Reinforcement learning
Lecture 1: Introduction

Hado van Hasselt
hado@google.com

Advanced deep learning and reinforcement learning, UCL
January 18, 2018

Admin

- RL lectures: mostly Thursday 9-11am, some exceptions
- Check Moodle for updates
- Use Moodle for questions
- Grading: assignments
- Background material:
  Reinforcement Learning: An Introduction, Sutton & Barto 2018
  `http://incompleteideas.net/book/the-book-2nd.html`
  Background for this lecture: chapters 1 and 3

Outline

What is reinforcement learning?

Core concepts

Agent components

Challenges in reinforcement learning

What is reinforcement learning?

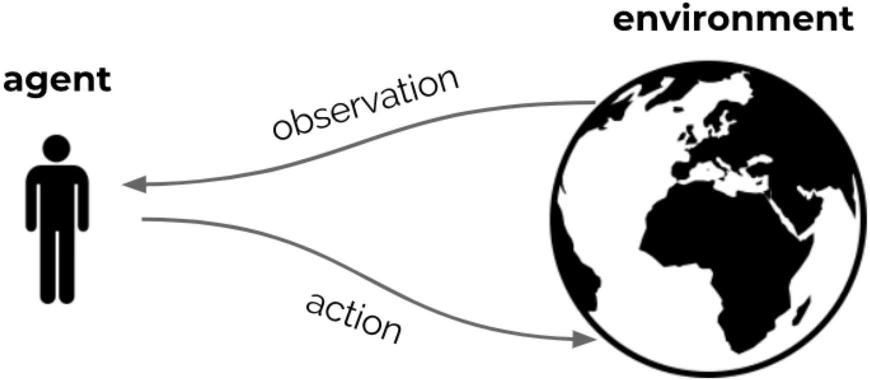
Motivation

- First, automation of repeated physical solutions
  - Industrial revolution (1750 - 1850) and Machine Age (1870 - 1940)
- Second, automation of repeated mental solutions
  - Digital revolution (1960 - now) and Information Age
- Next step: allow machines to find solutions themselves
  - AI revolution (now - ????)
- This requires learning autonomously how to make decisions

What is Reinforcement Learning?

- We, and other intelligent beings, learn by interacting with our 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 are goal-directed
- We can learn without examples of optimal behaviour

The Interaction Loop



What is Reinforcement Learning?

There are (at least) two distinct reasons to learn:

1. Find previously unknown solutions
   E.g., a program that can play Go better than any human, ever
2. Find solutions online, for unforeseen circumstances
   E.g., a robot that can navigate terrains that differ greatly from any expected terrain

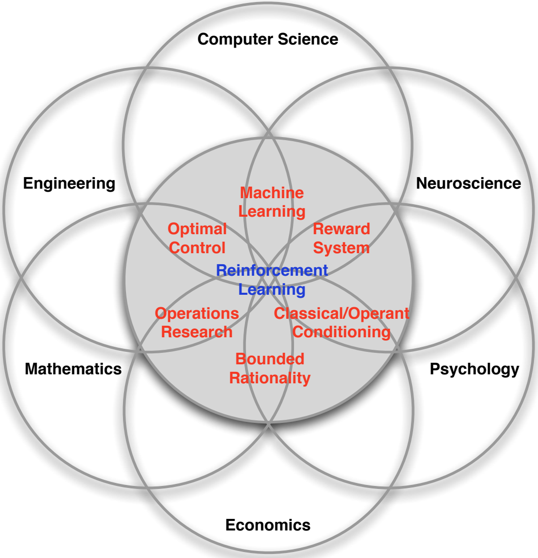
- Reinforcement learning seeks to provide algorithms for both cases
- Note that the second point is not (just) about generalization — it is about learning efficiently online, during operation

What is Reinforcement Learning?

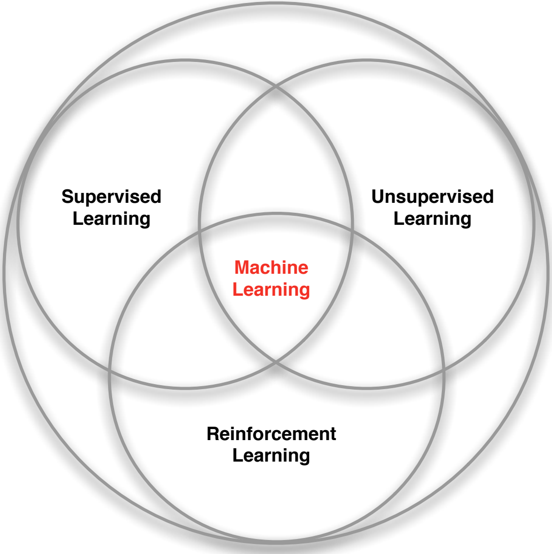
- Science of learning to make decisions from interaction
- This requires us to think about
  - ...time
  - ...(long-term) consequences of actions
  - ...actively gathering experience
  - ...predicting the future
  - ...dealing with uncertainty
- Huge potential scope

