COMP219:
Artificial Intelligence

Lecture 27: Reinforcement Learning

Revision Lecture

- Revision Lecture:
  - Date: Wednesday January 10, 2018
  - time: 10:00am
  - Location: CHAD-CHAD

Class Test 2

- 15th December, 15:00
- Again, based on first letter of last name:

  A-G → CHAD-ROTB

  H-Z → CTH-LTA
- What to study? **Everything except Prolog.**
- Example questions end of lecture

Overview

- Last time
  - Regression and classification with linear models; Non-parametric models: *K*-nearest neighbours

- Today
  - Reinforcement learning
    - General overview
    - N-armed bandit problem and Gittins index

- Learning outcomes covered today:

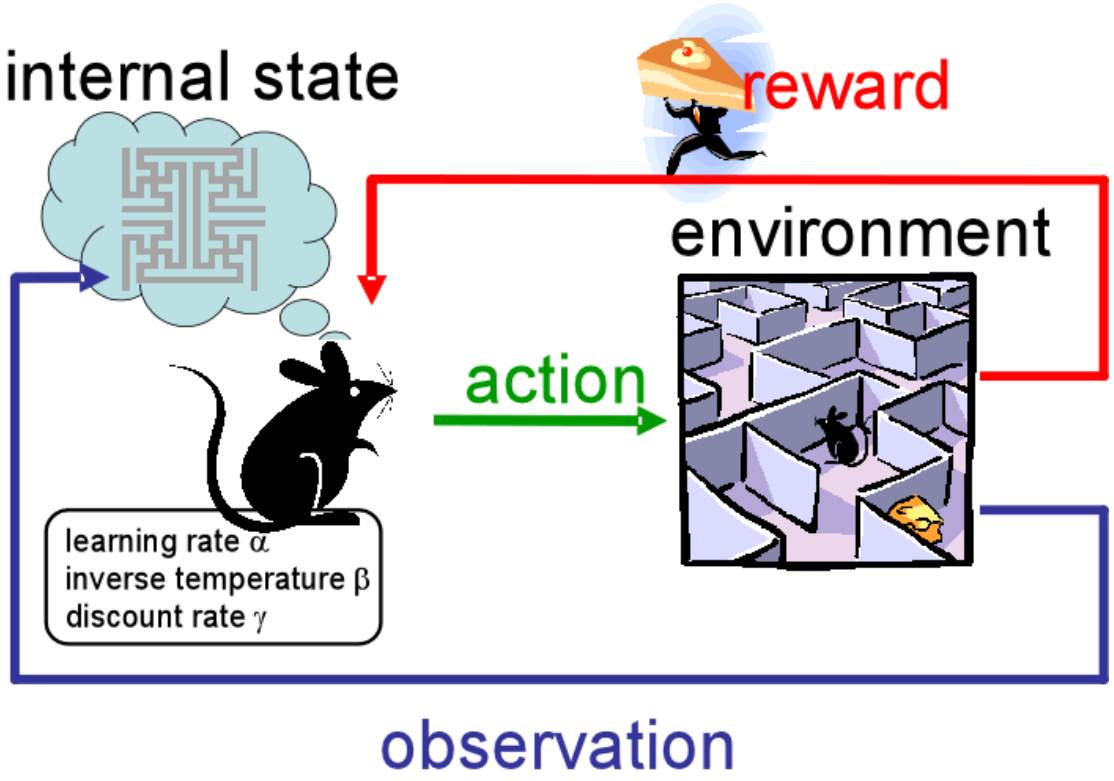
  Identify or describe the major approaches to learning in AI and apply these to simple examples

Reinforcement Learning (RL)

- A learning task:
  - agents learn what to do **without labelled examples**
  - learn from a series of reinforcements: **rewards** (and/or **punishments**)
- That is, RL is a problem, not one particular technique
  - but can 'approach a problem' by phrasing it as an RL problem
- Reinforcement learning has been studied by animal psychologists for over 60 years
  - Animals recognise pain and hunger as negative rewards, and pleasure and food intake as positive rewards
  - Foraging behaviour of bees
- Alan Turing proposed the reinforcement learning approach in 1948, but he thought it ineffective “at best a part of the teaching process”
- Arthur Samuel did the first successful work on machine learning (1959) which applied most of the modern reinforcement learning ideas

