Reinforcement learning (COMP-579)

- Instructors: Doina Precup and Isabeau Prémont-Schwartz
- TAs: Mark Bai, Ali Karimi, Gandharv Patil, Ray Luo and Shuyuan Zhang
- Class web page: http://www.cs.mcgill.ca/~dprecup/courses/rl.html
- Lectures split between Doina and Isabeau
- Lectures streamed on zoom and recorded on a best-effort only; questions only from in-person participants
- Office hours: to be posted on MyCourses
- Please use Ed for questions!

Outline

- Administrative issues
- What is reinforcement learning (RL)?
- Applications of RL
- If we have time: multi-arm bandits

Prerequisites

- Knowledge of programming in Python
- Probability, calculus, linear algebra; general comfort with math
- Knowledge of machine learning (McGill courses: COMP-0451, COMP-551, COMP-652)
- If in doubt about your background, contact Doina or Isabeau

Course material

- Required textbook: Sutton & Barto, Reinforcement learning: An Introduction, Second edition, 2019 (available online)

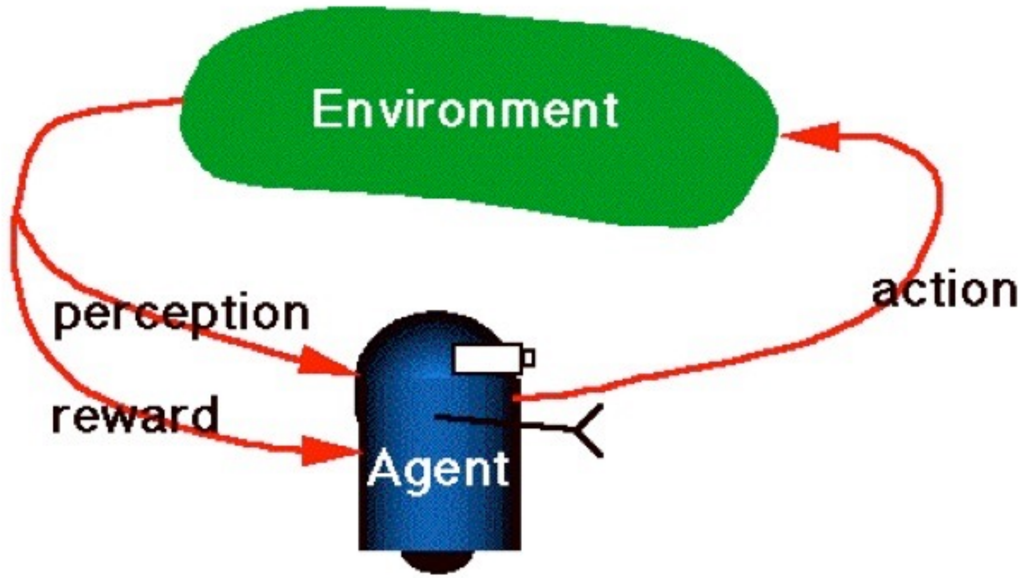
- Other required or suggested materials posted on the course web page

- Schedule posted on the web page; you MUST do the reading in order to really benefit from this course

Evaluation

- Project (40%): individual or in groups of up to 3;
- Three assignments (60%, dates posted on course web page)
- Assignments consist of a mix of theoretical and implementation/experimentation exercises.
- Specific instructions will be posted by the TA in charge of each assignment

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 $\langle x_1\, x_2 \ldots x_n, y\rangle$, 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 that supervised learning

A big success story: AlphaGo

ARTICLE

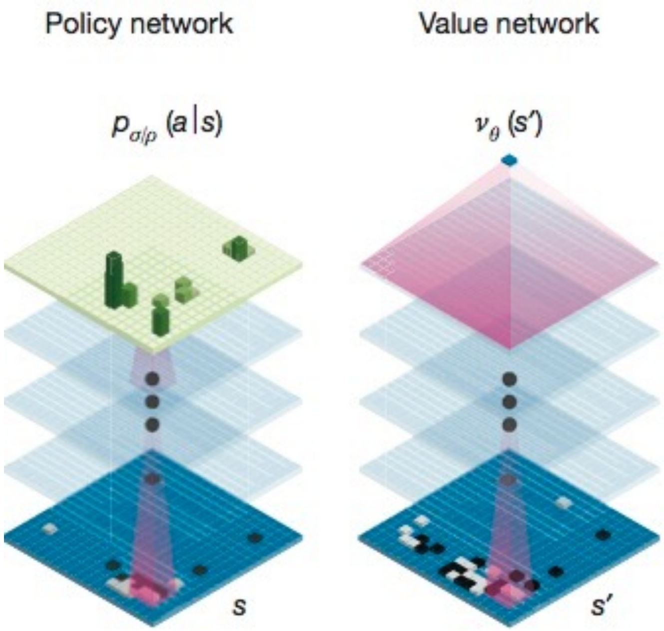
doi:10.1038/nature16961

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

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

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

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 find 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