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

Related Disciplines



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 can be delayed, not instantaneous
- Time matters
- Earlier decisions affect later interactions

Examples of decision problems

- Examples:
  - Fly a helicopter
  - Manage an investment portfolio
  - Control a power station
  - Make a robot walk
  - Play video or board games
- These are all reinforcement learning problems
  (no matter which solution method you use)

Video

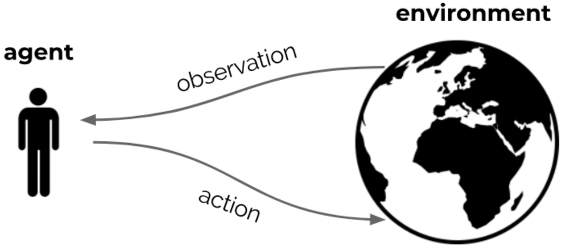
Atari

Core concepts

Core concepts of a reinforcement learning system are:

- Environment
- Reward signal
- Agent, containing:
  - Agent state
  - Policy
  - Value function (probably)
  - Model (optionally)

Agent and 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})

Rewards

- A reward R_t is a scalar feedback signal
- Indicates how well 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

Reinforcement learning is based on the reward hypothesis

Definition (Reward Hypothesis)

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

Do you agree?

Values

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

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

- Goal is then to maximize value, by picking suitable actions
- Rewards and values define desirability of a state or action (no supervised feedback)
- Note that returns and values can be defined recursively

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

Actions in sequential problems

- Goal: select actions to maximise value
- 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

Action values

- It is possible to condition the value on actions:

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

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

Agent components

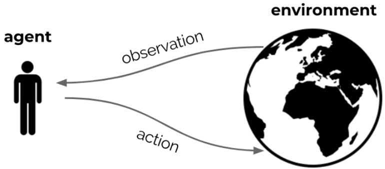
Agent components

- Agent state
- Policy
- Value function
- Model

State

- Actions depend on the state of the agent
- Both agent and environment may have an internal state
- In the simplest case, there is only one state (next lecture)
- Often, there are many different states — sometimes infinitely many
- The state of the agent generally differs from the state of the environment
- The agent may not even know the full state of the environment

Environment State



- The environment state is the environment's internal state
- It is not usually visible to the agent
- Even if it is visible, it may contain lots of irrelevant information

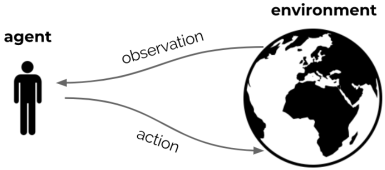
Agent State

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

$$\mathcal{H}_t = O_0, A_0, R_1, O_1, ..., O_{t-1}, A_{t-1}, R_t, O_t$$

- For instance, the sensorimotor stream of a robot
- This history can be used to construct an agent state S_t
- Actions depend on this state

Fully Observable Environments



Full observability:
Suppose the agent sees the full environment state

- observation = environment state
- The agent state could just be this observation:

$$S_t = O_t = \text{environment state}$$

- Then the agent is in a Markov decision process

Markov decision processes

Markov decision processes (MDPs) provide a useful mathematical framework

Definition

A decision process is Markov if

$$p(r, s \mid S_t, A_t) = p(r, s \mid \mathcal{H}_t, A_t)$$

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

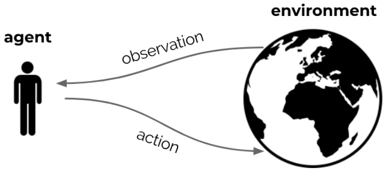
$$\mathcal{H}_t \to S_t \to \mathcal{H}_{t+1}$$

- Once the state is known, the history may be thrown away
- The environment state is typically Markov
- The history $\mathcal{H}_t$ is Markov

Partially Observable Environments

- Partial observability: The agent gets partial information
  - A robot with camera vision isn't told its absolute location
  - A poker playing agent only observes public cards
- Now the observation is not Markov
- Formally this is a partially observable Markov decision process (POMDP)
- The environment state can still be Markov, but the agent does not know it