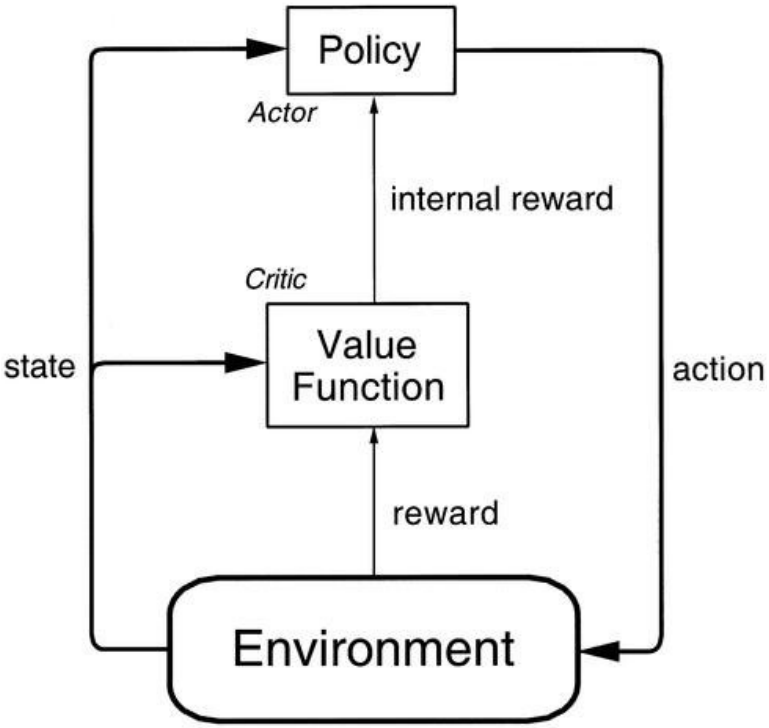
Reinforcement Learning Task

- The agent has to learn a policy that maps states to actions leading to maximum reward



Source: Julien Vitay

Reinforcement Learning Agent



Source: Julien Vitay

- Agent interacts with its environment and learns a policy which maximises the reward obtained from the environment (**optimal policy**)
- There are no labelled examples to learn from – the agent must discover whether an action is correct or not by observing rewards. Therefore it must try out all possibilities (exploration)
- Imagine playing a game whose rules you don't know: "you lose"
- The exploratory space can become huge

RL Agent Interacts with Environment

- RL agents need to interact with the environment. For example
  - *Games*: When a master chess player makes a move, the choice is informed both by planning (anticipating possible responses and counter- responses) and by immediate, intuitive judgments of the desirability of particular positions and moves
  - *Adaptive control*: An adaptive controller adjusts parameters of a control system in real time. The controller agent optimises the yield/cost/quality trade-off on the basis of specified margin costs without strictly following the set parameters originally suggested by engineers
  - *Mobile robots*: A mobile vacuum cleaning robot decides whether it should enter a new room in search of more dirt to clean or start trying to find its way back to its battery recharging station. It makes its decision based on how quickly and easily it has been able to find the recharger in the past

Elements of RL (I)

- **Policy π** defines the behaviour of the agent: which action to take in a given state to maximize the received reward in the long term
  - Stimulus-response rules or associations
  - Could be a simple lookup table or function, or need more extensive computation (e.g. search)
  - Can be probabilistic
- **Reward function r** defines the goal in a reinforcement learning problem: maps a state or action to a scalar number, the reward (or reinforcement). The RL agent's sole objective is to maximise the total reward it receives in the long run
  - Defines good and bad events
  - Cannot be altered by the agent but may inform change of policy
  - Can be probabilistic (expected reward)

