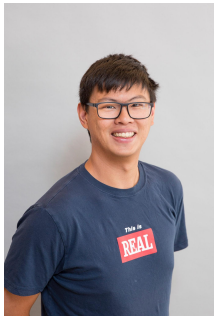
Reinforcement Learning for Games

Timothy Lillicrap & Blake Richards

Who made this possible?



Raymond Chua
McGill Uni and Mila

Mandana Samiei
McGill Uni and Mila

The tutorial repo is based on the mcts repo created by Surag Nair and colleagues.

Outline

1. Conceptual & concrete goals.
2. A bit of history.
3. What is RL? What is planning?
4. From AlphaGo to AlphaZero
5. Background for tutorials
6. Tutorials #1-10
7. What can't our algorithms do?
8. Planning with a learned model

Conceptual Goals

1. Understand how to use reinforcement learning and planning to solve 2-player zero-sum games.

2. Understand how to train policy and value functions for games.

3. Understand how value and policy functions can be used to play games and plan.

Along the way, build appreciation for how these machine learning and game theory concepts map onto ideas in neuroscience.

Concrete Goals

1. Create an Othello player that uses planning, in combination with value and policy networks, to play games.

2. Show that when pitted head to head, planning offers an advantage over players that does not plan.

A bit of history

In the 1990s researchers created chess playing AI that took games from the best chess players in the world.

Deep Blue made use of custom hardware to run the alpha-beta planning algorithm and famously bested Garry Kasparov, the world champion at the time.

However, Deep Blue could not easily be generalized to solve other complex board games.

It took almost 20 years more before AI was able to master the game of Go. Why?

Limitations of Deep Blue

Deep Blue relied on two key features of chess:

1. Effective board position value judgements formalized by expert human players.
2. An action space that was small enough that planning algorithms could be exhaustive enough.

These features limit the applicability of Deep Blue's algorithms.
For example, the board game Go has a *much* larger action space, and the value of board positions are less easily made explicit by experts.

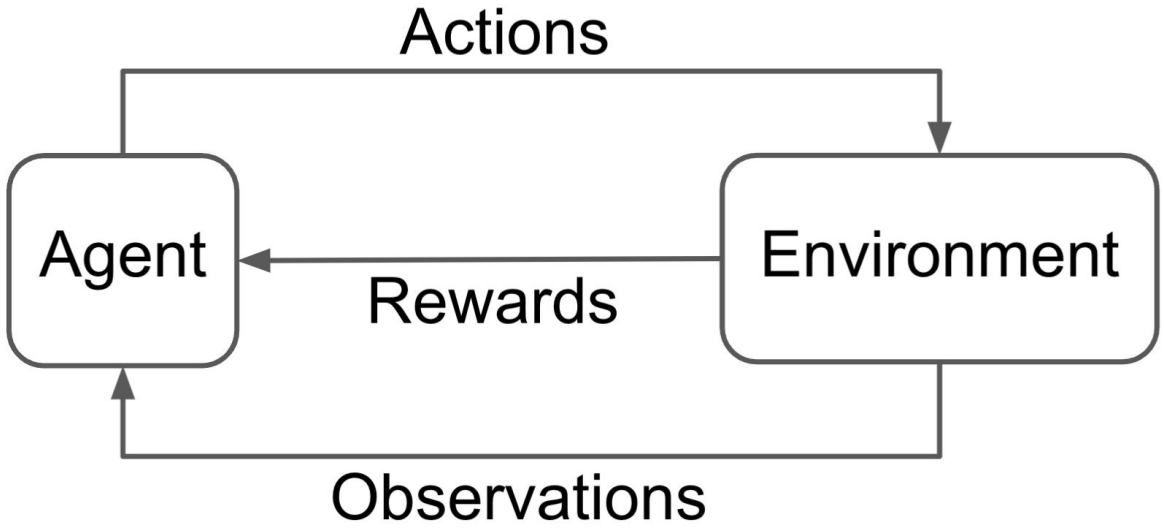
Combining planning with learning

In 2016 AlphaGo overcame both of these problems to best a top player, Lee Sedol, in a full game of Go.

