

Reinforcement learning (COMP-767)

- Instructor: Doina Precup
- Email: dprecup@cs.mcgill.ca
- Class web page: <http://www.cs.mcgill.ca/~dprecup/courses/rl.html>

Outline

- Administrative issues
- What is reinforcement learning (RL)?
- Multi-arm bandits

Prerequisites

- Knowledge of a programming language (preferably Python)
- Probability, calculus, linear algebra; general comfort with math
- Previous exposure to machine learning (McGill courses: COMP-551, COMP-652)
- If in doubt about your background, contact Doina

Course material

- Required textbook: Sutton & Barto, Reinforcement learning: An Introduction, Second edition, 2018 (available online)
- Supplementary textbooks: Csaba Szepesvari, "Algorithms for Reinforcement Learning", Morgan and Claypool, 2010, and Dimitri Bertsekas and John Tsitsiklis, "Neuro-dynamic programming", Athena Scientific, 1997.
- Other required or suggested materials posted on the course web page
- Schedule posted on the web page; you MUST do the reading in order to really benefit from this course

Evaluation

- Project (30%): individual or in groups of 2-3; email a brief description to Doina by March 19; poster session April 11; report due by April 25
- Four assignments (50%)
- Assignments (individual or in groups of 2) consist of choosing either to do an experimental notebook or a short report on a paper or theory question. Submissions will be made publicly available. You should be able to answer a few questions or do a short presentation on the topic.
- In-class midterm exam (20%) - March 14 (crib sheet allowed)

What is Reinforcement Learning?

- Agent-oriented learning—learning by interacting with an environment to achieve a goal
 - more **realistic** and **ambitious** than other kinds of machine learning
- Learning by trial and error, with only delayed evaluative feedback (reward)
 - the kind of machine learning most like natural learning
 - learning that can tell for itself when it is right or wrong
- The beginnings of a *science of mind* that is neither natural science nor applications technology

A big success story:AlphaGo



ARTICLE

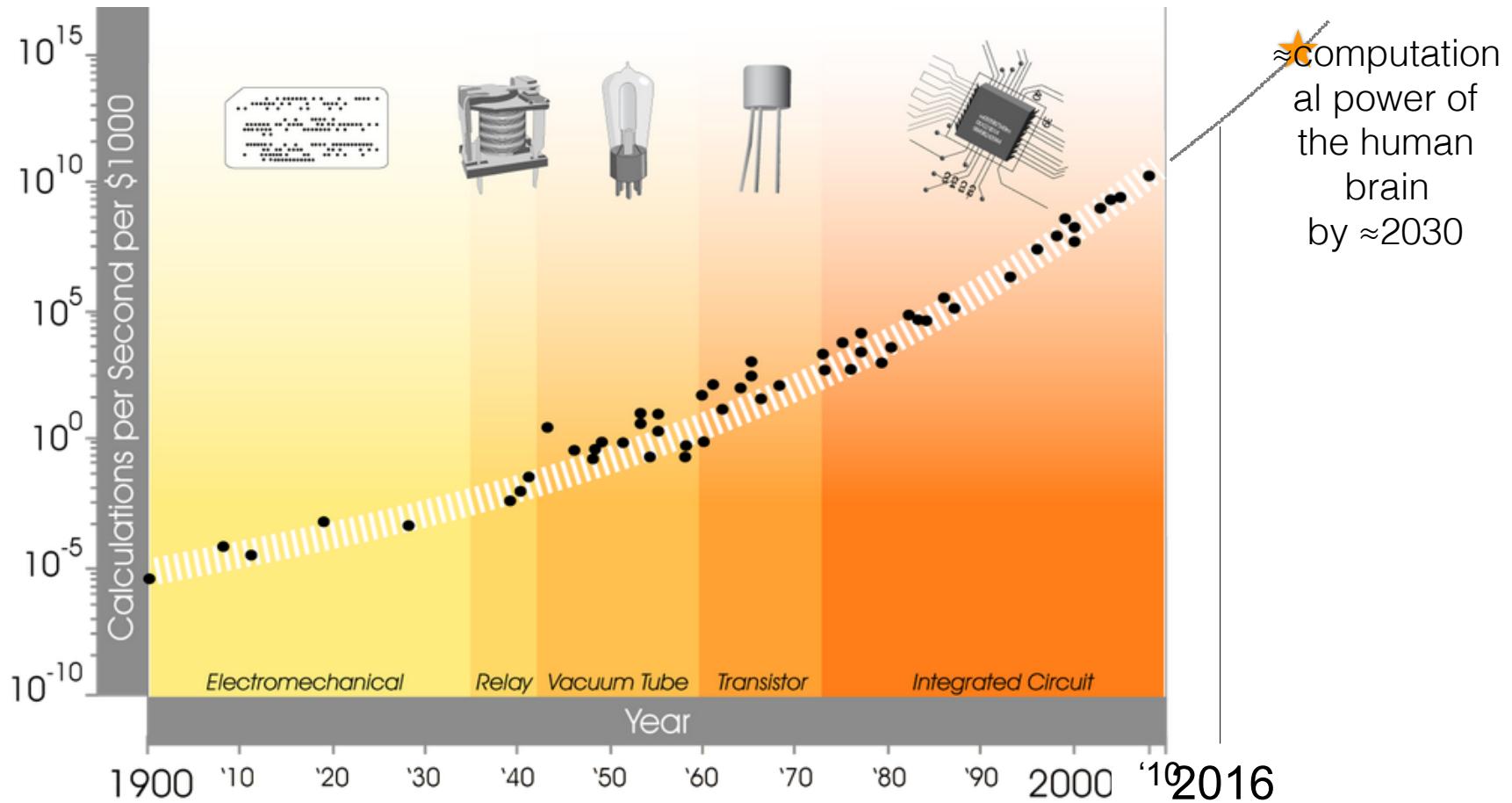
[doi:10.1038/nature16961](https://doi.org/10.1038/nature16961)

