Reinforcement Learning
Lecture 1: Introduction

S. M. Ahsan Kazmi

Me:

S. M. Ahsan Kazmi
Senior Lecturer (Associate Professor)
School of Information Science, Facility of Computer Science & Creative Technologies, The University of the West of England, UK.

email: a.kazmi@innopolis.ru

Research: 5G and beyond networks, Network virtualization, End to end Slicing, Distributed learning, Smart environments, etc.

Google Scholar: https://scholar.google.co.kr/citations?user=rsAINPYAAAAJ&hl=en

Researchgate: https://www.researchgate.net/profile/Sm_Kazmi

Office Hours: Please contact me by email to arrange a suitable time if you would like to discuss something about the module.

Outline

- Admin
- About Reinforcement Learning
- The Reinforcement Learning Problem
- Inside An RL Agent
- Agent Categories

Admin

Course Delivery Format

- Lecture:
  - 1 session per week
- Lab:
  - Will be announced during the lab session.

Course Grading Criteria

- Individual Labs: 30 points
  - 1 lab session per week
  - Up to 8-9 labs
  - 3 Assignments, 10 points each

- Exam: 20 points

- Project: 50 points- 3 to 4 members
  - Proposal Presentation: 5 points
  - Final Report: 15 points
  - Implementation: 25 points
  - Final Presentation: 5 points

Course Outline

- Week 1
  - Introduction to RL
- Week 2
  - Exploration and Exploitation
- Week 3
  - Markov Decision Processes
- Week 4
  - Planning by Dynamic Programming
- Week 5
  - Model-Free Prediction-I
- Week 6
  - Proposal Presentation (02/27)
- Week 7
  - Model-Free Prediction-II
- Week 8
  - Model-Free Control
- Week 9
  - Value Function Approximation & Policy Gradient Methods
- Week 10
  - Integrating Learning and Planning
  - State-of-the-art RL algorithms
- Week 11
  - Exam (04/03)
- Week 12-13
  - Support & Exam feedback
- Week 14
  - Presentations, Reports, and Code submissions (04/24)
- Week 15
  - Project Feedback

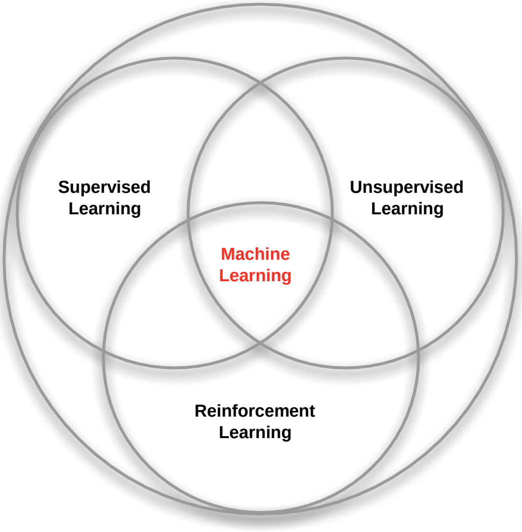
Text Books

- Reinforcement Learning: An Introduction, Sutton & Barto 2018
    - http://incompleteideas.net/book/the-book-2nd.html

- Algorithms for Reinforcement Learning, Szepesvari, Morgan and Claypool, 2010

About Reinforcement Learning

Branches of Machine Learning



What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent’s actions affect the subsequent data it receives

Reinforcement Learning?

What is reinforcement learning?

- People and animals learn by **interacting with the environment**
- This differs from certain other types of learning
  - It is **active** rather than passive
  - Interactions are often **sequential** — future interactions can depend on earlier ones
- We can learn **without examples** of optimal behavior
- We are **goal-directed**
- Instead, we optimize some **reward signal**

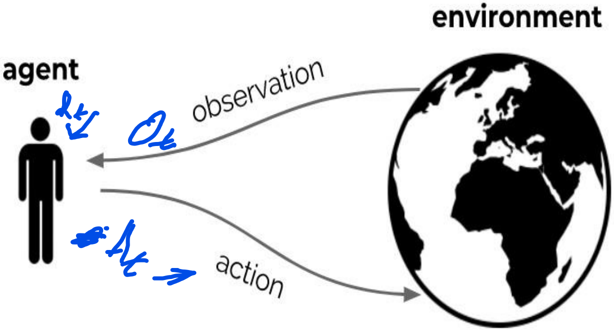
agent выпол действ → получ наблюд → получ награду

