CSC2541: Deep Reinforcement Learning

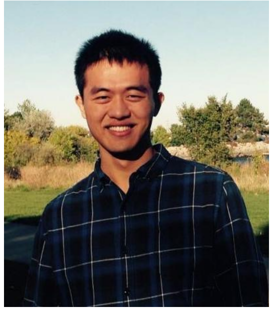
Lecture 1: Introduction

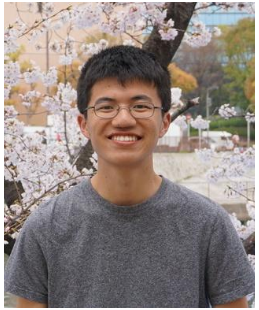
Slides borrowed from David Silver

Jimmy Ba

UNIVERSITY OF TORONTO

Logistics





- Instructor: Jimmy Ba   Teaching Assistants: Tingwu Wang, Michael Zhang
- Course website: TBD
- Office hours: after lecture. TA hours: TBD

Logistics

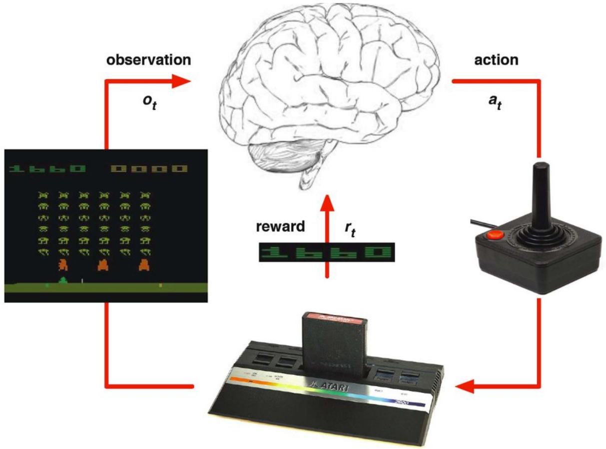
Grades breakdown:

- 20% seminar presentation
- 20% project proposal (Due Oct. 14th)
- 60% final project presentation and report (Due Dec. 16th)
- Suggested textbook: An Introduction to Reinforcement Learning, Sutton and Barto (available online)

Reinforcement learning

Learning to act through trial and error:

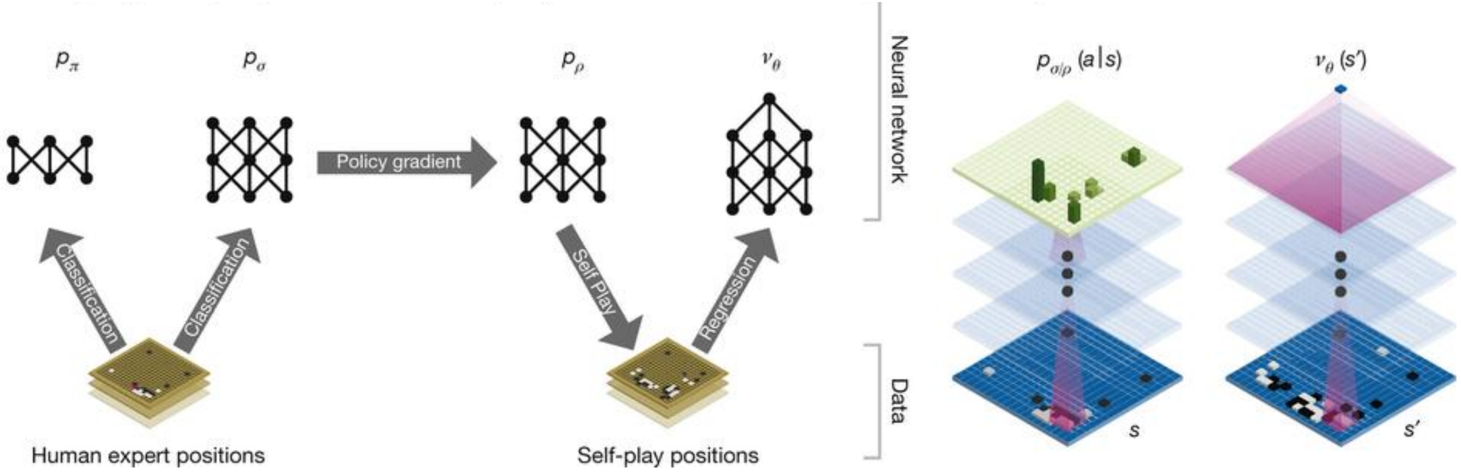
- An agent interacts with an environment and learns by maximizing a scalar reward signal.
- No models, labels, demonstrations, or any other human-provided supervision signal.



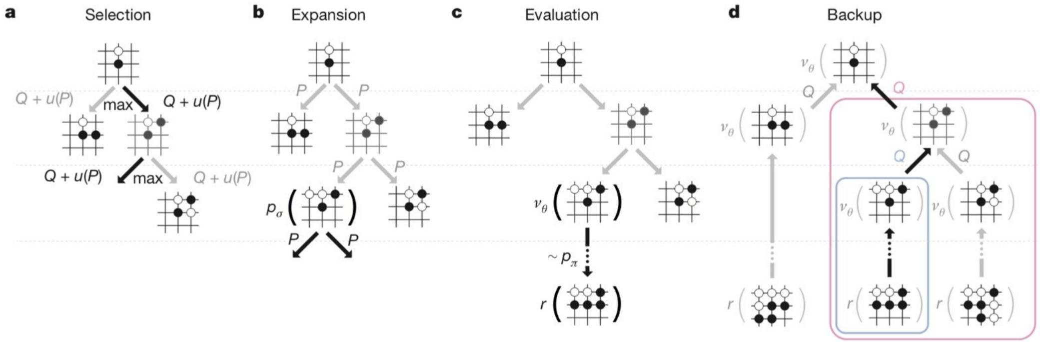
Mnih et. al., 2015

More success stories in reinforcement learning

More success stories in reinforcement learning



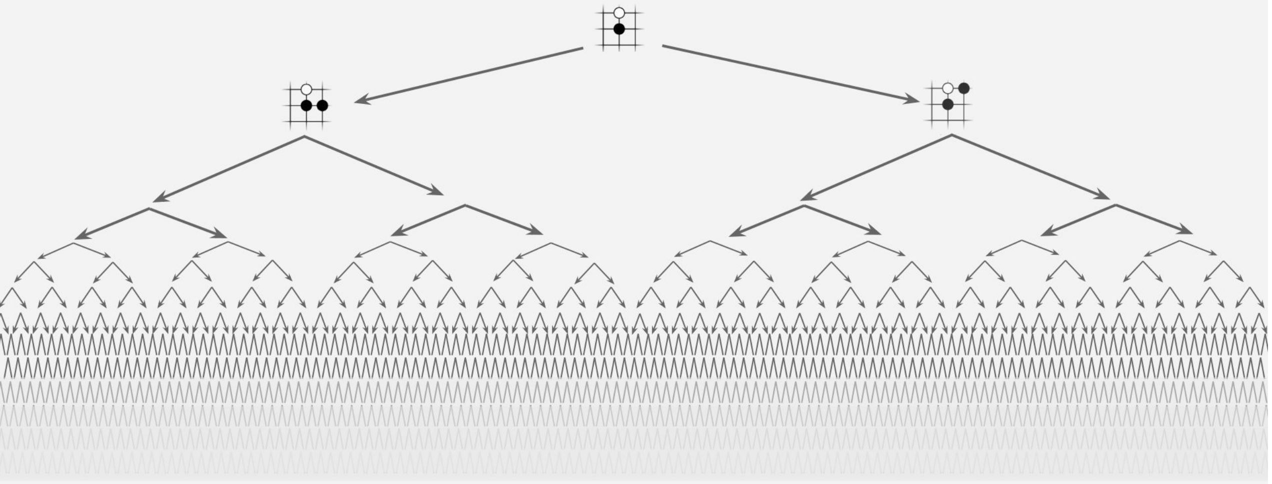
- Monte-Carlo:



David Silver

Preview case study

- We can think of the game of Go as a tree search problem.
  - Choose a move that has the highest chance of winning: argmax P(win | next_state)
  - We can run forward sampling algorithm to solve for this probability if we have the model of our opponent.