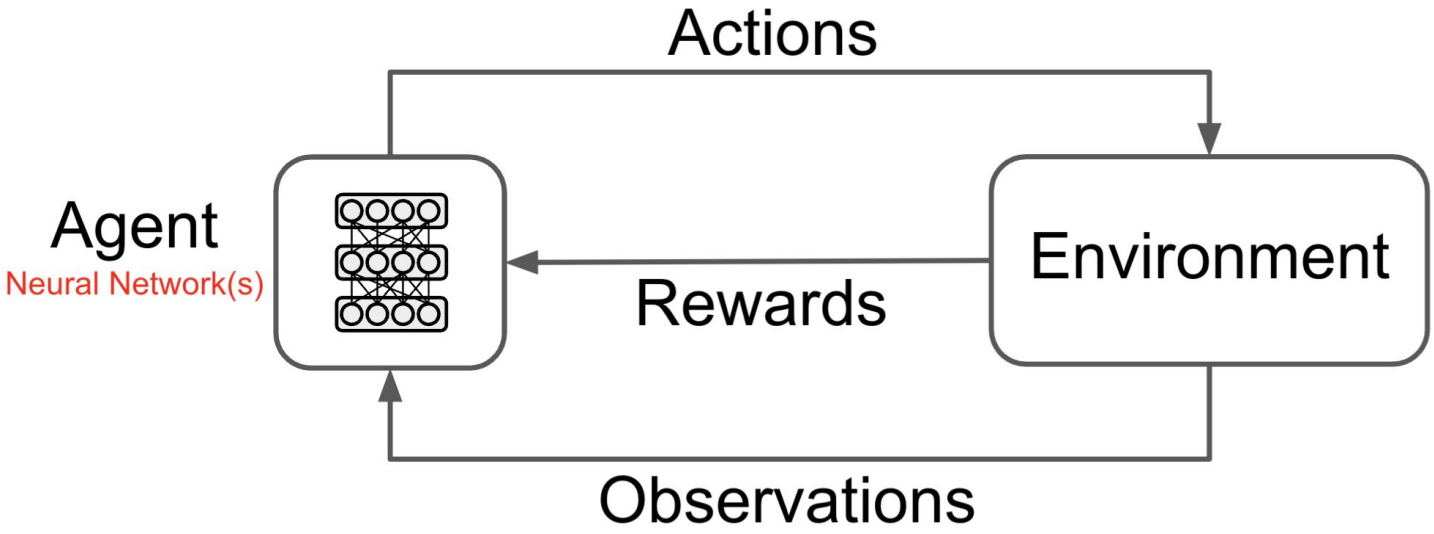
To do so, it combined a planning algorithm with reinforcement learning in order to overcome the two limitations of Deep Blue:

1. Instead of asking experts, AlphaGo learned how to value board positions by looking at games and their outcomes.
2. AlphaGo learned a "policy prior" by predicting expert moves, allowing it to massively shrink the necessary search space for planning.