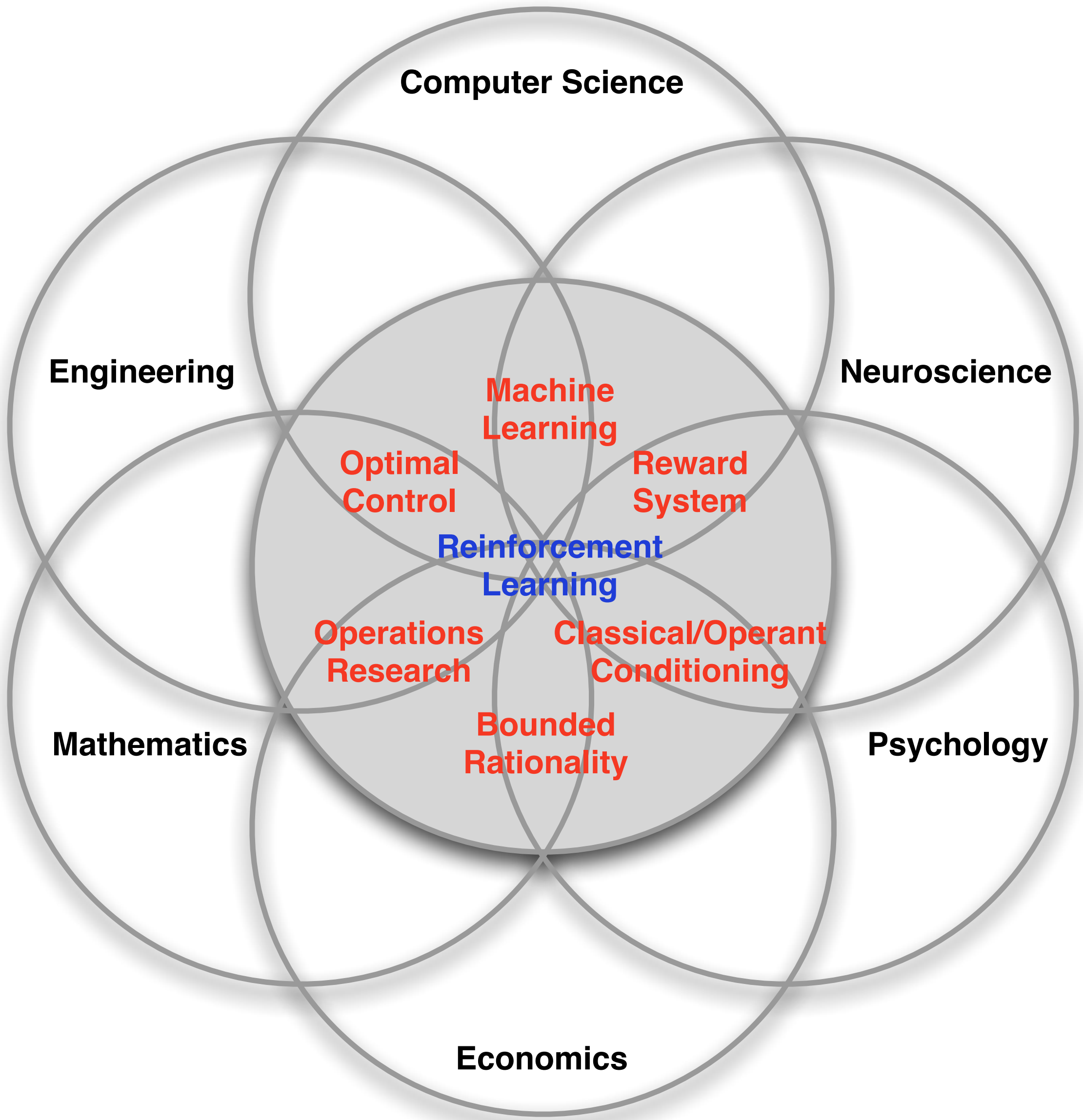
- The learner has to balance the need to *explore* its environment and the need to *exploit* its current knowledge

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

Basic Principles of Reinforcement Learning

- *All machine learning is driven to minimize prediction errors*
- In reinforcement learning, the algorithm makes *predictions* about the *expected future cumulative reward*
- These predictions should be consistent, i.e. similar to each other over time
- *Errors* are computed *between predictions made at consecutive time steps*
- *If the situation improved since last time step, pick the last action more often*

An Intersection Field!



