knowledge / policy representation of the agent
- Continuous or discrete quantity

History and Information State

Definition 1.1: History

The history is the past sequence of all observations, actions and rewards

$$\mathbb{H}_k = \{\boldsymbol{o}_0, r_0, \boldsymbol{a}_0, \ldots, \boldsymbol{a}_{k-1}, \boldsymbol{r}_k, r_k\} \tag{1.6}$$

up to the time step k.

If the current state $\boldsymbol{s}_k$ contains all useful information from the history it is called an information or Markov state:

Definition 1.2: Information state

A state $\boldsymbol{S}_k$ is called an information state if and only if

$$\mathbb{P}\left[\boldsymbol{S}_{k+1}|\boldsymbol{S}_k\right] = \mathbb{P}\left[\boldsymbol{S}_{k+1}|\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_k\right] . \tag{1.7}$$

- History is fully condensed in $\boldsymbol{s}_k$, i.e., $\boldsymbol{s}_{k+1}$ is only depending on $\boldsymbol{s}_k$.
- A given system can be fully described by $\boldsymbol{s}_k$.

Model Examples With Markov States

Linear time-invariant (LTI) state space model

$$\begin{aligned} \boldsymbol{s}_{k+1} &= \boldsymbol{A}\boldsymbol{s}_k + \boldsymbol{B}\boldsymbol{a}_k \,, \\ \boldsymbol{o}_k &= \boldsymbol{C}\boldsymbol{s}_k + \boldsymbol{D}\boldsymbol{a}_k \,. \end{aligned} \tag{1.8}$$

Nonlinear time-invariant state space model:

$$\begin{aligned} \boldsymbol{s}_{k+1} &= \boldsymbol{f}\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right) \,, \\ \boldsymbol{o}_k &= \boldsymbol{h}\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right) \,. \end{aligned} \tag{1.9}$$

Atari game:

$s_k =$

Degree of Observability

Full observability

- Agent directly observes environment state (e.g. $\boldsymbol{o}_k = \boldsymbol{I}\boldsymbol{s}_k$).
- If $\boldsymbol{s}_k$ is Markov: Markov decision process (MDP).

Partial observability

- Agent indirectly observes environment state (e.g. $\boldsymbol{o}_k = \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{s}_k$).
- If $\boldsymbol{s}_k$ is Markov: partial observable MDP (POMDP).
- Agent may reconstructs state information $\hat{\boldsymbol{s}}_k \approx \boldsymbol{s}_k$ (belief, estimate).

POMDP examples

- Technical systems with limited sensors (cutting costs)
- Poker game (unknown opponents' cards)
- Human health status

Action

- An action is the agent's degree of freedom in order to maximize its reward (i.e., the agent's output at each sample k).
- Major distinction:
  - Finite number of actions: $\boldsymbol{a}_k \in \{\boldsymbol{a}_{k,1}, \boldsymbol{a}_{k,2}, \ldots\} \in \mathbb{R}$ (finite-action-set, FAS)
  - Infinite number of actions: $\boldsymbol{a}_k \in \mathbb{R}$ (continuous-action-set, CAS)
  - Deterministic $\boldsymbol{a}_k$ or random $\boldsymbol{A}_k$ variable
  - Maybe state-dependent: $\boldsymbol{a}_k \in \mathcal{A}(\boldsymbol{s}_k)$
- Examples:
  - Take a card during Black Jack game (FAS)
  - Drive an autonomous car (CAS)
  - Buy stock options for your trading portfolio (FAS/CAS)

Remark on state and action spaces

Evaluating the state and action space framework (e.g., discrete vs. continuous) of a new RL problem should be always one of the first steps in order to choose appropriate solution algorithms.

Policy

- A policy π is the agent's internal strategy on picking actions.
- Deterministic policies: maps state and action directly:

$$\boldsymbol{a}_k = \boldsymbol{\pi}(\boldsymbol{s}_k)\,. \tag{1.10}$$

- Stochastic policies: maps a probability of the action given a state:

$$\boldsymbol{\pi}(\boldsymbol{A}_k|\boldsymbol{S}_k) = \mathbb{P}\left[\boldsymbol{A}_k|\boldsymbol{S}_k\right]\,. \tag{1.11}$$

- RL is all about changing π over time in order to maximize the expected return.

