Lecture 1: Introduction

EL2805: Reinforcement Learning

Alexandre Proutiere

KTH, The Royal Institute of Technology

Lecture 1: Outline

1. Admin
2. What is Reinforcement Learning? Why is it interesting?
3. Schedule of the course
4. Lecture 0: Basic notations and notions in probability

1. Admin

- Canvas:
  - Schedule
  - Lecture slides and notes / exercices
  - Python codes
  - Problems and solutions
- Grading: 2 reports on the computer labs (each 1.5 hp), 1 home work (1 hp), exam (3.5 hp)
- 14 lectures, 3 computer labs, 6 exercise sessions
- Teaching assistants: Yassir Jedra, Damianos Tranos, and Ingvar Max Ziemann (PhD students in EECS/IS/DCS)
- Contact: alepro@kth.se, tranos@kth.se, jedra@kth.se, ziemann@kth.se

Labs

Work by group of 2

- Lab 0: guidelines for python, and first examples (not graded).
  Solutions provided on canvas
- Labs 1 and 2: dynamic programming and Q learning, SARSA
  Lab 1: Nov. 14, Lab 2: Nov. 20
  Deadline for uploading your (individual) report on canvas:
  Nov 30, 11:59 PM
- Lab 3: Deep RL algorithms
  Lab 3: Dec. 4
  Deadline for uploading your (individual) report on canvas:
  Dec 14, 11:59 PM

Homework

Work by group of 2

- Text available Nov 20
- Deadline for uploading your (individual) report on canvas: **Dec 2, 11:59 PM**

Text books and resources

The course roughly follows:

R. Sutton, A. Barto, "Reinforcement learning: An introduction", 2nd edition, 2017 (available online)

Python codes of all examples in the book available online on github.

Other textbooks

- M. Puterman, "Markov decision processes: discrete stochastic dynamic programming", Wiley, 2014
- D. Bertsekas, "Dynamic Programming and Optimal Control", vol 1 and 2, Athena Scientific, 2012
- C. Szepesvari, "Algorithms for Reinforcement Learning", Morgan and Claypool, 2010 (available online)

Text books and resources

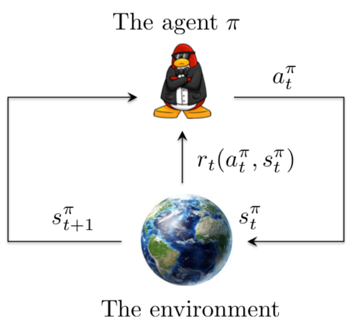
Examples of other contemporary courses on Reinforcement Learning:

- Introduction to RL, David Silver, UCL:
  `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html`
- Deep RL, Sergey Levine, UC Berkeley:
  `http://rail.eecs.berkeley.edu/deeprlcourse/`

All videos are on youtube ....

2. What is Reinforcement Learning?

Learning optimal sequential behaviour / control from interacting with the environment



Unknown state dynamics and reward function:

$s_{t+1}^\pi = F(s_t^\pi, a_t^\pi)$

$r(s_t^\pi, a_t^\pi)$

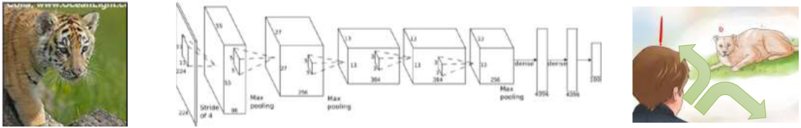
Reinforcement Learning: Applications

- Robotics
- Portfolio optimisation
- Playing games (better than humans)
- Self-driving cars
- Optimal communication protocols in radio networks
- Display ads
- Search engines
- ...

Reinforcement Learning is not new!

- Mathematical foundations: Bellman's Dynamic Programming (1952), Robbins-Monro algorithm (1951), Stochastic Gradient descent algorithm (Kiefer-Wolfowitz, 1952)
- Q-learning (Watkins 1989)
- RL using neural networks (Lin 1993), survey on Neural networks for Control (Miller-Sutton-Werbos 1995)
- ...