Initial successes: Games

- Learned the world's best player of Backgammon (Tesauro 1995)

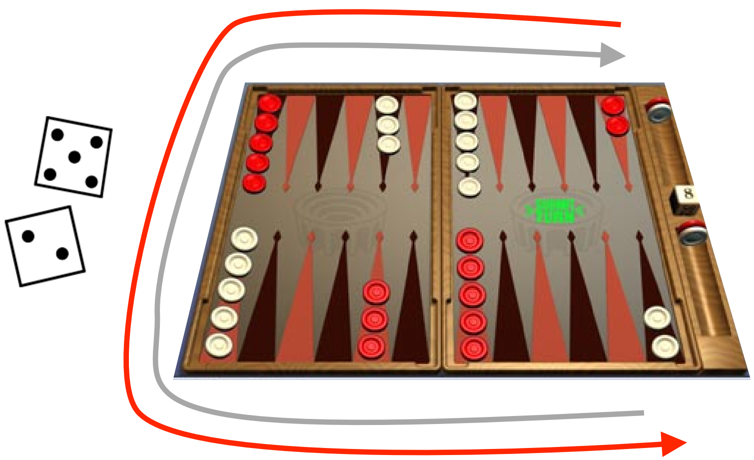
- Used to make strategic decisions in *Jeopardy!* (IBM's Watson 2011)

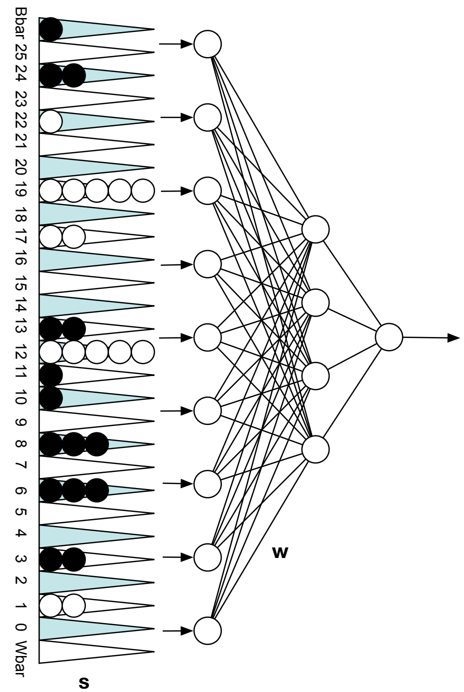
- Achieved human-level performance on Atari games from pixel-level visual input, in conjunction with deep learning (Google DeepMind 2015)

- In all these cases, performance was better than could be obtained by any other method, and was obtained without human instruction

Example: TD-Gammon

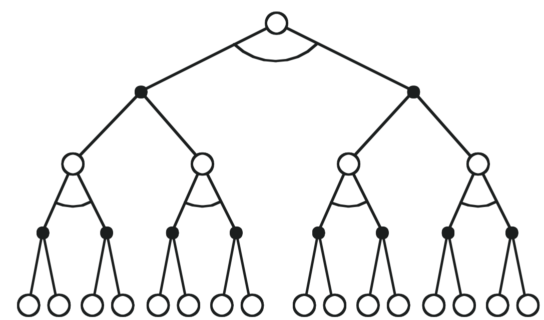
Tesauro, 1992-1995





estimated state value
(≈ prob of winning)

Action selection
by a shallow search



Start with a random Network

Play millions of games against itself

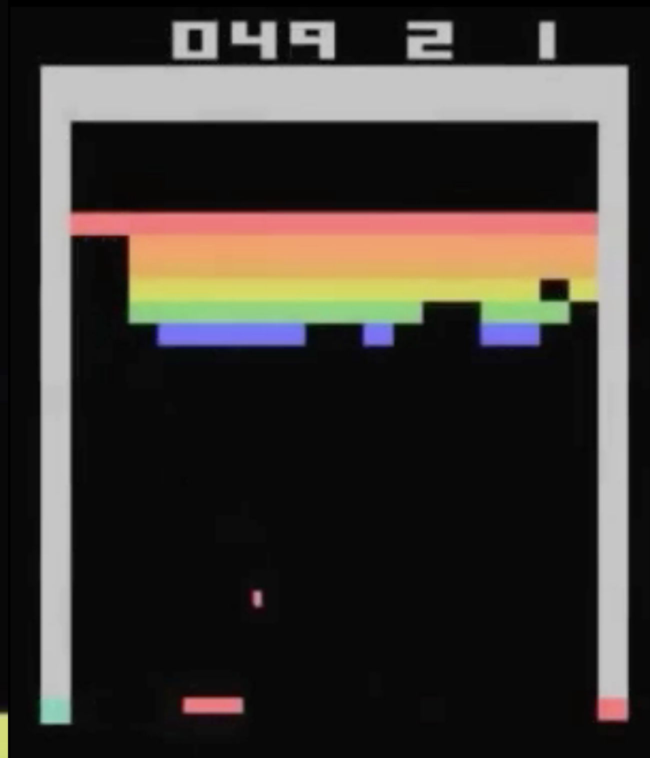
Learn a value function from this simulated experience

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 Learing Performance on Atari Games