Deterministic Policy Example

Task: Find optimal (i.e. shortest) path.

- The reward $r_k = -1$ if not in lower right corner.
- Agent's behavior is determined by specifying movement.
- Environment state is the location of the agent.

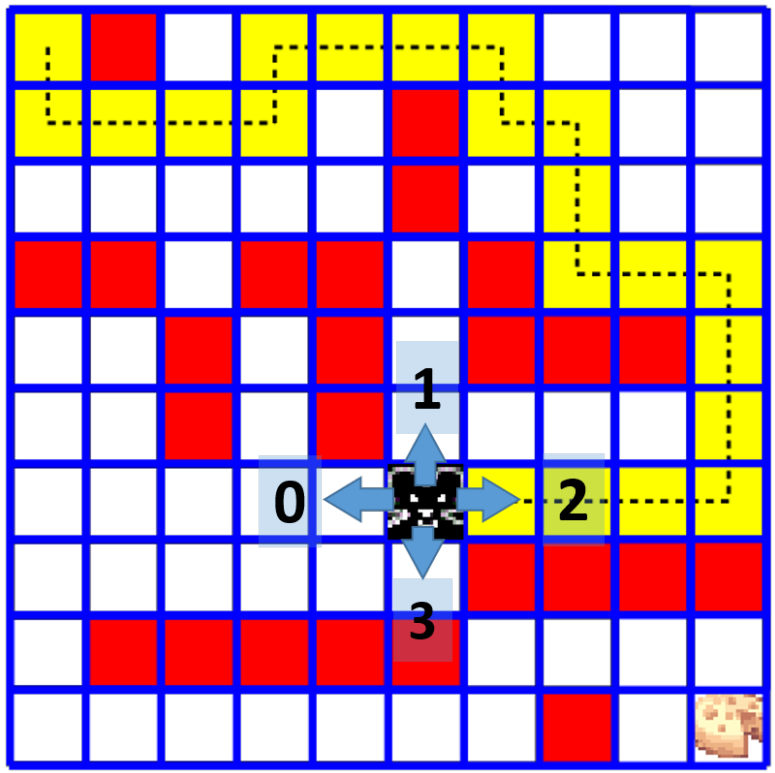


Fig. 1.9: Maze solving problem

Stochastic Policy Example

Task: Two-player game of extended rock-paper-scissors

- A deterministic policy can be easily exploited by the opponent.
- A uniform random policy would be instead unpredictable (assuming an ideal random number generator).

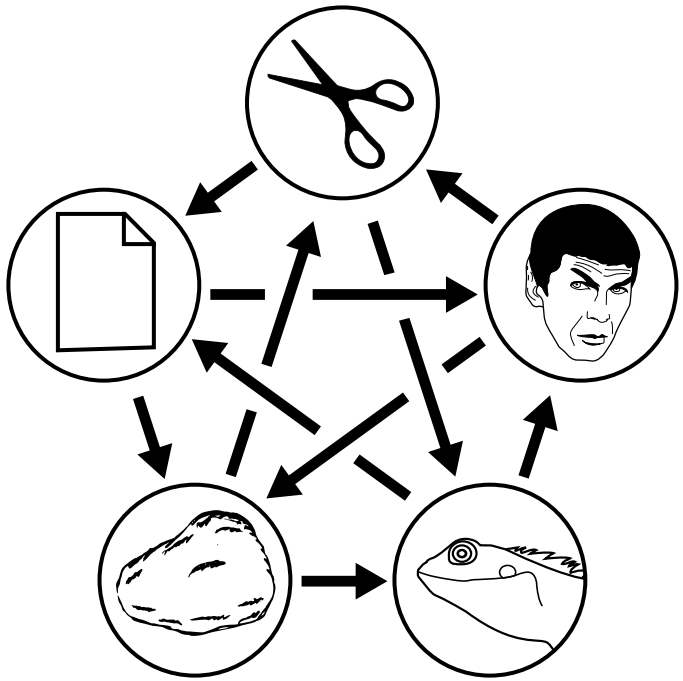


Fig. 1.10: Rock paper scissors lizard Spock game mechanics (source: www.wikipedia.org, by Diriector Doc CC BY-SA 4.0)

