Lecture 1: Introduction

Friday, October 22, 2021

Reinforcement Learning, Winter Term 2021/22

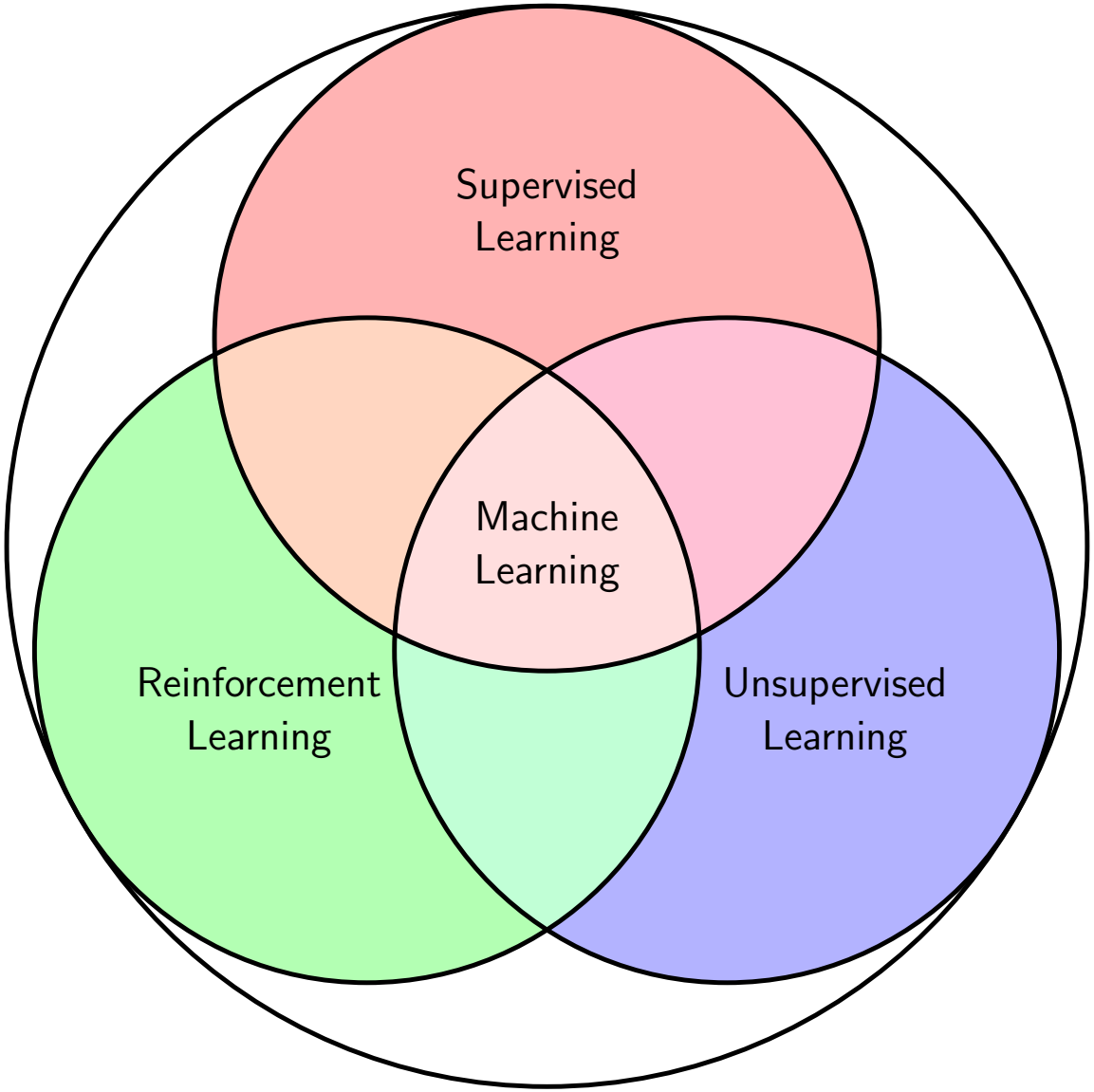
Joschka Boedecker, Gabriel Kalweit, Jasper Hoffmann

Neurorobotics Lab
University of Freiburg

Lecture Overview

Lecture Overview

Reinforcement Learning in Machine Learning



Discussion

What are characteristics of Reinforcement Learning? What are the differences to Supervised Learning?

