Reinforcement Learning

Lecture 1
Introduction

Stergios Christodoulidis

MICS Laboratory
CentraleSupélec
Université Paris-Saclay

https://stergioc.github.io/

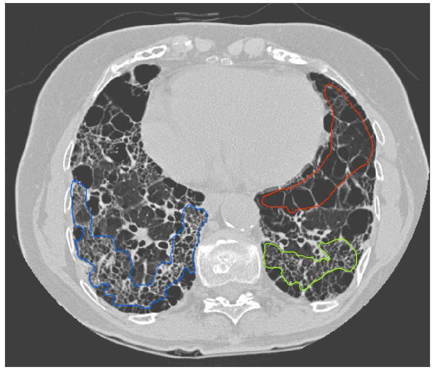
CentraleSupélec

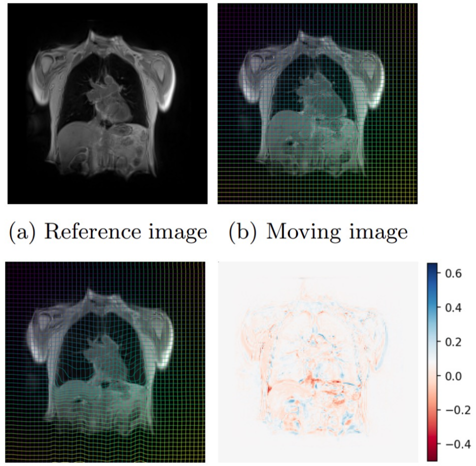
Short Introduction

- PhD degree in University of Bern
  - Medical Image Analysis
  - Diagnosis Support Systems

- Postdoctoral Researcher at GR
  - AI for Cancer
  - Treatment Decision Support
  - Understanding of biological

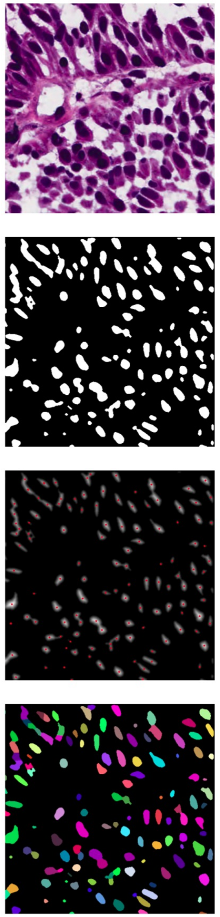
- Assistant Prof at CS MICS Lab
  - Machine Learning
  - Healthcare Applications





(a) Reference image (b) Moving image

(c) Deformed image (d) Difference



The Team

Stergios Christodoulidis

βiomathematics Research Group, MICS Laboratory

Office: Bâtiment Bouygues sb.132

Email: stergios.christodoulidis@centralesupelec.fr

Othmane Laousy

βiomathematics Research Group, MICS Laboratory

Office: Bâtiment Bouygues, MICS Laboratory

Email: othmane.laousy@centralesupelec.fr

Admin

- Every Tuesday 8:30 to 11:30
- Amphi F3.05, Breguet
- Information, slides and announcements will be posted at Edunao:
  - https://centralesupelec.edunao.com/course/view.php?id=3753

➢ Office hours: we will be available right after the lecture or, send us an email and we will find a good time to meet

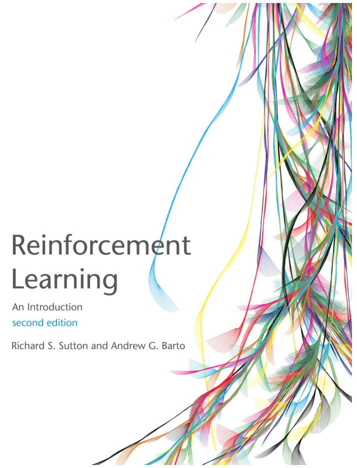
Learning Objectives of the Course

By the end of the course, you will be able to:

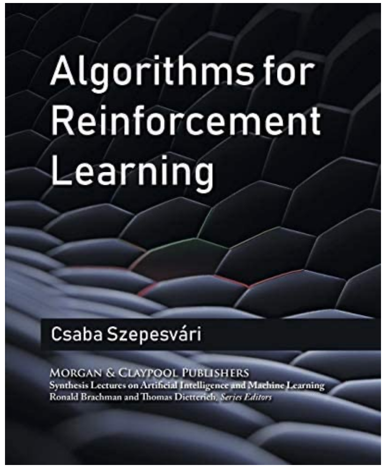
- Understand the major aspects of the Reinforcement Learning (RL) problem.
- Get an understanding of the trade-off between exploration and exploitation when dealing with RL problems.
- Learn the most important algorithms in order to be able to solve a particular RL problem.
- Get familiar with the available libraries.
- Evaluate several methods and select appropriate ones for solving the task at hand.

Reading Material

- An Introduction to Reinforcement Learning,
  - Sutton and Barto, MIT Press, 1998
  - Available free online!
  - http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html



- Algorithms for Reinforcement Learning,
  - Csaba Szepesvari, Morgan and Claypool, 2010
  - Available free online!
  - http://www.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf



Organization of the Course

- The course will be split in Lectures and Lab Sessions
- The assessment will be based on:
    - Submited Lab Sessions Notebooks
    - An Individual Assignement
    - A Group Project of 2 people (max 3)

Assessment in Detail

- Assessment will be based on Lab Sessions deliverables (40%), group project (60%) and for submited lecture notes (extra 10%).
- Labs:
  - All labs sessions notebooks needs to be submited (jupyter notebooks, 20%)
  - Individual Assignment (jupyter notebook, 20%)
- Group Project:
  - Project proposal document (single page document, 10%)
  - Final Submission (report, code, 50%)
- Lecture Notes:
  - Submited in any digital format (.md, .tex, .doc, +10%)

Schedule of the Lectures

- Part I: Elementary Reinforcement Learning
  - 01 Introduction to Reinforcement Learning & k-armed Bandits (3h)
  - 02 Markov Decision Processes (1h30)
  - 03 Dynamic Programming (1h30)
  - 04 Monte Carlo Methods (1h30)
  - 05 Temporal-Difference Learning (1h30)

➢ (2 weeks break)

- Part II: Reinforcement Learning in Practice
  - 06 Value Function Approximation (1h30)
  - 07 Policy gradient methods (1h30)
  - 08 Deep Learning Applications and Recent Advances of RL (Guest Lecturer, 1h30)

Schedule of the Lab Sessions

- Part I: Elementary Reinforcement Learning:
  - Lab 01 - Multi-armed bandits (greedy, ε-greedy, 1h30)
  - Lab 02 - Gridwold problem (dynamic programming , 1h30)
  - Lab 03 - Cart Pole (q-learning , 1h30)
  - Individual Assignment – Flappy Bird Environment (your solution , 1h30)

➢ (2 weeks break)

- Part II: Reinforcement Learning in Practice
  - Group Project time (2x1h30)
  - Lab 05 Deep Q-Learning (Guest Lecturer, 1h30)

Due Dates

- Part I Deliverables + Group Project Description: **22/02/2022 (Lecture 06 – Start of Part II)**
- Group Project: **08/03/2022 (Lecture 08 – End of the Course)**

