Theory and Methods for Reinforcement Learning

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 1: Introduction to Reinforcement Learning

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-618 (Spring 2023)

lions@epfl

License Information for Theory and Methods for Reinforcement Learning (EE-618)

- This work is released under a Creative Commons License with the following terms:
- **Attribution**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees must give the original authors credit.
- **Non-Commercial**
  - The licensor permits others to copy, distribute, display, and perform the work. In return, licensees may not use the work for commercial purposes – unless they get the licensor's permission.
- **Share Alike**
  - The licensor permits others to distribute derivative works only under a license identical to the one that governs the licensor's work.
- Full Text of the License

Acknowledgements

- Joint effort between ODI@ETHZ (Prof. Niao He) and lions@epfl (myself)
  - Anas Barakat, Batu Yardim, Jiawei Huang, Liang Zhang, Zebang Shen, Junchi Yang, Siqi Zhang, and Yifan Hu
  - Luca Viano, Igor Krawczuk, Ali Kavis, Ahmet Alacaoglu, Grigorios Chrysos, Pedro Abranches, Leello Dadi, Ali Ramezani, Stratis Skoulakis, Kimon Antonakopoulos, Fanghui Liu, Fabian Latorre, Thomas Pethick and Angeliki Kamoutsi

A paradigm shift in machine learning (ML) applications

- Self driving, industry automation, robotic manipulation, trading and finance,...



https://neptune.ai/blog/reinforcement-learning-applications



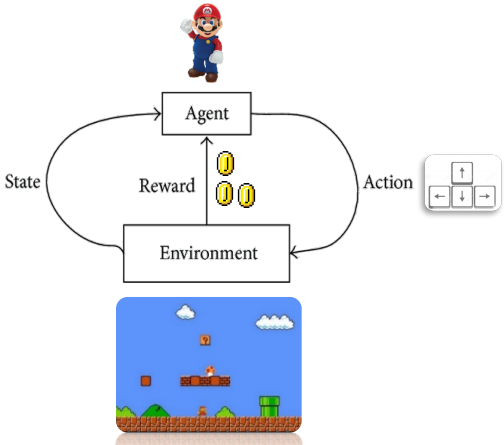
https://www.forbes.com/sites/bernardmarr/2022/12/28/what-does-chatgpt-really-mean-for-businesses/?sh=27bc344f7d1e

What is reinforcement learning (RL)?

- Classical definitions:
  - Sutton and Barto, 1998: Reinforcement learning is learning what to do – how to map situations to actions – so as to maximize a numerical reward signal.
  - WIKIPEDIA, 2023: Reinforcement learning is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward.

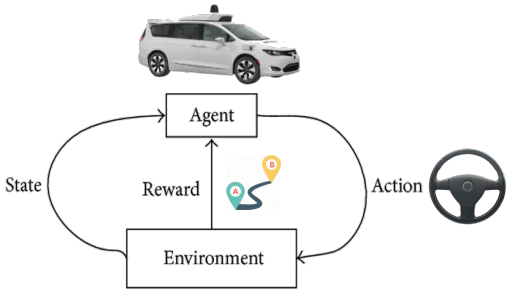
A common theme in RL

- An agent learns to act by interacting with an uncertain environment



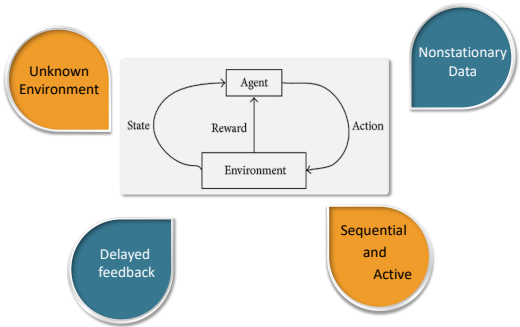
A common theme in RL

- An agent learns to act by interacting with an uncertain environment



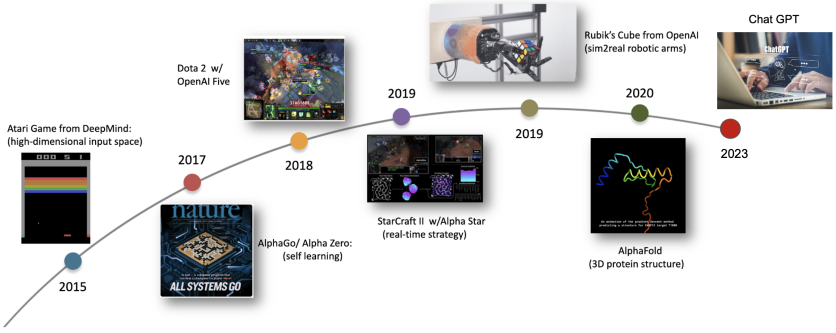
A common theme in RL

- An agent learns to act by interacting with an uncertain environment



Remarkable progress on ML applications

- Which one is not due to RL?



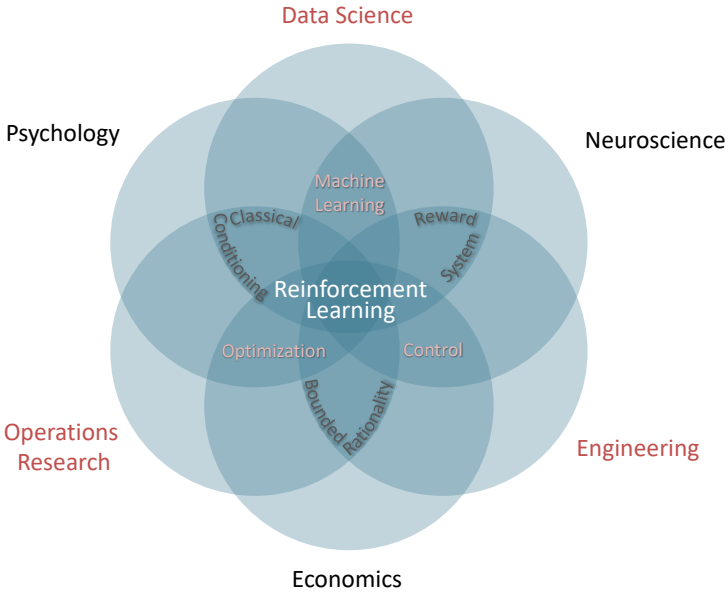
Perceptions of RL

- Roughly speaking...
  - For EE, it is control theory *mutatis mutandis*

    control $\rightarrow$ action

    controller $\rightarrow$ agent or policy

    system or plant $\rightarrow$ environment
  - For CS, it is an ML paradigm along with supervised and unsupervised learning.
  - For others?