Value Functions

- The state-value function is the expected return being in state $\boldsymbol{s}_k$ following a policy $\boldsymbol{\pi}$: $v_\pi(\boldsymbol{s}_k)$.
- Assuming an MDP problem structure the state-value function is

$$v_\pi(\boldsymbol{s}_k) = \mathbb{E}_\pi\left[G_k | \boldsymbol{S}_k = \boldsymbol{s}_k\right] = \mathbb{E}_\pi\left[\sum_{i=0}^{\infty} \gamma^i R_{k+i+1} \middle| \boldsymbol{s}_k\right]. \tag{1.12}$$

- The action-value function is the expected return being in state $\boldsymbol{s}_k$ taken an action $\boldsymbol{a}_k$ and, thereafter, following a policy $\boldsymbol{\pi}$: $q_\pi(\boldsymbol{s}_k, \boldsymbol{a}_k)$.
- Assuming an MDP problem structure the action-value function is

$$q_\pi(\boldsymbol{s}_k, \boldsymbol{a}_k) = \mathbb{E}_\pi\left[G_k | \boldsymbol{S}_k = \boldsymbol{s}_k, \boldsymbol{A}_k = \boldsymbol{a}_k\right] = \mathbb{E}_\pi\left[\sum_{i=0}^{\infty} \gamma^i R_{k+i+1} \middle| \boldsymbol{s}_k, \boldsymbol{a}_k\right]. \tag{1.13}$$

- A key task in RL is to estimate $v_\pi(\boldsymbol{s}_k)$ and $q_\pi(\boldsymbol{s}_k, \boldsymbol{a}_k)$ based on sampled data.

Model

- A model predicts what will happen inside an environment.
- That could be a state model $\boldsymbol{\mathcal{P}}$:

$$\boldsymbol{\mathcal{P}} = \mathbb{P}\left[\boldsymbol{S}_{k+1} = \boldsymbol{s}_{k+1} | \boldsymbol{S}_k = \boldsymbol{s}_k, \boldsymbol{A}_k = \boldsymbol{a}_k\right] . \tag{1.14}$$

- Or a reward model $\mathcal{R}$:

$$\mathcal{R} = \mathbb{P}\left[R_{k+1} = r_{k+1} | \boldsymbol{S}_k = \boldsymbol{s}_k, \boldsymbol{A}_k = \boldsymbol{a}_k\right] . \tag{1.15}$$

- In general, those models could be stochastic (as denoted above) but in some problems relax to a deterministic form.
- Using data in order to fit a model is a learning problem of its own and often called system identification.

Exploration and Exploitation

- In RL the environment is initially unknown. How to act optimal?
- Exploration: Find out more about the environment.
- Exploitation: Maximize current reward using limited information.
- Trade-off problem: what's the best split between both strategies?