Deep RL: end-to-end learning for complex systems

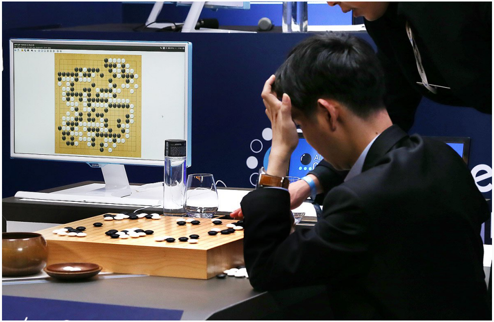


Raw Observation   Action

Learning to play Atari games from pixel observations (google deep mind 2015):
`https://www.youtube.com/watch?v=AVg_YIp09ps`

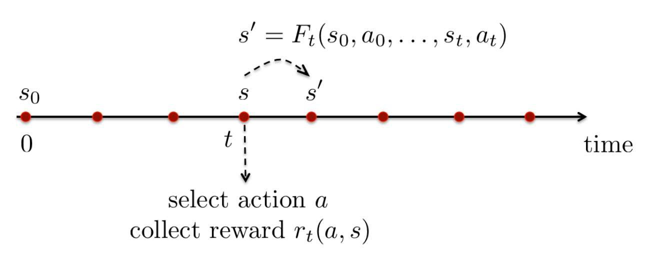
- From pixels to actions: perception, interpretation, and decisions are *jointly* learnt using deep learning models
- ... Nice but cannot exploit our expertise in these various subproblems

Deep RL: AlphaGo



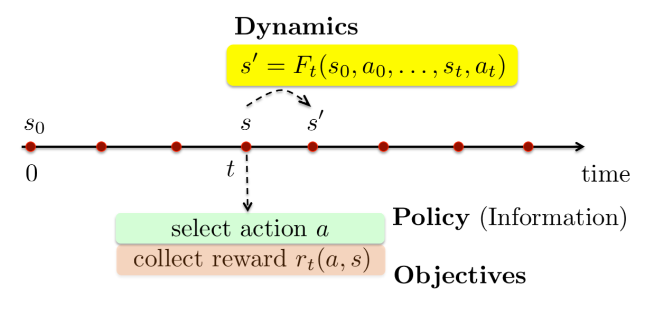
AlphaGo: A Deep RL algorithm beating Go world champion (google deep mind 2016)
Estimated number of positions in the game of go: 10^{172}
(Chess: $\sim 10^{43}$ according to Claude Shannon)

RL is Sequential Decision Making



Objective. Devise a sequential action selection / control policy maximising rewards

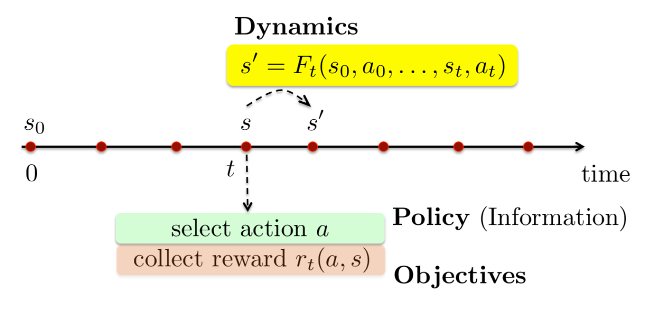
Sequential Decision Making



Problem definition

1. System dynamics
2. Set of available policies – available information or feedback to the decision maker
3. Reward structure