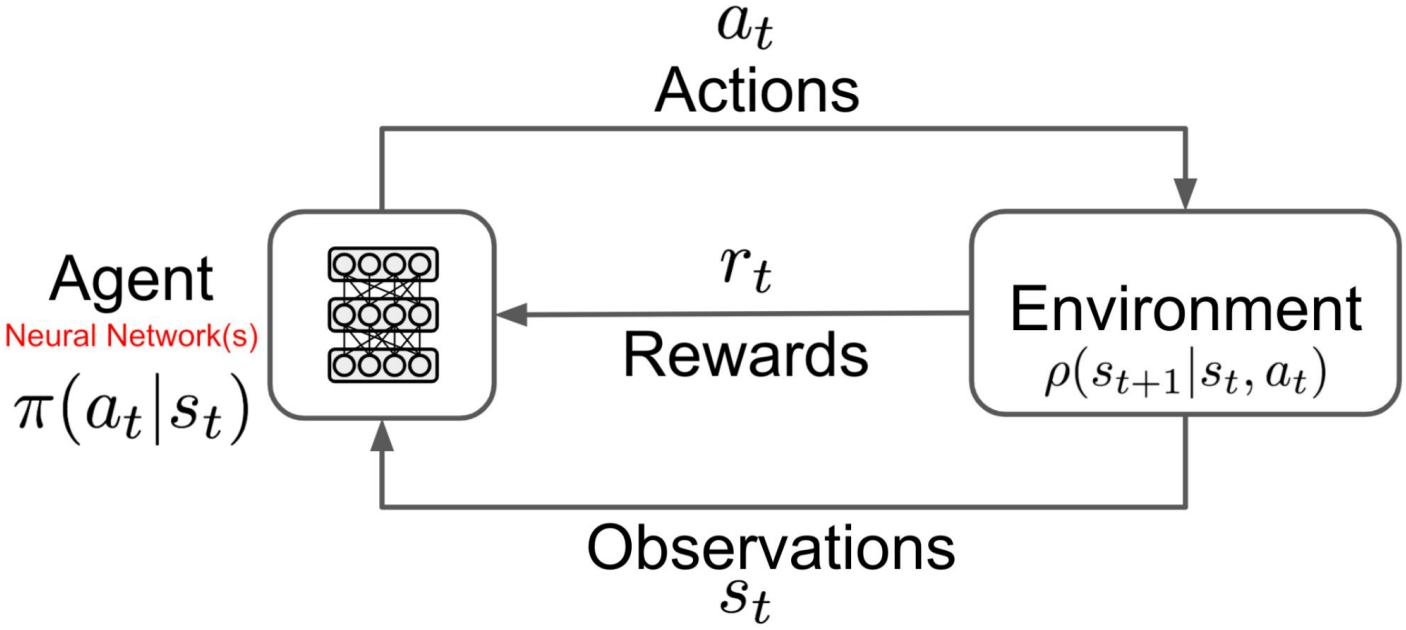
What is reinforcement learning?



What is deep reinforcement learning?



Formalizing the agent-environment loop



A single trial

$$
\begin{array}{ccccc}
r_0, & r_1, & r_2, & \ldots, & r_T \\
a_0, & a_1, & a_2, & \ldots, & a_T \\
\pi(a_0|s_0), & \pi(a_1|s_1), & \pi(a_2|s_2), & \ldots, & \pi(a_T|s_T) \\
s_0, & s_1, & s_2, & \ldots, & s_T \\
 & \rho(s_1|s_0,a_0), & \rho(s_2|s_1,a_1), & \ldots, & \rho(s_T|s_{T-1},a_{T-1})
\end{array}
$$

Time ⟶

Probability of trajectory τ

$$p_\theta(\tau) = \rho(s_0) \prod_{t=0}^{T} \rho(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t)$$

Measuring outcomes

Return for a single trial:

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$$

Objective function:

$$J(\theta) = \int_{\mathbb{T}} p_\theta(\tau) R(\tau) d\tau$$

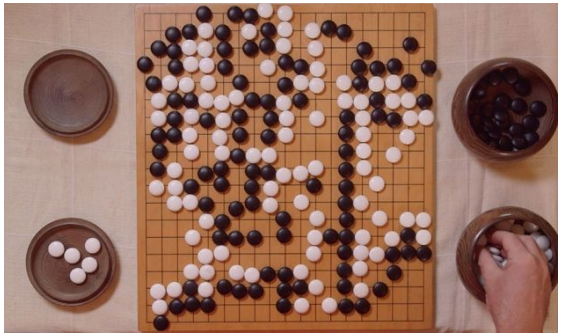
What is planning?