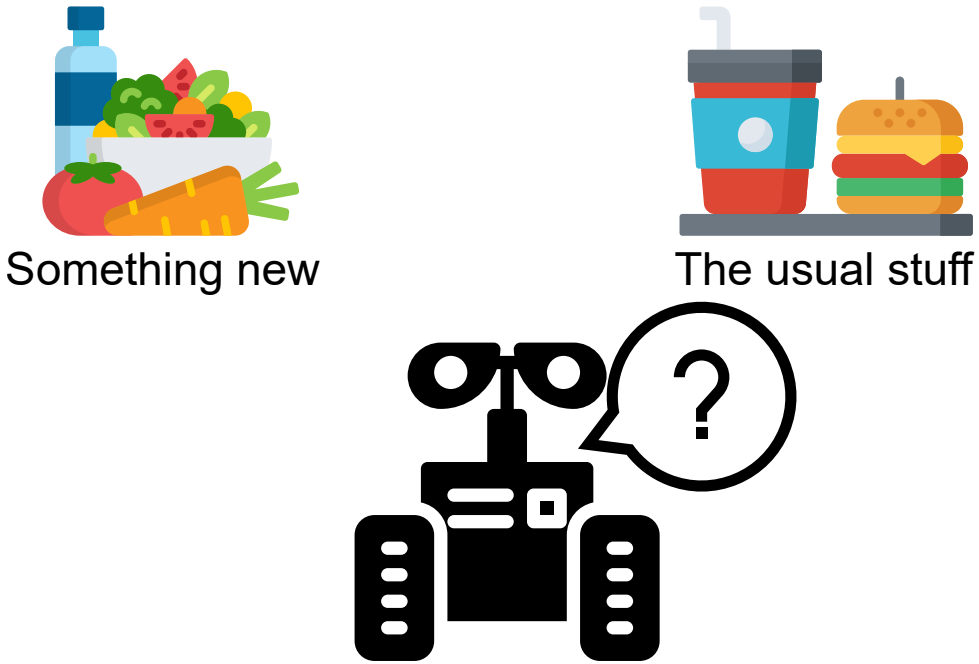


Fig. 1.11: The exploration exploitation dilemma

Table of Contents

Maze Example

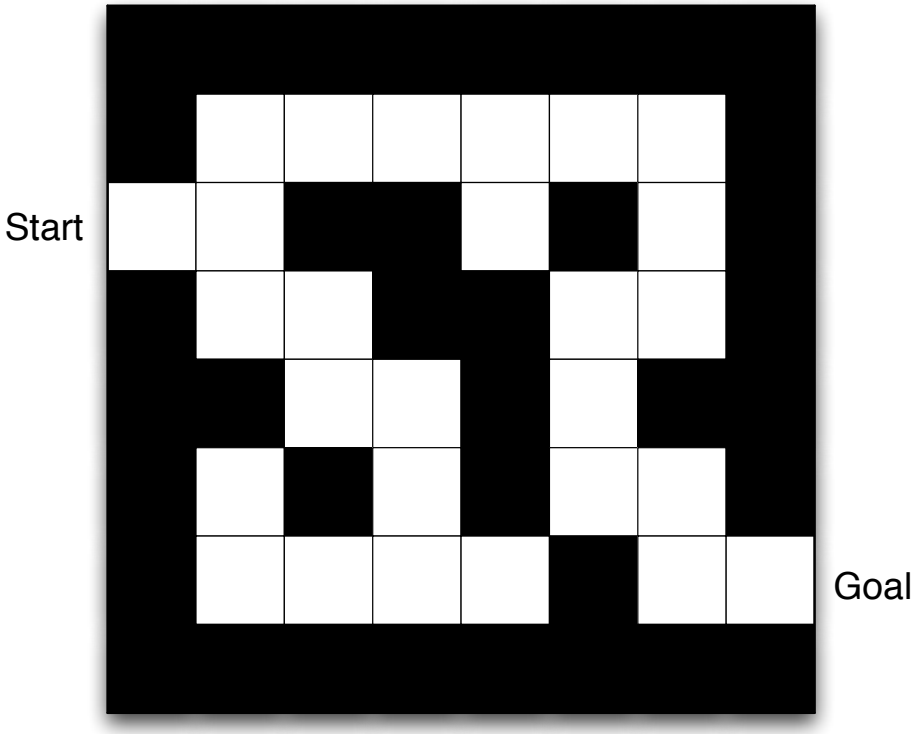


Fig. 1.12: Maze setup (source: D. Silver, Reinforcement learning, 2016. CC BY-NC 4.0)

Problem statement:

- Reward: $r_k = -1$
- At goal: episode termination
- Actions: $u_k \in \{N, E, S, W\}$
- State: Agent's location
- Deterministic problem (no stochastic influences)

