Lecture 1: Introduction to Reinforcement Learning

Hado van Hasselt

Outline

Class Information

- Thursdays 9:15 to 11:00am
- Website:
  `http://hadovanhasselt.wordpress.com/2016/01/12/ucl-course/`
- Group:
  `http://groups.google.com/group/csml-advanced-topics`
- {hado,modayil}@google.com
- TA: Zbigniew Wojna (zbigniewwojna@gmail.com)

Assessment

- Assessment will be 50% coursework, 50% exam
- Coursework
  - Assignment A: RL problem
  - Assignment B: Kernels problem
  - Assessment $= max(assignment1, assignment2)$
- Examination
  - A: 3 RL questions
  - B: 3 kernels questions
  - Answer any 3 questions

Textbooks

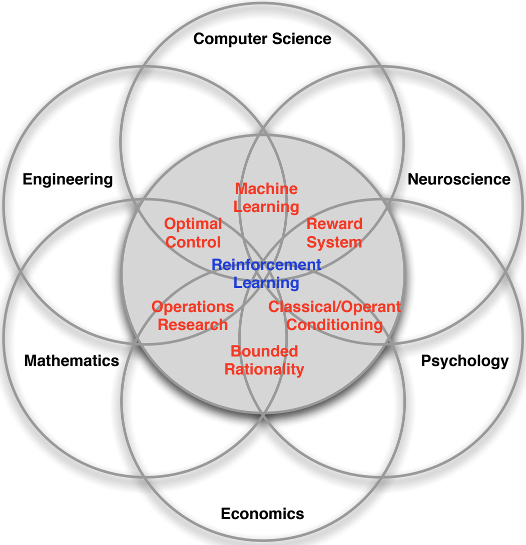
- An Introduction to Reinforcement Learning, Sutton & Barto
  - MIT Press, 1998
  - $\sim$ 30 pounds
  - Available free online!
  - `http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html`
- Algorithms for Reinforcement Learning, Szepesvari
  - Morgan and Claypool, 2010
  - $\sim$ 20 pounds
  - Available free online!
  - `http://www.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf`

What is Reinforcement Learning?

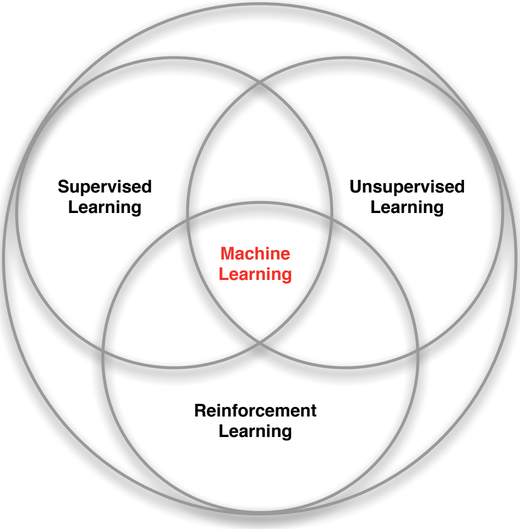
- Science of learning to make decisions, from experience
- This requires us to think about
    - ...predicting (long-term) consequences of actions
    - ...time
    - ...gathering experience
    - ...dealing with uncertainty
- Huge potential applicability

$$\text{RL} = \text{AI?}$$

Many Faces of Reinforcement Learning



Branches of Machine Learning



Characteristics of Reinforcement Learning

How does reinforcement learning differ from other machine learning paradigms?

- No supervision, only a reward signal
- Feedback is often delayed, not instantaneous
- Time really matters (sequential, non-i.i.d data)
- Agent's actions affect the subsequent data it receives

Examples of Reinforcement Learning

- Fly stunt manoeuvres in a helicopter
- Defeat the world champion at Backgammon
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Playing Atari games better than humans

Helicopter Manoeuvres

Atari

Rewards

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

$$R_{t+1} + R_{t+2} + R_{t+3} + \ldots$$

Reinforcement learning is based on the reward hypothesis

Definition (Reward Hypothesis)

All goals can be formalized as the outcome of maximizing a cumulative reward

Do you agree?