Characteristics of Reinforcement Learning

- Framework to solve sequential decision making problems
- No supervisor, trial-and-error
- Delayed feedback
- Behaviour affects subsequent data
- Exploration and exploitation

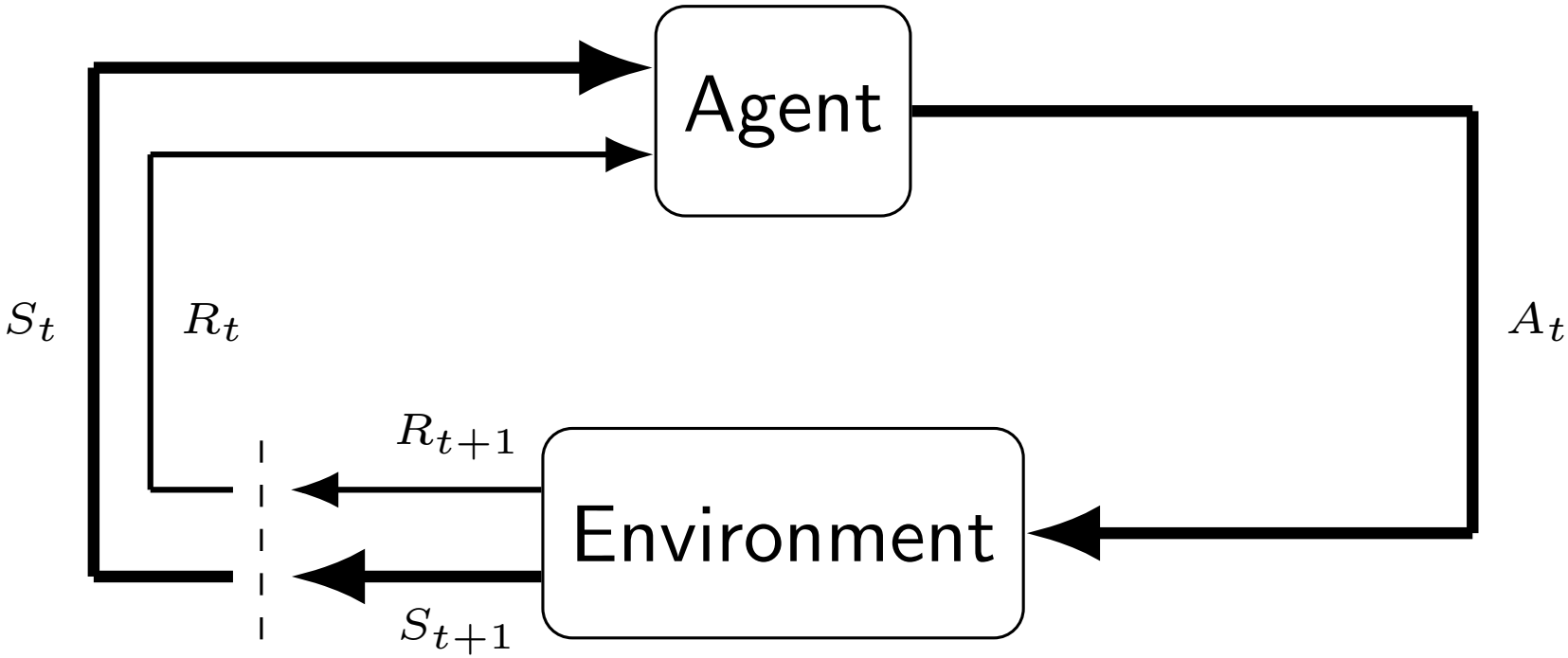
Examples

Videos in lecture recording.

Discussion

Where would you apply Reinforcement Learning?

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: $R_1, R_2, R_3, \ldots$

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.
- The agent aims at maximizing the expected discounted **cumulative reward** $G_t = R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-(t+1)} R_T$.
  T can be inifinite.

Reward Hypothesis

Everything 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

Exploration and Exploitation

- Unique 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?

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:

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

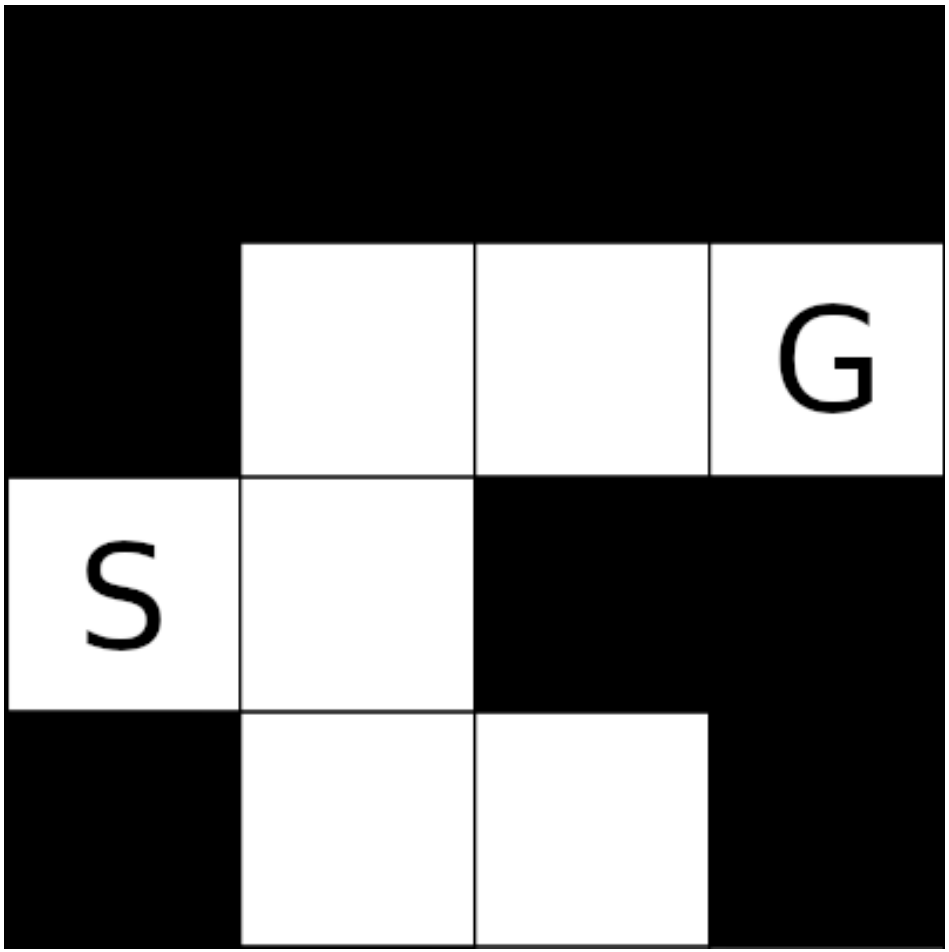
The future is independent of the past given the present.

Components of RL Systems

- Policy: defines the behaviour of the agent
  - is a mapping from a state to an action
  - can be stochastic: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$
  - or deterministic: $\pi(s) = a$
- Value-function: defines the expected value of a state or an action
  - $v_\pi(s) = \mathbb{E}[G_t|S_t = s]$ and $q_\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a]$
  - Can be used to evaluate states or to extract a good policy
- Model: defines the transitions between states in an environment
  - p yields the next state and reward
  - $p(s', r|s, a) = \mathsf{Pr}\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\}$

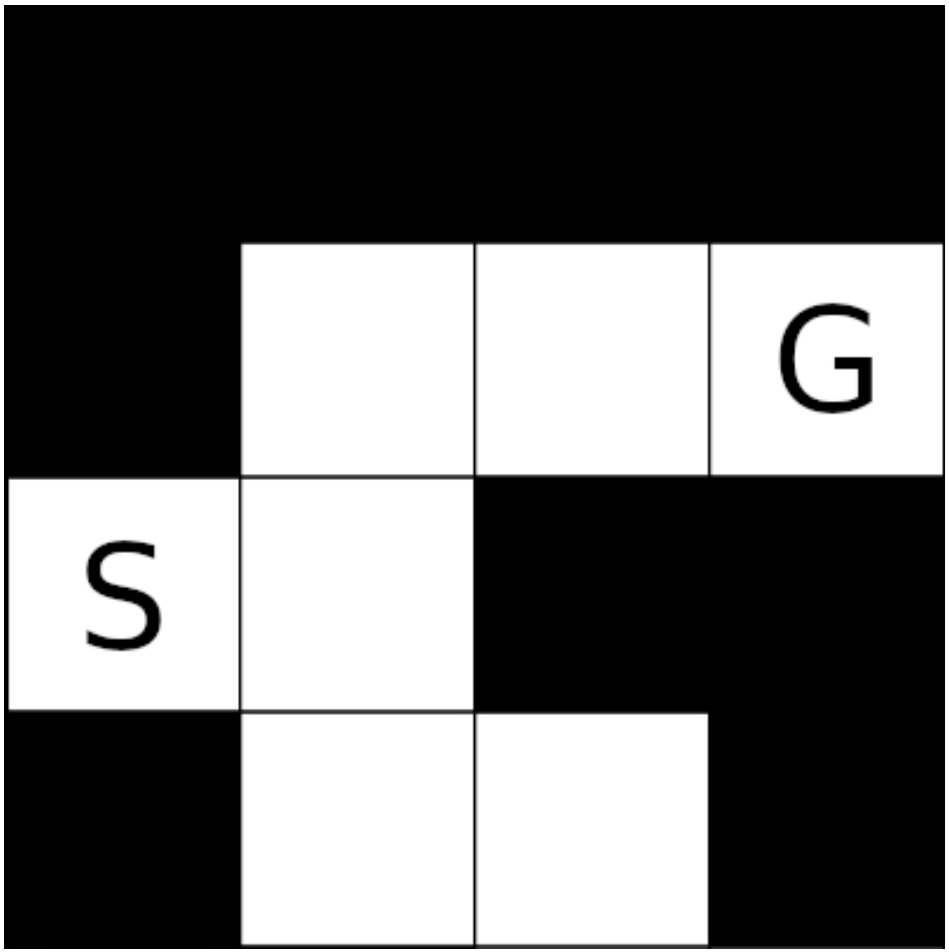
Maze Example: Policy

- Rewards: -1 per time step
- Actions: up, down, left, right
- States: location of the agent