Data Science
Psychology
Neuroscience
Machine Learning
Classical Conditioning
Reward System
Reinforcement Learning
Optimization
Control
Bounded Rationality
Operations Research
Engineering
Economics

Challenges to RL



The New York Times

One Giant Step for a Chess-Playing Machine

The stunning success of AlphaZero, a deep-learning algorithm, heralds a new age of insight — one that, for humans, may not last long.

What is frustrating about machine learning, however, is that the algorithms can't articulate what they're thinking. We don't know why they work, so we don't know if they can be trusted. AlphaZero gives every appearance of having discovered some important principles about chess, but it can't share that understanding with us. Not yet, at least. As human beings, we want more than answers. We want insight. This is going to be a source of tension in our interactions with computers from now on.

- Theoretical foundations are more important than ever

Challenges to RL



The New York Times

One Giant Step for a Chess-Playing Machine

The stunning success of AlphaZero, a deep-learning algorithm, heralds a new age of insight — one that, for humans, may not last long.

What is frustrating about machine learning, however, is that the algorithms can't articulate what they're thinking. We don't know why they work, so we don't know if they can be trusted. AlphaZero gives every appearance of having discovered some important principles about chess, but it can't share that understanding with us. Not yet, at least. As human beings, we want more than answers. We want insight. This is going to be a source of tension in our interactions with computers from now on.



- Theoretical foundations are more important than ever
- Common challenges with ML: Robustness, interpretability, scalability, reproducibility,...

Perceptions of our RL course (EE-618)

- Why are you taking this course?
  - Learn the basics of RL
  - Gain hands-on experience with RL implementations
  - Apply RL to my research
  - Might be useful for my future job
  - Just need the credits
  - Other reasons?
  - Let us know `https://go.epfl.ch/rl_form2023`

What are our learning objectives?

- By the end of the course, participants will be able to
  - Define the key features of RL that distinguishes it from standard ML
  - Identify the strengths and limitations of various RL algorithms
  - Understand the theoretical properties of RL algorithms
  - Recognize the common, connecting boundary of optimization and RL
  - Formulate and solve sequential decision-making problems by applying relevant RL tools
  - Generalize or discover "new" applications, algorithms, or theories of RL towards conducting research

What EE-618 is really about: Theory and methods



Simplex (Dantzig)

Stochastic Approximation (Robbins & Monro)

Interior Point Method (Karmarker, Nemirovski, etc.)

Accelerated GD (Nesterov)

Boom of SGD and first-order methods

Kingma and Ba, Adam (ICLR, 2015)

Linear Programming

Stochastic Programming

Convex Programming

Stochastic Optimization

Nonconvex Optimization

1940s 1950s 1980s 2000s 2020s



Dynamic Programming

Value-based Methods

Policy Optimization

Deep Reinforcement Learning

DeepMind, DQN (Nature, 2015)

Boom of policy gradient methods

Policy Iteration (Howard)

Linear Programming

Q-Learning (Watkins)

TD Learning & Actor-critic (Sutton, Barto, etc.)

What EE-618 is *not* really about: real-applications

- The following important topics are beyond the scope of this course
  - Coding tricks and super practical implementations of RL
  - Applications of RL in real-world
  - RL engineering
  - Building autonomous robots
  - Building autonomous driving systems
  - Building ChatGPT-like systems

Should I take this course?

- This course is right for you, if you...
  - want to understand the RL theory
  - have interest in performing RL research
  - have strong math background
  - want to gain hands-on experience with RL

- This course may not be right for you, if you...
  - **only** want to gain hands-on experience with RL
  - do not have interest in performing RL research
  - have only basic math background

What is left for today?

- A preview of the course...
  - Dynamic Programming
    - Value Iteration
    - Policy Iteration
    - Monte Carlo Methods
    - TD, SARSA, Q-learning
  - Linear Programming
    - REPS, Proximal Point
    - Applications to offline RL
  - Policy-based RL
    - Policy Gradient Method
    - Natural Policy Gradient Method
    - TRPO and PPO
  - Imitation Learning and Inverse RL
    - Behavior Cloning, GAIL
    - Interactive IL (DAgger, SMILe)
    - Max Margin and Max Entropy IRL
  - Markov Games
    - Fictitious Play
    - Policy Gradient
    - Nash Q-learning
  - Robust and Deep RL
    - Deep Q Network and Extensions
    - Deep Actor-Critic (A3C, DDPG, TD3)
    - Robust DDPG/TD3

Theory	Bellman Equations Policy Gradient Theorems Performance Difference Lemma	Stochastic Approximation Optimization and Game Theory Convergence Analysis

Markov Decision Processes (MDPs)

- MDPs are important for modeling sequential decision making problems.
- We can use MDPs to formally describe an environment for RL.
- We will use the following roadmap for MDPs [24]:
  1. Markov chains
  2. Markov reward processes
  3. Markov decision processes

Markov chains

Definition (Markov Chain)