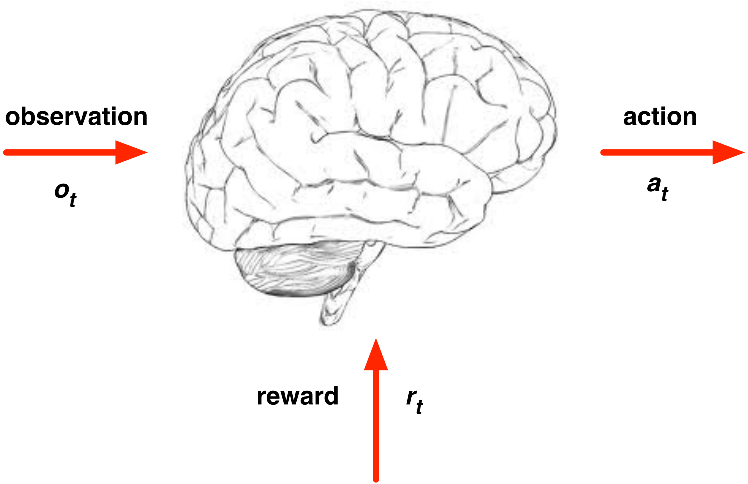
Examples of Rewards

- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - −ve reward for crashing
- Defeat the world champion at Backgammon
  - +/−ve reward for winning/losing a game
- Manage an investment portfolio
  - +ve reward for each $ in bank
- Control a power station
  - +ve reward for producing power
  - −ve reward for exceeding safety thresholds
- Make a humanoid robot walk
  - +ve reward for forward motion
  - −ve reward for falling over
- Play many different Atari games better than humans
  - +/−ve reward for increasing/decreasing score

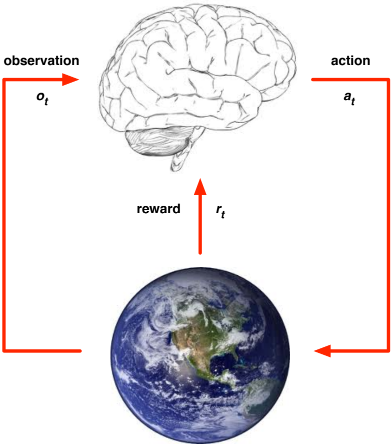
Sequential Decision Making

- Goal: *select actions to maximise 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)

Agent and Environment



Agent and Environment



- 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
  - Emits scalar reward R_t
- Rewards could be intrinsic (The agent defines its goals)

History and State

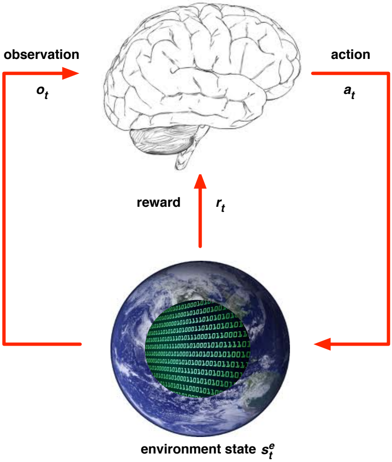
- A history is a sequence of observations, actions, rewards

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

- i.e. all observable variables up to time t
- i.e. the sensorimotor stream of a robot
- State is the information used to determine what happens next
- Formally, state is a function of the history:

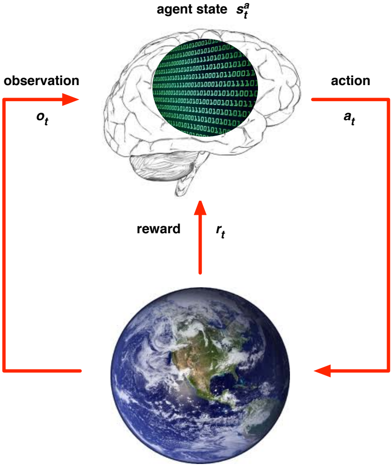
$$S_t = f(H_t)$$

Environment State



- The environment state S_t^e is the environment's private representation
- i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if S_t^e is visible, it may contain irrelevant information

Agent State



- The agent state is the agent's internal representation
- 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:

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

Information 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

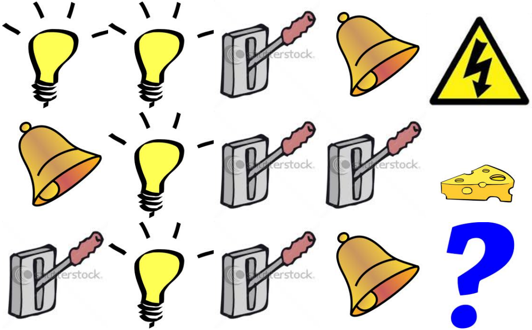
$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$

- "The future is independent of the past given the present"

$$H_t \rightarrow S_t \rightarrow H_{t+1}$$

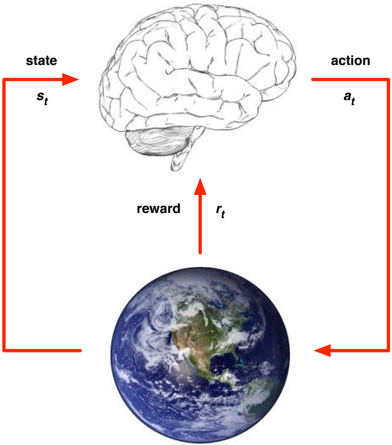
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future
- The environment state S_t^e is Markov
- The history H_t is Markov

Rat Example



- What if agent state = last 3 items in sequence?
- What if agent state = counts for lights, bells and levers?
- What if agent state = complete sequence?