Mastering the game of Go with deep neural networks and tree search

David Silver^{1*}, Aja Huang^{1*}, Chris J. Maddison¹, Arthur Guez¹, Laurent Sifre¹, George van den Driessche¹, Julian Schrittwieser¹, Ioannis Antonoglou¹, Veda Panneershelvam¹, Marc Lanctot¹, Sander Dieleman¹, Dominik Grewe¹, John Nham², Nal Kalchbrenner¹, Ilya Sutskever², Timothy Lillicrap¹, Madeleine Leach¹, Koray Kavukcuoglu¹, Thore Graepel¹ & Demis Hassabis¹

The first AI
Go player to
defeat a human
(9 dan)
champion

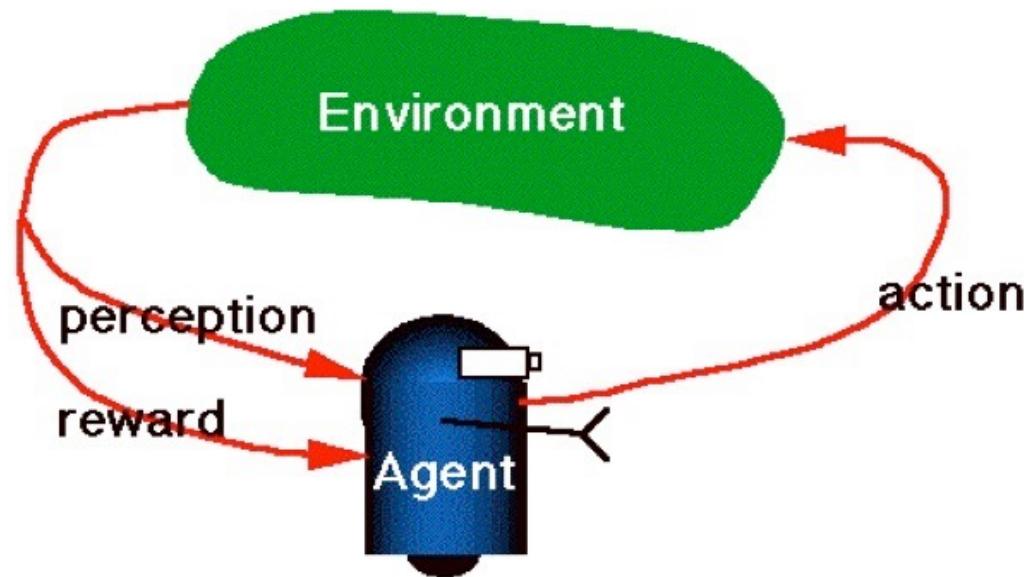
The computational revolution



Reinforcement Learning



Reward: Food or electric shock



Reward: Positive and negative numbers

- Learning by **trial-and-error**
- Numerical reward is often **delayed**

Contrast: Supervised Learning

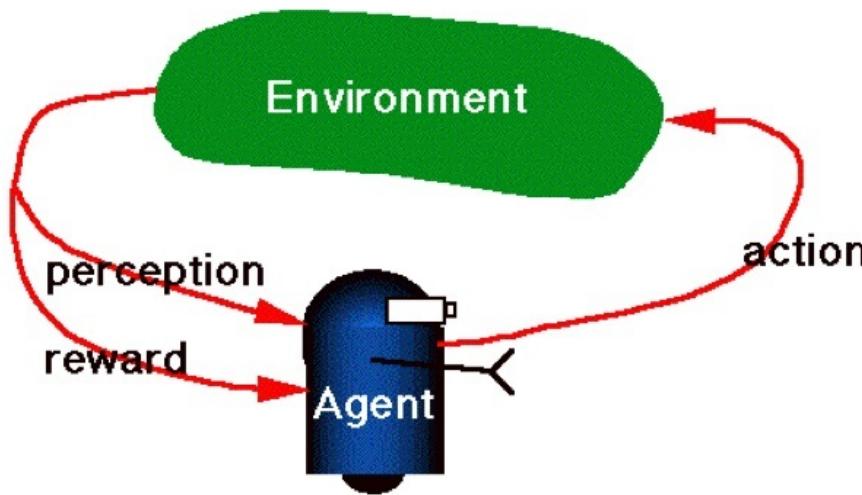
- Training experience: a set of *labeled examples* of the form $\langle x_1 x_2 \dots x_n, y \rangle$, where x_j are values for *input variables* and y is the *desired output*
- This implies the existence of a “teacher” who knows the right answers
- What to learn: A *function* mapping inputs to outputs which optimizes an objective function
- E.g. Face detection and recognition:



Contrast: Unsupervised learning

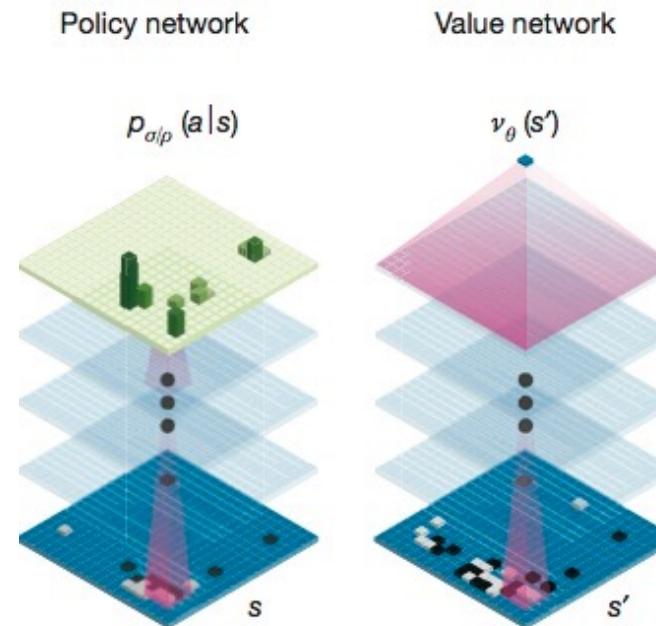
- Training experience: unlabelled data
- What to learn: interesting associations in the data
- E.g., clustering, dimensionality reduction, density estimation
- Often there is no single correct answer
- Very necessary, but significantly more difficult than supervised learning