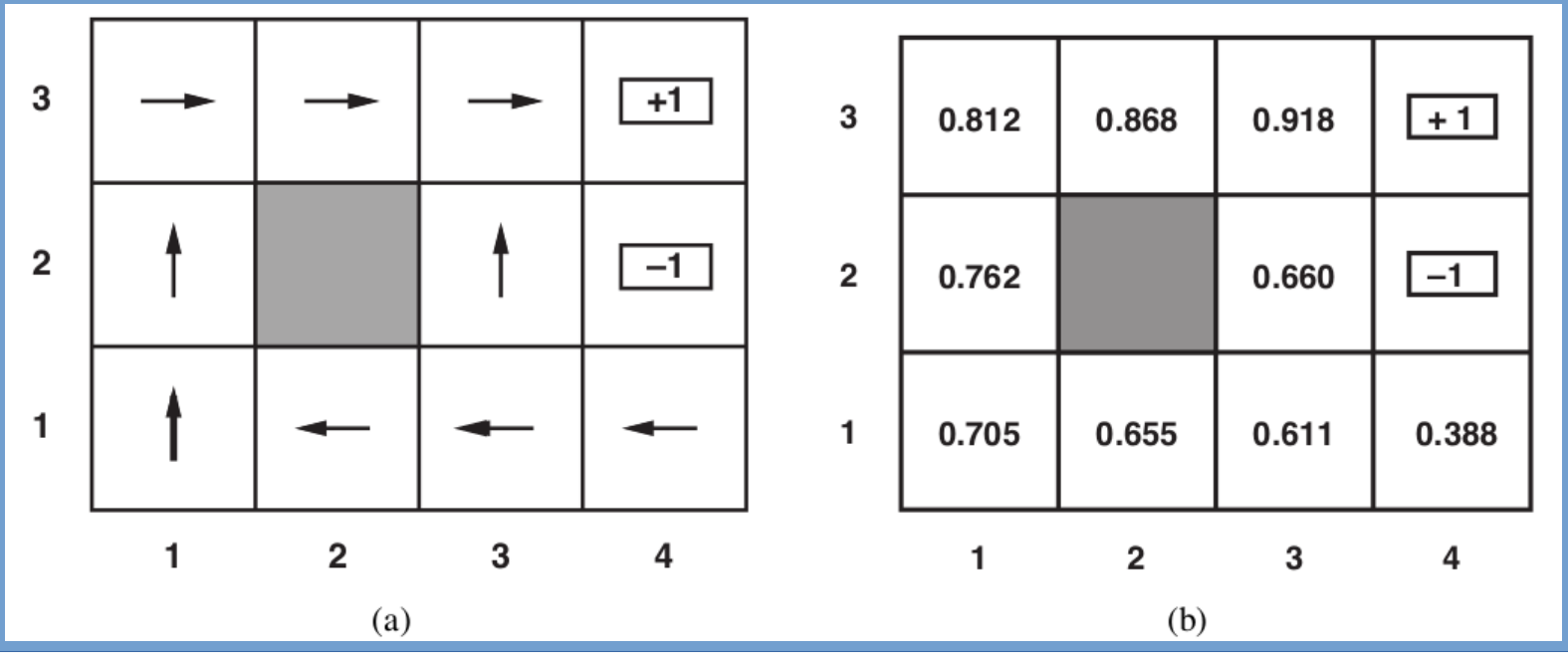
Elements of RL (II)

- **Value function** V defines the total amount of reward an agent can expect to accumulate over the future, starting from that state
  - What is good in the long run (reward function defines what is good *now*) considering the states (and rewards) that are likely to follow
  - A state may yield a **low reward** but have a **high value** (or the opposite)
  - e.g. immediate pain/pleasure vs. long term happiness
- **Transition model** M defines the transitions in the environment: action a taken in the state s_1 will lead to state s_2
  - Can be probabilistic

Elements of RL (II)

- **Value function** V defines the total amount of reward an agent can expect to accumulate over the future, starting from that state

Value function example.



Types of Reinforcement Learning

Reinforcement learning can be

- Passive – where the agent's policy is fixed and the task is to learn the utilities of states (or state-action pairs)
- Active – where the agent must also learn what to do, i.e. exploration

Passive Reinforcement Learning

- The agent's **policy π is fixed**: in state s it always executes $\pi(s)$
- Goal is to learn how good the policy is:
  - to **learn the value function $V^{\pi}(s)$**
- Agent does not know the reward function r or transition model M
- Agent executes a set of trials in the environment using its policy π
  - Starts in initial state s_0, experiences a sequence of states and rewards until it reaches a terminal state s_t
  - Agent uses information about rewards to learn the expected value $V^{\pi}(s_i)$ associated with each non-terminal state s_i

Active Reinforcement Learning

- A passive agent has a fixed policy determining behaviour, but an active agent must decide which actions to take…
- For instance ("model-based RL" or "adaptive dynamic programming"):
  - learn a complete model M with outcome probabilities for all actions
  - then learn the value function $\mathrm{V}(s)$
  - then, given the resulting V, decide which actions to take
- Issue: what if the learned model is incorrect…? Might perform sub-optimally!
- Active agent must trade-off between
  - **exploitation** (to maximise its reward),
  - **exploration** (to learn if there are better actions/states it has not found yet)
- **How to balance?** can't exploit all the time; can't explore all the time.

n-Armed Bandit Problem

- Model to reason about exploration vs exploitation
- A **one-armed bandit** is a slot machine:
  - A gambler can insert a coin, pull the lever and collect the winnings (if any)
- An **n-armed bandit** has *n* levers:
  - gambler must choose which lever to play on each successive time step…
  - he one that has paid off best?
  - Or the one that has not been tried?