Agent State



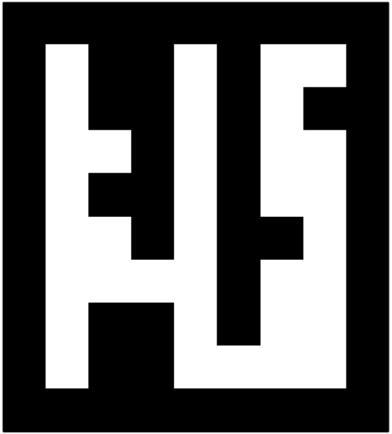
- The agent state is a function of the history
- The agent's action depends on its state
- For instance, $S_t = O_t$
- More generally:

$$S_{t+1} = f(S_t, A_t, R_{t+1}, O_{t+1})$$

  where f is a 'state update function'
- The agent state is typically much smaller than the environment state

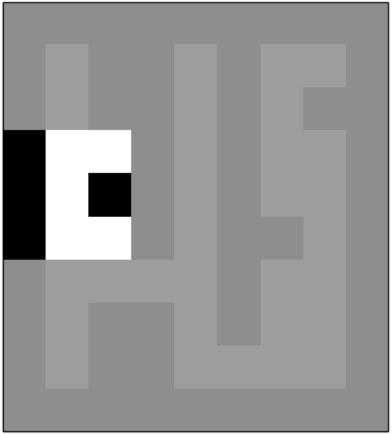
Agent State

The full environment state of a maze



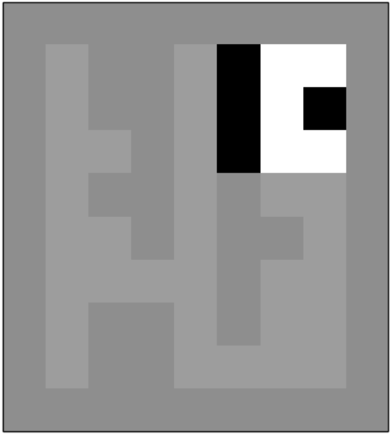
Agent State

A potential observation



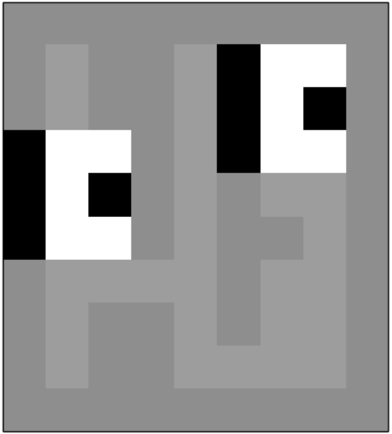
Agent State

An observation in a different location



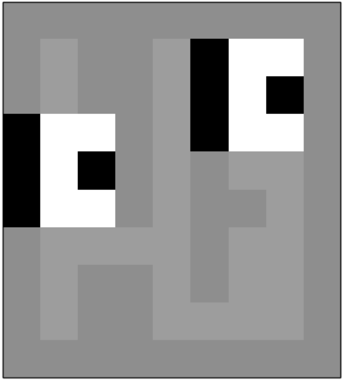
Agent State

The two observations are indistinguishable



Agent State

These two states are not Markov



How could you construct a Markov agent state in this maze (for any reward signal)?

Partially Observable Environments

- To deal with partial observability, agent can construct suitable state representations
- Examples of agent states:
  - Last observation: $S_t = O_t$ (might not be enough)
  - Complete history: $S_t = \mathcal{H}_t$ (might be too large)
  - Some incrementally updated state: $S_t = f(S_{t-1}, O_t)$
    (E.g., implemented with a recurrent neural network.)
    (Sometimes called 'memory'.)
- Constructing a Markov agent state may not be feasible; this is common!
- More importantly, the should state be contain enough informative for good policies, and/or good value predictions

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: $\pi(A|S) = p(A|S)$

Agent components

Agent components

- Agent state
- Policy
- Value function
- Model

Value Function

- The actual value function is the expected return

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}\left[G_t \mid S_t = s, \pi\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s, \pi\right]
\end{aligned}
$$

- We introduced a discount factor $\gamma \in [0, 1]$
  - Trades off importance of immediate vs long-term rewards
- The value depends on a policy
- Can be used to evaluate the desirability of states
- Can be used to select between actions

Value Functions

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

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

  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:

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

  This does not depend on a policy
- We heavily exploit such equalities, and use them to create algorithms

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

Agent components

Agent components

- Agent state
- Policy
- Value function
- Model

Model

- A model predicts what the environment will do next
- E.g., $\mathcal{P}$ predicts the next state