Maze Example: RL-Solution by Policy

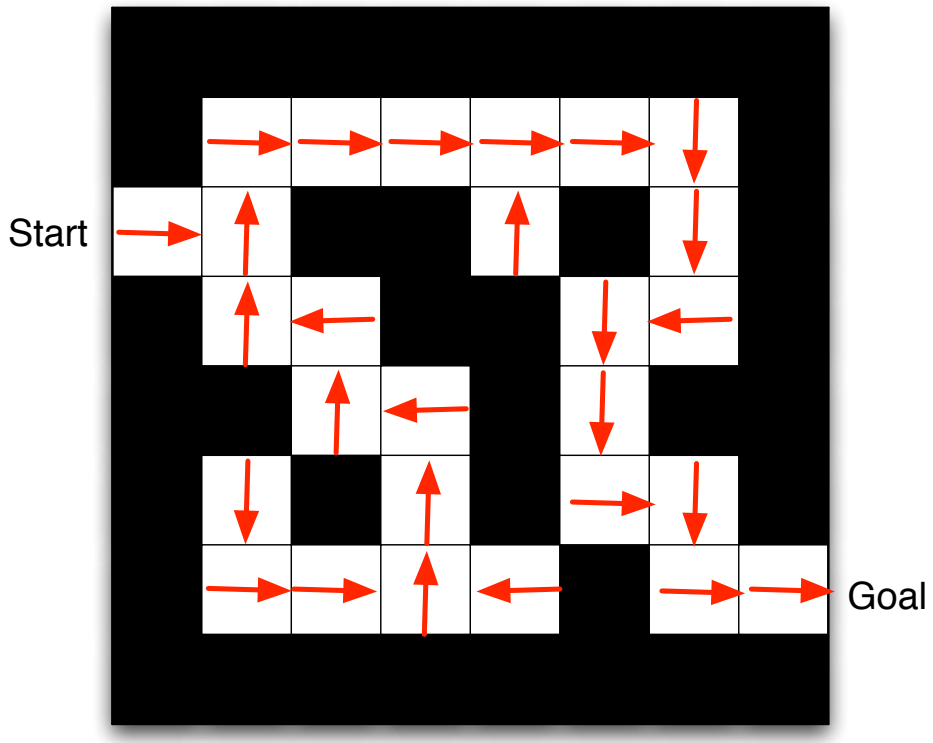


Fig. 1.13: Arrows represent policy $\pi(x_k)$ (source: D. Silver, Reinforcement learning, 2016. CC BY-NC 4.0)

Key characteristics:

- For any state there is a direct action command.
- The policy is explicitly available.