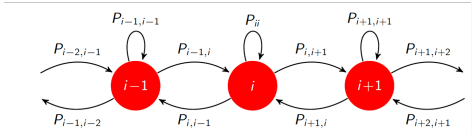
A (time-homogeneous) Markov chain is a stochastic process $\{X_0, X_1, \ldots\}$, taking values on a countable number of states, satisfying the so-called Markov property, i.e.,

$$P(X_{t+1} = j | X_t = i, X_{t-1}, \ldots, X_0) = P(X_{t+1} = j | X_t = i) = P_{ij}.$$

Markov Process

Markov process is a tuple $\langle \mathcal{S}, P \rangle$, where

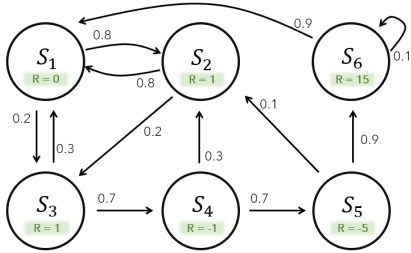
- $\mathcal{S}$ is the set of all possible states
- $P_{ss'} = P(s'|s)$: $\mathcal{S} \to \mathcal{S}$ is the transition model
- μ is the initial state distribution: $s_0 \sim \mu \in \Delta(\mathcal{S})$



Definition (Stationary distribution)

If a Markov chain is irreducible and aperiodic with finite states (i.e., ergodic), then there exists a unique stationary distribution $d^\star$ and $\{X_t\}$ converges to it, i.e., $\lim_{t\to\infty} P^t_{ij} = d_j, \forall i, j$. We can represent this via $d^\star = d^\star P$ where $[P]_{ij} = P_{ij}$ and $d^\star$ is a row vector. Hence, $d^\star$ is the left principal eigenvector of P.

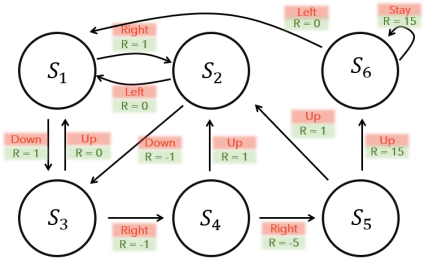
Markov chains with associated reward values



Markov reward processes

- $\mathcal{S}$ is the set of all possible states
- $P_{ss'} = P(s'|s)$: $\mathcal{S} \to \mathcal{S}$ is the transition model
- μ is the initial state distribution: $s_0 \sim \mu \in \Delta(\mathcal{S})$
- $r(s)$: $\mathcal{S} \to \mathbb{R}$ is the expected reward function
- γ is the discount factor: $\gamma \in [0, 1]$

Markov reward processes with actions



Markov Decision Process

A Markov decision process (MDP) is a Markov reward process with actions.

- $\mathcal{S}$ is the set of all possible states
- $\mathcal{A}$ is the set of all possible actions
- $P^a_{ss'} = P(s'|s,a)$: $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the tr. model
- $r(s,a)$: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function
- μ is the initial state distribution: $s_0 \sim \mu \in \Delta(\mathcal{S})$
- γ is the discount factor: $\gamma \in [0,1]$

Using MDPs to establish a performance criteria

Definition (Return)

Acting on MDPs results in immediate rewards $r(s_t, a_t)$. Accumulating these rewards, we obtain the return.

Finite horizon or fixed time horizon T

- Cumulative reward:

$$\mathbb{E}\left[\sum_{t=0}^{T-1} r(s_t, a_t)\right]$$

- Average reward:

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} r(s_t, a_t)\right]$$

Infinite horizon

- Discounted reward:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

- Average reward:

$$\liminf_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} r(s_t, a_t)\right]$$

Finite Horizon vs Infinite Horizon

- Does the finite horizon matter?
  - What is the best action in the MDP below, if we are at $t = T - 1$ or $t = T - 2$?

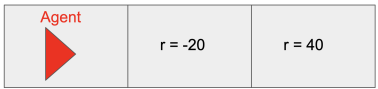


Figure: An example of finite horizon MDP

  - Think to a basketball match for a real world example.

- For sake of simplicity, we focus on the discounted infinite horizon setting in this course.

Why discounted return?

- The discount factor γ routinely appears in MDPs with infinite horizon:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

- We can use $\gamma \in (0, 1)$ to trade off the importance of past and present rewards.

Observations:
- If $\gamma = 1$, the total reward may be infinite, e.g., when the Markov process is cyclic.
- With $\gamma \in (0, 1)$, assuming bounded rewards, i.e., $r < \infty$, the return will always be finite.

From MDPs to policies

What is our goal?

Find a behavior or rule to make decisions that maximize the expected return.

- A **policy** selects an action based on the history $h_t := (s_{0:t}, a_{0:t-1}) := (s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t)$
  - It is a mapping $\pi : \mathcal{S} \to \mathcal{A}$ or $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, where Δ is the appropriate probability simplex.

Deterministic Policy

- Stationary policy $\pi : \mathcal{S} \to \mathcal{A}$, $a_t = \pi(s_t)$
- Markov policy $\pi_t : \mathcal{S} \to \mathcal{A}$, $a_t = \pi_t(s_t)$
- History-dependent policy $\pi_t : \mathcal{H} \to \mathcal{A}$
  - $\mathcal{H}$ is the set of all possible trajectories
  - $a_t = \pi_t(h_t)$

Randomized Policy:

- Stationary policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, $a_t \sim \pi(\cdot|s_t)$
- Markov policy $\pi_t : \mathcal{S} \to \Delta(\mathcal{A})$, $a_t \sim \pi_t(\cdot|s_t)$
- History-dependent policy $\pi_t : \mathcal{H} \to \Delta(\mathcal{A})$
  - $\mathcal{H}$ is the set of all possible trajectories
  - $a_t \sim \pi_t(h_t)$

Remarks:

- The **infinite horizon** objective can be maximized by a *stationary policy*.
- The **finite horizon** objective needs instead of a *Markov policy*.