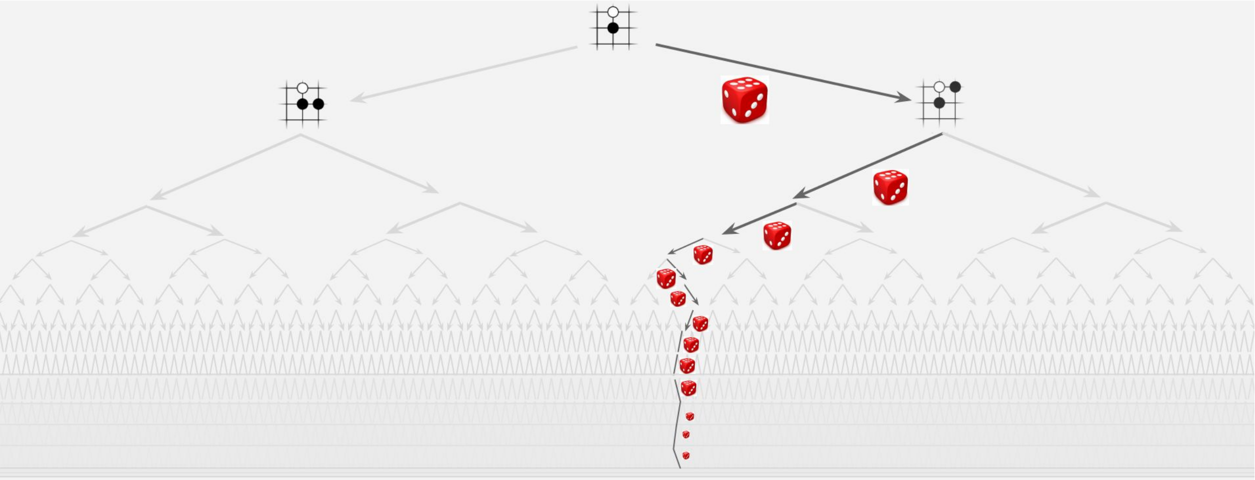


- The tree is too **wide**: too many branches at each node, which makes the summation over all those states infeasible.
- The tree is too **deep**: initial condition of the message passing algorithm is at the bottom of the tree.

Silver et. al., 2016

Preview: AlphaGo case study

- We can think of the game of Go as a tree search problem.
  - Monte-Carlo rollouts can reduce the breath of the tree.
  - It does not help much if the proposal distribution is bad.

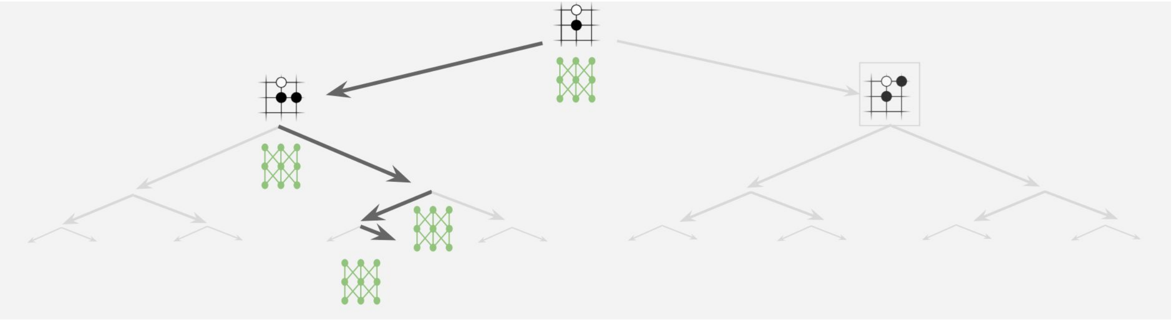


- The tree is too **wide**: too many branches at each node, which makes the summation over all those states infeasible.
- The tree is too **deep**: initial condition of the message passing algorithm is at the bottom of the tree.

Silver et. al., 2016

Preview: AlphaGo case study

- We can think of the game of Go as a tree search problem.
  - Monte-Carlo rollouts + neural network learnt on expert moves, i.e. policy network
  - The policy network helps MC rollouts to not waste computational resources on “**bad**” moves.

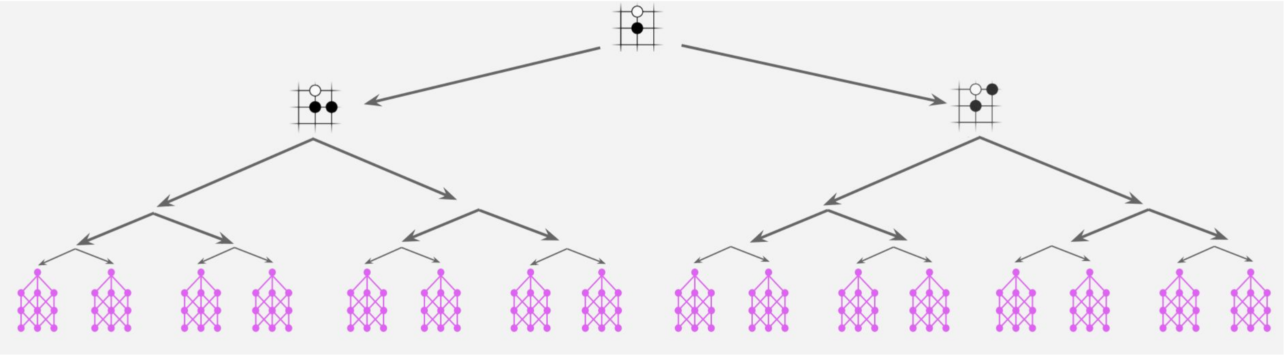


- policy network cut down the **breath** of the search tree.
- The tree is too **deep**: initial condition of the message passing algorithm is at the bottom of the tree.

Silver et. al., 2016

Preview: AlphaGo case study

- We may not want to unroll all the way to the leaves of the tree.
  - Use a neural network to **approximate** the future condition, i.e. value network
  - The value network learns the probability of winning at each node of the tree.