Acknowledments

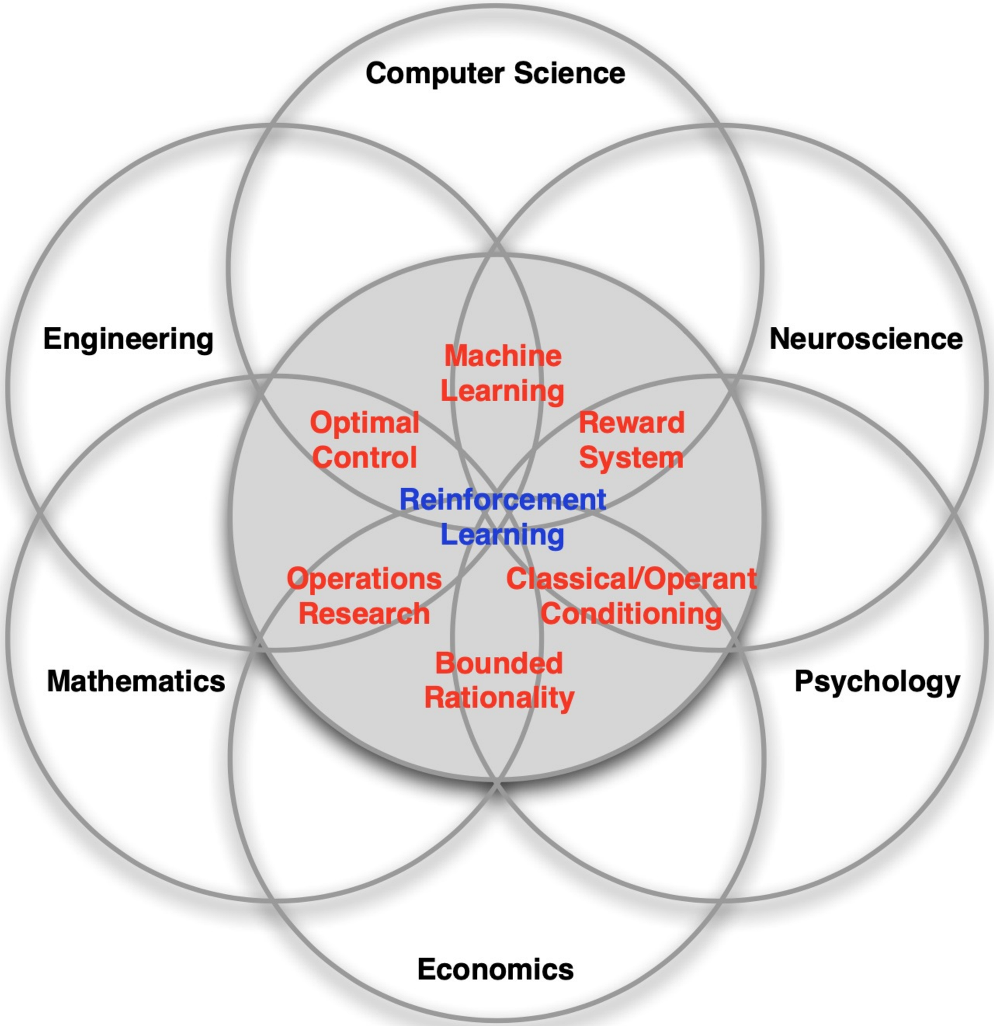
- The lecture slides are using parts of the structure and material by:
    - David Silver (Deep Mind, UCL)
    - Hado van Hasselt (Deep Mind)
    - https://deepmind.com/learning-resources

Today's Lecture

- Introduction to Reinforcement Learning
- The Reinforcement Learning problem
- Inside an RL agent
- Examples
- Exploration vs Exploitation
- Multi-armed bandits
- Greedy and ε-Greedy Agents
- Policy Gradients
- UCB Algorithms

What is Reinforcement Learning?

What is Reinforcement Learning?



[David Silver, IRL, UCL 2015]

What is Reinforcement Learning?

- Science and framework of learning to make decisions from interaction
- This requires us to think about
    - ...time
    - ...(long-term) consequences of actions
    - ...actively gathering experience
    - ...predicting the future
    - ...dealing with uncertainty
- Huge potential scope
- A formalization of the AI problem

What is Reinforcement Learning?

- People and animals learn by interacting with their environment.
- This type of learning can be defined as active learning and it based on sequencial interactions
  - Future interactions depend on earlier ones
- The learning is typically goal-oriented
- Instead of learning with examples, a reward is optimized.

RL characteristics

- There is no supervision, only a reward signal.
- Feedback might be delayed.
- Time matters (sequential, non i.i.d. data)
- Agent’s actions affect the subsequent data it receives

Why Reinforecement Learning?

There are distinct reasons to learn:

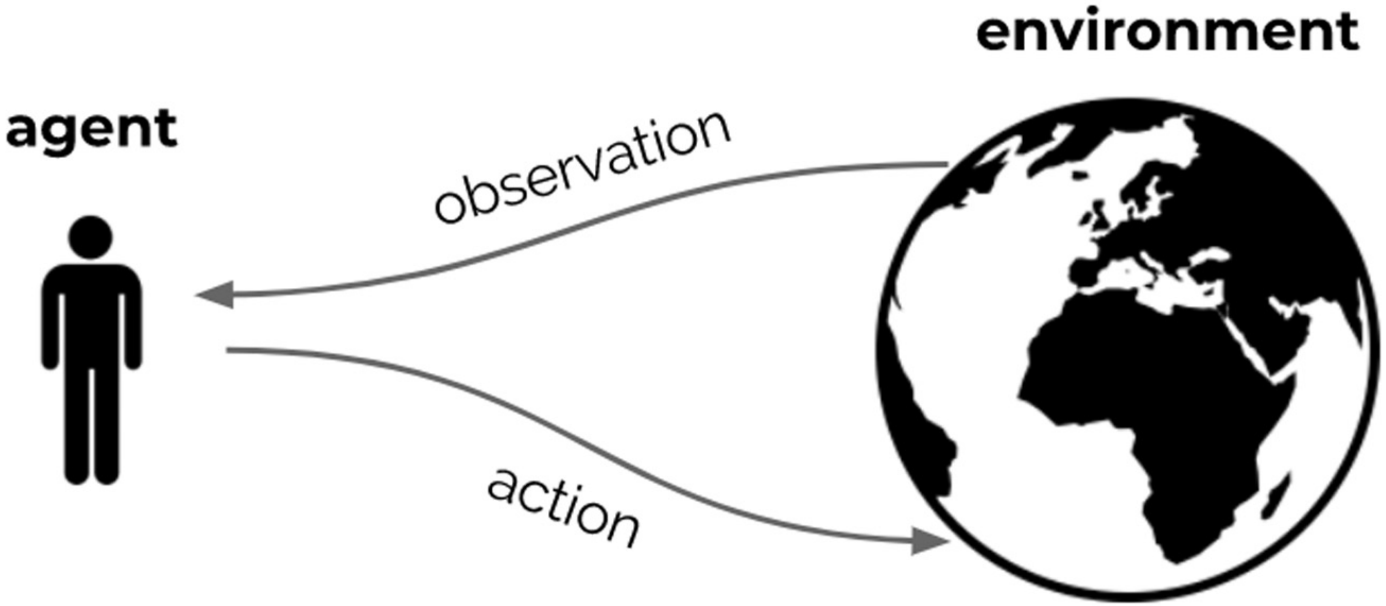
1. Find solutions
   - A program that plays chess really well
   - A manufacturing robot with a specific purpose
2. Adapt online, deal with unforeseen circumstances
   - A chess program that can learn to adapt to you
   - A robot that can learn to navigate unknown terrains

- Reinforcement learning can provide algorithms for both cases
- Note that the second point is not (just) about generalization — it is about continuing to learn efficiently online, during operation