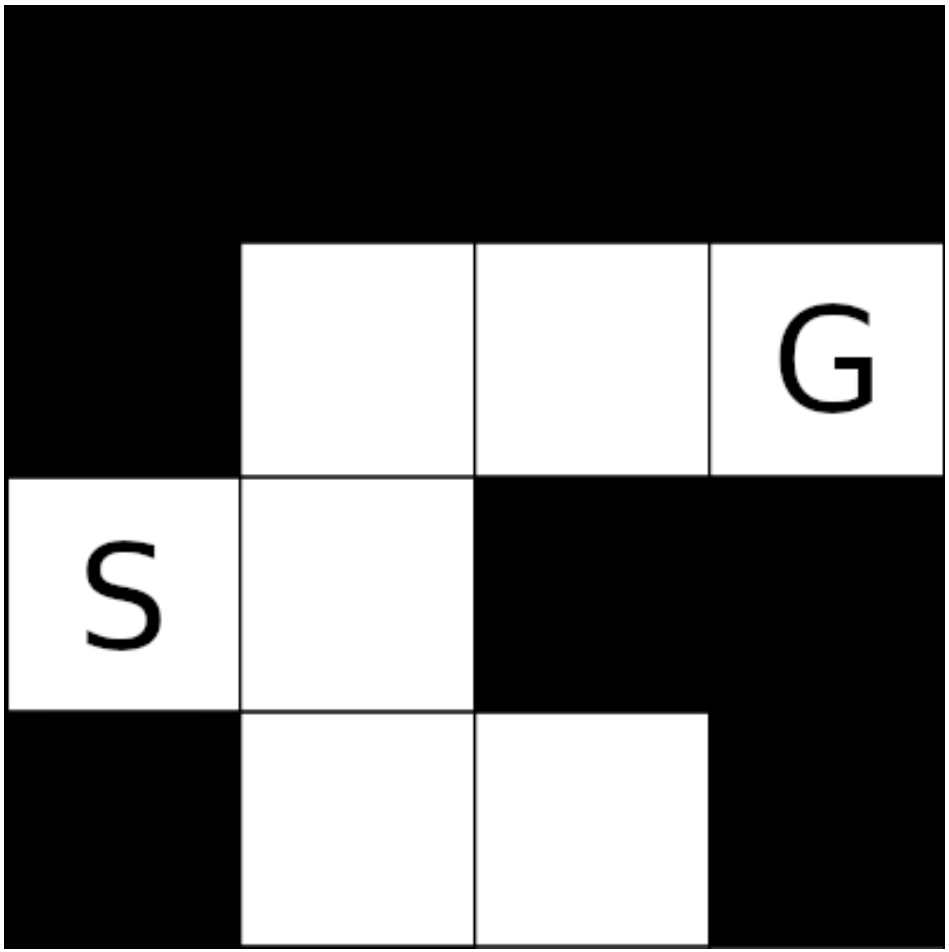
Maze Example: Value-function

- Rewards: -1 per time step
- Actions: up, down, left, right
- States: location of the agent