Value functions: Towards provably "good" decisions

Definition (State-value function)

$$V^{\pi}(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s,\ \pi\right]$$

Remarks:

- State-value function represents the expected return starting at the state under policy π
- For convenience, we may drop the π in RHS when it is clear from the context

Solving MDPs to determine an optimal policy

The ultimate goal in RL

To find an **optimal policy** $\pi^\star \in \Pi$ such that

$$V^{\pi^\star}(s) = V^\star(s) := \max_{\pi \in \Pi} V^\pi(s), \forall s \in \mathcal{S}.$$

Remark:
- The optimal policy may not be unique, while $V^\star$ is unique.

Solving MDPs to determine an optimal policy

The ultimate goal in RL

To find an **optimal policy** $\pi^\star \in \Pi$ such that

$$V^{\pi^\star}(s) = V^\star(s) := \max_{\pi \in \Pi} V^\pi(s), \forall s \in \mathcal{S}.$$

Remark: ○ The optimal policy may not be unique, while $V^\star$ is unique.

Key Questions

- **Q1**: Does the optimal policy $\pi^\star$ exist?
- **Q2**: How to evaluate my current policy π, i.e., how to compute $V^\pi(s)$?
- **Q3**: If $\pi^\star$ exists, how to improve my current policy π, i.e., how to find $\pi^\star$?

Solving MDPs to determine an optimal policy

The ultimate goal in RL

To find an **optimal policy** $\pi^\star \in \Pi$ such that

$$V^{\pi^\star}(s) = V^\star(s) := \max_{\pi \in \Pi} V^\pi(s), \forall s \in \mathcal{S}.$$

Remark: ○ The optimal policy may not be unique, while $V^\star$ is unique.

Key Questions

- **Q1**: Does the optimal policy $\pi^\star$ exist?
- **Q2**: How to evaluate my current policy π, i.e., how to compute $V^\pi(s)$? ***–policy evaluation***
- **Q3**: If $\pi^\star$ exists, how to improve my current policy π, i.e., how to find $\pi^\star$? ***–policy improvement***

Bellman optimality conditions

- The optimal value function $V^\star$ is the **fixed point** of the following equation:

$$V^\star(s) = \max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^\star(s') \right].$$

Remarks:

- This requirement is also known as **Bellman optimality conditions**.
- Here, we assume that there exists a **deterministic** optimal policy.
- Fixed-point perspective motivates Value Iteration and Policy Iteration methodologies.

Value Iteration (VI)

Algorithm: Value Iteration (VI) for solving MDPs

Start with an arbitrary guess V_0 (e.g., $V_0(s) = 0$ for any s)
for each iteration t **do**
  Update V_t for any s as follows

$$V_{t+1}(s) = \max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V_t(s') \right]. \tag{1}$$

end for

Remarks:

- Recall that the optimal value function $V^\star$ is the **fixed point** of the following equation:

$$V^\star(s) = \max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^\star(s') \right].$$

- **Value iteration** can be therefore viewed as a fixed-point iteration.
- The course will go deeper into an operator view of this update.

Policy Iteration (PI)

Algorithm: Policy Iteration (PI) for solving MDPs

Start with an arbitrary policy guess π_0.
for each iteration t **do**
(Step 1: Policy evaluation) Compute V^{π_t} via an option below:

(Option 1) Initialize V and iteratively apply policy value iteration,
$V^{\pi_t}(s) \leftarrow \mathbb{E}_{a\sim\pi_t(\cdot|s)}\left[r(s,a)+\gamma\sum_{s'\in\mathcal{S}}P(s'|s,a)V(s')\right]$, until convergence;

(Option 2) Use the closed-form solution: $V^{\pi_t}=(I-\gamma P^{\pi_t})^{-1}R^{\pi_t}$.

(Step 2: Policy improvement) Update the current policy π_t by the greedy policy

$$\pi_{t+1}(s)=\arg\max_{a\in\mathcal{A}}\left[r(s,a)+\gamma\sum_{s'\in\mathcal{S}}P(s'|s,a)V^{\pi_t}(s')\right]. \tag{2}$$

end for

Remarks:

- Recall that we assume that there exists a **deterministic** optimal policy.
- Greedy policy achieves the optimal deterministic policy.

Comparison

Algorithm	Value Update	Policy Update
Value Iteration (VI)	$V_{t+1} = \max_{a \in \mathcal{A}} \Big[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'\|s,a) V(s') \Big]$	None
Policy Iteration (PI)	$V_{t+1} = \mathbb{E}\Big[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'\|s,a) V(s') \| \pi_t \Big]$	Greedy Policy

Algorithm	Per iteration cost	Number of iterations	Output
Value Iteration (VI)	$\mathcal{O}\big(\|\mathcal{S}\|^2 \|\mathcal{A}\|\big)$	$T = \mathcal{O}\Big(\frac{\log(\epsilon^{-1}(1-\gamma))}{\log \gamma}\Big)$	V_T such that $\|V_T - V^\star\| \leq \epsilon$
Policy Iteration (PI)	$\mathcal{O}\big(\|\mathcal{S}\|^3 + \|\mathcal{S}\|^2 \|\mathcal{A}\|\big)$	$T = \mathcal{O}\big(\frac{\|\mathcal{S}\|(\|\mathcal{A}\|-1)}{1-\gamma}\big)$	$V^\star$ and $\pi^\star$

Observations:

- VI and PI are broadly dynamic programming approaches.
- PI converges in finite number of iterations [27] whereas VI does not [25].
- These solution mythologies assume that the transition dynamics is known!
- These solution mythologies are broadly known as model-based RL.

From dynamic programming to optimization via the Bellman optimality conditions

Lemma

The optimal value function $V^\star(s)$ is an element of the following set

$$\left\{ V \in R^{|\mathcal{S}|} : V(s) \geq r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V(s') \right\}.$$

Derivation:

- Recall the Bellman optimality conditions

$$V^\star(s) = \max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^\star(s') \right].$$

- As a result, for any s, a, we have that

$$V^\star(s) \geq \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^\star(s') \right].$$

- The equality is achieved for maximizing actions only (i.e., $V^\star$ is the element-wise minimum).