Space Invaders

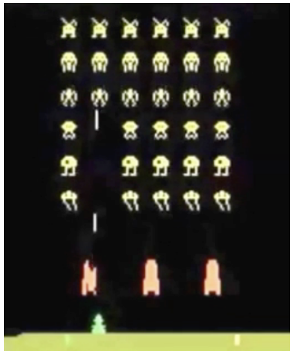


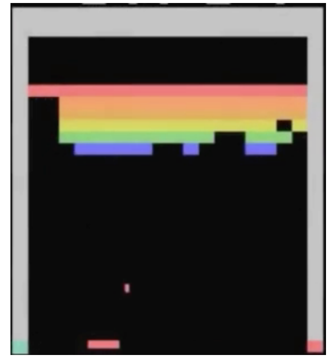
Breakout

Enduro

RL + Deep Learning, applied to Classic Atari Games

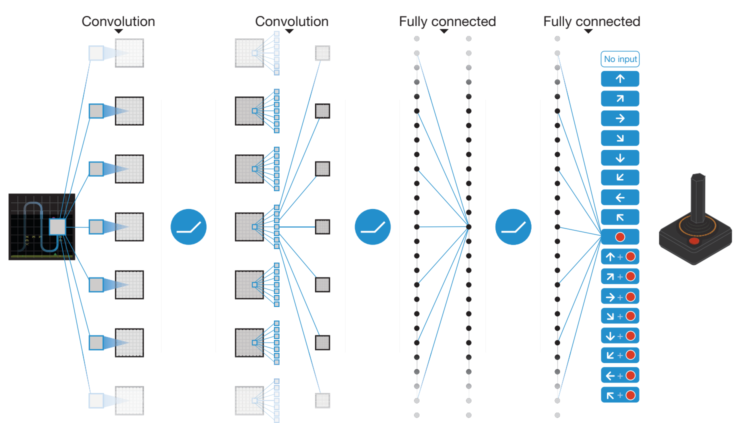
Google Deepmind 2015, Bowling et al. 2012




- 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

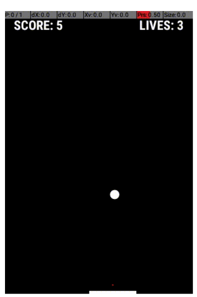


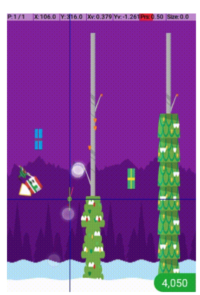
to predictions of final score for each of 18 joystick actions

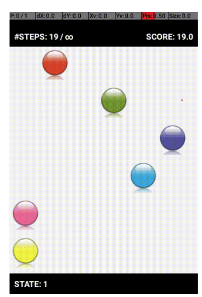
- Learned to play better than all previous algorithms and at human level for more than half the games

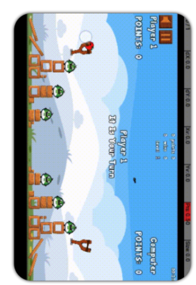
Same learning algorithm applied to all 49 games! w/o human tuning

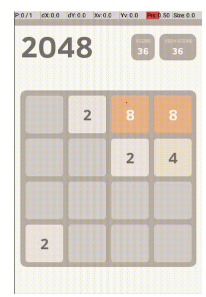
RL can produce agents that play complex games!