205 2 1
0
OXYGEN
ACTIVISION
0 0
99
10600
3
PLAYER
ENEMY
80
0121
2 299
ACTIVISION
0045 0000

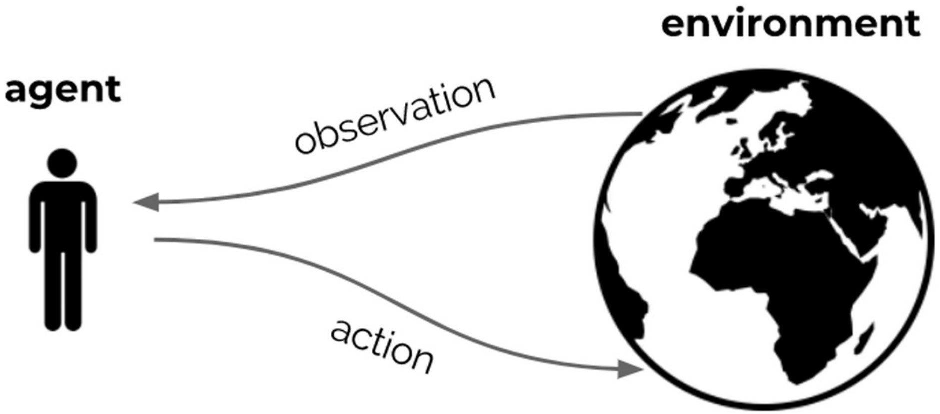
Formalizing the Reinforcement Learning Problem

The RL Problem



[Hado van Hasselt, 2021]

The Agent and the Environment



- At each step t the agent:
  - Receives observation O_t (and reward R_t)
  - Executes action A_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1} (and reward R_{t+1})

[Hado van Hasselt, 2021]

Reward

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t — defines the goal
- The agent's goal is to maximize cumulative reward

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots$$

- We call this the return.
- The Reinforcement Learning paradigm is based on the reward hypothesis

Hypothesis
Any goal can be formalized as the outcome of maximizing a cumulative reward

Examples of Rewards

- Fly a helicopter
  - Reward: air time (+), crashing (-) …
- Manage an investment portfolio
  - Reward: gain (+), …
- Control a heating system
  - Reward: efficiency(+), cost(-), …
- Make a robot walk
  - Reward: distance covered(+), speed (+/-), out of battery(-) …
- Play video games
  - Reward: score (+), win (+), …

Values

- We call the expected cumulative reward, from a state s, the value

$$\begin{aligned} \mathrm{v}(s) &= \mathbb{E}\left[G_t | S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + R_{t+2} + R_{t+3} + \cdots \, | S_t = s\right] \end{aligned}$$

- The value depends on the actions the agent takes
- Goal is to maximize value, by picking suitable actions
- Rewards and values define utility of states and action (no supervised feedback)
- Returns and values can be defined recursively

$$\begin{aligned} &G_t = R_{t+1} + G_{t+1} \\ &\mathrm{v}(s) = \mathbb{E}\left[R_{t+1} + \mathrm{v}(S_{t+1}) | S_t = s\right] \end{aligned}$$

Action Values

- It is also possible to condition the value on actions:

$$q(s, a) = \mathbb{E}\left[G_t | S_t = s, A_t = a\right]$$

Maximizing Value by Taking Actions

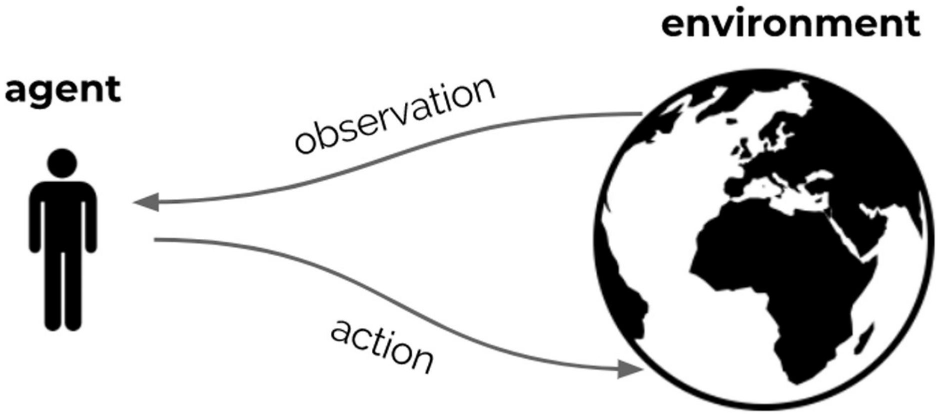
- Goal: select actions to maximize total future reward
- Actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
    - A financial investment (may take months to mature)
    - Refueling a helicopter (might prevent a crash in several hours)
    - Blocking opponent moves (might help winning chances many moves from now)
- A mapping from states to actions is called a policy

Core Consepts

The reinforcement learning formalism includes:

- Reward signal (specifies the goal)
- Environment (dynamics of the problem)
- Agent, containing:
  - Agent state
  - Policy
  - Value function estimate?
  - Environment Model?

Environment State



- The environment state S_t^e is the environment's internal state
- It is usually invisible to the agent
- Even if S_t^e is visible, it may contain lots of irrelevant information

[Hado van Hasselt, 2021]

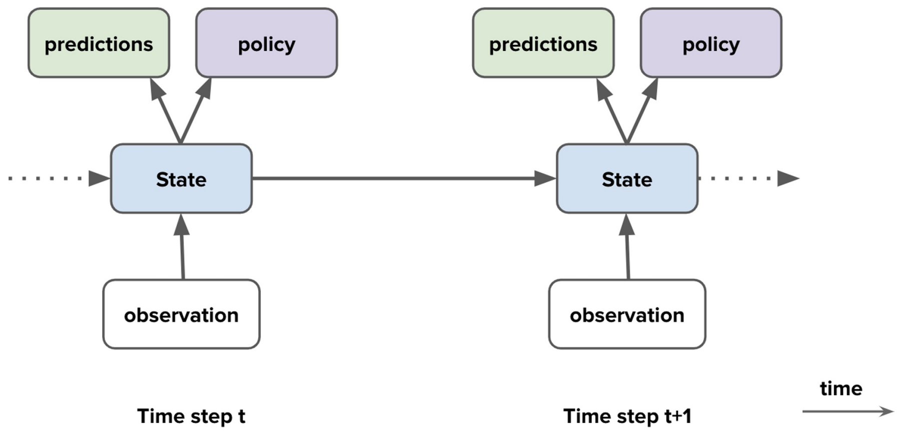
Agent State

- The history is the sequence of observations, actions, rewards

$$H_t = O_1, R_1, A_1, \dots, A_{t-1}, O_t, R_t$$

- For instance, the sensorimotor stream of a robot
- This history is used to construct the agent state S_t^a
- State is the information used to determine what happens next
- Formally, state is a function of the history:

$$S_t^a = f(H_t)$$



[Hado van Hasselt, 2021]

Information or Markov State

- An information state (a.k.a. Markov state) contains all useful information from the history.