Solving MDPs via linear programming (LP)

Primal LP for Finding $V^\star$

Let $\mu(s), s \in \mathcal{S}$ be the initial distribution (or any positive weights). We have

$$\begin{aligned} \min_{V} \quad & \sum_{s\in\mathcal{S}} \mu(s) V(s) \\ \text{s.t.} \quad & V(s) \geq \; r(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) V(s') \quad \forall\; s \in \mathcal{S},\; a \in \mathcal{A}, \end{aligned} \tag{P}$$

where $V^\star \leftarrow \arg\min$.

Solving MDPs via linear programming (LP)

Primal LP for Finding $V^\star$

Let $\mu(s), s \in \mathcal{S}$ be the initial distribution (or any positive weights). We have

$$\begin{aligned} \min_{V} \quad & \sum_{s\in\mathcal{S}} \mu(s) V(s) \\ \text{s.t.} \quad & V(s) \geq r(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) V(s') \quad \forall\ s \in \mathcal{S},\ a \in \mathcal{A}, \end{aligned} \tag{P}$$

where $V^\star \leftarrow \arg\min$.

Remarks:

- The optimal value function $V^\star$ is the unique solution to the above LP.
- Number of decision variables: $|\mathcal{S}|$, number of constraints: $|\mathcal{S}| \times |\mathcal{A}|$.

Towards the dual LP: Occupancy measure

Definition

Given any policy π, we define the occupancy measure of the policy π as:

$$\lambda(s,a) = (1-\gamma)\mathbf{E}\left[\sum_{t=0}^{T}\gamma^t \mathbf{1}(s_t = s, a_t = a)|\pi\right]. \tag{3}$$

Remark: ○ Intuitively it is the discounted visitation frequency of a certain state action pair under policy π.

Solving MDPs via the dual LP

Dual LP formulation

Given the primal LP (P), its dual can be written as follows:

$$\begin{aligned} \max_{\lambda \in \Delta} \quad & \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} r(s,a) \lambda(s,a) \\ \text{s.t.} \quad & \sum_{a \in \mathcal{A}} \lambda(s,a) = \mu(s) + \gamma \sum_{s' \in \mathcal{S}, a' \in \mathcal{A}} P(s|s',a') \lambda(s',a') \quad \forall\ s \in \mathcal{S} \end{aligned} \tag{D}$$

where $\lambda^\star \leftarrow \arg\max$.

Solving MDPs via the dual LP

Dual LP formulation

Given the primal LP (P), its dual can be written as follows:

$$\begin{aligned} \max_{\lambda \in \Delta} \quad & \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} r(s,a) \lambda(s,a) \\ \text{s.t.} \quad & \sum_{a \in \mathcal{A}} \lambda(s,a) = \mu(s) + \gamma \sum_{s' \in \mathcal{S}, a' \in \mathcal{A}} P(s|s',a') \lambda(s',a') \quad \forall\ s \in \mathcal{S} \end{aligned} \tag{D}$$

where $\lambda^\star \leftarrow \arg\max$.

Remarks:

- Number of decision variables: $|\mathcal{S}| \times |\mathcal{A}|$, number of constraints: $|\mathcal{S}|$.
- The solution $\lambda^\star$ to (D) corresponds to the state-action occupancy of the optimal policy.
- The LP formulation is due to Manne [16]

Model-free RL: Basics

- When the **transition dynamics** is unknown, we can use the state-action value function.

Definition (State-action value function/Quality-function)

$$Q^{\pi}(s,a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t) \mid s_0 = s,\ a_0 = a,\ \pi\right]$$

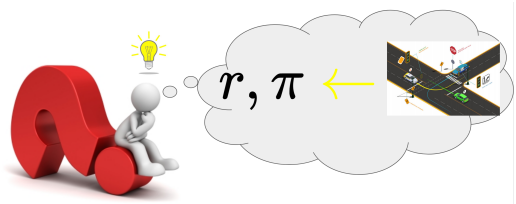
Remarks:

- Given a state action value function, the greedy policy can be determined as follows:

$$\pi(s) = \mathrm{argmax}_{a\in\mathcal{A}}\ Q(s,a).$$

- The optimal state-value function $Q^{\pi^\star}$ can be found via Q-Learning (see Lecture 2).
- Model-free is arguably more popular than model-based RL in ML.
- A new consideration in this setting is the sample complexity (vs. computational complexity)!

Learning from demonstrations



- The reward function is unknown or is difficult to design in real world problems.
- It is easier/more natural to use "demonstrations" by experts.

Imitation learning (IL) vs inverse reinforcement learning (IRL)

- Setting:
  - Given an expert's demonstrations $\{(s_i, \pi_{\mathsf{E}}(s_i))\}$ (offline trajectories or online queries)
  - Reward signal is unobserved
  - Transition model may be known or unknown

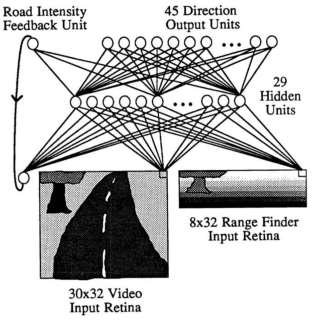
- Goals and approaches:
  - Recover the expert's policy π_{E} directly: imitation learning (IL)
  - Recover the expert's latent reward function $r^{\mathsf{E}}(s, a)$: inverse reinforcement learning (IRL)

A historic application

- Inverse Reinforcement Learning has been formally introduced by [20].



(a)



(b)

Figure: One of the first imitation learning systems using neural networks.

- ALVINN: Autonomous Land Vehicle In a Neural Network, 1989 [23].

https://www.youtube.com/watch?v=2KMAAmkz9go&t=205s.