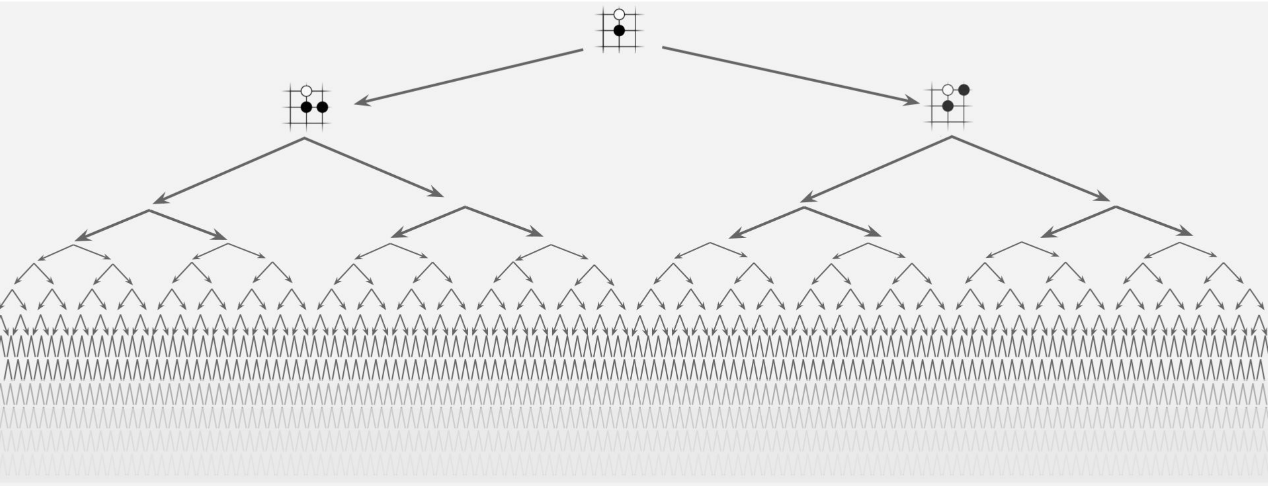


- policy network cut down the **breath** of the search tree.
- Value network cut down the **depth** of the search tree.

Silver et. al., 2016

Preview: AlphaGo case study

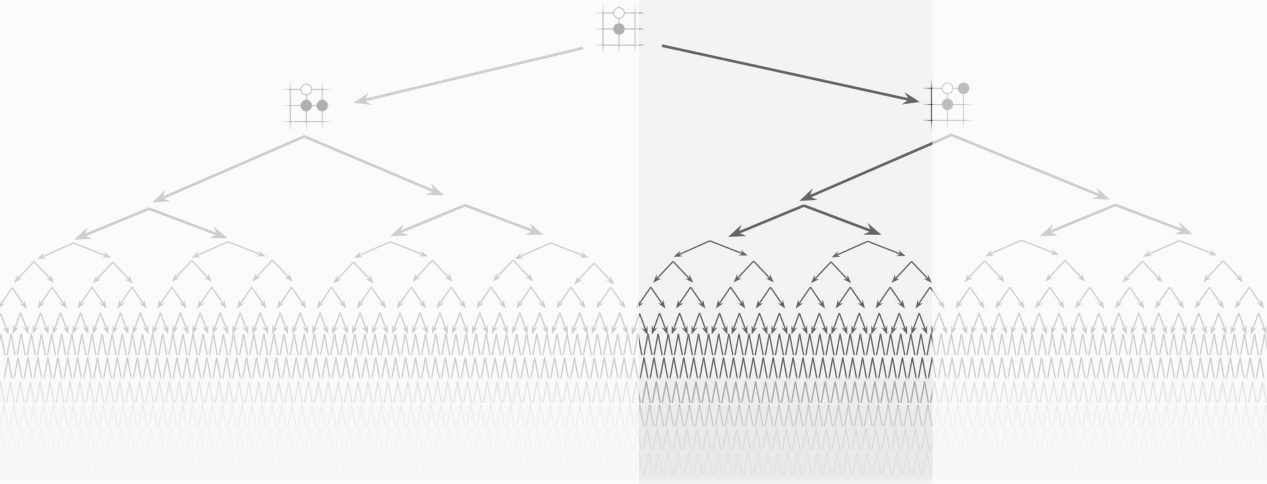
- Use both policy and value networks to significantly reduce the inference computation.



Silver et. al., 2016

Preview: AlphaGo case study

- Use both policy and value networks to significantly reduce the inference computation.

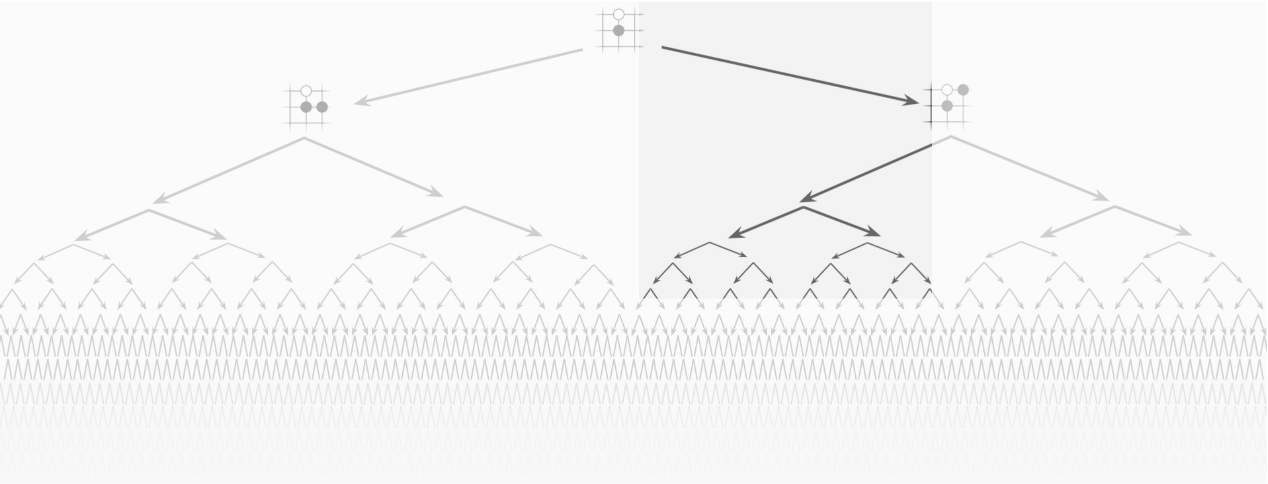


- policy network cut down the **breath** of the search tree.