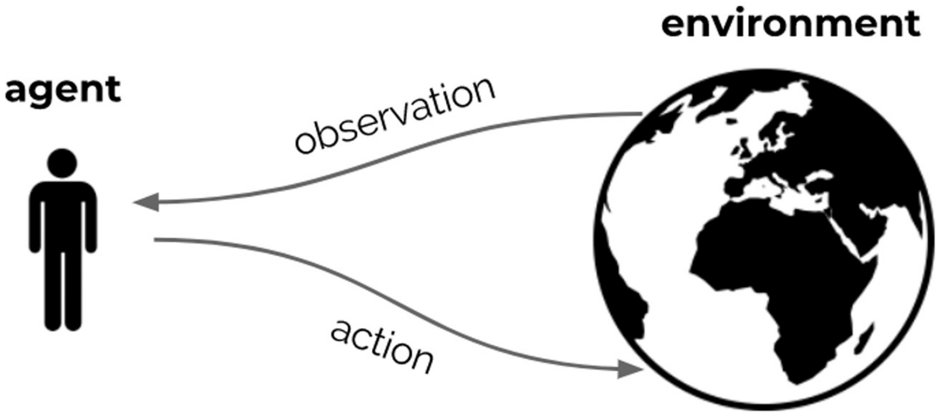
Definition

A state S_t is Markov if and only if:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1 \dots S_t]$$

- "The future is independent of the past given the present"
- Doesn't mean it contains everything, just that adding more history doesn't help
- Once the state is known, the history may be thrown away
- The environment state S_t is Markov
- The history H_t is Markov

Fully Observable Environments



Full observability

- Suppose the agent sees the full environment state
  - Agent state = environment state = information state
  - Formally, this is a Markov decision process (MDP)

$$S_t^a = S_t^e = O_t$$

[Hado van Hasselt, 2021]

Partially Observable Environments

- Partial observability means that agent indirectly observes environment:
  - A robot with camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now agent state $\neq$ environment state
- Formally this is a partially observable Markov decision process (POMDP)
- Agent must construct its own state representation S_t^a, e.g.
  - Complete history: $S_t^a = H_t$
  - Beliefs of environment state: $S_t^a = (\mathbb{P}[S_t^e = s^1], \dots, \mathbb{P}[S_t^e = s^n])$
  - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

Major Components of an RL Agent

Major Components of an RL Agent

- An RL agent may include one or more of these components:
  - Policy: agent’s behavior function
  - Value function: how good is each state and/or action
  - Model: agent’s representation of the environment

Policy

- A policy is the agent's behavior
- It is a map from state to action, e.g.
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s]$

Value Function

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore, to select between actions, e.g.

$$\mathrm{v}_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s]$$

Model

- A model predicts what the environment will do next
- $\mathcal{P}$ predicts the next state
- $\mathcal{R}$ predicts the next (immediate) reward, e.g.

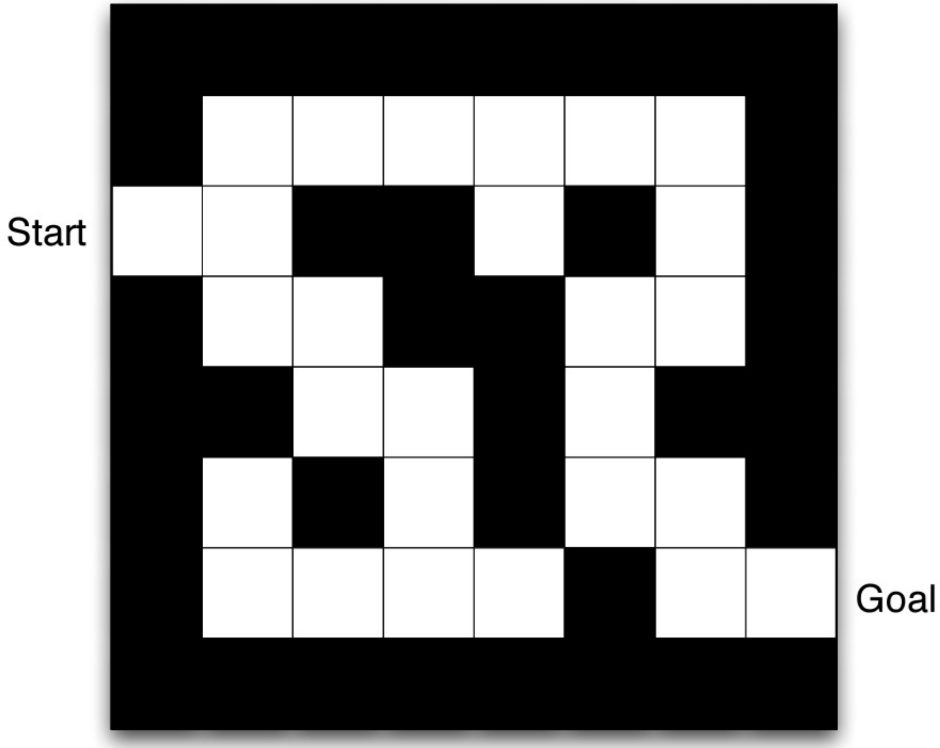
$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

$$\mathcal{R}_{s}^{a} = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

Maze Example

Maze Example (1/4)

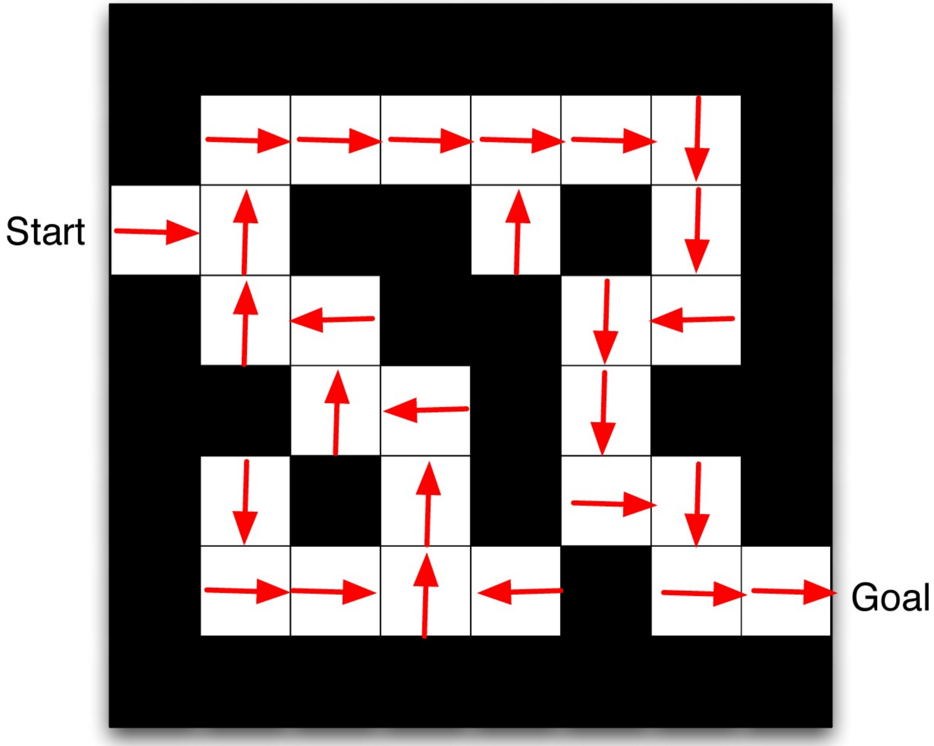
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent’s location



[Hado van Hasselt, 2021]

Maze Example (2/4)

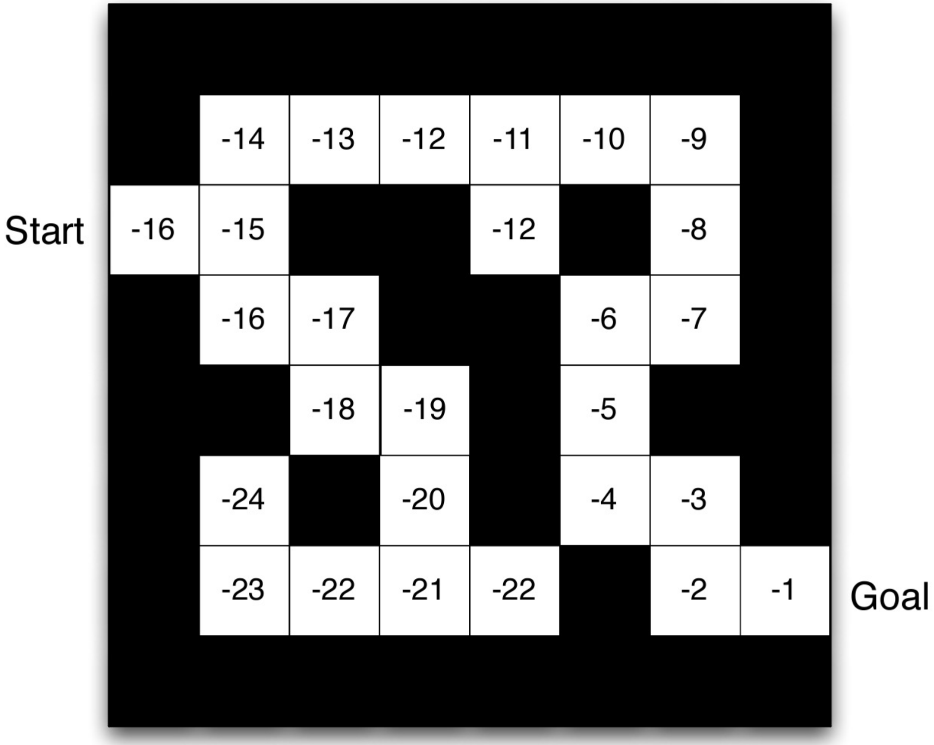
- Arrows represent policy $\pi(s)$ for each state s