One of the latests applications

- Large language models: ChatGPT



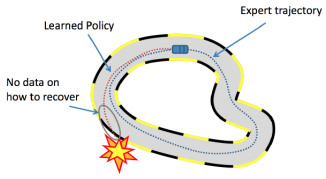
https://www.forbes.com/sites/bernardmarr/2022/12/28/what-does-chatgpt-really-mean-for-businesses/?sh=27bc344f7d1e

- The last training step is based on Reinforcement Learning from Human Feedback (RLHF) (see [21]).
- A recent work [30] shows a close connection between IRL and RLHF.

Another variation along the theme: Behavioral cloning and interactive IL

- Behavioral cloning (BC) is a supervised learning approach to learning from demonstrations
  - Given an expert's demonstrations $\{(s_i, \pi_*(s_i))\}$ (offline trajectories or online queries)
  - Fix a loss: $\mathcal{L} : \mathcal{A} \to \mathbb{R}$
  - Output $\pi^\star \in \operatorname{argmin}_\pi \sum_i^N \mathcal{L}(a_i, \pi(s_i))$ with a_i, s_i in the dataset provided by the expert.

- BC can result in cascading errors
  - Any error at a state can accumulate over an episode.
  - It can have catastrophic consequences...



- Solution: *Interactive IL* allows to query the expert policy from a particular state

Figure: `https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c`

Taxonomy of approaches for learning from demonstrations

Method	Reward learning	Access to environment	Interactive demonstrations	Pre-collected demonstrations
Behavioural Cloning	NO	NO	NO	YES
Imitation Learning	NO	YES	NO	YES
Interactive IL	NO	YES	YES	MAYBE
Inverse RL	YES	YES	NO	YES

Remarks:

- BC avoids interaction with the environment, but can suffer from cascading errors.
- IL helps with the cascading errors but requires (expensive) expert queries
- IRL explains the expert's behavior but has poor sample complexity and scalability.

Motivation for Robust (I)RL

- Mismatches between the settings of the learning and the deployment
- Mismatches between the settings of the expert and the learner
- Example: transfer the driving skills among different road conditions, traffic dynamics and car brands

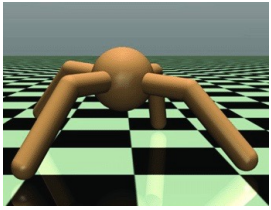


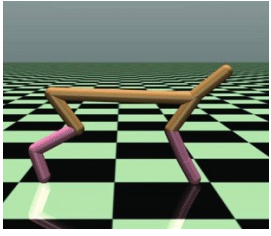


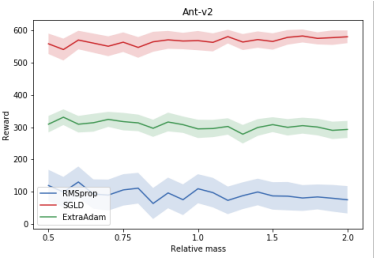
Figure: A Toyota Prius[1] and Bugatti la voiture noir[2] have arguably different dynamics!

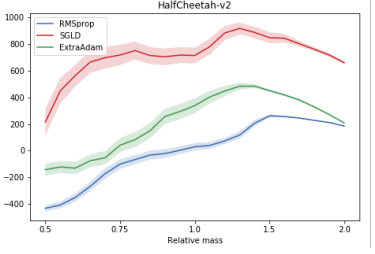
[1] https://www.autobild.de/artikel/toyota-prius-3-hybridauto-als-gebrauchtwagen-16425701.html,
[2] https://www.autobild.de/artikel/toyota-prius-3-hybridauto-als-gebrauchtwagen-16425701.html

Application: Noisy action robust reinforcement learning[1]









[1] K. Parameswaran, Y-T. Huang, Y-P. Hsieh, P. Rolland, C. Shi, V. Cevher, "Robust Reinforcement Learning via Adversarial Training with Langevin Dynamics" NeurIPS 2020.

Markov games

- A Markov game involves multiple agents, optimizing their individual rewards



- **State** + **Ryu's action** + **Ken's action** $\rightarrow$ **New State**
  - **State** $\leftarrow$ *Fighters' Positions* + *Fighters' Life*
  - **Actions for Ryu** $\leftarrow$ *jump, kick, punch,..* (resp. Ken)
  - **Reward for Ryu** $\leftarrow$ 1 *if Ken is KO,* 0 *otherwise* (resp. Ken)
- The course will cover algorithms for Markov games and robustness in the context of Nash equilibrium
- Also see EE-735: Online Learning in Games

Sample complexity to solve Markov games

	sample complexity
[7]	$\tilde{\mathcal{O}}((\lvert A\rvert \vee \lvert B\rvert)^{10.75}\lvert S\rvert^{1.25}\epsilon^{-12.5})$
[28]	$\tilde{\mathcal{O}}((\lvert A\rvert^3 + \lvert B\rvert^3)\lvert S\rvert^{10.5}\epsilon^{-8})$
[28]	$\tilde{\mathcal{O}}((\lvert A\rvert^3 + \lvert B\rvert^3)\lvert S\rvert^{4.5}\epsilon^{-4}C^{-10})$
Reflected NAC [3]	$\tilde{\mathcal{O}}(\lvert S\rvert^3(\lvert A\rvert \vee \lvert B\rvert)^3\epsilon^{-4})$
Reflected NAC [3]	$\tilde{\mathcal{O}}(\lvert S\rvert^3(\lvert A\rvert \vee \lvert B\rvert)\epsilon^{-2})$

- Sample complexity is a major challenge in Markov games.
- It provides a lowerbound for the computational complexity, which is roughly $(\lvert A\rvert \vee \lvert B\rvert)\times$ higher
- Developing sample efficient algorithms is an active research direction!