In machine learning, planning is the process of predicting some aspect of the future by thinking ahead.

Planning can take myriad forms, but perhaps the clearest examples involve using a single-step model of the environment to simulate actions and state changes in "imagination".

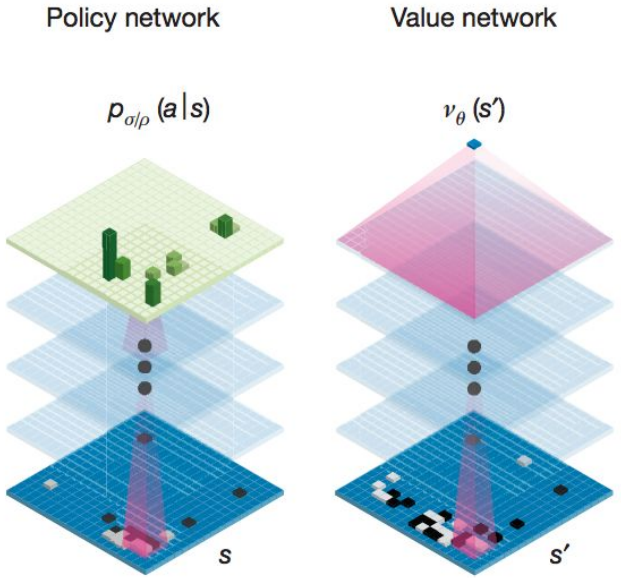
$$\rho\left(s_{t+1} \middle| s_t, a_t\right)$$

Combining Deep Networks with Planning



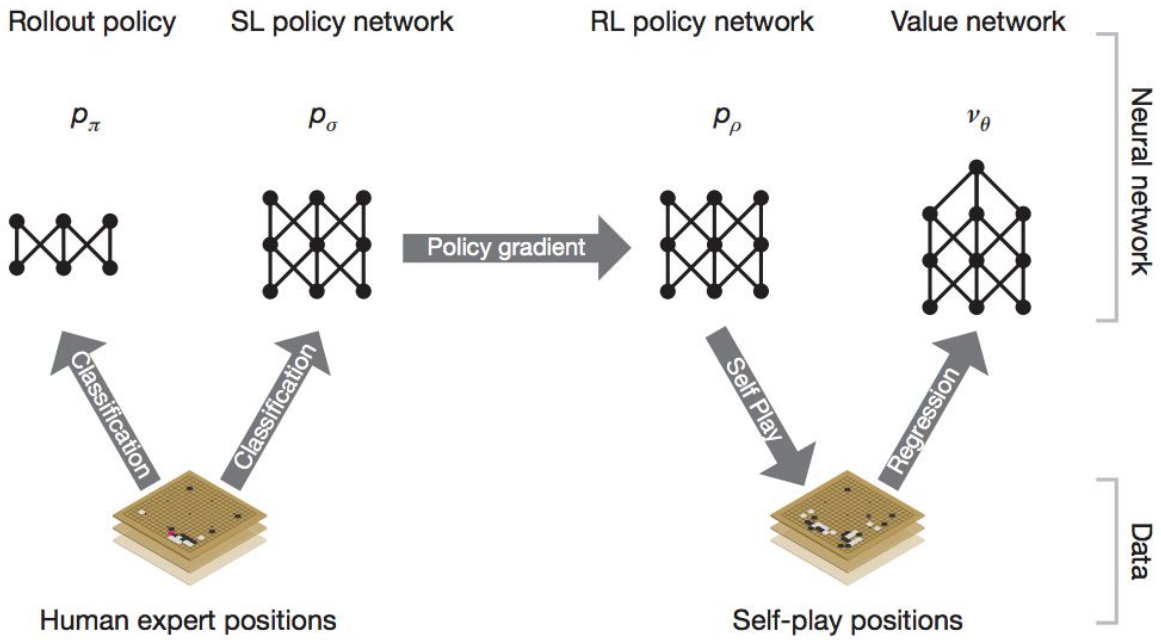
$$\rho(s_{t+1}|s_t, a_t)$$

Use environment model to plan!