[Hado van Hasselt, 2021]

Maze Example (3/4)

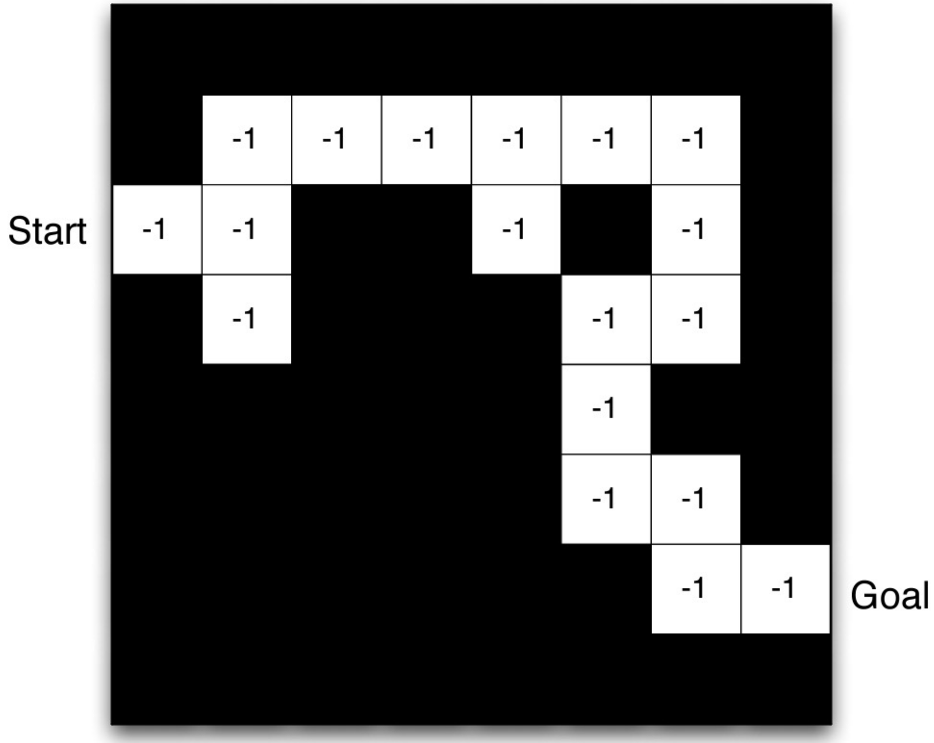
- Numbers represent value $v_{\pi}(s)$ of each state s



[Hado van Hasselt, 2021]

Maze Example (4/4)

- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect
- Grid layout represents transition model $\mathcal{P}^a_{ss'}$
- Numbers represent immediate reward R^a_s from each state s (same for all a)



[Hado van Hasselt, 2021]

Agent Categories (1/2)

- Value Based
  - No Policy (Implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function

Agent Categories (2/2)

- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Optionally Policy and/or Value Function
  - Model

Subproblems of the RL problem

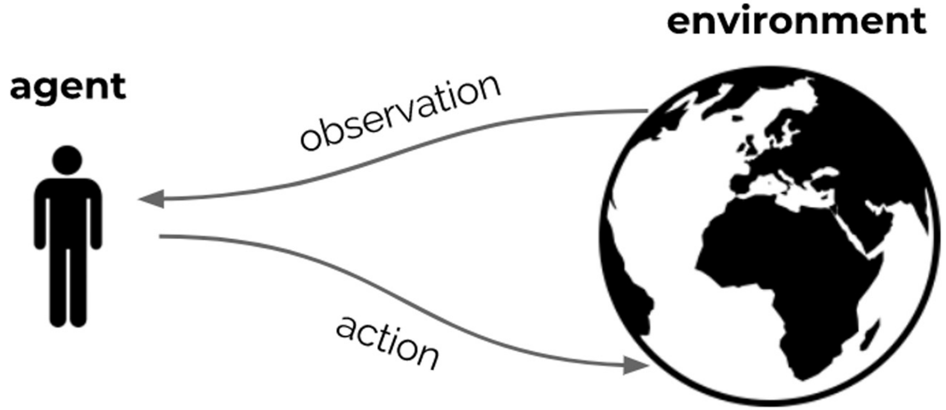
Prediction and Control

- Prediction: evaluate the future (for a given policy)
- Control: optimize the future (find the best policy)
- These can be strongly related.
- If we could predict everything, do we need anything else?

Learning Agent Components

- All components are functions
  - Policies, π: $\mathcal{S} \rightarrow \mathcal{A}$ (or to probabilities over $\mathcal{A}$)
  - Value functions, v: $\mathcal{S} \rightarrow R$
  - Models, m: $\mathcal{S} \rightarrow \mathcal{S}$ and/or r: $\mathcal{S} \rightarrow R$
  - State update, u: $\mathcal{S} \times \mathcal{O} \rightarrow \mathcal{S}$
- E.g., we can use neural networks, and machine learning techniques to learn them.
- Deep reinforcement learning is a rich and active research field

Summary



- Reinforcement learning is the science of learning to make decisions
- Agents can learn a policy, value function and/or a model
- The general problem involves taking into account time and consequences
- Decisions affect the reward, the agent state, and environment state
- Learning is active: decisions impact data

[Hado van Hasselt, 2021]

Exploration vs Exploitation

Exploration vs. Exploitation Dilemma

- Online decision-making involves a fundamental choice:
  - Exploitation Make the best decision given current information
  - Exploration Gather more information
- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decisions

Examples

- Restaurant Selection
  - Exploitation Go to your favorite restaurant
  - Exploration Try a new restaurant
- Online Banner Advertisements
  - Exploitation Show the most successful advertisement
  - Exploration Show a different advertisement
- Oil Drilling
  - Exploitation Drill at the best-known location
  - Exploration Drill at a new location
- Game Playing
  - Exploitation Play the move you believe is best
  - Exploration Play an experimental move