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

The interaction loop

- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step



Goal: optimise sum of rewards, through repeated interaction

Sequential Decision Making

- Goal: **select actions to maximise value**
- Actions may have long-term consequences
- Reward may be delayed
- It may be better to sacrifice immediate rewards to gain more long-term reward
- Examples:
  - Refueling a helicopter (might prevent a crash in several hours)
  - Defensive moves in a game (may help chances of winning later)
  - Learning a new skill (can be costly & time-consuming at first)
- A mapping from states to actions is called a **policy**

The reward hypothesis

- The agent's job is to maximize cumulative reward
- A reward R_t is a scalar feedback signal that indicates how well the agent is doing at step t

- Reinforcement learning is based on the reward hypothesis:

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

Examples of RL problems & Rewards

- Fly a helicopter → **Reward**: air time, inverse distance, ...
- Manage an investment portfolio → **Reward**: gains, gains minus risk, ...
- Control a power station → **Reward**: efficiency, ...
- Make a robot walk → **Reward**: distance, speed, ...
- Play video or board games → **Reward**: win, maximise score, ...

If the goal is to learn via interaction, these are all reinforcement learning problems
(Irrespective of which solution you use)

Rewards & Return

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

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

- We call this the **return**

Values

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

$$\boxed{\begin{aligned} v(s) &= \mathrm{E}\left[G_t \mid S_t = s\right] \\ &= \mathrm{E}\left[R_{t+1} + R_{t+2} + R_{t+3} + \ldots \mid S_t = s\right] \end{aligned}}$$

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

$$G_t = R_{t+1} + G_{t+1}$$

$$v(s) = \mathrm{E}\left[R_{t+1} + v(S_{t+1}) \mid S_t = s\right]$$

Action values

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

$$\begin{aligned} q(s, a) &= \mathbb{E}\left[G_t \mid S_t = s, A_t = a\right] \\ &= \mathbb{E}\left[R_{t+1} + R_{t+2} + R_{t+3} + \dots \mid S_t = s, A_t = a\right] \end{aligned}$$

- We will talk in-depth about state and action values later

Core concepts

The reinforcement learning formalism includes

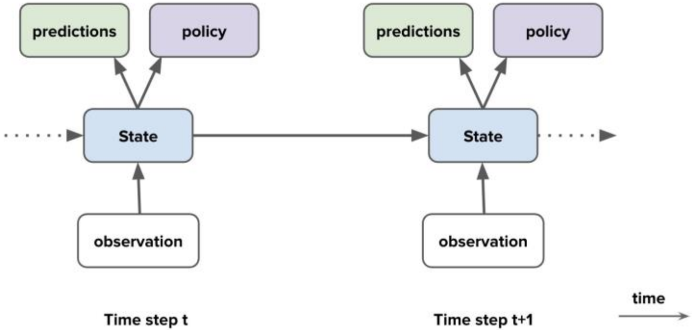
- **Environment** (dynamics of the problem)
- **Reward** signal (specifies the goal)
- **Agent**, containing:
    - Agent state
    - Policy
    - Value function estimate
    - Model
- We will now go into the agent

Inside the Agent: the Agent State

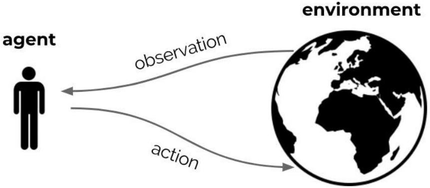
Agent components

Agent components

- **Agent state**
- Policy
- Value functions
- Model



Environment State



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

Agent State

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

$$H_t = O_0, A_0, R_1, O_1, \ldots, O_{t-1}, A_{t-1}, R_t, O_t$$

- For instance, the sensorimotor stream of a robot
- This history is used to construct the **agent state** S_t
  - i.e. whatever information the agent uses to pick the next action
  - i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history:

$$\boxed{S_t^a = f(H_t)}$$

Inside the Agent: the Policy

Agent components

Agent components

- Agent state
- **Policy**
- Value function
- Model

Policy

- A **policy** defines the agent's behaviour
- It is a map from agent state to action
- Deterministic policy: $A = \pi(S)$
- Stochastic policy: $\pi(A|S) = p(A|S)$