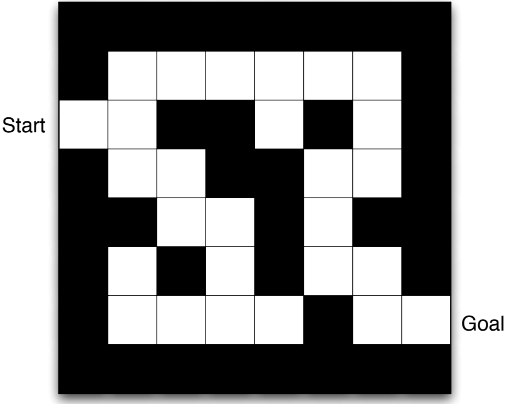
$$\mathcal{P}(s, a, s') \approx p\left(S_{t+1} = s' \mid S_t = s, A_t = a\right)$$

- E.g., $\mathcal{R}$ predicts the next (immediate) reward

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

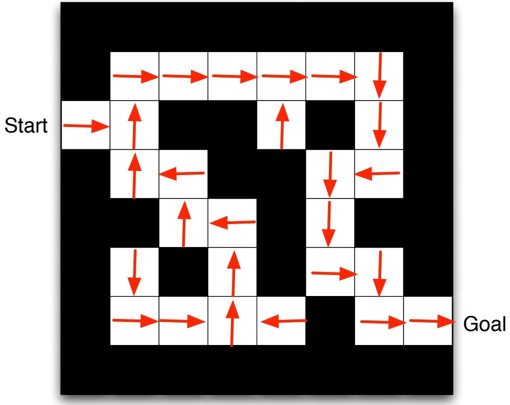
- A model does not immediately give us a good policy - we would still need to plan
- We could also consider stochastic (generative) models

Maze Example



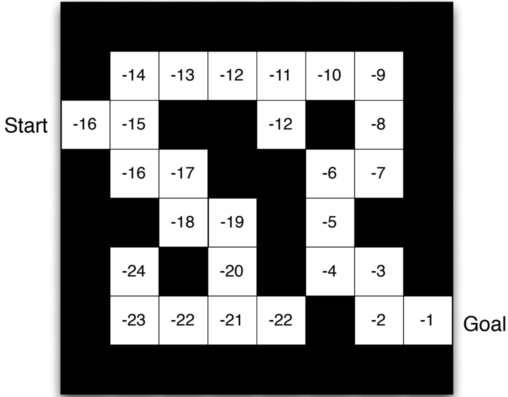
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze Example: Policy



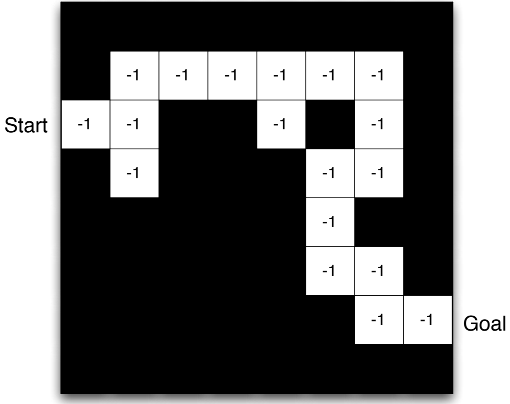
- Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value Function



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

Maze Example: Model



- Grid layout represents partial 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)

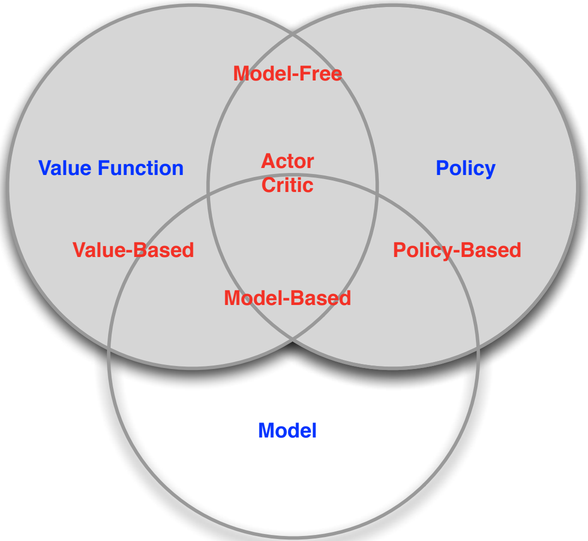
Categorizing agents

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

Categorizing agents

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

Agent Taxonomy



Challenges in reinforcement learning

Learning and Planning

Two fundamental problems in reinforcement learning

- Learning:
  - The environment is initially unknown
  - The agent interacts with the environment
- Planning:
  - A model of the environment is given
  - The agent plans in this model (without external interaction)
  - a.k.a. reasoning, pondering, thought, search, planning