Computational framework



- At every time step t , the agent perceives the *state* of the environment
- Based on this perception, it chooses an *action*
- The action causes the agent to receive a *numerical reward*
- Find a way of choosing actions, called a *policy* which *maximizes the agent's long-term expected return*

Example: AlphaGo



- Perceptions: state of the board
- Actions: legal moves
- Reward: +1 or -1 at the end of the game
- Trained by playing **games against itself**
- Invented **new ways of playing** which seem superior

Key Features of RL

- The learner is not told what actions to take, instead it finds out what to do by *trial-and-error search*

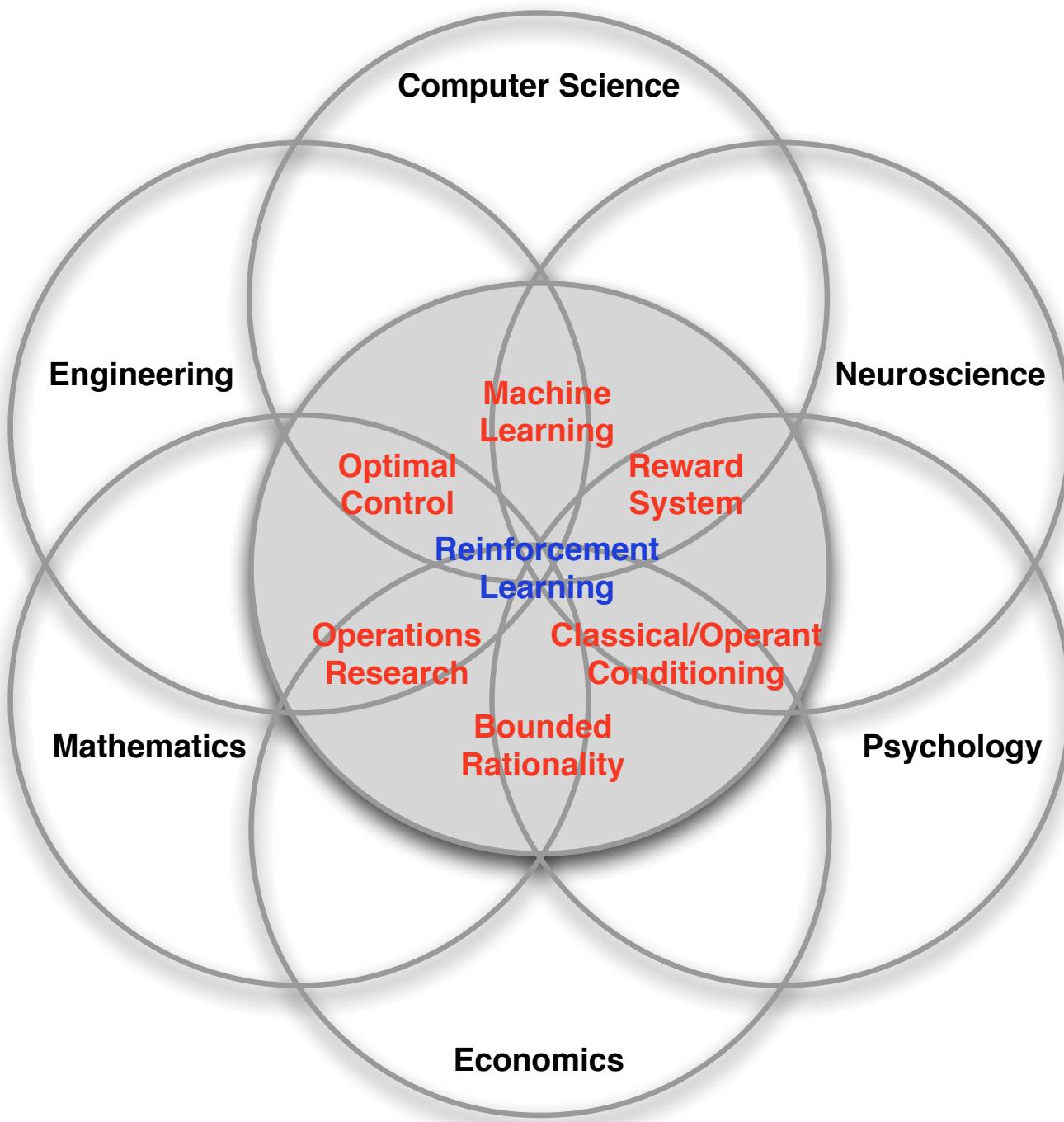
Eg. Players trained by playing thousands of simulated games, with no expert input on what are good or bad moves