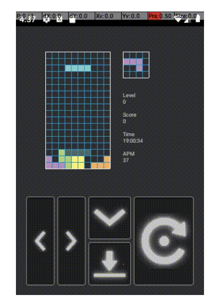








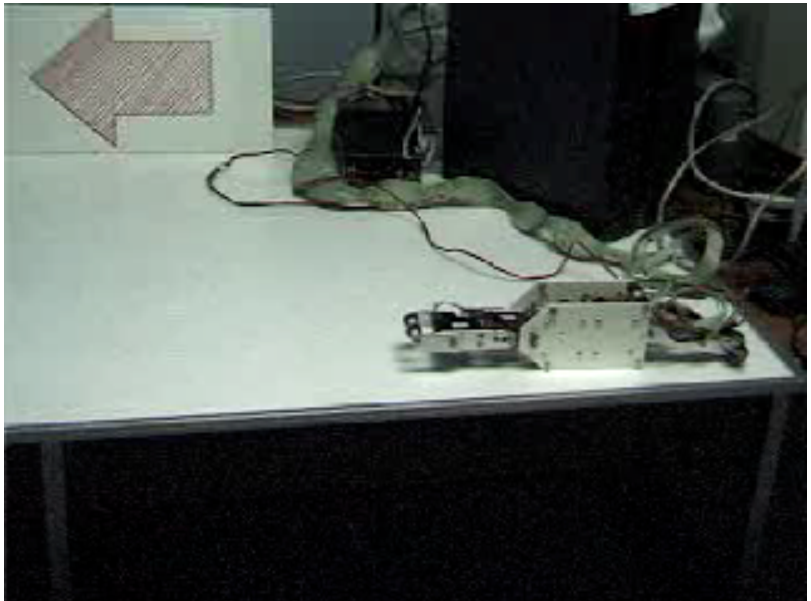




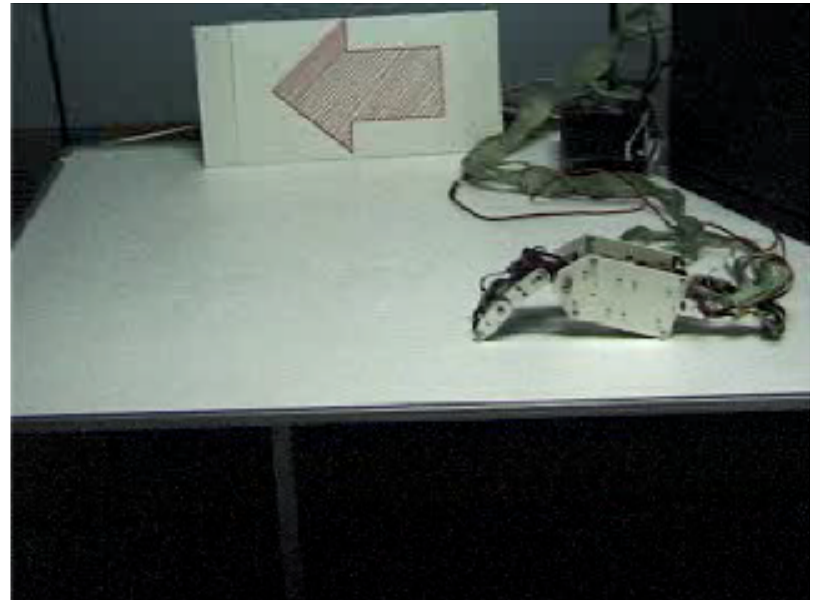
More successes: Complex control tasks

- 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)
- Control of tokamak plasma reactors
- In all these cases, performance was better than could be obtained by any other method, and was obtained without human instruction