Maze Example: RL-Solution by Value Function

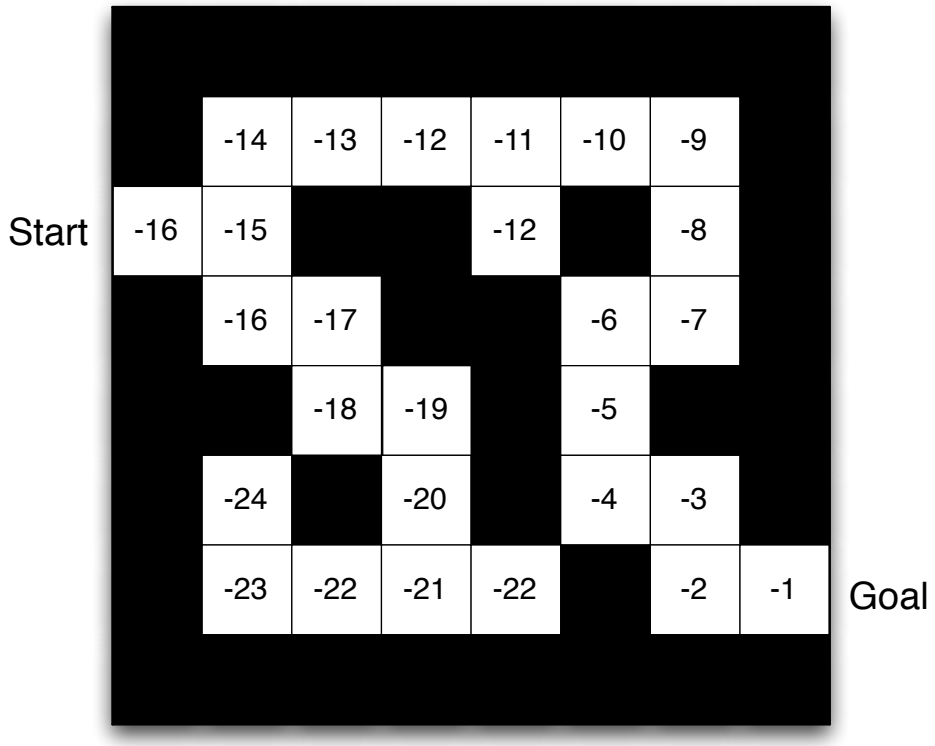


Fig. 1.14: Numbers represent value $v_\pi(x_k)$ (source: D. Silver, Reinforcement learning, 2016. CC BY-NC 4.0)

Key characteristics:

- The agent evaluates neighboring maze positions by their value.
- The policy is only implicit.

Maze Example: RL-Solution by Model Evaluation

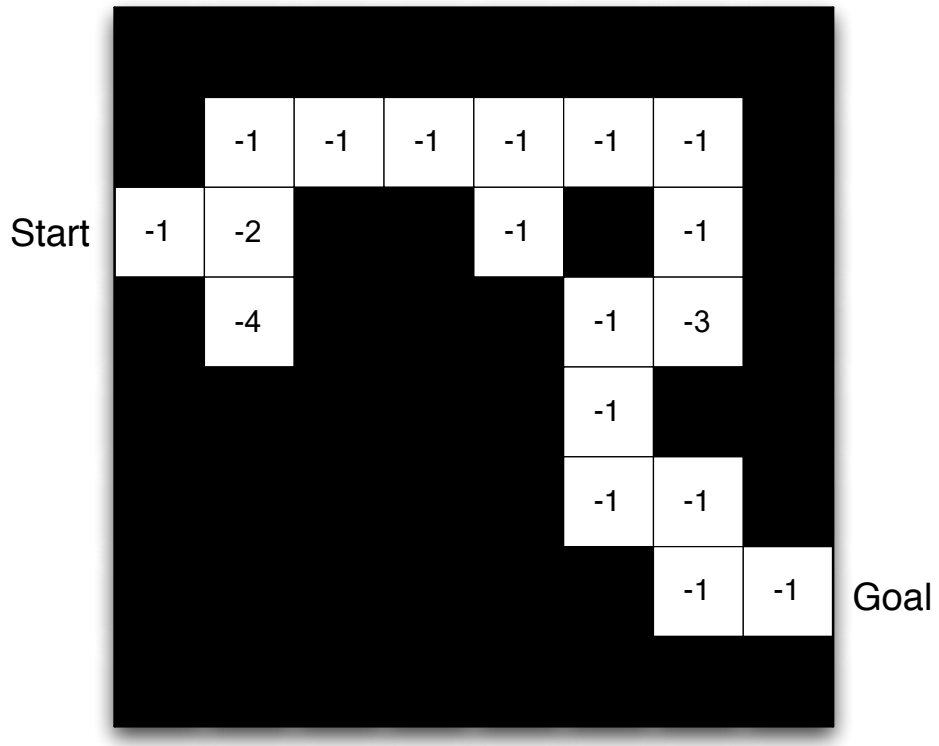


Fig. 1.15: Grid layout represents state model $\mathcal{P}$ and numbers depict the estimate by the reward model $\mathcal{R}$. (source: D. Silver, Reinforcement learning, 2016. CC BY-NC 4.0)

Key characteristics:

- Agent uses internal model of the environment.
- The model is only an estimate (inaccurate, incomplete).
- The agent interacts with the model before taking the next action (e.g., by numerical optimizers).