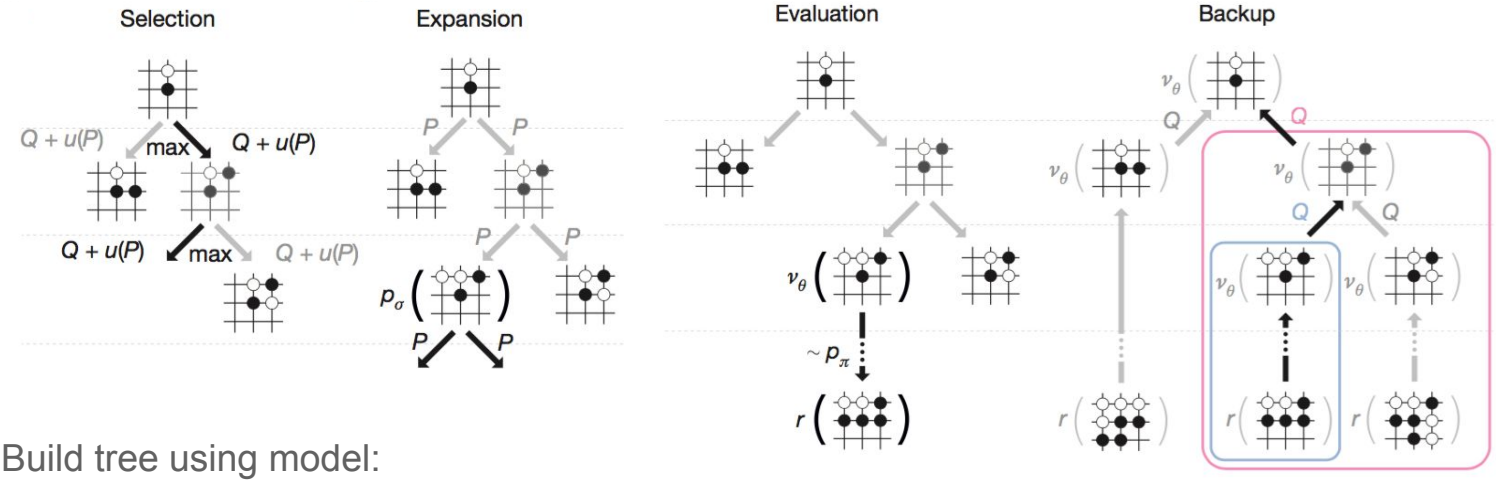
Silver, Huang et al., *Nature*, 2016

Training Policy and Value Networks



Silver, Huang et al., *Nature*, 2016

Plan with an environment model & MCTS



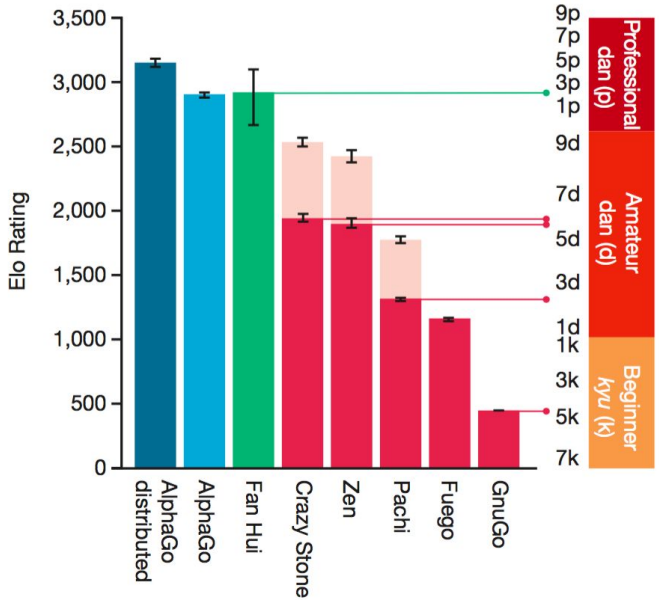
MCTS
=
Monte Carlo
Tree Search

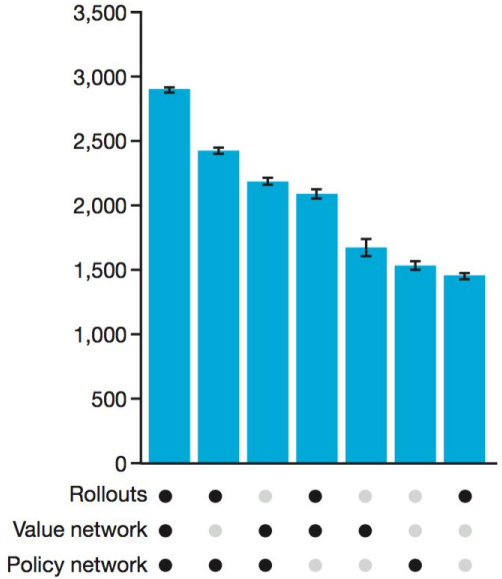
Build tree using model:

$$\rho(s_{t+1}|s_t, a_t)$$

Silver, Huang et al., *Nature*, 2016