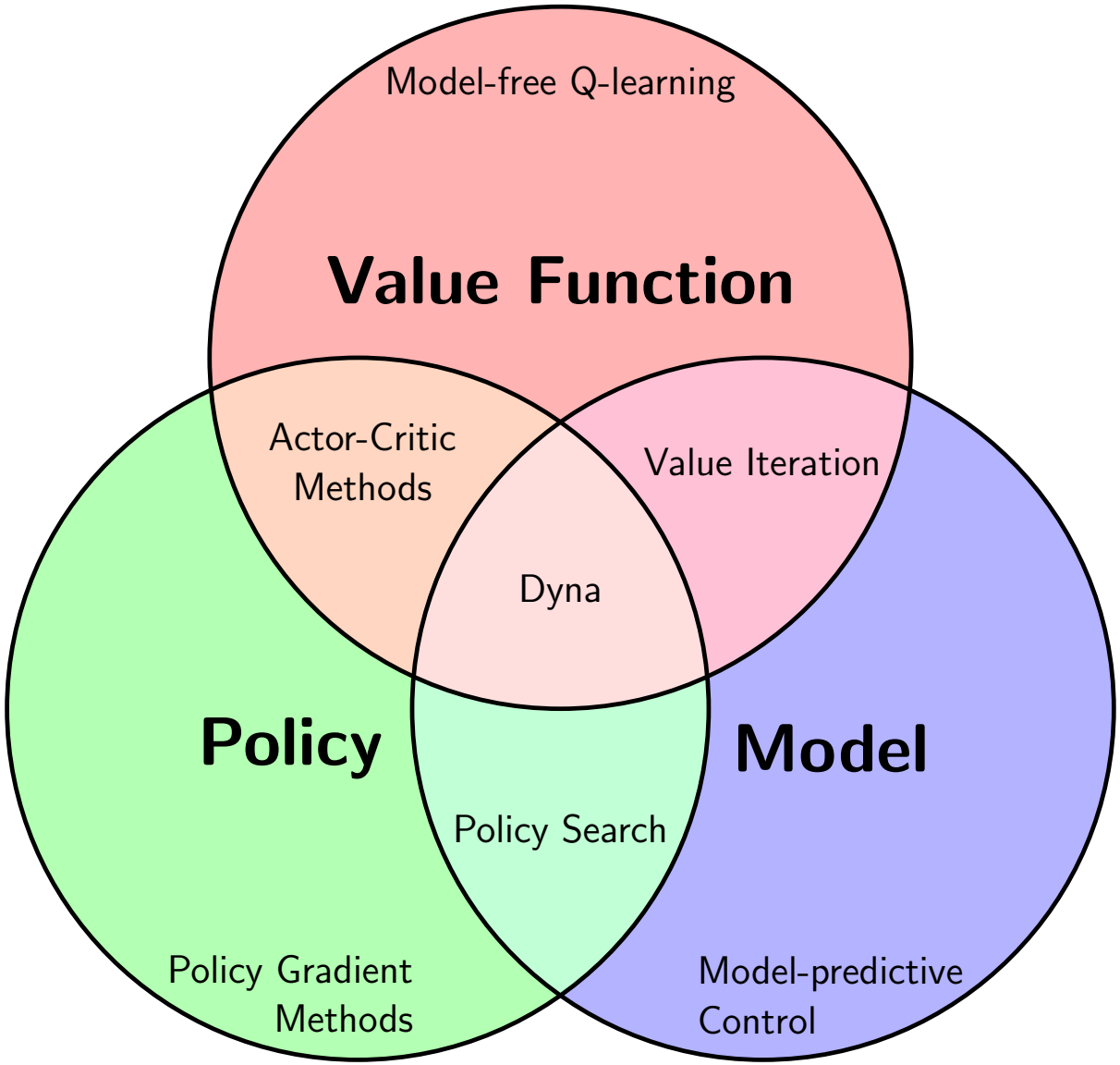


Maze Example: Model

- Rewards: -1 per time step
- Actions: up, down, left, right
- States: location of the agent



RL Overview


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

Course Outline

1. Introduction
2. Markov Decision Processes
3. Dynamic Programming
4. Monte-Carlo Methods
5. Temporal-Difference Methods
6. Planning and Learning
7. On-policy Prediction and Control with Function Approximation
8. Off-policy Methods with Function Approximation
9. Pre-Winter Break Q&A
10. Eligibility Traces
11. Policy Gradient Methods
12. Advanced Value-based Methods with Function Approximation
13. Inverse Reinforcement Learning
14. Overview of RL research in our group
15. Invited Talk (hopefully)

Lecture Overview

Organisation

- 6 ECTS
- Flipped classroom with pre-recorded lectures
- Q & A online via Zoom, every week on Friday, 14:00 (s.t.) - 15:30
- Format of the exam: open-book, open-internet online ILIAS exam (90 min, date: TBA by Dec 31, 2021, likely time-period: March 2022)
- Course material and **forum** on ILIAS: `https://ilias.uni-freiburg.de/goto.php?target=crs_2376878&client_id=unifreiburg`

Exercises

- Exercises are not mandatory, but highly recommended!
- Short walkthrough of sample solutions for most exercise sheets after the Q & A
- Please answer questions of others and ask your own questions in the **forum**

Exercises

- 6 ECTS $= 180$ working hours
- Lectures ≈ 30 hours
- In addition: exercise sheets and exam preparation
- Time management options:

7-9 hours per ex.	little exam prep.	**RECOMMENDED**
5-6 hours per ex.	more exam prep.	MINIMUM
0 hours per ex.	∞ exam prep.	**IMPOSSIBLE**