Silver et. al., 2016

Preview: AlphaGo case study

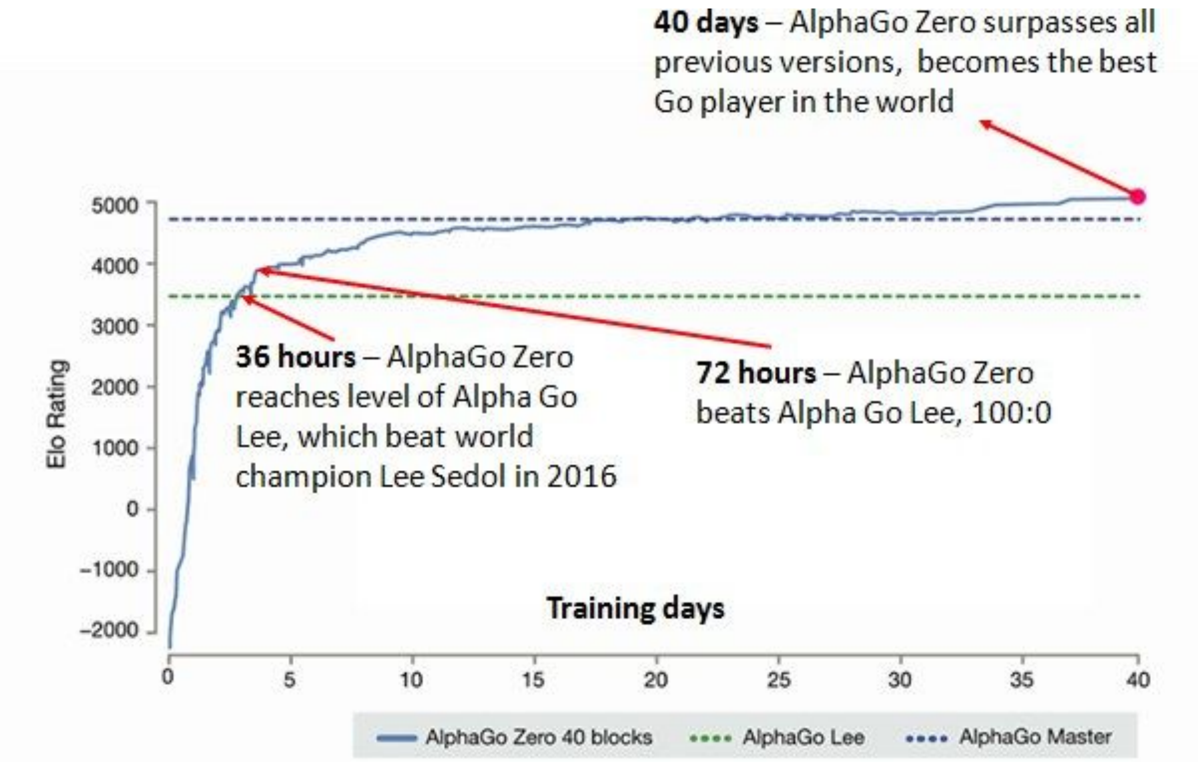
- Use both policy and value networks to significantly reduce the inference computation.



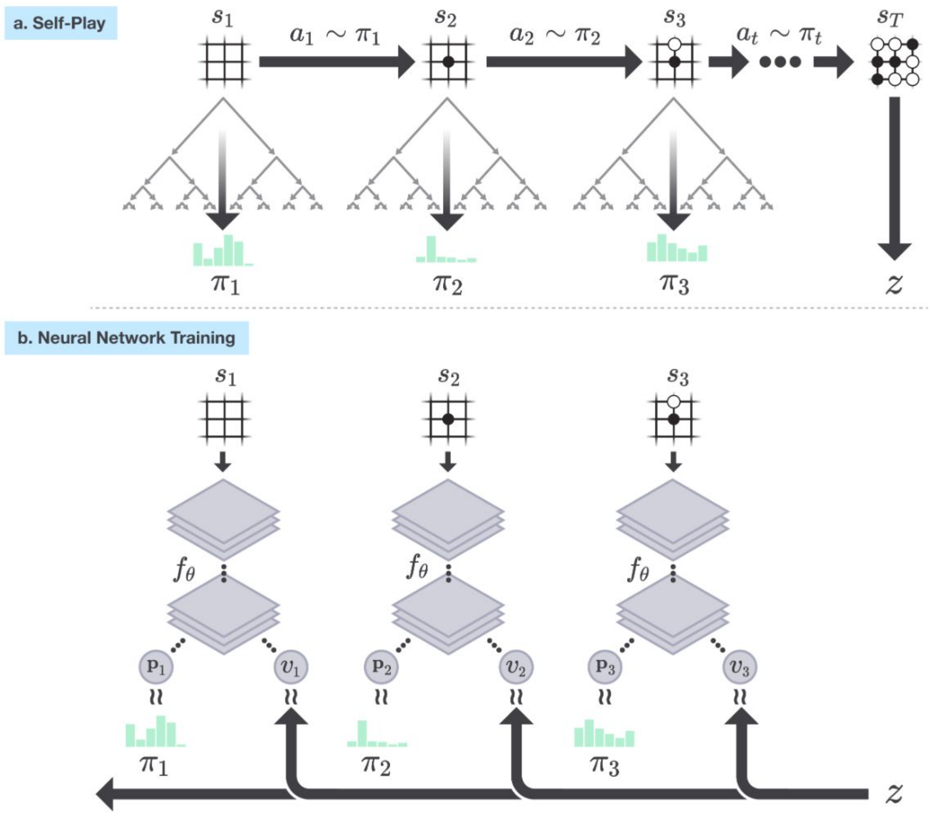
- policy network cut down the **breath** of the search tree.
- Value network cut down the **depth** of the search tree.

Silver et. al., 2016

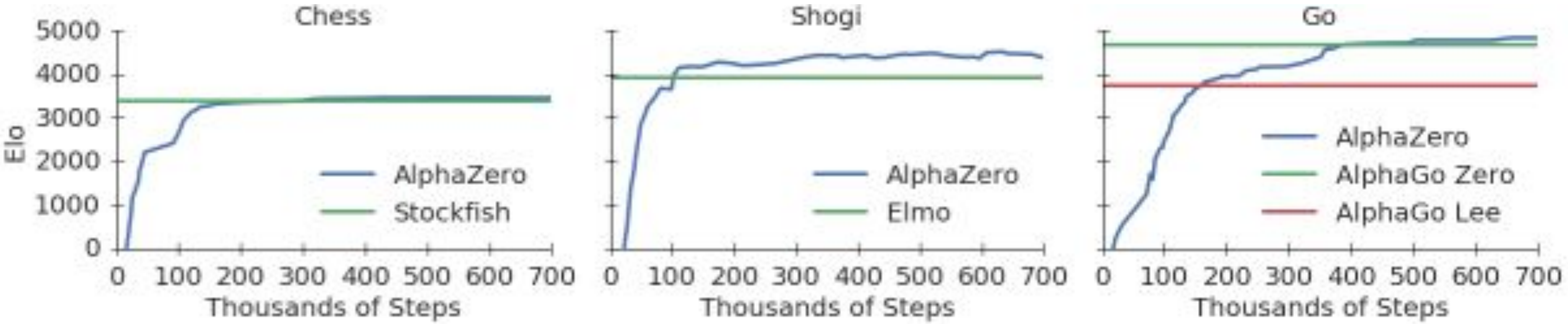
More success stories in reinforcement learning