- The environment is *stochastic*
- The *reward may be delayed*, so the learner may need to sacrifice short-term gains for greater long-term gains

Eg. Player might get reward only at the end of the game, and needs to assign credit to moves along the way

- The learner has to balance the need to *explore* its environment and the need to *exploit* its current knowledge

Eg. One has to try new strategies but also to win games

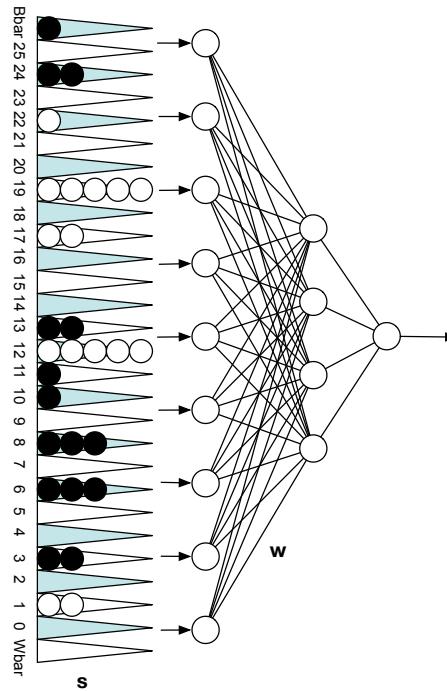
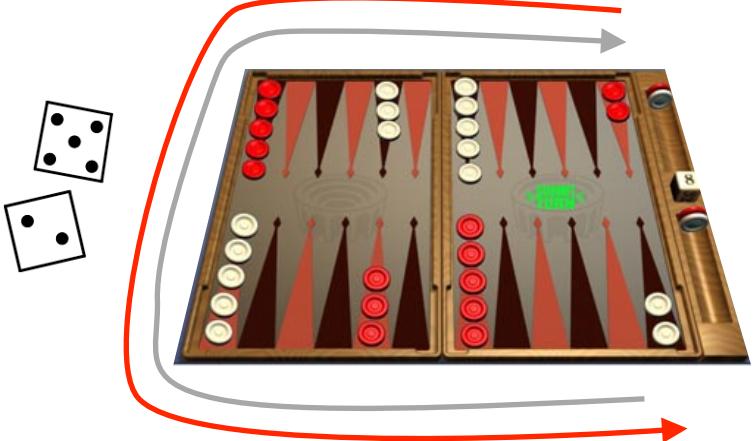


Some RL Successes

- Learned the world's best player of Backgammon (Tesauro 1995) 
- Learned acrobatic helicopter autopilots (Ng, Abbeel, Coates et al 2006+)
- Widely used in the placement and selection of advertisements and pages on the web (e.g., A-B tests)
- Used to make strategic decisions in *Jeopardy!* (IBM's Watson 2011)
- Achieved human-level performance on Atari games from pixel-level visual input, in conjunction with deep learning (Google Deepmind 2015)
- In all these cases, performance was better than could be obtained by any other method, and was obtained without human instruction

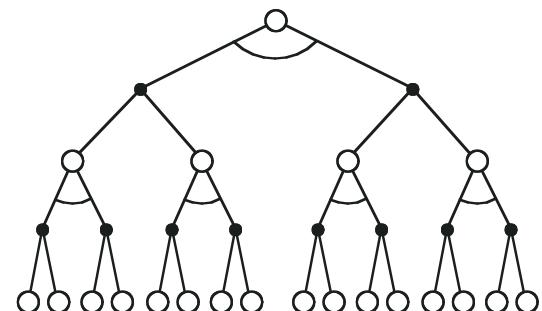
Example: TD-Gammon

Tesauro, 1992-1995



estimated state value
(≈ prob of winning)