Prediction and Control

- Prediction: evaluate the future (for a given policy)
- Control: optimize the future (find the best policy)
- These are strongly related:

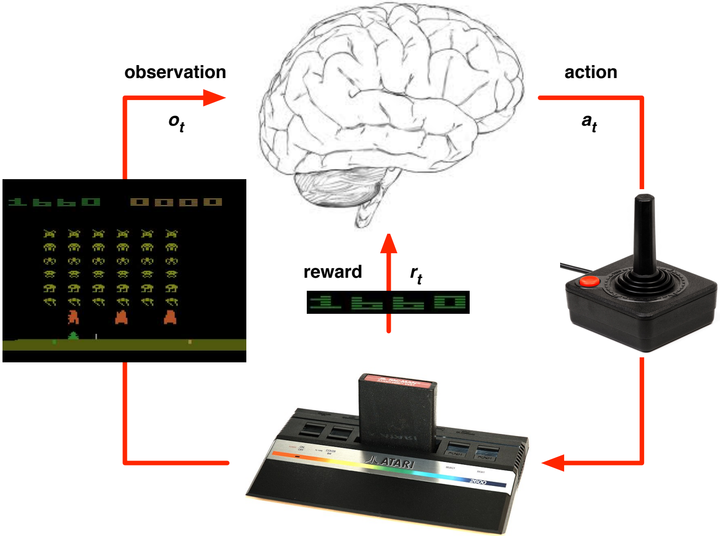
$$\pi_*(s) = \underset{\pi}{\operatorname{argmax}}\ v_\pi(s)$$

- If we could predict everything do we need anything else?

Learning the components of an agent

- All components are functions
  - Policies map states to actions
  - Value functions map states to values
  - Models map states to states and/or rewards
  - State updates map states and observations to new states
- We could represent these functions as neural networks, then use deep learning methods to optimize these
- Take care: we often violate assumptions from supervised learning (iid, stationarity)
- Deep reinforcement learning is a rich and active research field
- (Current) neural networks are not always the best tool (but they often work well)

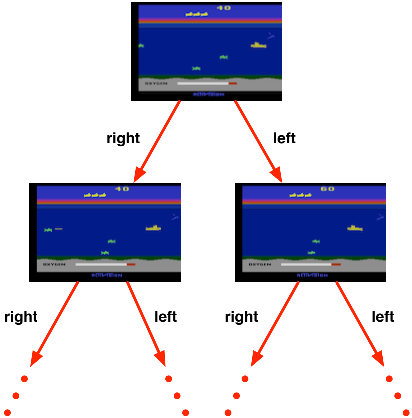
Atari Example: Reinforcement Learning



- 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
- 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
- Later versions add noise, to break algorithms that rely on determinism



Exploration and Exploitation

- We learn by trial and error
- The agent should discover a good policy
- ...from new experiences
- ...without sacrifycing too much reward along the way

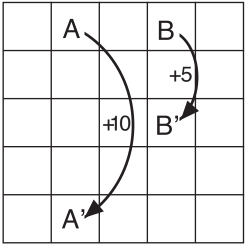
Exploration and Exploitation

- Exploration finds more information
- Exploitation exploits known information to maximise reward
- It is important to explore as well as exploit
- This is a fundamental problem that does not occur in supervised learning

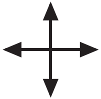
Examples

- Restaurant Selection
  - Exploitation: Go to your favourite restaurant
  - Exploration: Try a new restaurant
- 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

Gridworld Example: Prediction



(a)



Actions

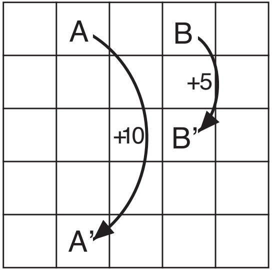
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

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

What is the value function for the uniform random policy?

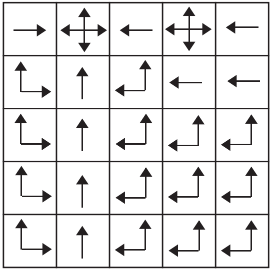
Gridworld Example: Control



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

- In this course, we discuss how to learn by interaction
- The focus is on understanding core principles and learning algorithms

Topics include

- Exploration, in bandits and in sequential problems
- Markov decision processes, and planning by dynamic programming
- Model-free prediction and control (e.g., Q-learning)
- Policy-gradient methods
- Challenges in deep reinforcement learning
- Integrating learning and planning
- Guest lectures by Vlad Mnih and David Silver

Video

Locomotion