More success stories in reinforcement learning



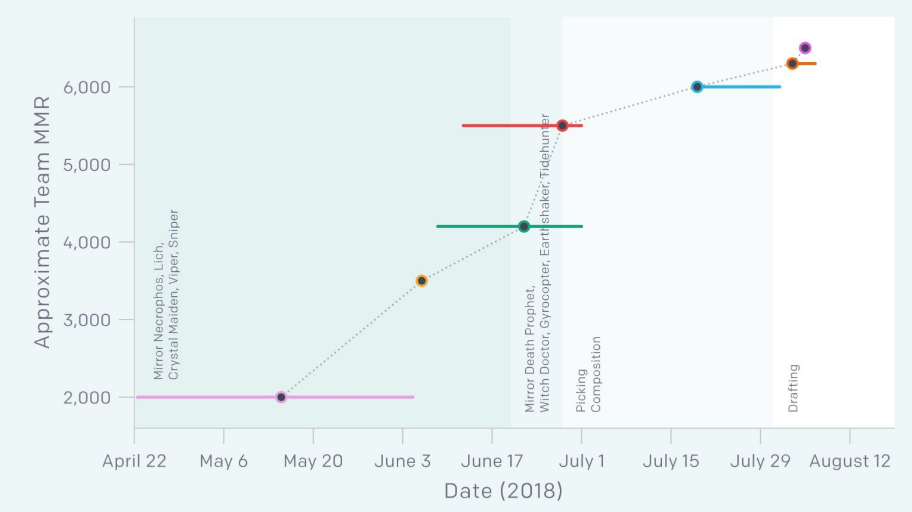
More success stories in reinforcement learning



More success stories in reinforcement learning

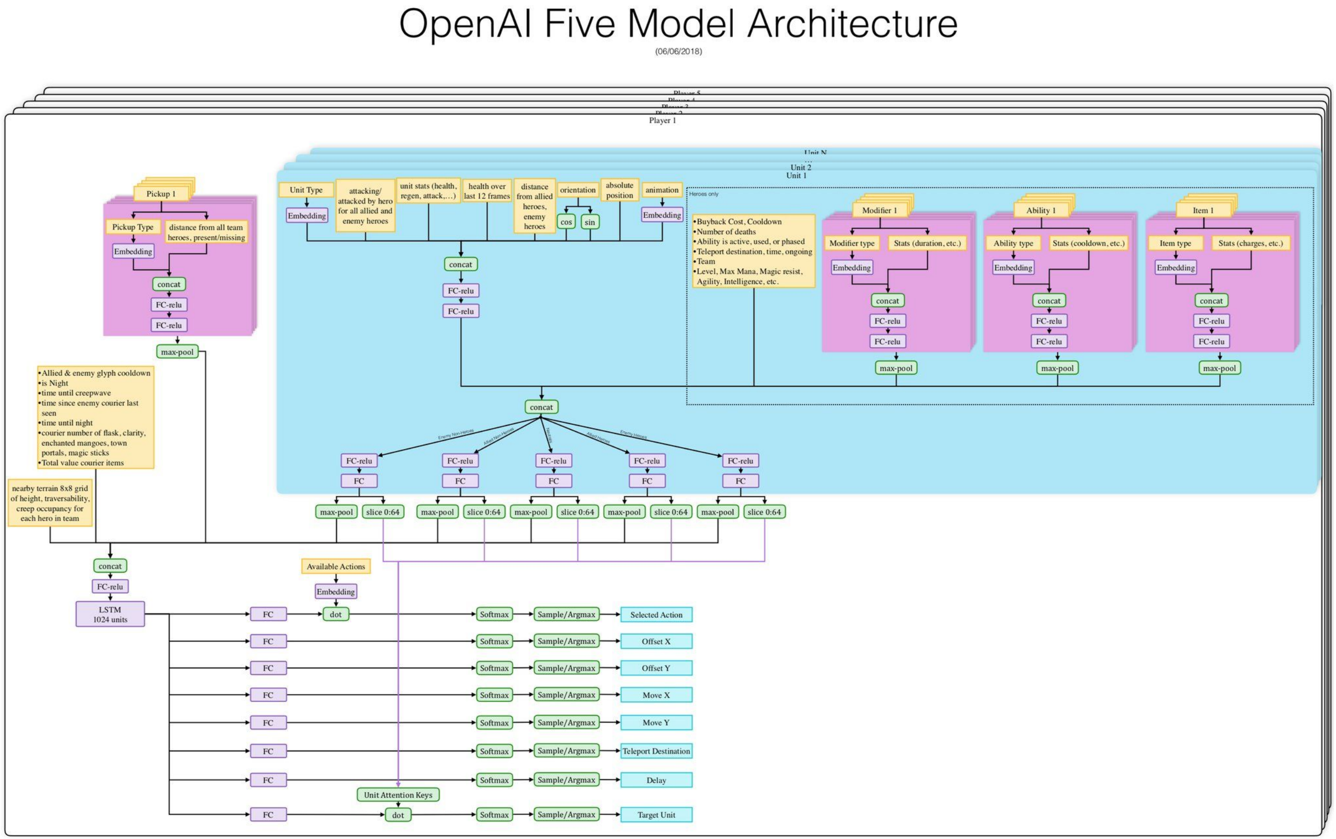
DOTA2 and OpenAI Five

- Partially observable game states



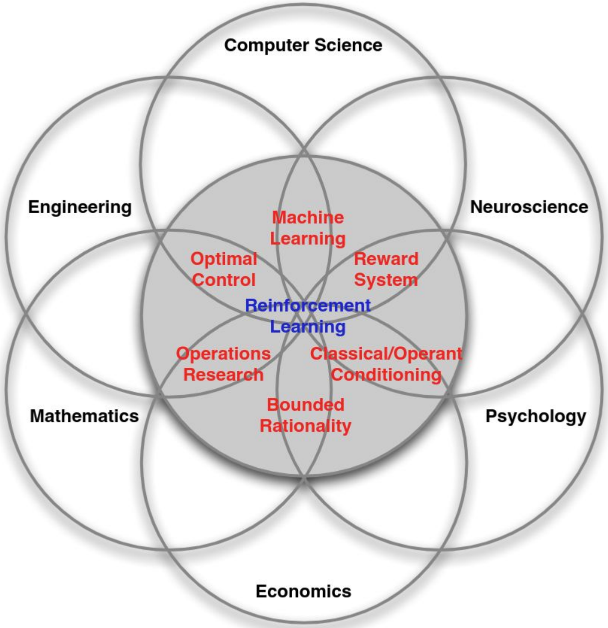
More success stories in reinforcement learning

1024 single layer LSTM:



Reinforcement learning

Learning to act through trial and error:



Reinforcement learning

Learning to act through trial and error:

- An agent interacts with an environment and learns by maximizing a scalar reward signal.
- No models, labels, demonstrations, or any other human-provided supervision signal.
- Feedback is delayed, not instantaneous
- Agent’s actions affect the subsequent data it receives (data not i.i.d.)