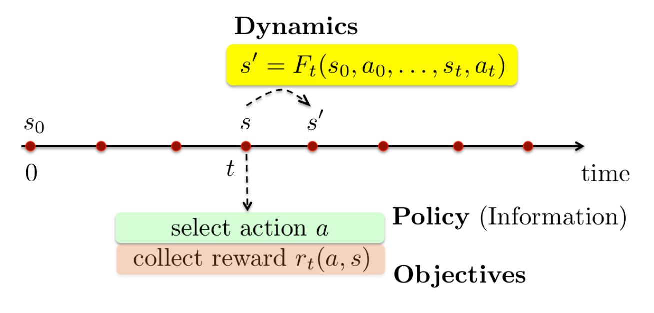
Sequential Decision Making



Dynamics. A few examples:

- Linear: $s_{t+1} = As_t + Ba_t$
- Deterministic and stationary: $s_{t+1} = F(s_t, a_t)$
- Markovian: $\mathbb{P}(s_{t+1} = s' | h_t, s_t = s, a_t = a) = p_t(s'|s, a)$ where $\sum_{s'} p_t(s'|s, a) = 1$; homogenous if $p_t(s'|s, a) = p(s'|s, a)$

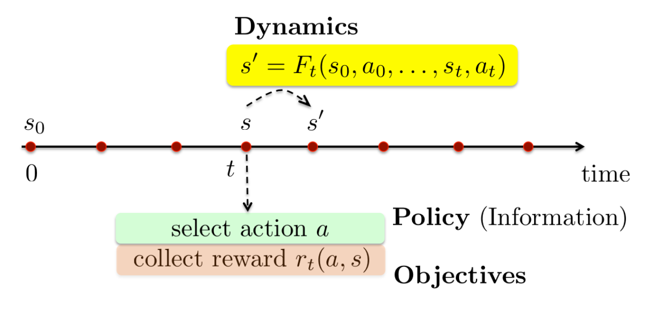
Sequential Decision Making



Information - Set of policies. A few examples:

- Markov Decision Process (MDP)
  - Fully observable state and reward
  - Known reward distribution and transition probabilities
  - a_t function of $(s_0, a_0, r_0, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$

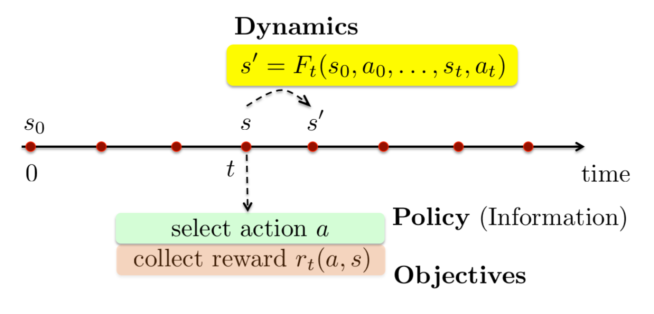
Sequential Decision Making



Information - **Set of policies.** A few examples:

- Partially Observable Markov Decision Process (POMDP)
  - Partially observable state: we know z_t with known $\mathbb{P}[s_t = s | z_t]$
  - Observed rewards
  - Known reward distribution and transition probabilities
  - a_t function of $(z_0, a_0, r_0, \ldots, z_{t-1}, a_{t-1}, r_{t-1}, z_t)$

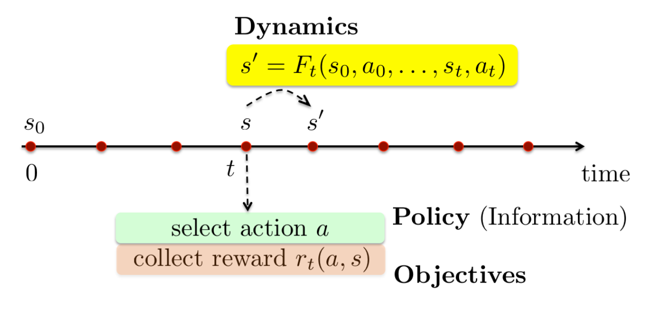
Sequential Decision Making



Information - **Set of policies.** A few examples:

- Reinforcement learning
  - Observable state and reward
  - Unknown reward distribution
  - Unknown transition probabilities
  - a_t function of $(s_0, a_0, r_0, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$

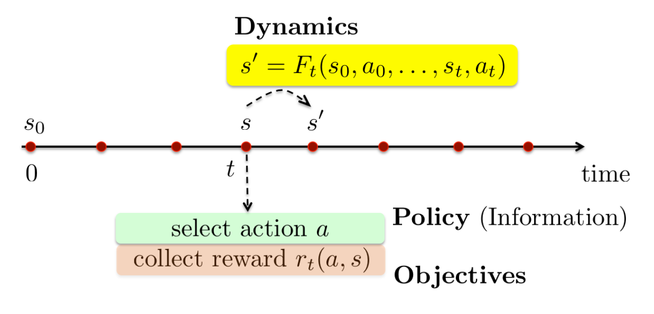
Sequential Decision Making



Information - Set of policies. A few examples:

- Adversarial problems
  - Observable state and reward
  - Arbitrary and time-varying reward function and state transitions
  - a_t function of $(s_0, a_0, r_0, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$

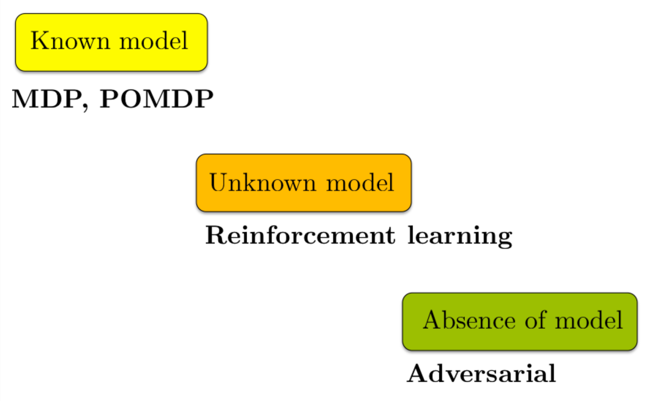
Sequential Decision Making



Objectives. A few examples:

- Finite horizon: $\max_\pi \mathbb{E}[\sum_{t=0}^{T} r_t(s_t^\pi, a_t^\pi)]$
- Infinite horizon discounted: $\max_\pi \mathbb{E}[\sum_{t=0}^{\infty} \lambda^t r_t(s_t^\pi, a_t^\pi)]$
- Infinite horizon average: $\max_\pi \liminf_{T\to\infty} \frac{1}{T}\mathbb{E}[\sum_{t=0}^{T} r_t(s_t^\pi, a_t^\pi)]$

Problem classification

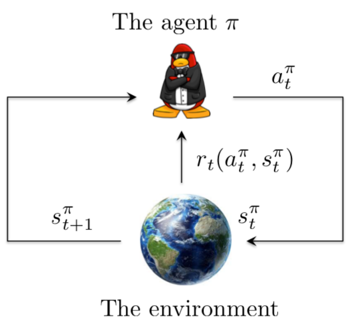


Selling an item

You need to sell your house, and receive offers sequentially. Rejecting an offer has a cost of 10 kSEK. What is the rejection/acceptance policy maximising your profit?

MDP. Offers are i.i.d. with known distribution
Reinforcement learning. (Bandit optimisation) Offers are i.i.d. with unknown distribution
Adversarial problem. The sequence of offers is arbitrary

In this course: RL for Markov Decision Process (MDP)



State dynamics:
$s_{t+1}^\pi = F_t(s_t^\pi, a_t^\pi)$

- History at t: $h_t^\pi = (s_1^\pi, a_1^\pi, \dots, s_{t-1}^\pi, a_{t-1}^\pi, s_t^\pi)$
- Markovian environment: $\mathbb{P}[s_{t+1}^\pi = s' | h_t^\pi, s_t^\pi = s, a_t^\pi = a] = p(s'|s, a)$
- Stationary deterministic rewards (for simplicity): $r_t(s, a) = r(s, a)$

Example

Playing pacman (Google Deepmind experiment, 2015)

State: the current displayed image
Action: right, left, down, up
Feedback: the score and its increments + state

What is to be learnt and optimized?