Action selection
by a shallow search



Start with a random Network

Play millions of games against itself

Learn a value function from this simulated experience

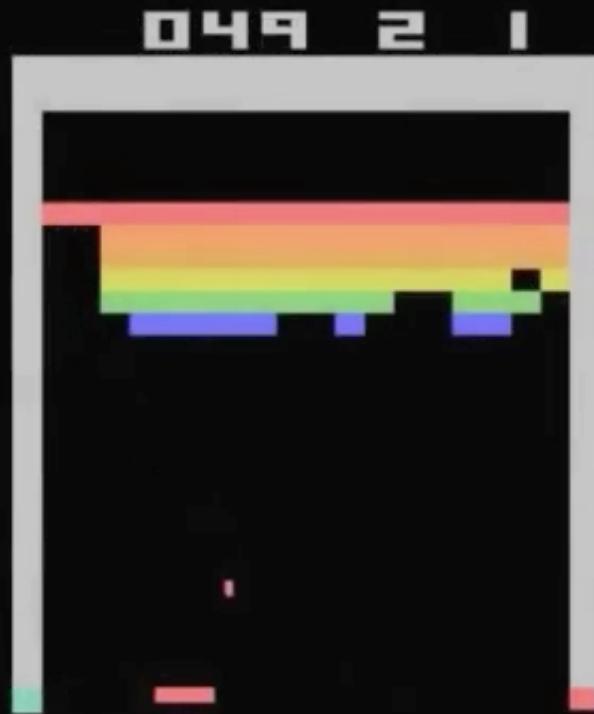
Six weeks later it's the best player of backgammon in the world

Originally used expert handcrafted features, later repeated with raw board positions

RL + Deep Learning Performance on Atari Games



Space Invaders



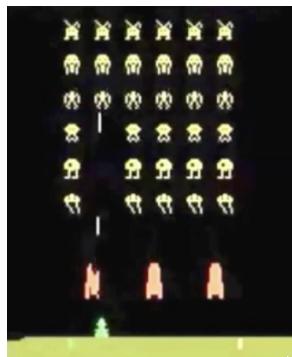
Breakout



Enduro

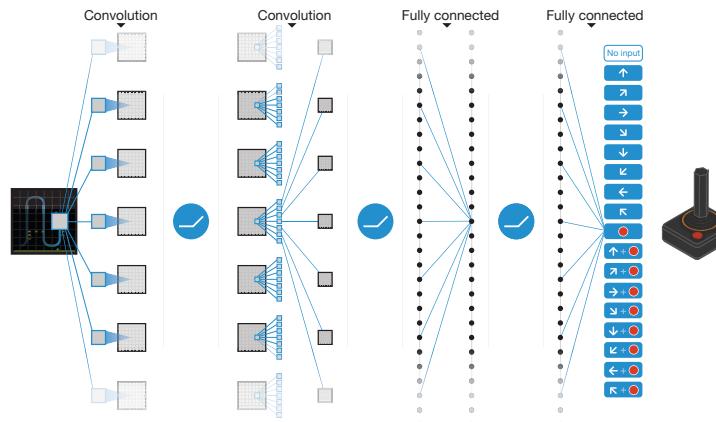
RL + Deep Learning, applied to Classic Atari Games