Reward

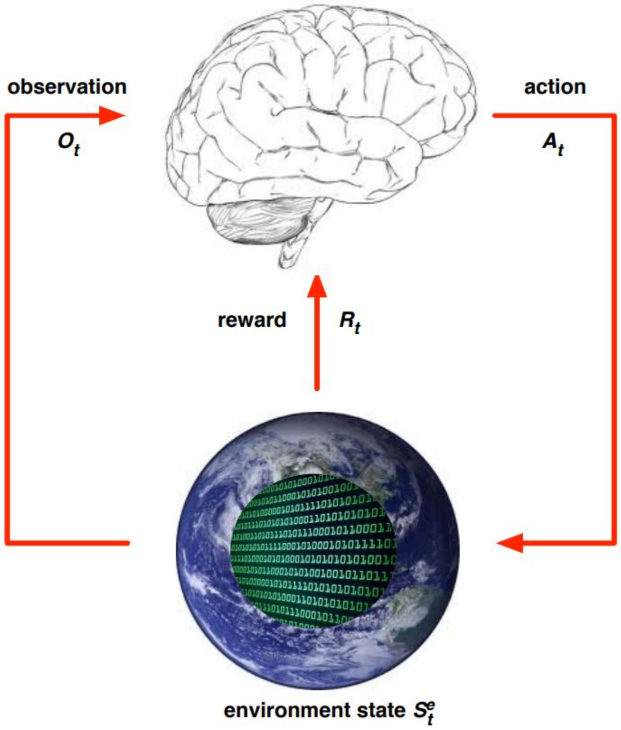
Reward hypothesis: All goals can be described by the maximization of the expected cumulative reward.

- A reward $R_t \in \mathbb{R}$ is a scalar feedback signal
- Indicates how well agent is doing at timestep t
- The agent's job is to maximise cumulative reward

Sequential decision making

- Goal: select actions to maximize total future reward
- Actions may have long term consequences
- Reward maybe delayed
- Might be better to sacrifice short-term gain for more long-term reward

Agent and Environment

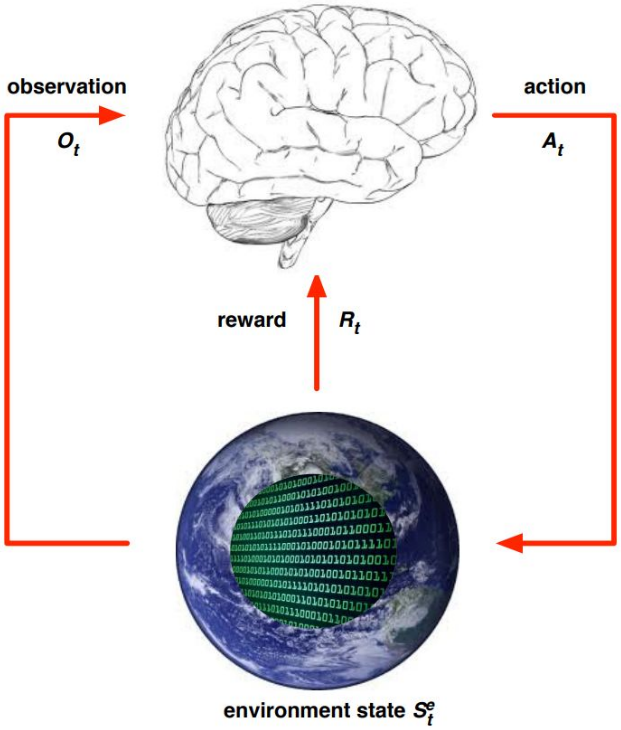


- At each step t, the agent:
  - Receives observation O_t
  - Executes action A_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits scalar reward R_t
  - Emits observation O_{t+1}

History and states

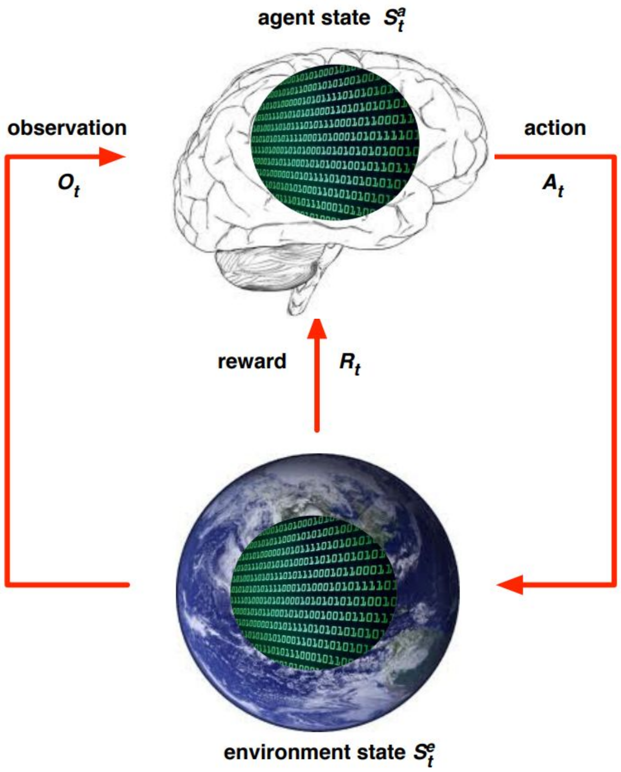
- History is the sequence of observations, actions, rewards up to timestep t
  - $H_t = \{O_1, R_1, A_1, O_2, R_2, A_2, \ldots, R_{t-1}, A_{t-1}, O_t\}$
- History consists of all the observable variables up to t
- State is defined as a function of the history $S_t = f(H_t)$ that is used to determine what happens next.
- Related concept: trajectory is the sequence of observation and action pairs
  - $\tau = \{O_1, A_1, O_2, A_2, \ldots, A_{t-1}, O_t\}$

Environment state



- The environment state S^e_t is the internal representation used by the environment.
- E.g. Sensor data on robot may contain joint angle and velocity. The environment keeps track of the acceleration and other information.
- The environment state is not observable in general.

Agent state

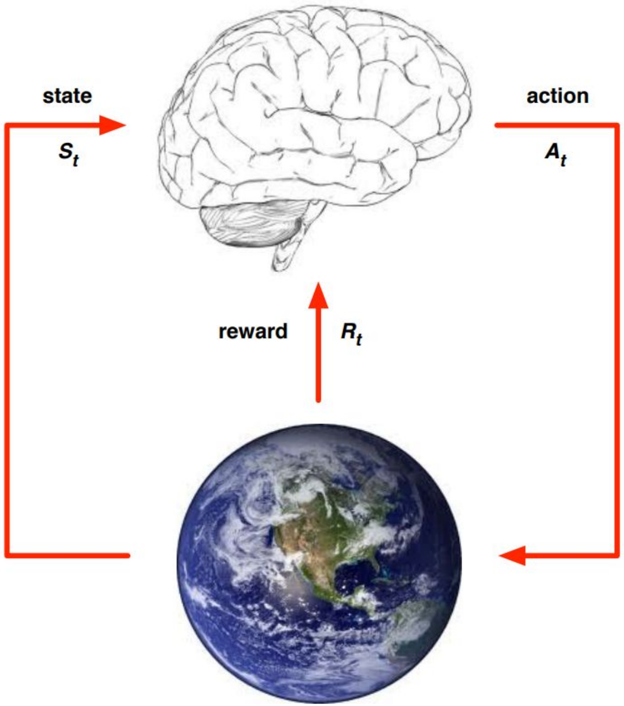


- The environment state S^a_t is the internal representation used by the agent.
- E.g. LSTM hidden states the agent uses to estimate of the true environment state.
- It can be any function of the history.