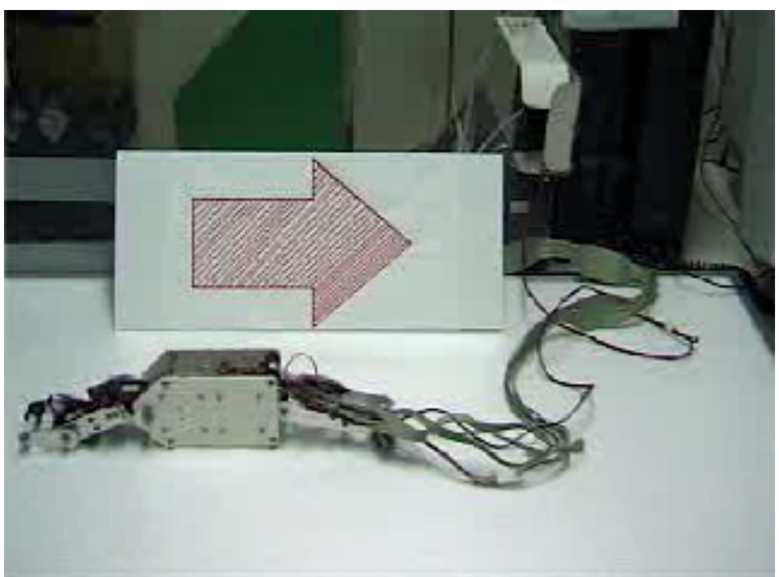
Example: Hajime Kimura's RL Robots



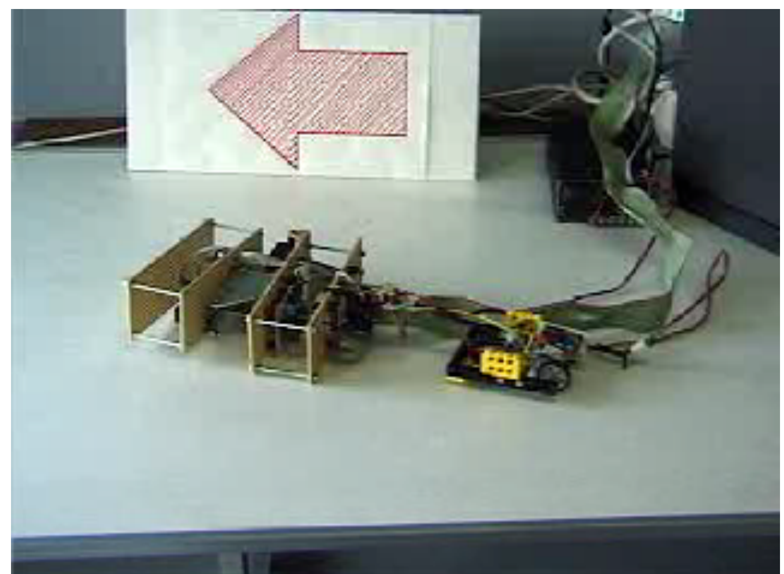
Before



After

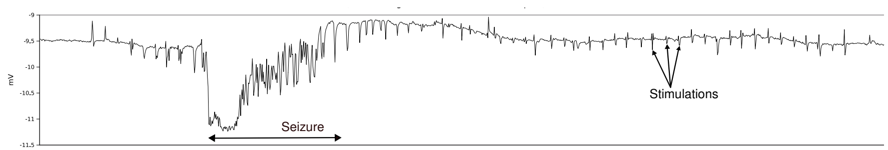


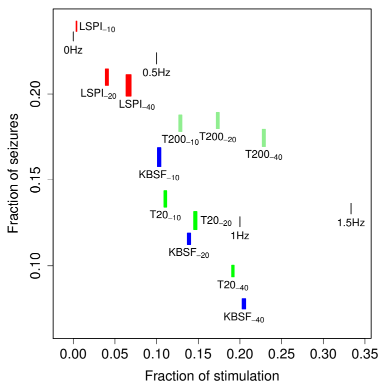
Backward



New Robot, Same algorithm

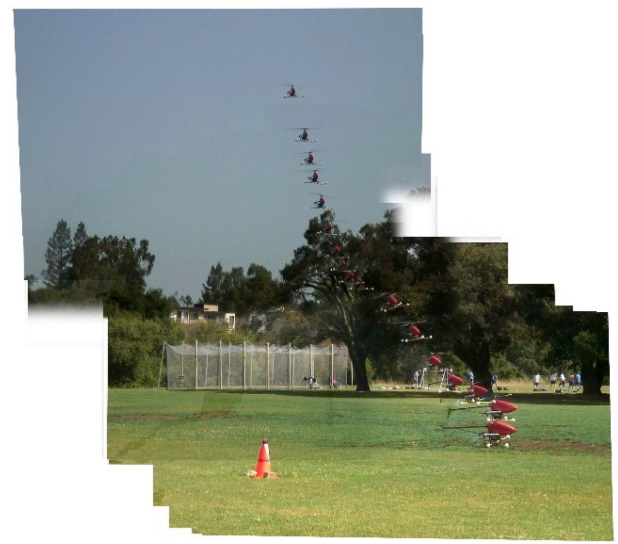
RL can solve many problems!