Training Policy and Value Networks





Silver, Huang et al., *Nature*, 2016

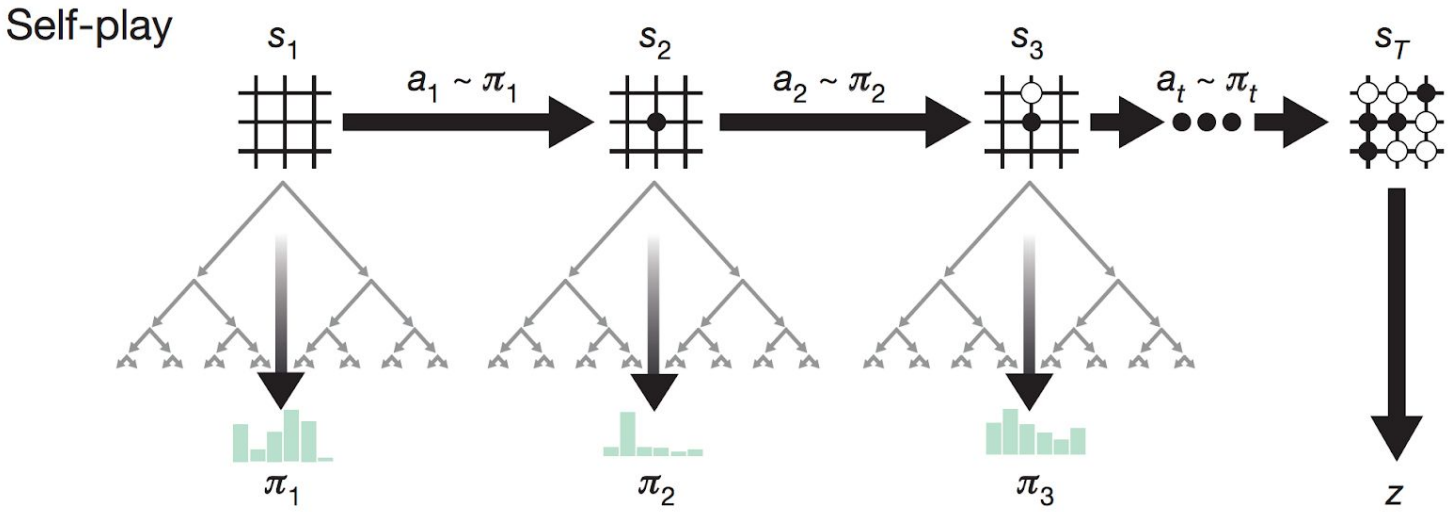
From AlphaGo to AlphaZero

The AlphaGo algorithm quickly evolved from an algorithm that used engineers in the design and training loop, to an algorithm that learned to play Go automatically from self-play. This algorithm was called AlphaGoZero.