- Slide taken from Hannah Bast

Team

Organizers of the Course

Joschka Boedecker

Gabriel Kalweit

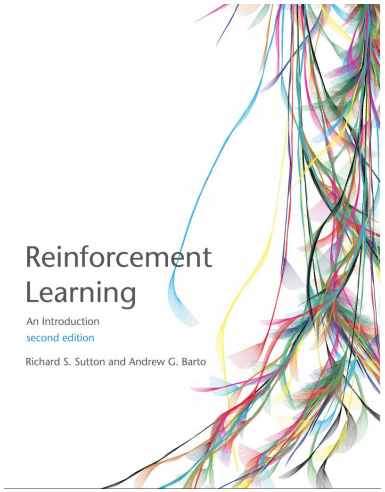
Jasper Hoffmann

Andreas Saelinger

Links to other lectures

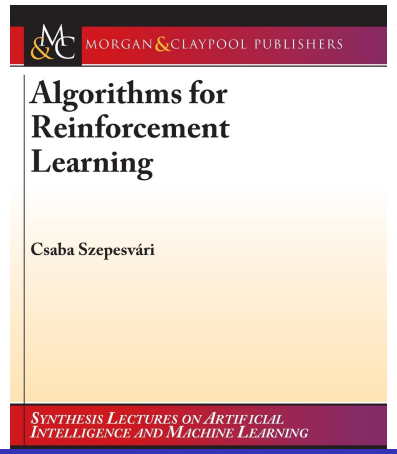
- Machine Learning Lecture (Summer term, recommended background)
- Deep Learning Lecture (Winter term, recommended background)
- Deep Learning Lab (Summer term)

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`



Acknowledgement

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

Lecture Overview

Multi-armed Bandits

- A multi-armed Bandit is a tuple $\langle \mathcal{A}, p \rangle$, where:
  - $\mathcal{A}$ is a finite set of actions (or *arms*) and
  - $p(r|a) = \Pr\{R_t = r | A_t = a\}$ is an unknown probability distribution over rewards
- In each time step t, the agent selects an action A_t
- The environment then generates reward R_t
- The agent aims at maximizing the cumulative reward

Multi-armed Bandits

- The value of action a is the expected reward $q(a) = \mathbb{E}[R_t | A_t = a]$
- If the agent knew q, it could simply pick the action with highest value to solve the problem
- Assume we knew optimal action a_*. Then the total regret is defined as:

$$L_t = \mathbb{E}\left[\sum_{t=1}^{T} q(a_*) - q(a_t)\right]$$

- Maximize cumulative reward $\equiv$ minimize total regret

Estimation of q

- We can estimate q given samples by $Q_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} R_t \mathbb{1}_{A_t=a}$, where $N_t(a)$ is the number of times a was taken in t time steps.
- Let R_i now denote the reward received after the ith selection of this action, and let Q_n denote the estimate of its action-value after it has been selected $n-1$ times.
- Equally, we can use an incremental implementation:

$$Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$$

- Generally, we can use some step size α:

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

- A constant α can also be used for non-stationary problems (i.e. problems for which the reward probabilities change over time)
- Having an estimate of q, we can do greedy action selection by:

$$A_t = \arg\max_a Q_t(a)$$

ϵ-greedy

- One of the simpliest ways to explore:
  - With probability $(1 - \epsilon)$ select the greedy action
  - With probability ϵ select a random action
- Can be coupled with a decay schedule for ϵ, so as to explore less when the estimate is quite accurate after some time
- Exemplary schedule: $\epsilon_{t+1} = (1 - \frac{t}{T})\epsilon_{\text{init}}$, where t is the current round and T is the maximum considered number of rounds

Softmax Action Selection

- Instead of choosing equally among actions, like ϵ-greedy, we can pick actions proportional to their value.
- This is called Softmax Action Selection and is modeled by a Boltzmann distribution over the Q-values:

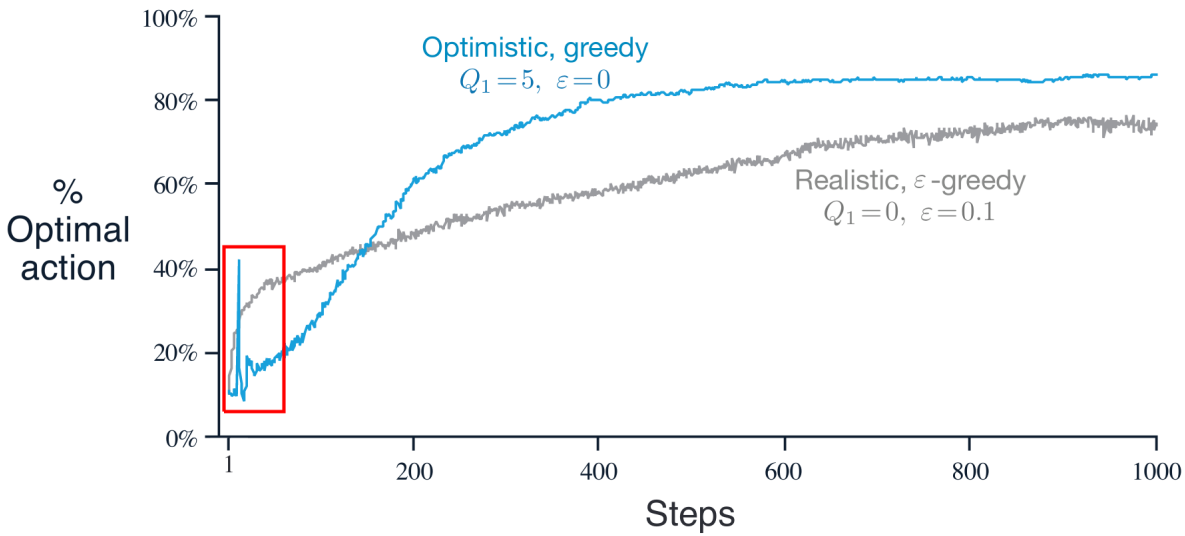
$$\frac{\exp(\frac{Q_t(a)}{\tau})}{\sum_{i=1}^{k} \exp(\frac{Q_t(i)}{\tau})},$$