n-Armed Bandit Problem cont'd

- *n*-armed bandit problem is a formal model for real problems in many domains
  - e.g. in marketing (which ad to show)
- Exploration is risky: uncertain payoffs
- But failure to explore means never discovering worthwhile actions
- To formulate an *n*-armed bandit problem properly, we must define what we mean by *optimal*

Gittins Index

- The **Gittins index** is a measure of the reward that can be achieved by a sequence of actions from the present state onwards with the probability that it will be terminated in the future
- *n*-armed bandit problem – it is possible to calculate a Gittins index for *n*-armed bandit machine:
- Gittins index = a function of the *number of times* a bandit has been played and *how much* it has paid out
- Indicates how worthwhile it is to invest more

Gittins, J.C. (1989). *Multi-armed bandit allocation indices*. Wiley-Interscience Series in Systems and Optimization. Chichester: John Wiley & Sons, Ltd. ISBN 0-471-92059-2.

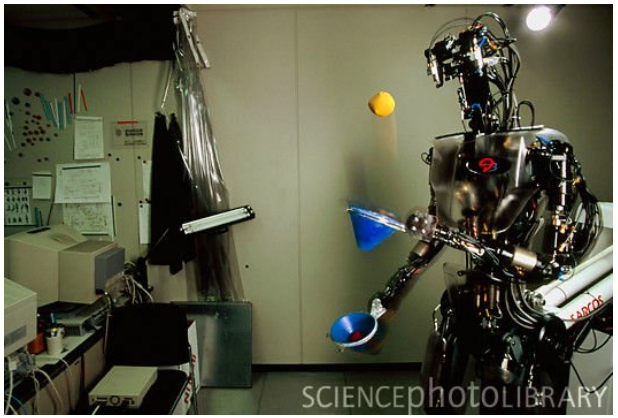
https://en.wikipedia.org/wiki/Gittins_index

RL Applications: Games

- It is very hard for a human to provide accurate and consistent evaluations of a large number of positions to train an evaluation function
- 1959 – Arthur Samuel applied RL to checkers
- 1992 – Gerald Tesauro's TD-GAMMON used RL techniques to find the optimal strategy to play backgammon: learn from self-play alone
- Recent successes:
  - Atari games: https://www.youtube.com/watch?v=TmPfTpjtdgg
  - Go
  - Poker

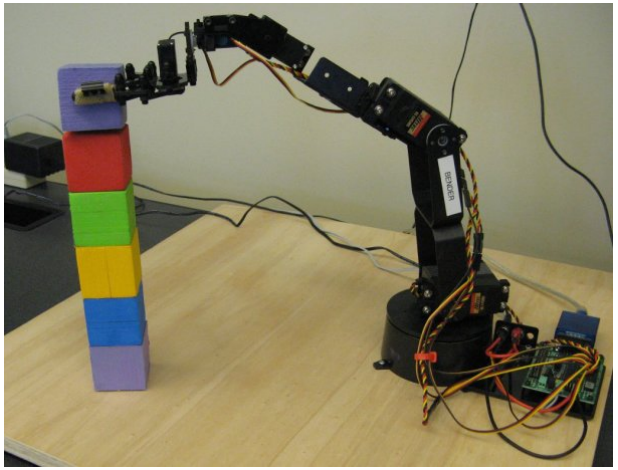
- Rewards may be fairly frequent (e.g. in table tennis, each point is a reward) or only at the end of the game (e.g. chess)
- The main problem for RL is that the reward (e.g. win or loss) could be delayed too much, e.g. a game that never ends

RL Applications: Robotics

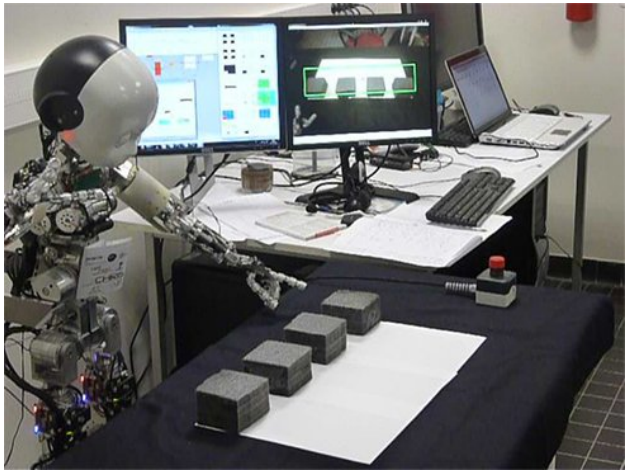


Motor control

Navigation and exploration



Sequence learning



Decision making

Source: Julien Vitay

Reinforcement Learning Possibilities

- Because of its potential for eliminating hand coding of control strategies, RL is one of the most active areas of machine learning research
- Applications in robotics promise to be especially valuable – will need methods for handling continuous, high-dimensional, partially observable environments in which successive behaviours may consist of millions of primitive actions