Fully Observable Environments



Full observability: agent directly observes environment state

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

- Agent state = environment state = information state
- This is a Markov decision process (MDP)

Partially Observable Environments

- Formally, MDPs fulfill

$$\mathbb{P}[S_{t+1}, R_{t+1} \mid S_t, A_t] = \mathbb{P}[S_{t+1}, R_{t+1} \mid H_t, A_t]$$

- Partial observability: agent gets partial information
  - A robot with camera vision isn't told its absolute location
  - A poker playing agent only observes public cards
- Now observation is not Markov
- Formally this is a partially observable Markov decision process (POMDP)

Partially Observable Environments

- Agent can construct a state representation S_t^a, e.g.
  - Last observation: $S_t^a = O_t$
  - Complete history: $S_t^a = H_t$
  - Beliefs of environment state: $S_t^a = (\mathbb{P}[S_t^e = S^1], ..., \mathbb{P}[S_t^e = S^n])$
  - Recurrent neural network: $S_t^a = f(S_{t-1}^a, O_t)$
- Constructing a Markov agent state may not be feasible
- This is the common case!
- In practice, 'Markov' is not viewed as boolean

Major Components of an RL Agent

- An RL agent may include one or more of these components:
  - Policy: agent's behaviour function
  - Value function(s): predictions about the future
    (Typically cumulative reward, but can be more general)
  - Model: agent's representation of the environment

Policy

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

Value Function

- Value function is the expected future reward

$$v_\pi(s) = \mathbb{E}\left[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots \mid S_t = s, \pi\right]$$

- Used to evaluate the goodness/badness of states
- Can be used to select between actions
- $\gamma \in [0, 1]$ is called the discount factor
  - Trades off importance of immediate vs long-term rewards

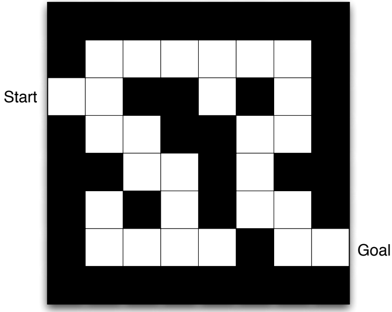
Example: Value Function in Atari

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.

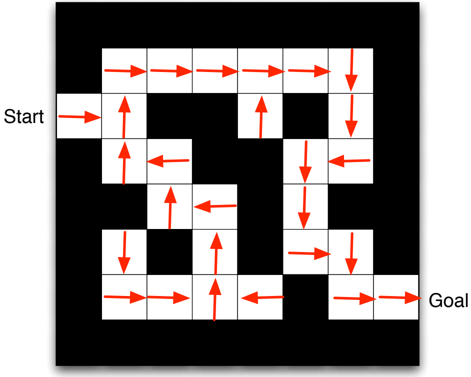
$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}_{ss'}^{a} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s'\right]$$

Maze Example



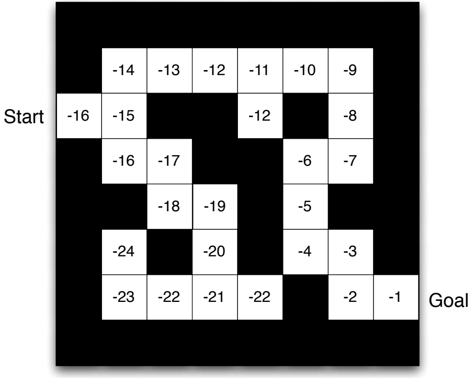
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze Example: Policy



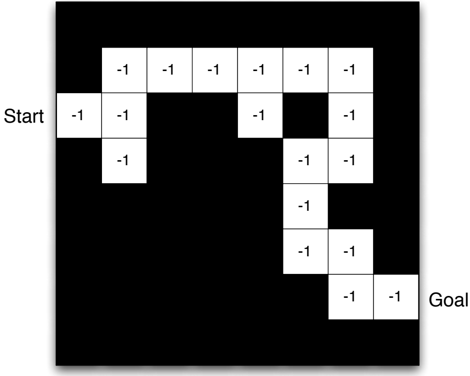
- Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value Function



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

Maze Example: Model



- Grid layout represents transition model $\mathcal{P}^a_{ss'}$
- Numbers represent immediate reward $\mathcal{R}^a_{ss'}$ from each state s (same for all a and s' in this case)