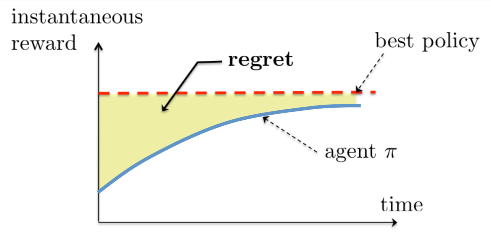
- MDP: The state dynamics $p(\cdot|s, a)$ and the reward function $r(a, s)$ are unknown
  Information available at time t under π:

$$s_1^\pi, a_1^\pi, r_1(a_1^\pi, s_1^\pi), \ldots, s_{t-1}^\pi, a_{t-1}^\pi, r_{t-1}(a_{t-1}^\pi, s_{t-1}^\pi), s_t^\pi$$

- Objective: maximise the cumulative reward

$$\sum_{t=1}^{T} \mathbb{E}[r_t(s_t^\pi, a_t^\pi)] \quad \text{or} \quad \sum_{t=1}^{\infty} \lambda^t \mathbb{E}[r_t(s_t^\pi, a_t^\pi)]$$

Regret



- Difference between the cumulative reward of an "Oracle" policy and that of agent π
- Regret quantifies the price to pay for learning!
- Exploration vs. exploitation trade-off: we need to probe all actions to play the best later ...

Bellman's equation

Objective: max the average discounted reward $\sum_{t=1}^{\infty} \lambda^t \mathbb{E}[r(s_t^\pi, a_t^\pi)]$
Assume the transition probabilities and the reward function are known

- Value function: maps the initial state s to the corresponding maximum reward $v(s)$
- Bellman's equation:

$$v(s) = \max_{a \in A} \left[r(s,a) + \lambda \sum_j p(j|s,a) v(j) \right]$$

- Solve Bellman's equation. The optimal policy is given by:

$$a^\star(s) = \arg\max_{a \in A} \left[r(s,a) + \lambda \sum_j p(j|s,a) v(j) \right]$$

A first RL algorithm: Q-learning

What if the transition probabilities and the reward function are unknown?

- Q-value function: the max expected reward starting from state s and playing action a:

  $$Q(s,a) = r(s,a) + \lambda \sum_j p(j|s,a) \max_{b \in A} Q(j,b)$$

  Note that: $v(s) = \max_{a \in A} Q(s,a)$
- Algorithm: update the Q-value estimate sequentially so that it converges to the true Q-value

Q-learning

1. **Initialisation:** select $Q \in \mathbb{R}^{S \times A}$ arbitrarily, and s_0
2. **Q-value iteration:** at each step t, select action a_t (each state-action pair must be selected infinitely often)
   Observe the new state s_{t+1} and the reward $r(s_t, a_t)$
   Update $Q(s_t, a_t)$:

$$\begin{aligned} Q(s_t, a_t) :=& Q(s_t, a_t) \\ &+ \alpha_t \left[r(s_t, a_t) + \lambda \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right] \end{aligned}$$

Q-learning: demo

The crawling robot ...

https://www.youtube.com/watch?v=2iNrJx6IDEo

3. Schedule of the course

Part 1 Introduction and Elements of Probability

Part 2 Markov Decision Processes and Dynamic Programming

Part 3 The RL problems: definitions and performance metrics for RL agents

Part 4 Policy evaluation: Monte Carlo and TD learning methods

Part 5 SARSA and Q learning algorithms

Part 6 Learning with function approximation (Deep RL)

Part 7 Policy gradient algorithms

Part 8 Actor-critic algorithms

Part 9 Exploration in RL

Lecture 14: Summary of the course