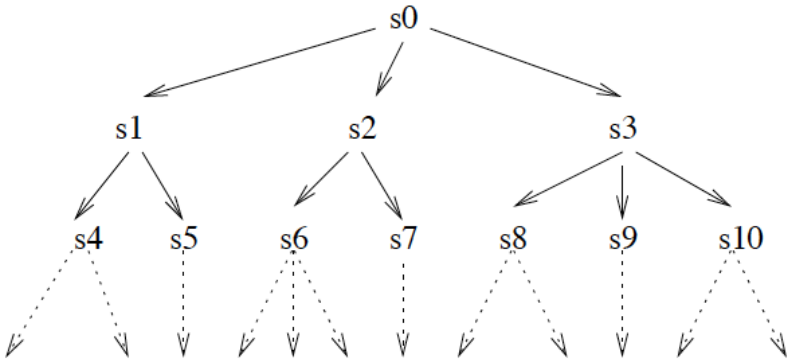
We have considered 3 types of learning

- Supervised learning
  - Agent learns a function from observing example input-output pairs
- Unsupervised learning
  - Learn patterns in the input without explicit feedback
  - Most common task is clustering
- Reinforcement learning
  - Learn from a series of reinforcements: rewards or punishments

- We note the existence of other approaches for addressing machine learning methods, but we conclude our study here

Class test 2 example questions

1. A search tree is given below.



A breadth-first search of the tree would output the nodes in the following order

- ☐ **A.** $s_0, s_1, s_2, s_3, s_4,$
- ☐ **B.** $s_0, s_1, s_4,$
- ☐ **C.** $s_0, s_2, s_6, s_3,$
- ☐ **D.** $s_4, s_5, s_6, s_7,$
- ☐ **E.** $s_0, s_1, s_2, s_4,$

Class test 2 example questions

2. Compare the advantages and disadvantages of depth-first search and breadth-first search.

Class test 2 example questions

3. Consider the following set of rules and facts in a rule-based system:

```
R1: IF hot AND smoky THEN ADD fire
R2: IF alarm_beeps THEN ADD smoky
R3: IF fire THEN DO switch_ sprinklers_on ADD sprinklers_on
F1: alarm_beeps;
F2: hot
```

Use backward chaining to say whether the goal "`switch_sprinklers_on`" can be satisfied or not

Class test 2 example questions

4. The following are a set of clauses in propositional logic.

1. $\neg q \vee \neg p \vee r$
2. $p \vee t$
3. q
4. s

By applying the resolution inference rule to some of these clauses a next possible step is:

☐ **A.** $\neg q \vee r \ \ [1, 2]$

☐ **B.** $q \vee r \vee t \ \ [1, 2]$

☐ **C.** $\neg q \vee \neg p \vee r \vee s \ \ [1, 4]$

☐ **D.** $p \vee q \vee t \ \ [2, 3]$

☐ **E.** $\neg q \vee r \vee t \ \ [1, 2]$

Class test 2 example questions

5. Which of the following is a learning model that summarises data with a set of parameters of fixed size (independent of the number of training examples)?

- ☐ **A.** Parametric model.
- ☐ **B.** Non-parametric model.
- ☐ **C.** Reinforcement model.
- ☐ **D.** Regular model.
- ☐ **E.** Irregular model.

Summary

- Reinforcement learning
  - Agent task, elements of RL
  - Passive vs active RL
  - N-armed bandit problem and Gittins index
  - Applications of RL
- Further reading RL:
  - R. S. Sutton, A. G. Barto: *Reinforcement Learning: An Introduction*. MIT Press, 1998 http://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html
  - Reinforcement Learning: State-of-the-Art. Editors: Wiering, Marco, van Otterlo, Martijn (Eds.) https://link.springer.com/book/10.1007%2F978-3-642-27645-3
- Further ML resources:
  - Russel & Norvig…!
  - Christopher Bishop. Pattern Recognition and Machine Learning
  - Goodfellow, Bengio & Courville. Deep Learning
  - Andrew Ng's Coursera course on Machine learning.

- Next time
  - Jan. 10th, 10am: revision lecture