Then, since AlphaGo was based on learning rather than expert opinions, it was possible to quickly generalize further.

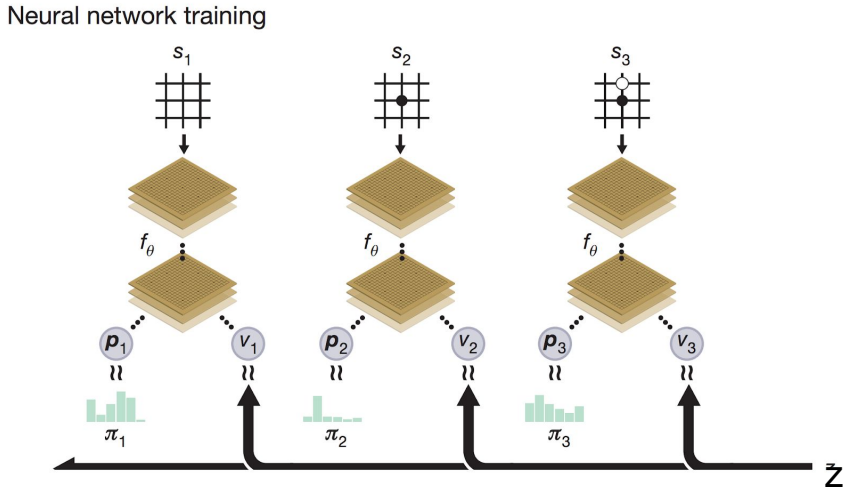
AlphaZero was able to learn Go, chess, and shogi from scratch with no human knowledge about the game except for the rules of the game.

Learning from self-play



Silver, Schrittwieser, Simonyan, et al. *Nature*, 2017

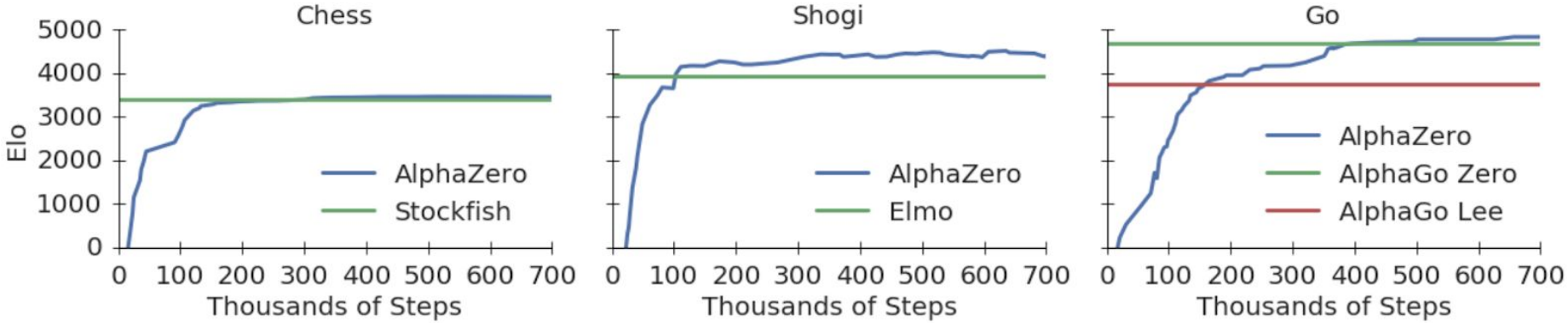
Learning from self-play



$$(\boldsymbol{p}, v) = f_\theta(s) \text{ and } l = (z - v)^2 - \boldsymbol{\pi}^{\mathrm{T}} \log \boldsymbol{p} + c\|\theta\|^2$$