Markov state

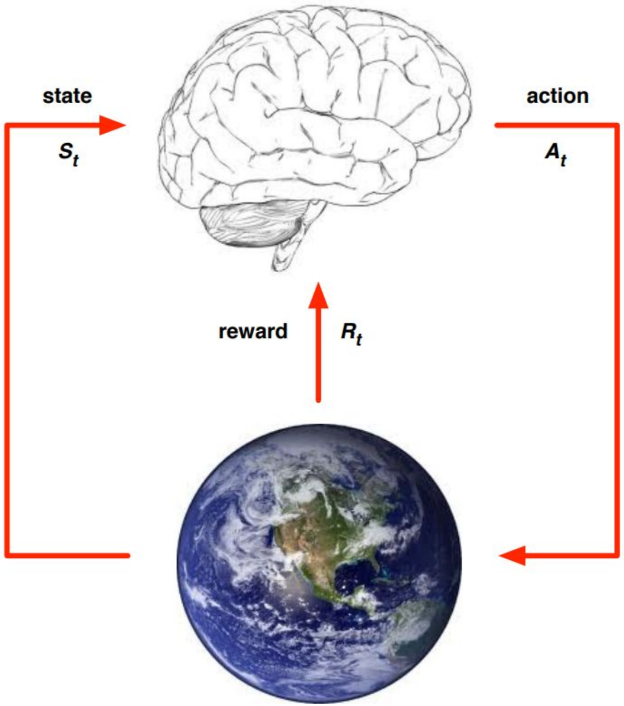
- A Markov state contains full information from the history
- A state S_t is Markov if and only if :
  - $P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, S_2, \ldots, S_t)$
  - i.e. the future is independent of the past given the present.
- Once the current state is known, the history can be thrown away
- The history H_t itself is Markov.

Fully observable environments



- The agent directly observe the **environment state** S^e_t.
  - O_t = S_t = S^e_t
  - And environment state is Markov
- Formally this turns into a **Markov Decision Process (MDP)**.

Partially observable environments

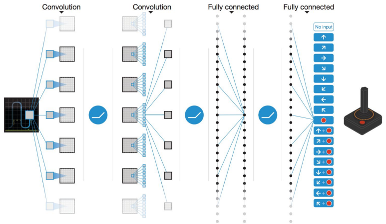


- The agent do not observe the **environment state** S^e_t.
  - O_t \= S^e_t
  - But environment state is Markov
- Formally this turns into a **Partially Observable Markov Decision Process (POMDP)**.

Which of the following are POMDPs?

Playing Poker

Self-play Go

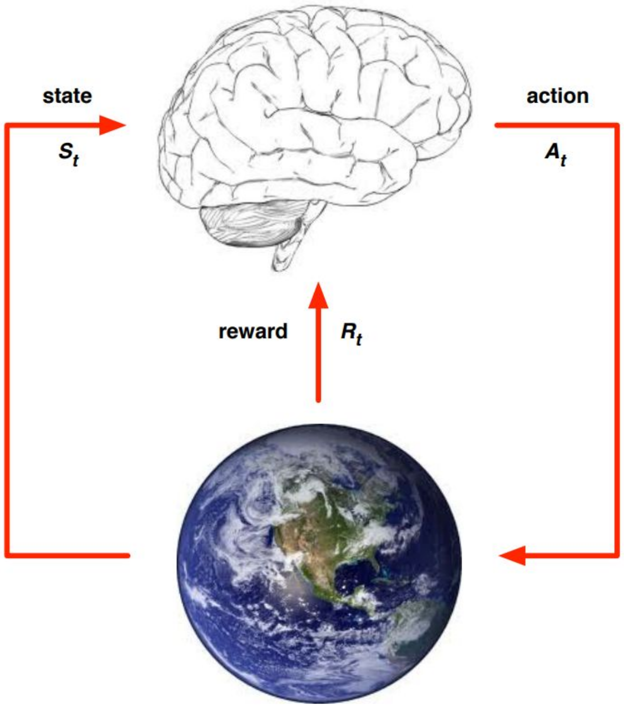


Learning Atari

games from pixels

Stock trading from historical data

Major component of an RL agent



- Policy: agent's behaviour function
- Value function: how good is each state and/or action.
- Dynamic model: agent's representation of the environment.

Policy

- A policy π is the agent's behaviour.
- It maps from the agent's state space to the action space.
    - Deterministic policy: $a = \pi(s)$
    - Stochastic policy: $\pi(s) = P(A_t = a | S_t = s)$

Value function

- Value function is a prediction of the future reward.
- Used to evaluate the goodness/badness of the state.
    - $v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \mid S_t = s \right]$
- We can use value function to choose the actions.

Model

- A model predicts what will happen next in the environment.
  - **Dynamics model** predicts the next state given the current state and the action.
  - **Reward model** predicts the immediate reward given the state and the action.

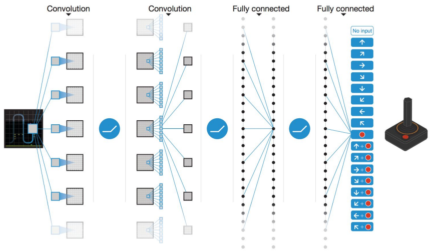
$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}_s^a = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

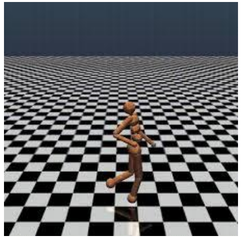
AlphaGo vs game of life

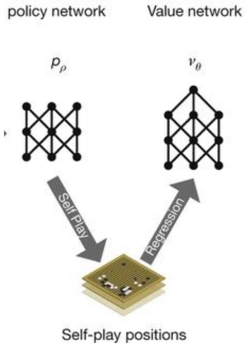
- Given environment model vs learn everything from scratch
- Discrete action space vs continuous action space
- Discrete state space vs continuous state space
- Single goal vs multi-goal
- Clean reward signal vs noisy reward signal

Category of RL agents

- Value-based
  - DQN Atari agents
- Policy-based
  - Locomotion control
- Actor Critic
  - AlphaGo