The critical need for function approximation

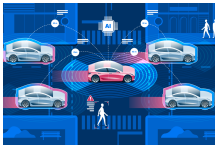


Figure: Self-driving car with continuous state/action space[3]

- Lookup tables can only handle small and finite state and action spaces:
  - The optimal value function V is an array of size $|S|$.
  - The optimal Q-function Q is a matrix of size $|S| \times |A|$.
  - A policy $\pi(a|s)$ is a matrix of size $|S| \times |A|$.
- How can we handle very large or continuous state and action spaces ?

[3] http://selfdrivingcars.space/?p=68

Value or Q-function approximation

- To handle continuous spaces, V and Q must be approximated using fewer parameters.
- We can adapt LP and dynamics methods to use these approximations.

Example (Linear function approximation)

Given an *embedding* ϕ that maps $s \in S$ to $\mathbb{R}^d$ with $d \ll |S|$, we can approximate the value function with a linear function $V_w(s) = w^\top \phi(s)$ or $Q_w(s) = w^\top \phi(s, a)$.

Example (Nonlinear function approximation)

We can use neural networks to approximate V or Q [15].

- Nonlinear parameterizations can demand sophisticated optimization technologies.

Remark:
- We can similarly parameterize the policy π_θ with parameters θ.

Working with the policy functions

- We can also write the basic RL objective in terms of policies:

$$\pi^\star = \arg\max_{\pi:\pi\in\Delta} J(\pi) := \sum_s \mu(s) V^\pi(s)$$

  - $\mu(s)$ is the initial state distribution.

Lemma (Performance difference [12])

It holds that for any policies π_1, π_2 and denoting with λ^{π_1} the occupancy measure of the policy π_1:

$$J(\pi_1) - J(\pi_2) = \mathbb{E}_{s\sim\lambda^{\pi_1}}\left[\langle Q^{\pi_2}(s,\cdot), \pi_1(\cdot|s) - \pi_2(\cdot|s)\rangle\right] = \langle Q^{\pi_2}(s,\cdot), \pi_1(\cdot|s) - \pi_2(\cdot|s)\rangle_{\lambda^{\pi_1}},$$

where $\langle\cdot,\cdot\rangle_\lambda$ is the weighted inner product.

- This perspective is a building block towards developing gradient-based policy optimization algorithms:

RL	Optimization
π	x
$J(\pi)$	$f(x)$
$Q^\pi(s,a)$	$\nabla f(x)$
$J(\pi^\star) - J(\pi) = \frac{1}{1-\gamma}\langle Q^\pi, \pi^\star - \pi\rangle_{\lambda^\star}$ [13]	$f(x^\star) - f(x) \geq \langle f(x), x^\star - x\rangle$

RL vs. optimization

RL	Optimization
Natural policy gradient [2], Policy mirror descent [13]	Mirror Descent [18, 5, 17]
Conservative policy iteration (CPI) [12]	Frank-Wolfe [8, 11]
Politex [1]	Dual averaging [19]
Actor-critic [29]	Two-time scale GDA [14, 9]
Natural actor-critic [10]	Two-time scale GDA with entropic setup in the primal [14, 9, 5, 17]
REPS/Q-REPS [22, 4]	Proximal point in the dual LP [26, 18, 17]

- Challenges to optimization in finding the optimal policy $\pi^\star$ in RL:
  - Exact computation of value functions V^π, Q^π not practical.
    - The algorithms need to work with state transition samples $(s_t, a_t, s_{t+1}, a_{t+1})$.
  - Samples are heavily coupled: i.e., *Markovian* data [6].
    - iid assumption of **stochastic optimization** does not hold.
  - For policy optimization, objective $J(\pi)$ is a non-convex function of π even in the tabular case [2].
- Optimization-based RL has advantages: Mirror descent obtains $\mathcal{O}\left(\frac{|S||A|}{(1-\gamma)^4\epsilon^2}\right)$-sample complexity [13].

What's Beyond?

- Episodic RL
- Strategic Exploration in RL
- Batch and Offline RL
- Safety in RL
- Multi-task RL
- Preference-based RL
- Causal RL
- Partially Observable Markov Decision Process (POMDP)
- ... ...

Where to go from here?

- Recent workshops:
  - Deep Reinforcement Learning Workshop, NeurIPS 2021, December 13
    https://sites.google.com/view/deep-rl-workshop-neurips2021
  - Offline Reinforcement Learning Workshop, NeurIPS 2021, December 14
    https://offline-rl-neurips.github.io/2021/
  - Ecological Theory of Reinforcement Learning Workshop, NeurIPS 2021, December 14
    https://sites.google.com/view/ecorl2021/
  - Political Economy of Reinforcement Learning Systems, NeurIPS 2021, December 14
    https://perls-workshop.github.io/
  - Decision Awareness in Reinforcement Learning, ICML 2022, July 22
    https://icml.cc/virtual/2022/workshop/13463
  - Responsible Decision Making in Dynamic Environments, ICML 2022, July 23
    https://icml.cc/virtual/2022/workshop/13453
  - 3rd Offline Reinforcement Learning Workshop: Offline RL as a "Launchpad", NeurIPS 2022, December 2
    https://nips.cc/virtual/2022/workshop/49971
  - Deep Reinforcement Learning Workshop, NeurIPS 2022, December 9
    https://nips.cc/virtual/2022/workshop/49989
  - European Workshop on Reinforcement Learning, 2022, 19-21 September
    https://ewrl.wordpress.com/ewrl15-2022/

Where to go from here? (Continued)

- Seminars:
  - RL Theory Virtual Seminar: https://sites.google.com/view/rltheoryseminars/
  - Control Meets Learning: https://sites.google.com/view/control-meets-learning/
- Simons Institute RL Program:
  - Theory of RL: https://simons.berkeley.edu/programs/rl20
  - Learning and Games: https://simons.berkeley.edu/programs/games2022

Logistics

- **Credits:** 3
- **Lectures:** Thursday 9:00-12:00 (INM011)
- **Project hours:** Thursday 09:00-12:00 (INM011)
- **Prerequisites:** Previous coursework in optimization, probability theory, and linear algebra is required (i.e., EE-556 Math of Data). Familiarity with deep learning and programming in python is useful.
- **Grading:** Project (cf., syllabus)
- **Moodle:** My courses > Genie electrique et electronique (EL) > Master > EE-618

  syllabus & course outline & project examples
- **TA's**: Luca Viano (head TA), Angeliki Kamoutsi, Yongtao Wu, Zhenyu Zhu, Andrej Janchevski, Leello Dadi, Pedro Abranches, Fabian Latorre.