Silver, Schrittwieser, Simonyan, et al. *Nature*, 2017

Learning from self-play: chess, Go, shogi



Silver, Hubert, Schrittwieser, et al. *Nature*, 2017

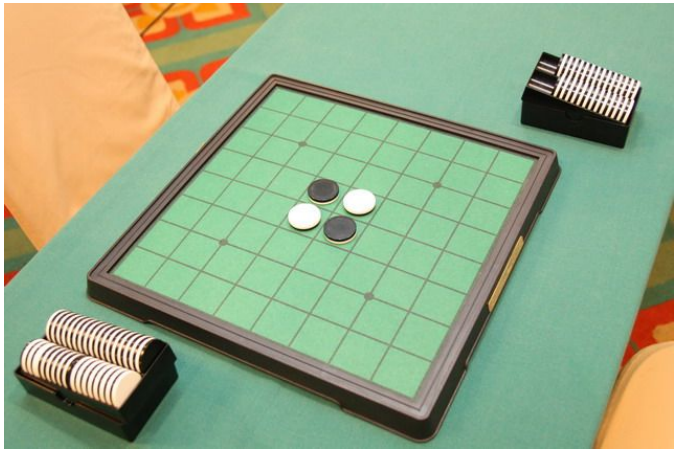
Background for tutorials: Othello/Reversi

We're going to build an Othello player. Othello is:

1. Simpler than Go or Chess.

2. Still a challenging and interesting game with a huge number of possible states.

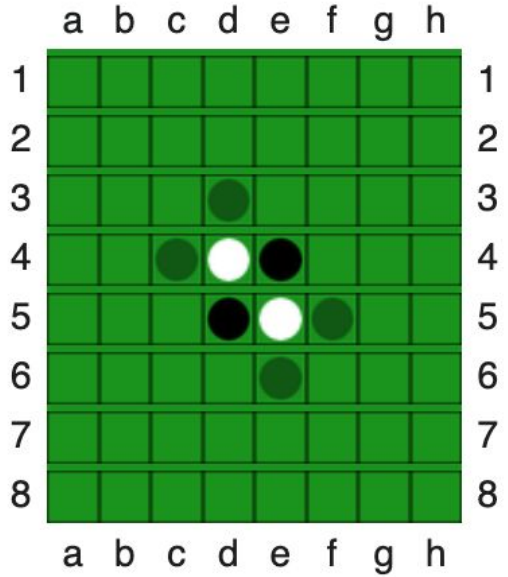
3. Can be played on smaller (6x6) board configuration for speed of simulations and learning.



The rules of Othello/Reversi

1. Each turn a player (black or white) places a single piece on the board.
2. Each piece must be placed on the board and so that there exists at least one straight (horizontal, vertical, or diagonal) occupied line between the new piece and another piece of the same color, with one or more contiguous pieces of the opponent between them.
3. The pieces in between are flipped over.
4. The player with the most pieces on the board when there are no more legal moves to play wins.

You can play here: https://www.othelloonline.org/



Tutorial #1
A game loop for RL

Let's play Othello!

Goal: Setup a game loop with two players for reinforcement learning experiments.

What do we need to learn to play a turn based game?

Game loop for a single episode:

```
while (not finished):
    Player 1 chooses move
    Step board state
    Player 2 chooses move
    Step board state

compute episode return R
```

A single episode can be viewed from 2 perspectives. From the perspective of Player 1 we have the following data:

$s_0, a_0, s_1, a_1, s_2, a_2, s_3, a_3, ..., s_T, a_T, R$

Creating a random player

For a game with a discrete set of possible moves:

1. Query the game to find which actions are legal and how many there are (*N*).
2. Set the probability of each of these actions to 1/*N*.
3. Sample a move from a multinomial on *N* possible actions.
4. Return the chosen action.

$$\pi_\theta(a_t|s_t)$$

Exercise

- Call a game loop for Othello.
- Complete an agent that plays *random moves*.
- Generate games including wins and losses.