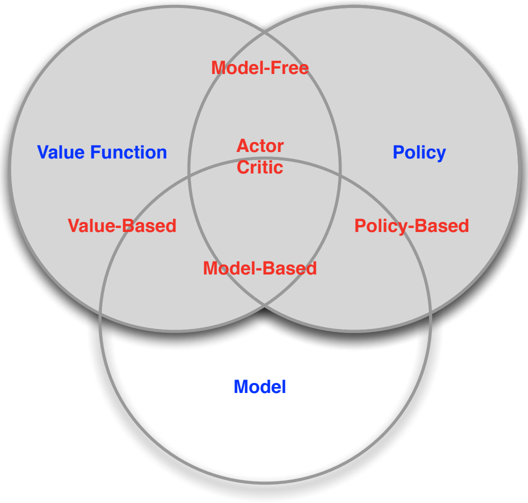
Categorizing RL agents (1)

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

Categorizing RL agents (2)

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

RL Agent Taxonomy

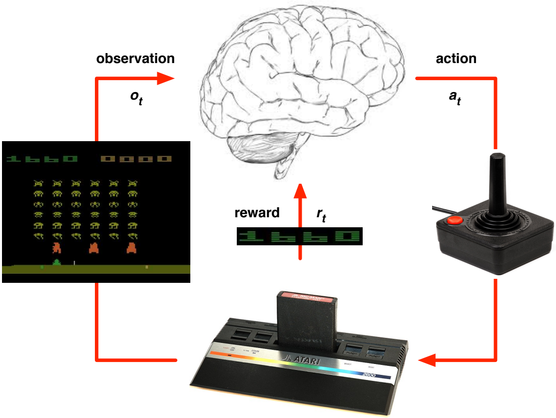


Learning and Planning

Two fundamental problems in sequential decision making

- Reinforcement Learning:
  - The environment is initially unknown
  - The agent interacts with the environment
- Planning:
  - A model of the environment is known
  - The agent performs computations with its model (without any external interaction)
  - a.k.a. deliberation, reasoning, introspection, pondering, thought, search

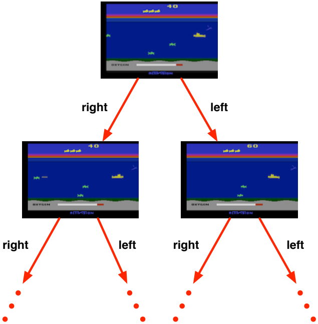
Atari Example: Reinforcement Learning



- Rules of the game are unknown
- Learn directly from interactive game-play
- Pick actions on joystick, see pixels and scores

Atari Example: Planning

- Rules of the game are known
- Can query emulator
  - perfect model inside agent's brain
- If I take action *a* from state *s*:
  - what would the next state be?
  - what would the score be?
- Plan ahead to find optimal policy
  - e.g. tree search



Exploration and Exploitation (1)

- Reinforcement learning is like trial-and-error learning
- The agent should discover a good policy
- ...from its experiences of the environment
- ...without losing too much reward along the way

Exploration and Exploitation (2)

- *Exploration* finds more information about the environment
- *Exploitation* exploits known information to maximise reward
- It is usually important to explore as well as exploit

Examples

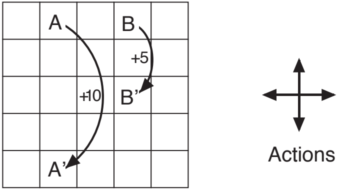
- Restaurant Selection
  - Exploitation Go to your favourite restaurant
  - Exploration Try a new restaurant
- Online Banner Advertisements
  - Exploitation Show the most successful advert
  - Exploration Show a different advert
- Oil Drilling
  - Exploitation Drill at the best known location
  - Exploration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

Prediction and Control

- Prediction: evaluate the future
  - Given a policy
- Control: optimize the future
  - Find the best policy
- These are strongly related:

$$\pi_*(s) = \underset{\pi}{\operatorname{argmax}}\ v_\pi(s)$$

Gridworld Example: Prediction



(a)

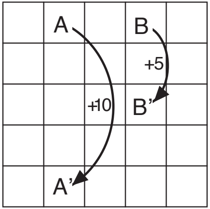
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

Reward is -1 when bumping into a wall, $\gamma = 0.9$

What is the value function for the uniform random policy?

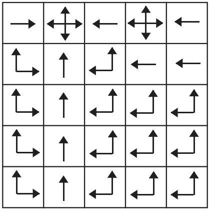
Gridworld Example: Control



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) V^*



c) π^*

What is the optimal value function over all possible policies?
What is the optimal policy?

Course Outline

- Part I: Elementary Reinforcement Learning
  1. Introduction to RL
  2. Exploration and Exploitation
  3. Markov Decision Processes
  4. Planning by Dynamic Programming
  5. Model-Free Prediction
  6. Model-Free Control
- Part II: Reinforcement Learning in Practice
  1. Value Function Approximation
  2. Policy Gradient Methods
  3. Integrating Learning and Planning
  4. Case study - RL in games