RL Agent Taxonomy

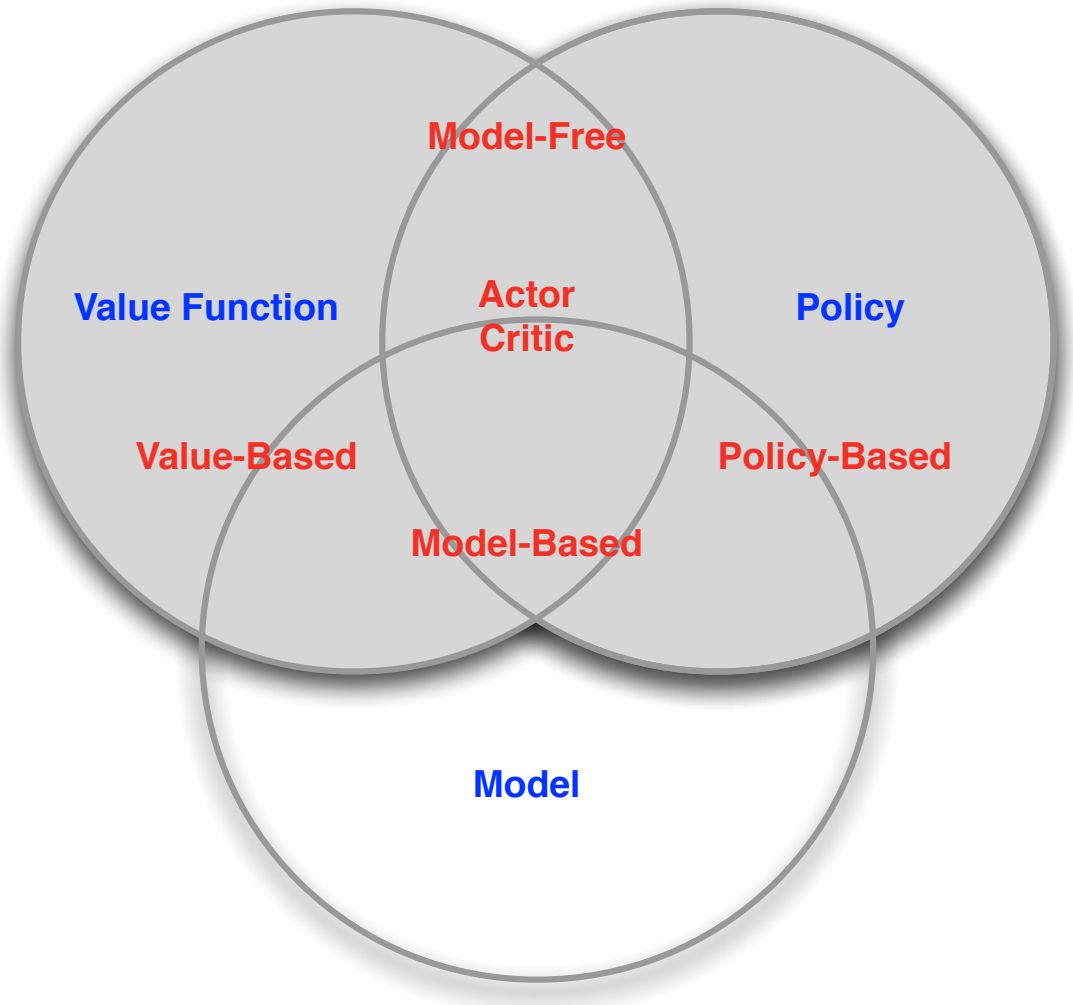


Fig. 1.16: Main categories of reinforcement learning algorithms (source: D. Silver, Reinforcement learning, 2016. CC BY-NC 4.0)

Table of Contents

RL vs. Planning

Two fundamental solutions to sequential decision making:

- Reinforcement learning:
  - The environment is initially unknown.
  - The agents interacts with the environment.
  - The policy is improved based on environment feedback (reward).
- Planning:
  - An a priori environment model exists.
  - The agents interacts with its own model.
  - The policy is improved based on the model feedback ('virtual reward').

Remark on learning and planning

Above the two extreme cases are confronted:

- RL = only learning without using available pre-knowledge.
- Planning = iterating on a model without improving it based on data.

Can this lead to efficient and optimal solutions?

Summary: What You've Learned Today

- Understanding the role of RL in machine learning and optimal sequential decision making.
- Become acquainted with the basic RL interaction loop (agent, environment, interpreter).
- Finding your way around the basic RL vocabulary.
- Internalize the significance of proper reward formulations (design parameter).
- Differentiate solution ideas on how to retrieve an optimal agent behavior (policy).