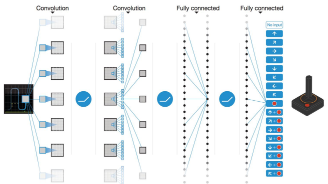


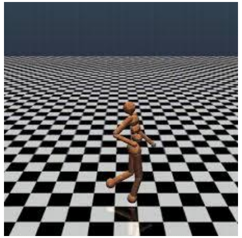




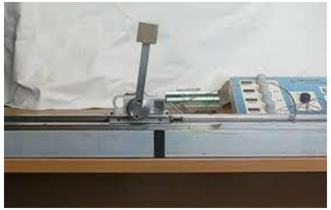
Category of RL agents

- Model-free agents
  - Does not learn any model

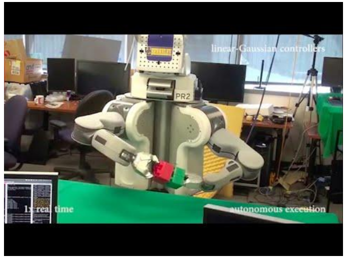




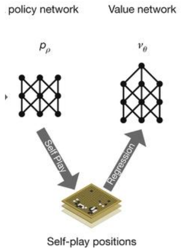
- Model-based agents



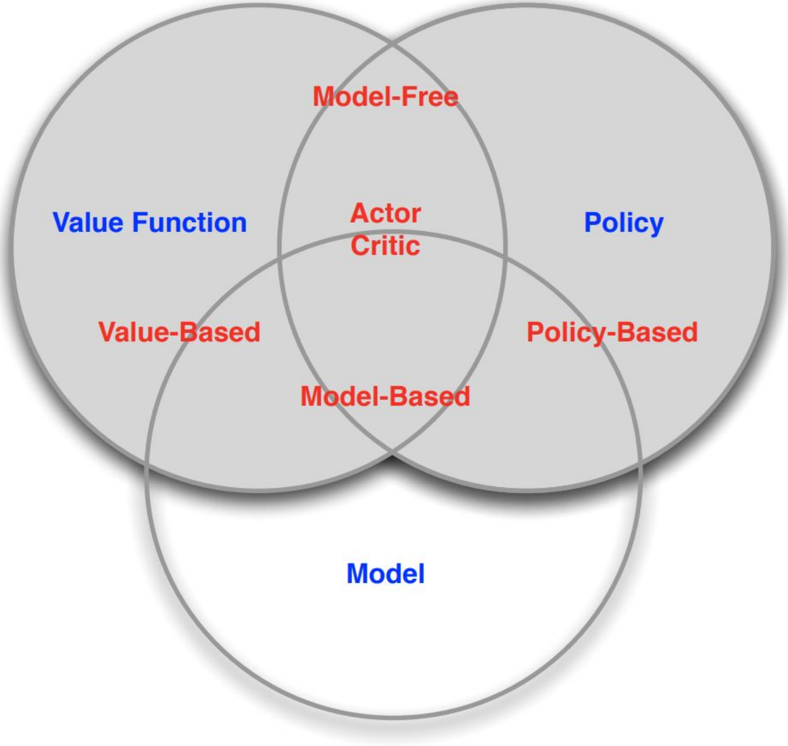
PILCO



guided policy search



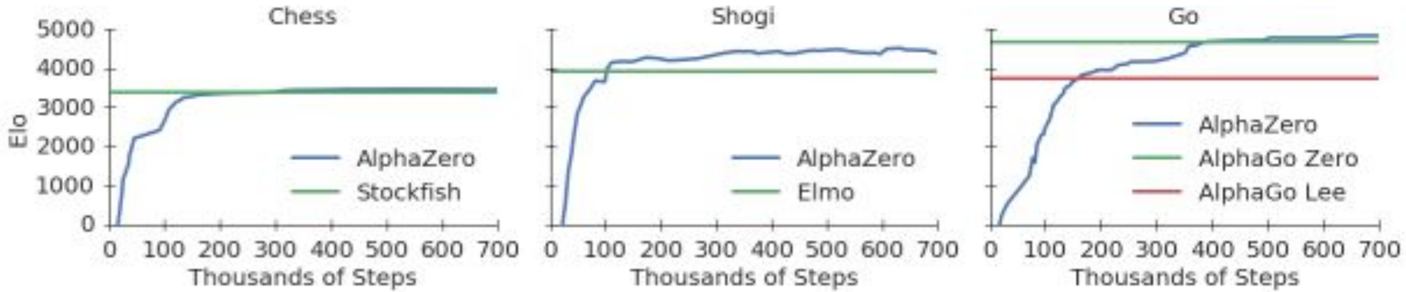
RL Agent Taxonomy



Challenges in reinforcement learning

- 200 million frames per game
- 40 days of human playing time

Challenges in reinforcement learning



- 5 - 23 million games
- 300-1000 years of human playing time

Challenges in reinforcement learning

OpenAI Five play copies of itself … **180 years** of gameplay data **each day** … **consuming 128,000 CPU cores and 256 GPUs**. … reward which is positive **when something good has happened** (e.g. an allied hero gained experience) and negative when something bad has happened (e.g. an allied hero was killed). ... applies our **Proximal Policy Optimization** algorithm ...

Learning

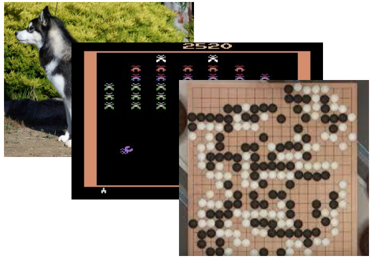
- How can we learn about the environment effectively?
    - How to act optimally under the current history
    - Learn about the rules of the game.
    - Learn about how the game state are affected by the agent’s action.
    - Exploration vs exploitation.

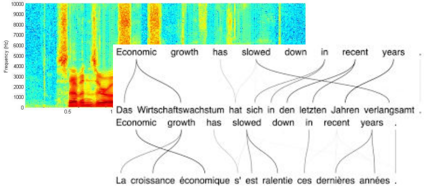
Planning

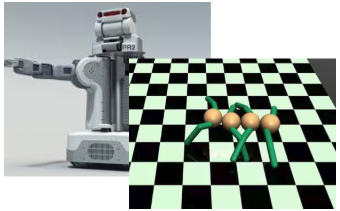
- If the environment model is known
  - How to act optimally under the model without interacting with the environment.
  - Decide when and how to learn in the model vs when to update the history from the environment.
  - i.e. reasoning, introspection, thoughts, search

Inductive bias and generalization

Inputs:







Inductive bias:

Convolutional Neural Networks (CNNs) for spatial reasoning

Recurrent Neural Networks (RNNs) and attention mechanism for sequential reasoning

Graph neural networks for motor skills and optimal control

How to design neural network policy, value function and model that can generalize to many environments.

Learning from experts

- Learning the environment model as well as the optimal behaviour is the Holy Grail of RL.
    - It can be very challenging, so we may consider additional learning signals.
- Learning from demonstrations.
    - First vs third person imitation learning.
    - Inverse reinforcement learning
- Learning from additional goal specification.
    - Learn to solve subgoals, divide and conquer.
    - Learn from expert preference.

Multi-agent system

- Multi-agent environments
  - Self-play
  - Cooperation vs competition
  - Coopetition
  - Population-based training
  - Evolutionary algorithms

Suggest reading

- Introduction: Sutton & Barto, Chapter 1-2
- Other lecture slides:
  - Berkeley Deep RL, Levin, Abbeel
  - UCL Deep RL, Silver
- Dynamic programming: Sutton & Barto, Chapter 3-5

Prerequisites

- CSC411, CSC412, CSC321 or equivalent
- Coursera
  - Geoff Hinton on Coursera
  - Andrew Ng on Coursera

Logistics

Seminar presentation

- We have about 8 weeks for seminar presentation. There is going to be major theme each week. E.g. Natural policy gradient, Imitation learning, Multi-agent systems and Inverse RL.
- 10 mins per student. No more than 8 slides. 5 mins for questions.
- Focus on the main ideas, previous works and future directions
- Coordinate with the other students presenting that week

Logistics

Course project

- Form a group, indicate group member contributions
- Submit proposals and final reports in NIPS format
- Project proposal 2 pages excluding references
- Work with the TAs to refine the project ideas
- 10 mins project presentation during the last two lectures
- Final course project report 4-8 pages excluding references