  where τ is called the *temperature*.
- High temperatures cause the actions to be all (nearly) equiprobable.
- Low temperatures cause a greater difference in selection.

Optimistic Initial Values

- Initialize $Q_0(a)$ to high values for all $a \in \mathcal{A}$
- The result is that all actions are tried several times before the value estimates converge
- Encourages exploration only in the beginning, which is especially bad for non-stationary problems

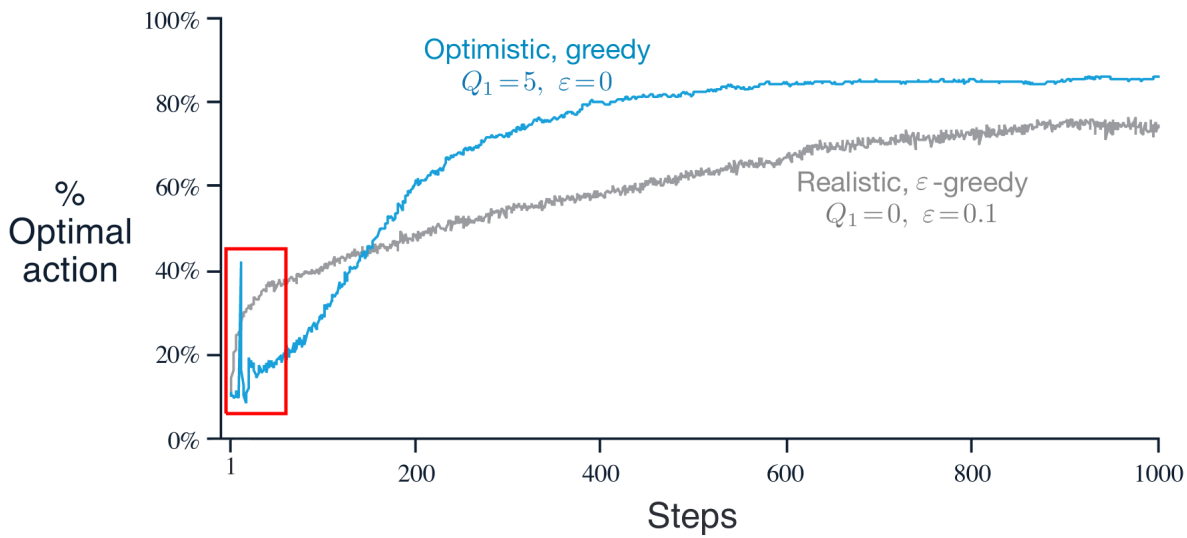
Question



Question (5 min)

The results should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? In other words, what might make this method perform particularly better or worse, on average, on particular early steps?