Principles

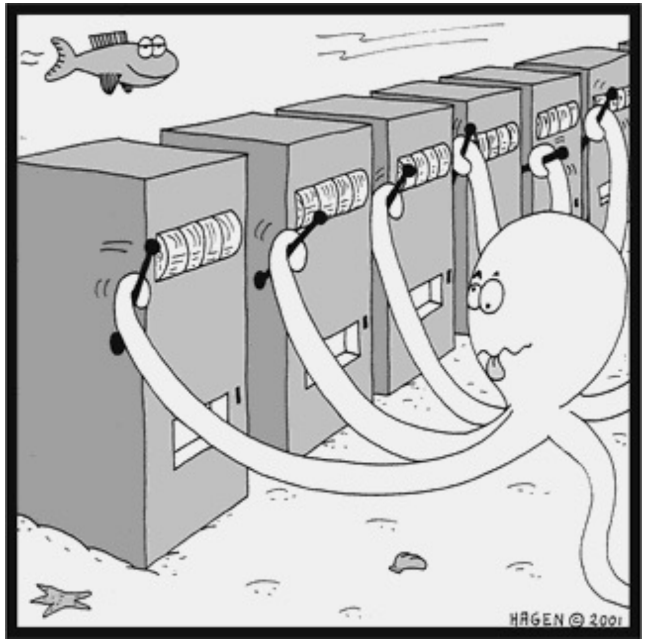
- Naive Exploration
  - Add noise to greedy policy (e.g., ε-greedy)
- Optimistic Initialization
  - Assume the best until proven otherwise
- Optimism in the Face of Uncertainty
  - Prefer actions with uncertain values
- Probability Matching
  - Select actions according to probability they are best Information
- State Search
  - Lookahead search incorporating value of information

Principles

- Naive Exploration
  - Add noise to greedy policy (e.g., ε-greedy)
- Optimistic Initialization
  - Assume the best until proven otherwise
- Optimism in the Face of Uncertainty
  - Prefer actions with uncertain values
- Probability Matching
  - Select actions according to probability they are best Information
- State Search
  - Lookahead search incorporating value of information

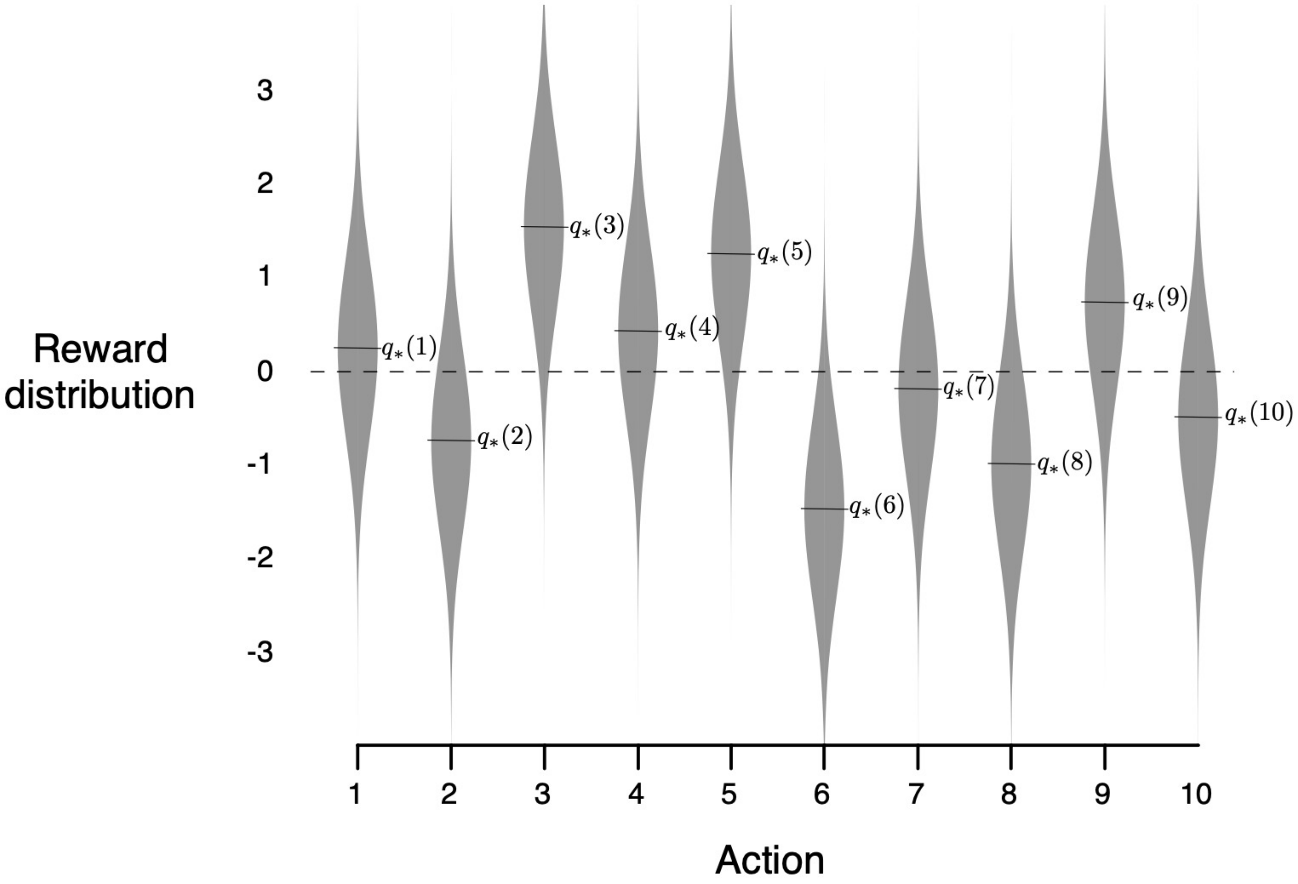
Multi-Armed Bandits

- A multi-armed bandit is a set of distributions $< \mathcal{A}, \mathcal{R} >$
- $\mathcal{A}$ is a (known) set of actions (or “arms")
- $\mathcal{R}^a(r) = \mathbb{P}[r|a]$ is an unknown probability distribution over rewards
- At each step t the agent selects an action $a_t \in \mathcal{A}$
- The environment generates a reward $r_t \sim \mathcal{R}^{a_t}$
- The goal is to maximize cumulative reward $\sum_{\tau=1}^{t} r_\tau$



[David Silver, IRL, UCL 2015]

10-armed Testbed



[An Introduction to Reinforcement Learning, Sutton and Barto]

Values and Regret

- The action-value is the expected reward for action a,

$$Q(a) = \mathbb{E}[\mathcal{R} \mid A = a]$$

- The optimal value V^* is

$$V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$$

- The regret is the opportunity loss for one step

$$I_t = \mathbb{E}[V^* - Q(a_t)]$$

- The regret for the optimal action is zero

Regret

- We want to minimize total regret:

$$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} \mathrm{V}^* - Q(a_\tau)\right]$$

- Maximize cumulative reward ≡ minimize total regret
- The summation spans over the full ‘lifetime of learning’

Algorithms

- We will discuss several algorithms:
  - Greedy, ε-Greedy
  - UCB
  - Policy Gradient

Greedy Algorithms

Greedy Algorithm

- One of the simplest algorithm
- Select action with highest value:

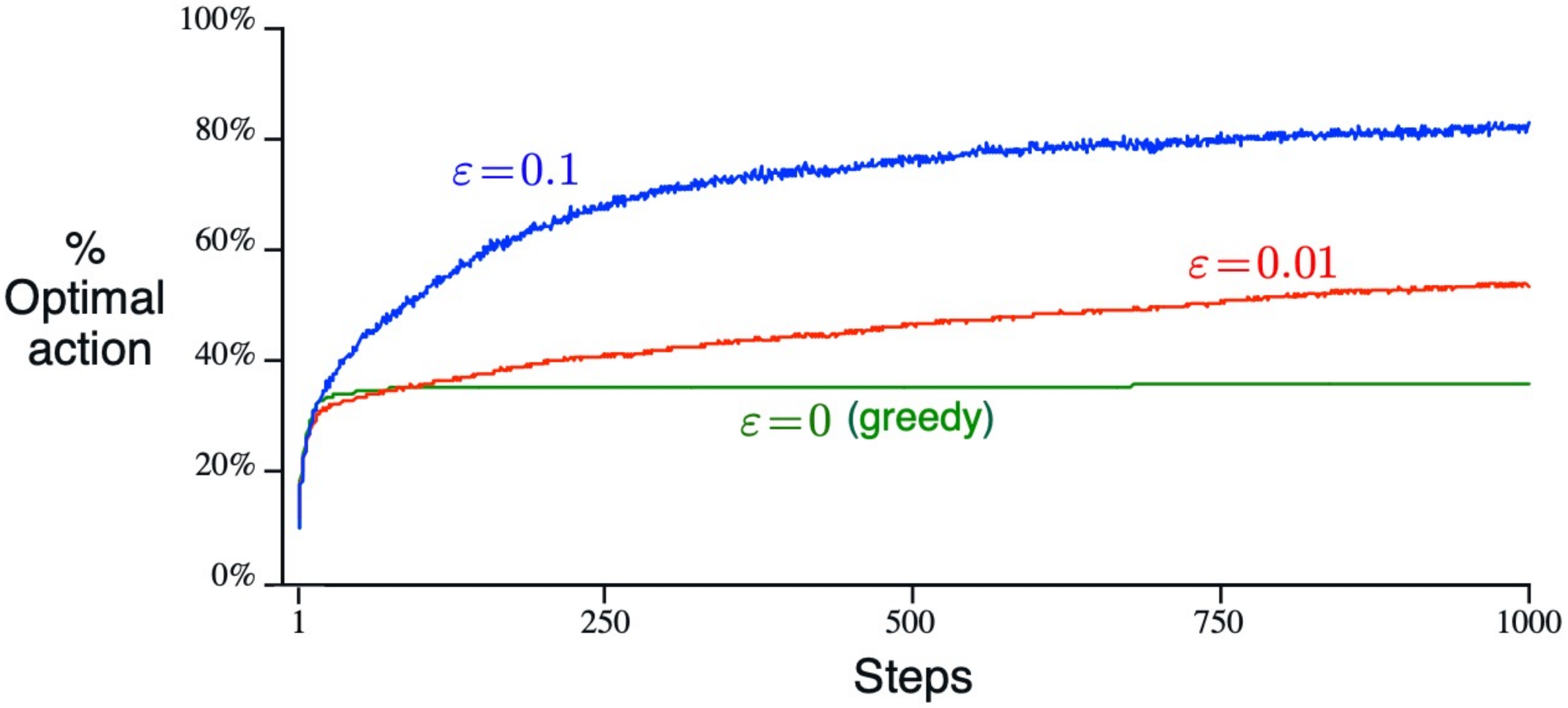
$$A_t = \underset{a \in \mathcal{A}}{argmax}\, Q_t(a)$$

- Greedy can lock onto a suboptimal action forever

ε-Greedy Algorithm

- The ε-greedy algorithm:
  - With probability 1 −ε select greedy action: $A_t = \underset{a \in \mathcal{A}}{argmax}\, Q_t(a)$
  - With probability ε select a random action
- ε-greedy continues to explore

Greedy vs ε-Greedy

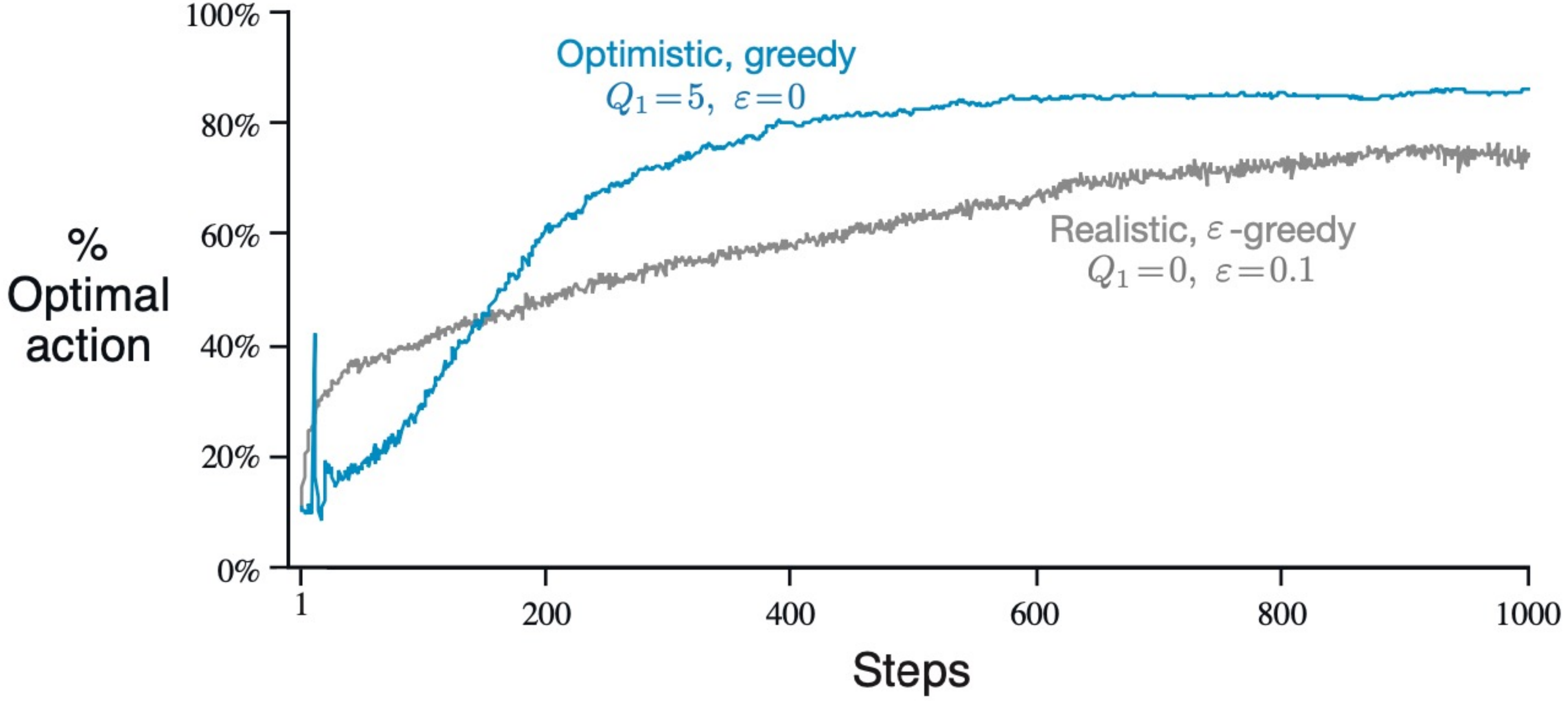


[An Introduction to Reinforcement Learning, Sutton and Barto]

Optimistic Initialization

- Simple and practical idea: initialize $Q(a)$ to high value
- Update action value at every time step
- Value decreases over time
- Encourages systematic exploration early on
- But can still lock onto suboptimal action

Optimistic vs Realistic Initialization



[An Introduction to Reinforcement Learning, Sutton and Barto]