Google Deepmind 2015, Bowling et al. 2012



- Learned to play 49 games for the Atari 2600 game console, without labels or human input, from self-play and the score alone

mapping raw
screen pixels

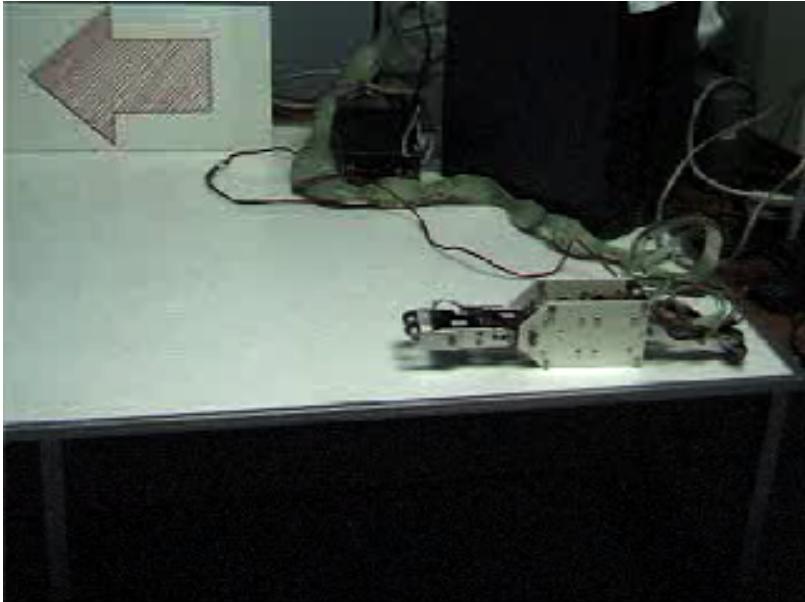


to predictions
of final score
for each of 18
joystick actions

- Learned to play better than all previous algorithms and at human level for more than half the games