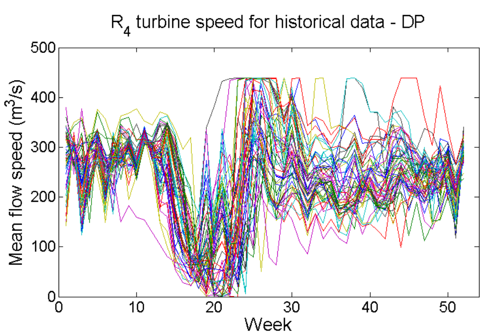


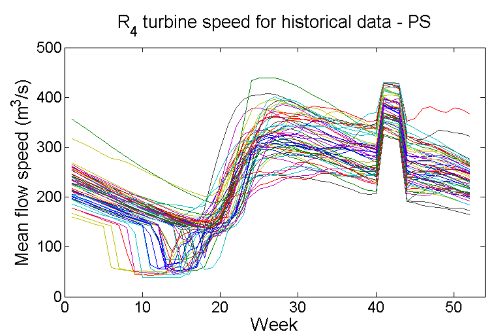
Epileptic seizure control

Guez et al, 2008
Barreto et al, 2011, 2012



Helicopter control

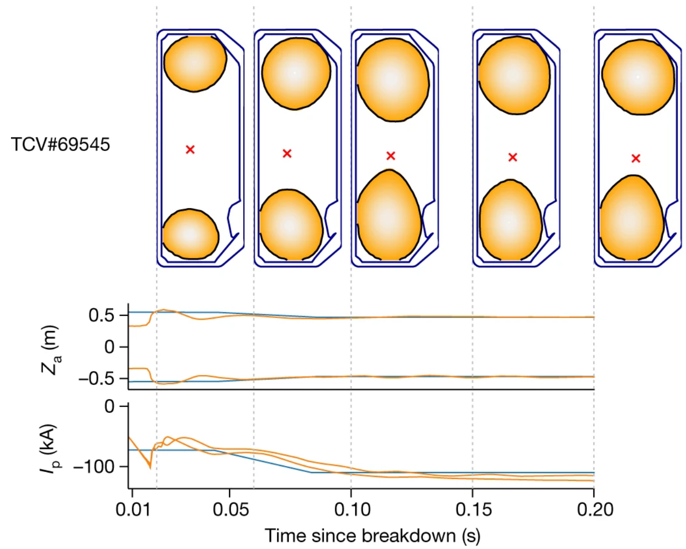


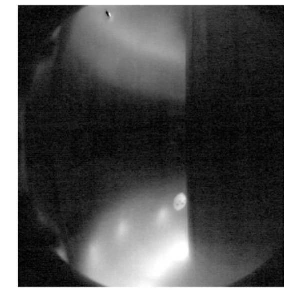


Power plant optimization
Grinberg et al, 2014

Recent Successes: Complex Control Tasks

Bellemare et al, Nature, 2020

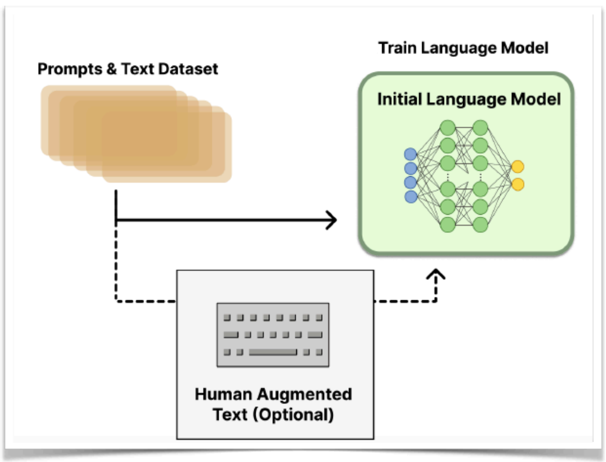




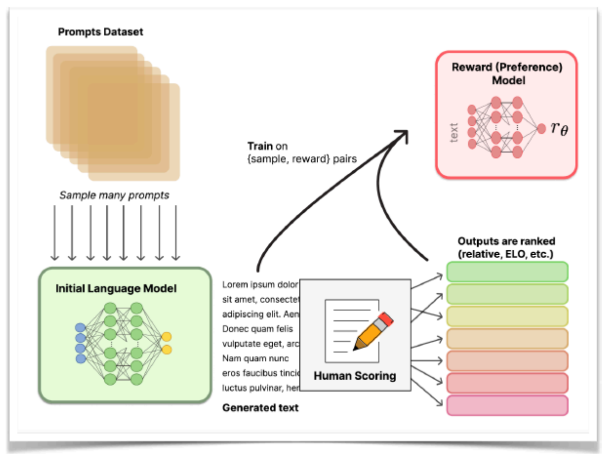
Degrave et al, Nature, 2022

Recent Successes: Chat Bots, RLHF

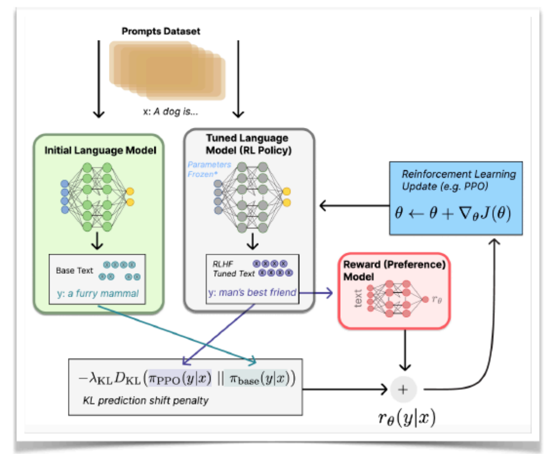
1. Language model pretraining



2. Reward model training



3. Fine-tuning with RL



Recent/Future Successes: Exa-Scale Search for Molecules

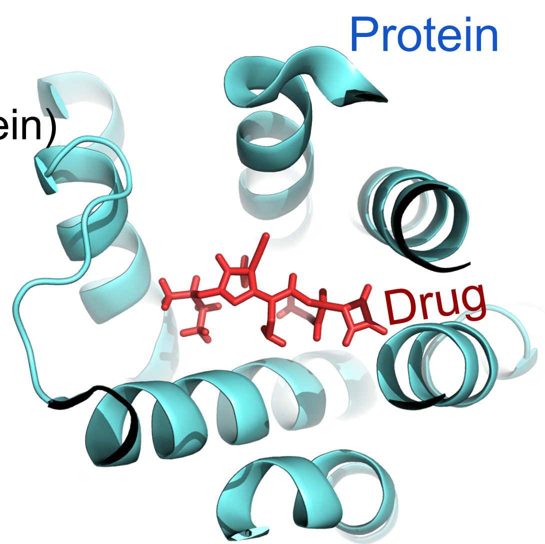
find drugs that bind to protein(s)

>$10^{16\sim20}$ space (simplified + for *one* protein)

most molecules are *bad*:

- not chemically feasible
- not binders
- toxic

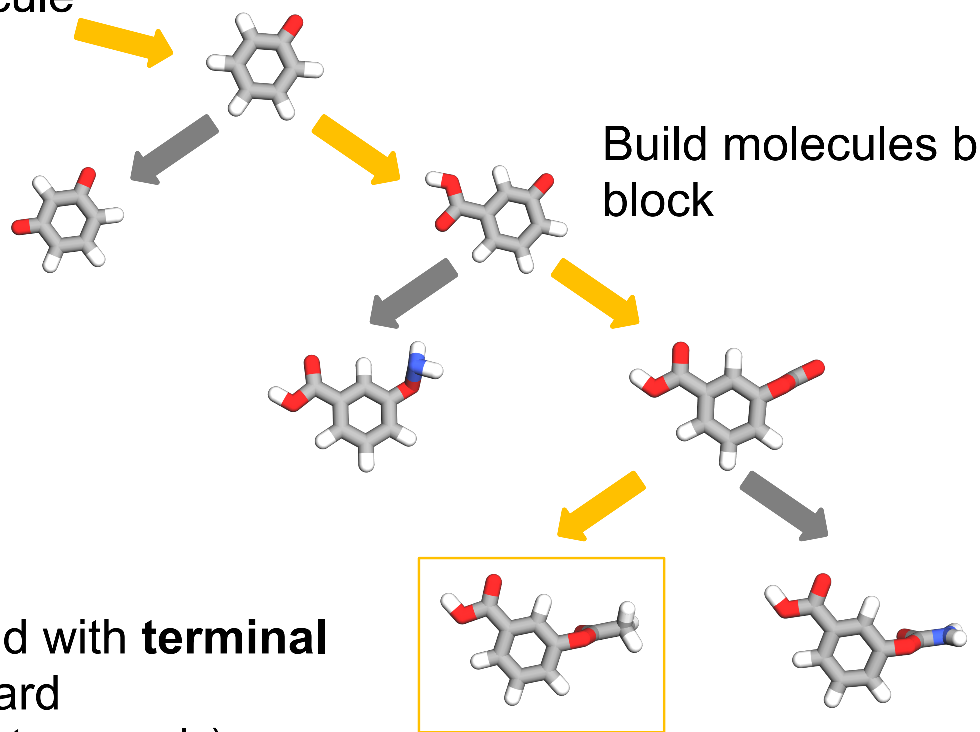
Needles in a haystack!



Molecule Search as Reinforcement Learning

“empty molecule”

Build molecules block by block



Episodes end with **terminal** positive reward
(no intermediate rewards)

Bengio et al, NeurIPS’2021

Mila

Recap: What is Reinforcement Learning?

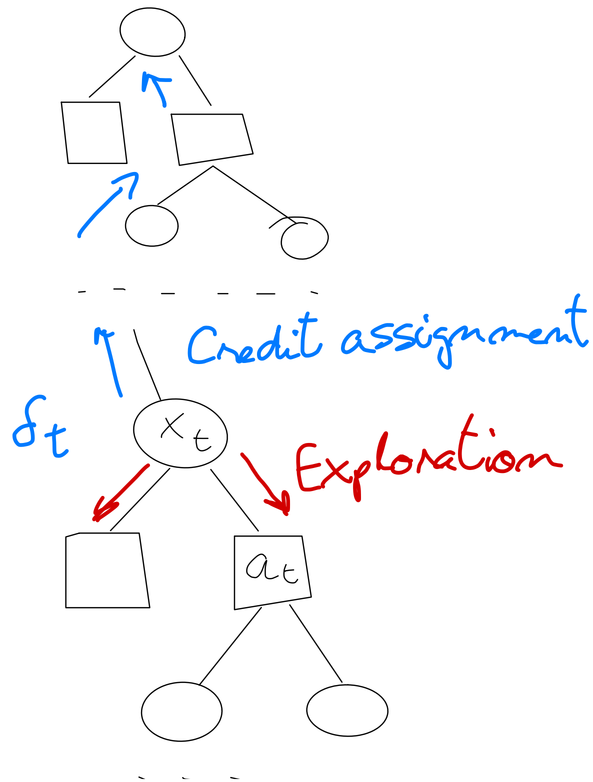
- Agent-oriented learning—learning by interacting with an environment to achieve a goal
  - more realistic and ambitious than other kinds of machine learning
- Learning by trial and error, with only delayed evaluative feedback (reward)
  - the kind of machine learning most like natural learning
  - learning that can tell for itself when it is right or wrong
- The beginnings of a *science of mind*

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

How to think about RL more systematically?

- At time t, agent receives an observation from set $\mathcal{X}$ and can choose an action from set $\mathcal{A}$ (think finite for now)
- Goal of the agent is to maximize long-term return



More details

- Circles represent random variables
- Squares represent decision variables
- Rewards are numbers received as part of the observation

More on decision making

- For simplicity, we are assuming a discrete time scale t=0, 1, …
- If the tree has no structure at all, nothing can be learned!
- Different flavours of RL algorithms make different assumptions about the structure of the tree
- Assumptions allow past experience to inform future decisions
- Next time: bandits - tree is a single node!