Upper Confidence Bound

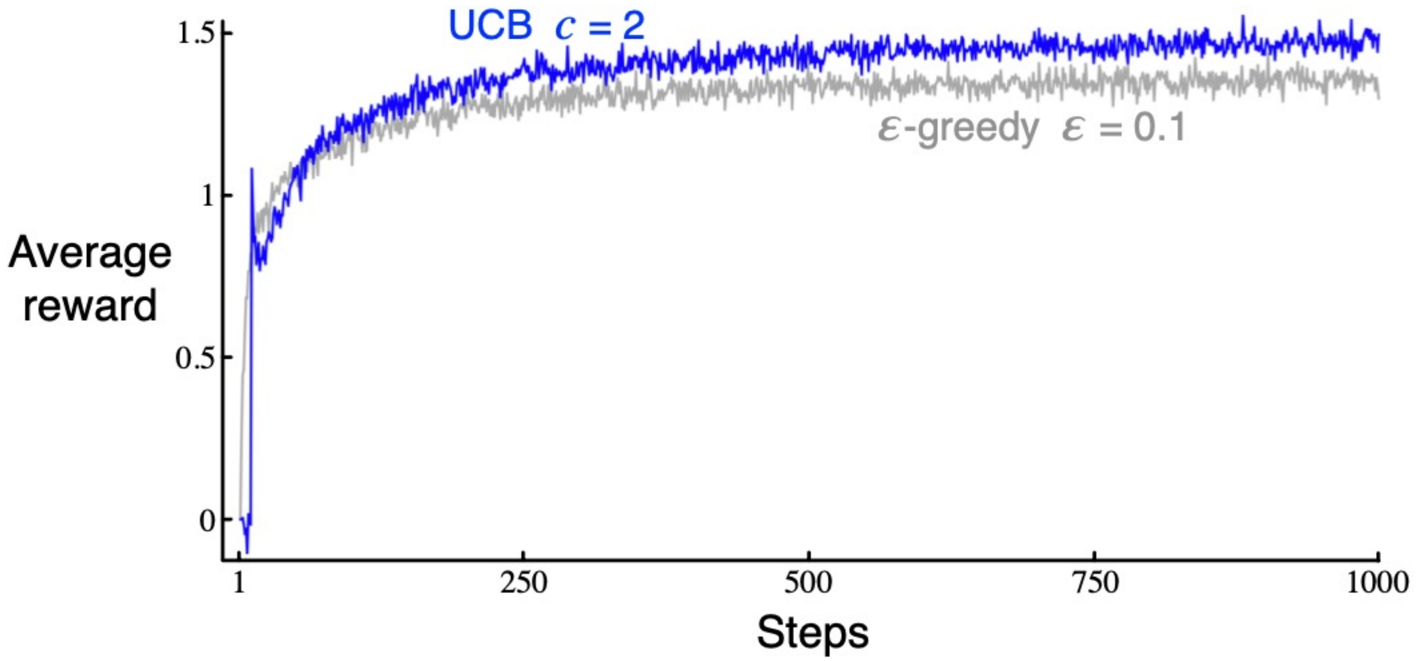
Upper Confidence Bound (UCB)

- The ε-Greedy algorithm performs exploration without any preference (random).
- Why not explore in a more explicit way?
- UCB selects the actions with the most uncertain value function estimates!

$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right]$$

- $N_t(a)$ denotes the times action a was selected prior to time t
- Eventually, the square-root term is a measure of uncertainty

UCB vs ε-Greedy



[An Introduction to Reinforcement Learning, Sutton and Barto]

Policy Gradient

Policy Gradient Methods

- So far, we have considered methods that estimate action values to select actions.
- Then we select the actions with the highest action-value.
- Instead, can learn policies $\pi(a)$ directly?
- We can consider learning a numerical preference for each action:

$$H_t(a) \in \mathbb{R}$$

Gradient Bandits

- Considering a set of action preferences $H_t(a)$ we can define a policy using the softmax distribution

$$\pi_t(a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

- In this configuration, the larger the preference the more often this action is selected.

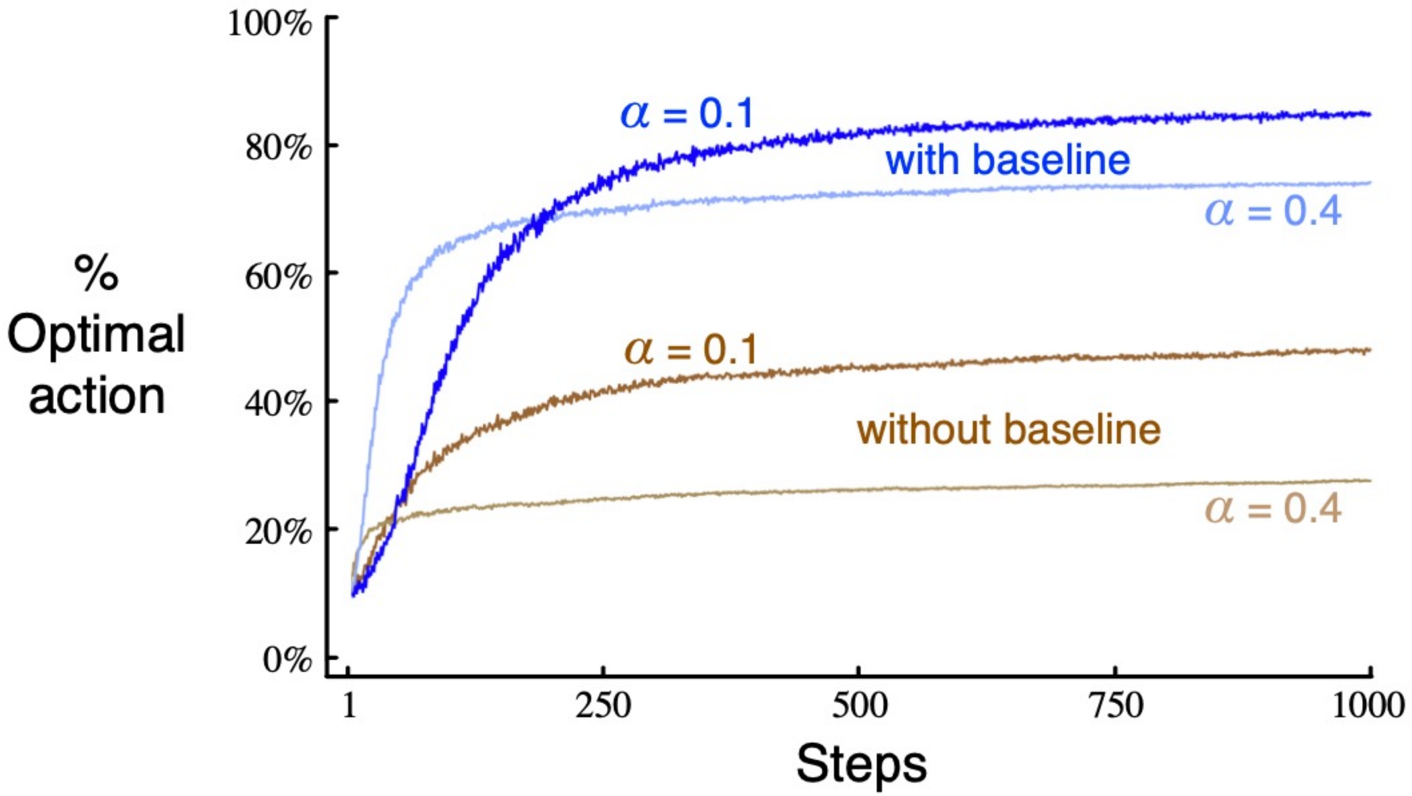
Stohastic Gradient Ascent

- There is a natural learning algorithm for soft-max actions preferences.
- This is the stochastic gradient ascent.
- On each step after selecting an action A_t and receiving a reward R_t the action preferences can be updated using:

$$H_{t+1}(A_t) = H_t(A_t) + \gamma(R_t - \bar{R}_t)\big(1 - \pi_t(A_t)\big)$$
$$H_{t+1}(a) = H_t(a) - \gamma(R_t - \bar{R}_t)\pi_t(a), \qquad for\ all\ a \neq A_t$$

- with $\bar{R}_t$ being the average of the rewards up to but not including time t (baseline)

Gradient Bandits with and without Baseline



[An Introduction to Reinforcement Learning, Sutton and Barto]

Conclusion

- Have covered several principles for exploration/exploitation
- Each principle was developed in bandit setting
- Same principles can be extended to the MDP setting