Same learning algorithm applied to all 49 games!
w/o human tuning

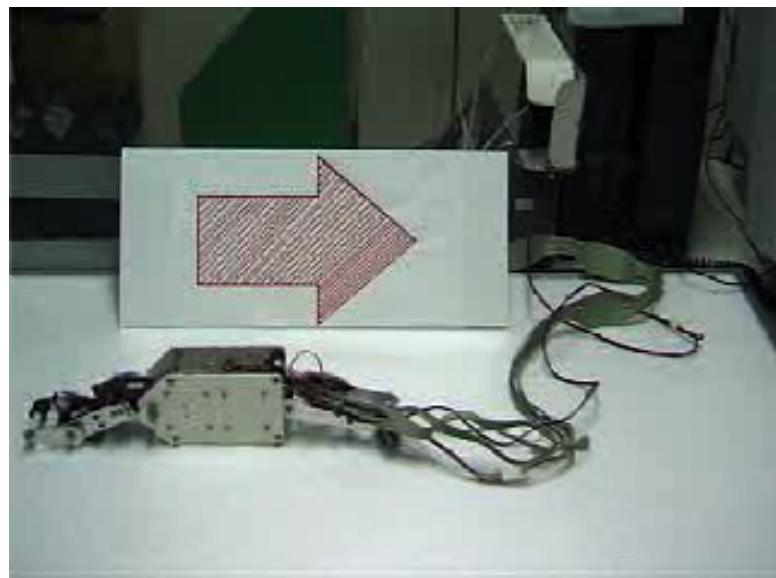
Example: Hajime Kimura's RL Robots



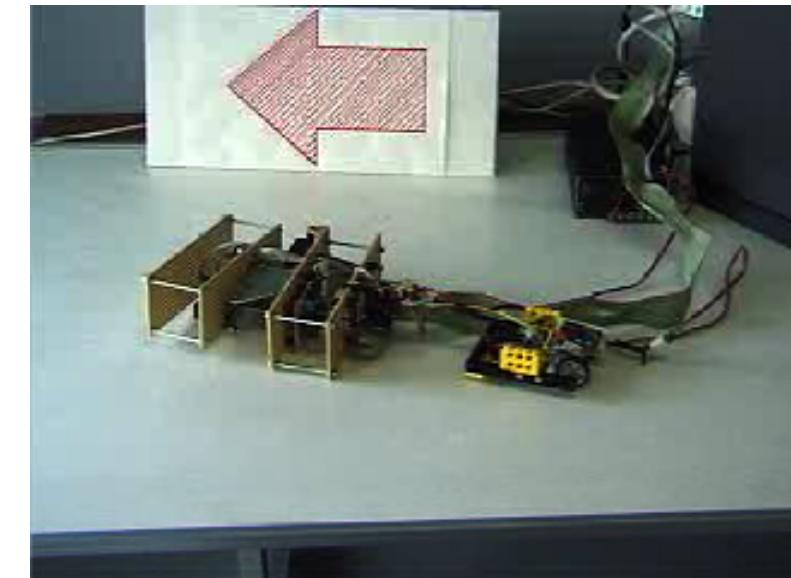
Before



After

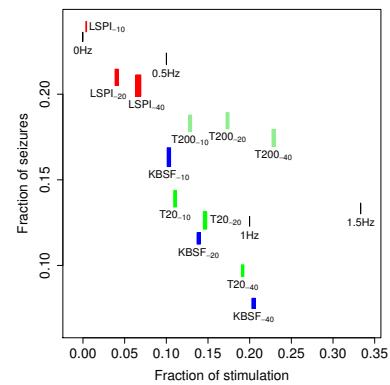
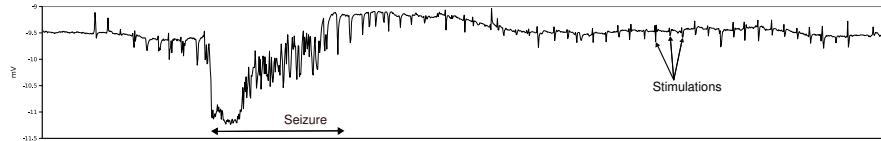


Backward



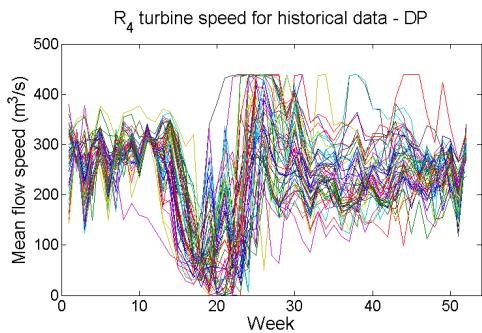
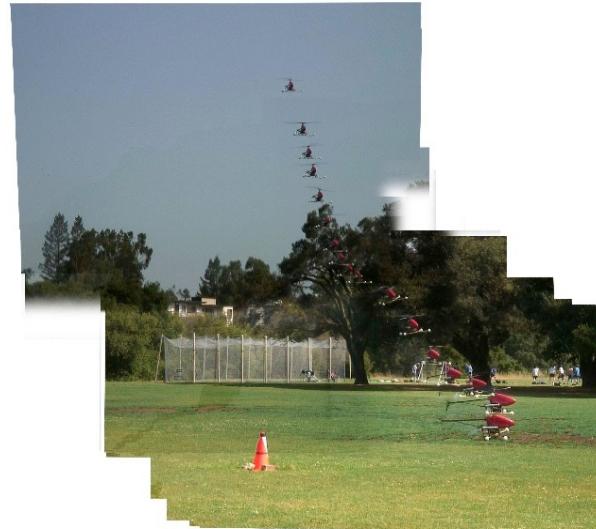
New Robot, Same algorithm

Applications: Few trajectories

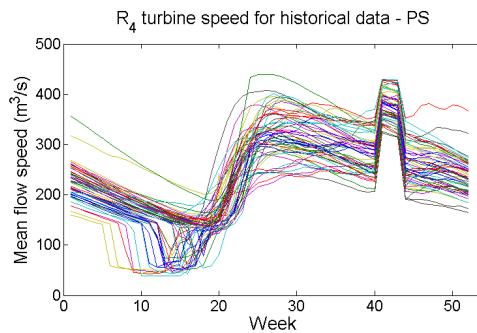


Epileptic seizure control

Our lab: Guez et al, 2008
Barreto et al, 2011, 2012



Power plant optimization
Our lab: Grinberg et al, 2014



Helicopter control
; Our lab: Barreto, Gehring et al, 2013

Signature challenges of RL

- Evaluative feedback (reward)
- Sequentiality, delayed consequences
- Need for trial and error, to explore as well as exploit
- Non-stationarity
- The fleeting nature of time and online data