References I

[1] Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvari, and Gellért Weisz.
Politex: Regret bounds for policy iteration using expert prediction.
In *International Conference on Machine Learning*, pages 3692–3702. PMLR, 2019.
56

[2] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan.
Optimality and approximation with policy gradient methods in markov decision processes.
In *Conference on Learning Theory*, pages 64–66. PMLR, 2020.
56

[3] Ahmet Alacaoglu, Luca Viano, Niao He, and Volkan Cevher.
A natural actor-critic framework for zero-sum markov games.
In *International Conference on Machine Learning*, pages 307–366. PMLR, 2022.
52

[4] Joan Bas-Serrano, Sebastian Curi, Andreas Krause, and Gergely Neu.
Logistic q-learning.
arXiv preprint arXiv:2010.11151, 2020.
56

[5] Amir Beck and Marc Teboulle.
Mirror descent and nonlinear projected subgradient methods for convex optimization.
Operations Research Letters, 31(3):167–175, 2003.
56

References II

[6] Jalaj Bhandari, Daniel Russo, and Raghav Singal.
A finite time analysis of temporal difference learning with linear function approximation.
In *Conference On Learning Theory*, pages 1691–1692. PMLR, 2018.
56

[7] Constantinos Daskalakis, Dylan J. Foster, and Noah Golowich.
Independent policy gradient methods for competitive reinforcement learning, 2021.
52

[8] Marguerite Frank, Philip Wolfe, et al.
An algorithm for quadratic programming.
Naval research logistics quarterly, 3(1-2):95–110, 1956.
56

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter.
Gans trained by a two time-scale update rule converge to a local nash equilibrium.
In *Advances in neural information processing systems*, pages 6626–6637, 2017.
56

[10] Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang.
A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic.
arXiv preprint arXiv:2007.05170, 2020.
56

References III

[11] Martin Jaggi.
Revisiting frank-wolfe: Projection-free sparse convex optimization.
In *International Conference on Machine Learning*, pages 427–435. PMLR, 2013.
56

[12] Sham Kakade and John Langford.
Approximately optimal approximate reinforcement learning.
In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.
55, 56

[13] Guanghui Lan.
Policy mirror descent for reinforcement learning: Linear convergence, new sampling complexity, and generalized problem classes.
arXiv preprint arXiv:2102.00135, 2021.
55, 56

[14] Tianyi Lin, Chi Jin, and Michael Jordan.
On gradient descent ascent for nonconvex-concave minimax problems.
In *International Conference on Machine Learning*, pages 6083–6093. PMLR, 2020.
56

[15] Fanghui Liu, Luca Viano, and Volkan Cevher.
Understanding deep neural function approximation in reinforcement learning via epsilon-greedy exploration.
arXiv preprint arXiv:2209.07376, 2022.
54

References IV

[16] A. Manne.
Linear programming and sequential decisions.
Management Science, 6(3):259–267, 1960.
40, 41

[17] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro.
Robust stochastic approximation approach to stochastic programming.
SIAM Journal on optimization, 19(4):1574–1609, 2009.
56

[18] Arkadi Semenovic Nemirovski and David Borisovich Yudin.
Problem complexity and method efficiency in optimization.
1983.
56

[19] Yurii Nesterov.
Primal-dual subgradient methods for convex problems.
Mathematical programming, 120(1):221–259, 2009.
56

[20] A. Y. Ng and S. J. Russell.
Algorithms for inverse reinforcement learning.
In *International Conference on Machine Learning (ICML)*, 2000.
45

References V

[21] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al.

Training language models to follow instructions with human feedback.

arXiv preprint arXiv:2203.02155, 2022.

46

[22] Jan Peters, Katharina Mulling, and Yasemin Altun.

Relative entropy policy search.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

56

[23] Dean A. Pomerleau.

Alvinn: An autonomous land vehicle in a neural network.

In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1989.

45

[24] Martin L Puterman.

Markov Decision Processes: Discrete Stochastic Dynamic Programming.

John Wiley & Sons, Inc., 1994.

20

[25] Satinder Singh Richard and Richard C. Yee.

An upper bound on the loss from approximate optimal-value functions.

In *Machine Learning*, pages 227–233, 1994.

35

References VI

[26] R Tyrrell Rockafellar.

Monotone operators and the proximal point algorithm.

SIAM journal on control and optimization, 14(5):877–898, 1976.

56

[27] Bruno Scherrer.

Improved and generalized upper bounds on the complexity of policy iteration.

Mathematics of Operations Research, 41(3):758–774, 2016.

35

[28] Chen-Yu Wei, Chung-Wei Lee, Mengxiao Zhang, and Haipeng Luo.

Last-iterate convergence of decentralized optimistic gradient descent/ascent in infinite-horizon competitive markov games, 2021.

52

[29] Yue Wu, Weitong Zhang, Pan Xu, and Quanquan Gu.

A finite time analysis of two time-scale actor critic methods.

arXiv preprint arXiv:2005.01350, 2020.

56

[30] Banghua Zhu, Jiantao Jiao, and Michael I Jordan.

Principled reinforcement learning with human feedback from pairwise or k-wise comparisons.

arXiv preprint arXiv:2301.11270, 2023.

46