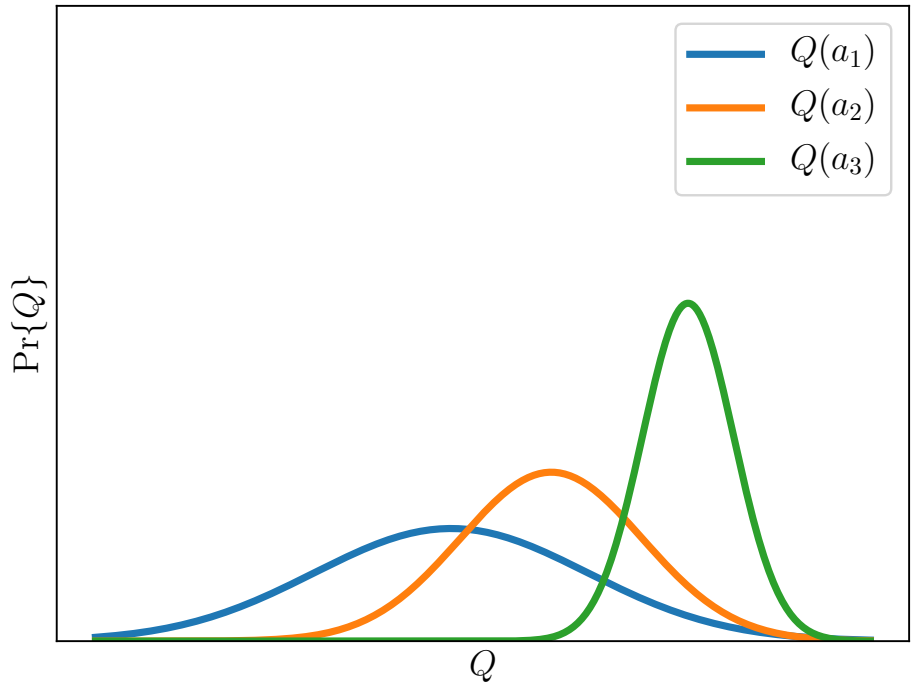
Solution



Solution

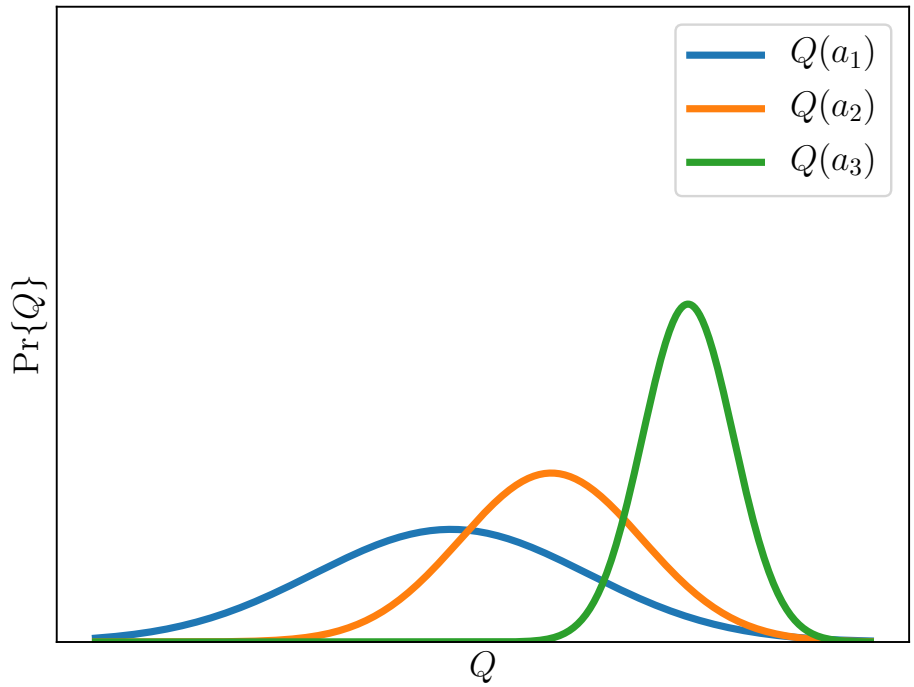
In the beginning, all action-values are overly optimistic. Therefore, each execution will reduce the value of the corresponding action. The value of the best action, however, will reduce the least in expectation. Thus, it will be focused by the greedy policy for some time – hence the peak. This will in turn reduce the value of the best action more, which is the reason for the drop. This procedure will repeat, but with smaller peaks the more the values converge.

Optimism in the Face of Uncertainty Principle



Which action should we pick?

Optimism in the Face of Uncertainty Principle



Which action should we pick?

Optimism in the Face of Uncertainty Principle

The more uncertain we are about an action-value, the more important it is to explore that action – since it could turn out to be the best action.

Upper Confidence Bound

- Exploration is needed because there is always uncertainty about the accuracy of the action-value estimates
- Idea: take into account how close their estimates are to being maximal and the uncertainties in those estimates
- For example by the *Upper Confidence Bound* (UCB) action selection:

$$A_t = \arg\max_a Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$$

- Each time a is selected, the uncertainty is presumably reduced
- On the other hand, each time an action other than a is selected, t increases but $N_t(a)$ does not
- One implementation of the *Optimism in the Face of Uncertainty Principle*

Contextual Bandits

- A contextual bandit is a tuple $\langle \mathcal{A}, p_{\mathcal{S}}, p_{\mathcal{R}} \rangle$, where:
  - $p_{\mathcal{S}}(s) = \Pr\{S_t = s\}$ is an unknown distribution over states (or *contexts*) and
  - $p_{\mathcal{R}}(r|s,a) = \Pr\{R_t = r | S_t = s, A_t = a\}$ is an unknown probability distribution over rewards
- In each time step t, the environment generates s_t and the agent selects action a_t
- The environment then generates reward r_t
- Example: Imagine a slot machine that changes its display color each time it changes its action-values
  - If you have no information about the task, the described bandit algorithms might not work very well
  - If you know the color, however, you can have different policies for different colors
- If actions are allowed to affect the next state as well as the immediate reward, then we have the full reinforcement learning problem