Inside the Agent: Value Estimates

Agent components

Agent components

- Agent state
- Policy
- Value function
- Model

Value Function

- The actual value function is the expected return

$$v_\pi(s) = \mathrm{E}\left[G_t \mid S_t = s, \pi\right]$$

$$= \mathrm{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s, \pi\right]$$

- We introduced a **discount factor** $\gamma \in [0, 1]$
    - Trades off the importance of immediate vs long-term rewards
- The value depends on a policy

Value Functions

- The return has a recursive form $G_t = R_{t+1} + \gamma G_{t+1}$
- Therefore, the value has as well

$$
\begin{aligned}
v_\pi(s) &= \mathrm{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t \sim \pi(s)\right] \\
&= \mathrm{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t \sim \pi(s)\right]
\end{aligned}
$$

Here $a \sim \pi(s)$ means a is chosen by policy π in state s (even if π is deterministic)

- This is known as a **Bellman equation** (Bellman 1957)
- A similar equation holds for the optimal (=highest possible) value:

$$
v_*(s) = \max_a \mathrm{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right]
$$

This does **not** depend on a policy

Value Function approximations

- Agents often approximate value functions
- We will discuss algorithms to learn these efficiently
- With an accurate value function, we can behave optimally
- With suitable approximations, we can behave well, even in intractably big domains

Inside the Agent: Models

Agent components

Agent components

- Agent state
- Policy
- Value function
- **Model**

Model

- A **model** predicts what the environment will do next
    - E.g., P predicts the next state

$$\mathsf{P}(s, a, s') \approx p(S_{t+1} = s' \mid S_t = s, A_t = a)$$

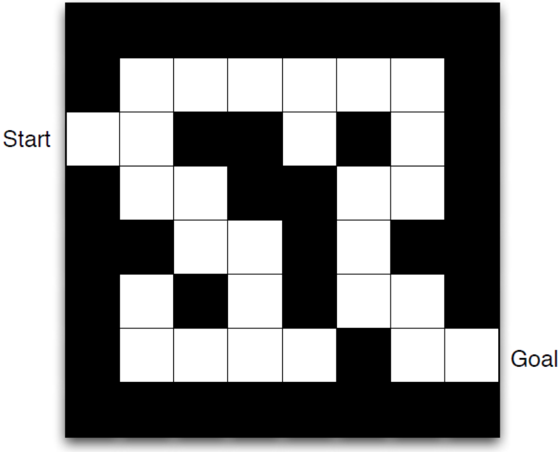
    - E.g., R predicts the next (immediate) reward

$$\mathsf{R}(s, a) \approx \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

- A model does not immediately give us a good policy - we would still need to plan

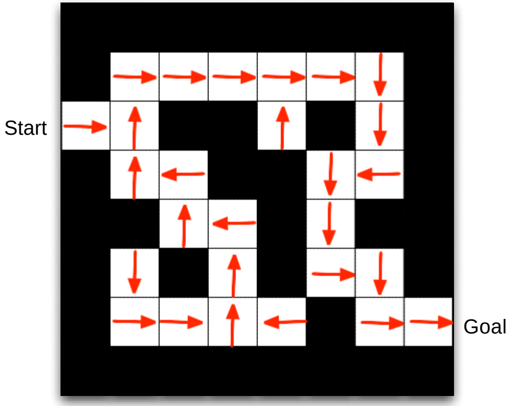
An Example

Maze Example



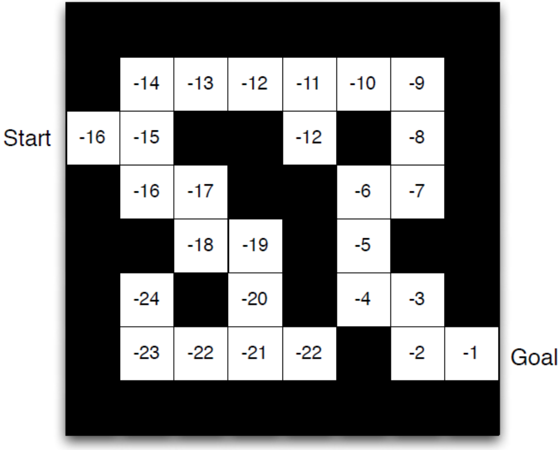
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze Example: Policy



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

Maze Example: Value Function



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

Questions?

End of Lecture