Reinforcement Learning for Games

Timothy Lillicrap & Blake Richards

Tutorial #6
Plan using Monte Carlo rollouts

Where will imagination take us?

Goal: Learn the core idea behind using simulated rollouts to understand the future.

Imagining the future using a model

We can string together calls to our policy function with calls to a model of the game to imagine states in the future.

Then, we can call our value function to estimate how good they are:

$$s_t \xrightarrow[\pi_\theta(a|s)]{} a_t \xrightarrow[p(s'|s,a)]{} s_{t+1} \xrightarrow[\pi_\theta(a|s)]{} a_{t+1} \xrightarrow[p(s'|s,a)]{} s_{t+2} \rightarrow V(s_{t+2})$$

Exercise

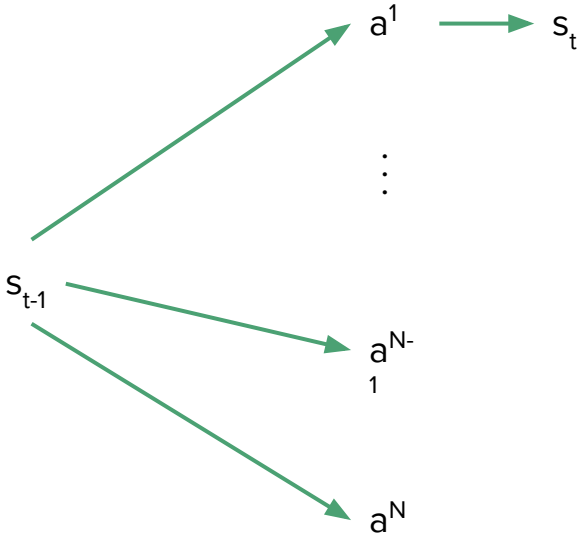
- Complete a loop to run a Monte Carlo simulation using the policy network.
- Compute the value function at the end of *k=3* steps for a given board position.

Tutorial #7
Play with planning

Plan to win!

Goal: Learn how to use simple Monte Carlo planning to play games.

Use Monte Carlo imaginations to play



$$s_t \xrightarrow[p(s'|s,a)]{\pi_\theta(a|s)} a_t \rightarrow s_{t+1} \xrightarrow[p(s'|s,a)]{\pi_\theta(a|s)} a_{t+1} \rightarrow s_{t+2} \rightarrow V(s_{t+2})$$

$$s_t \xrightarrow[p(s'|s,a)]{\pi_\theta(a|s)} a_t \rightarrow s_{t+1} \xrightarrow[p(s'|s,a)]{\pi_\theta(a|s)} a_{t+1} \rightarrow s_{t+2} \rightarrow V(s_{t+2})$$

Average Values

Use Monte Carlo imaginations to play

Now that we can imagine the future and estimate its value, we'll use this capacity to create an agent that plans:

for i in 1 to *k*:

Choose the *i*th ranked action a^i for the root state s_t according to our policy.
Run *N* Monte Carlo rollouts from s_{t+1} follows from the application of a^i
Average the estimated values for each rollout to get: V_i^{AVG}
Build an array of $[V_i^{AVG}, a^i]$ pairs.

To act, choose the action associated with the highest value.

Exercise

- Incorporate Monte Carlo simulations into an agent.
- Do this by choosing the best 3 actions suggested by the policy function and running 10 MC simulations for a depth of 3 from each of these.
- Choose the best of these 3 moves according to the MC rollouts (i.e. which one has the highest estimated value).
- Run the resulting player versus the random, value-based, and policy-based players.

Where to next?

A glimpse of the future

Goal: Understand some of the open problems for the next phase of learning+planning algorithms.

What can't our algorithms do?

The last half decade has given rise to a growing set of powerful reinforcement learning algorithms that are capable of competing with top human players for a widening set of games.

Variations of these are being applied throughout game playing, and we're beginning to see strong solutions to Poker, Starcraft, DotA, Hanabi, and others.

There are, however, clear limitations to the AlphaX algorithms.

What can't our algorithms do?

Limitations of the AlphaX algorithms include:

1. They use a perfect model of the environment.
2. MCTS plans step-by-step.
3. MCTS plans in a strictly forward mode.

Humans appear to have the capacities to plan in ways that AlphaX algorithms cannot:

1. They learn models of the environment.
2. They appear to plan in temporally abstract jumps.
3. They appear to be able to plan in mixes of forward and backward modes.

We have made some progress on #1

Limitations of the AlphaX algorithms include:

1. ***They use a perfect model of the environment.***
2. MCTS plans step-by-step.
3. MCTS plans in a strictly forward mode.

Humans appear to have the capacities to plan in ways that AlphaX algorithms cannot:

1. ***They learn models of the environment.***
2. They appear to plan in temporally abstract jumps.
3. They appear to be able to plan in mixes of forward and backward modes.

Learning models of the world

Reinforcement learning algorithms are increasingly making use of learned models of the environment. The essential idea is to train a model that makes predictions about about how actions lead to future states, values, actions, or rewards.

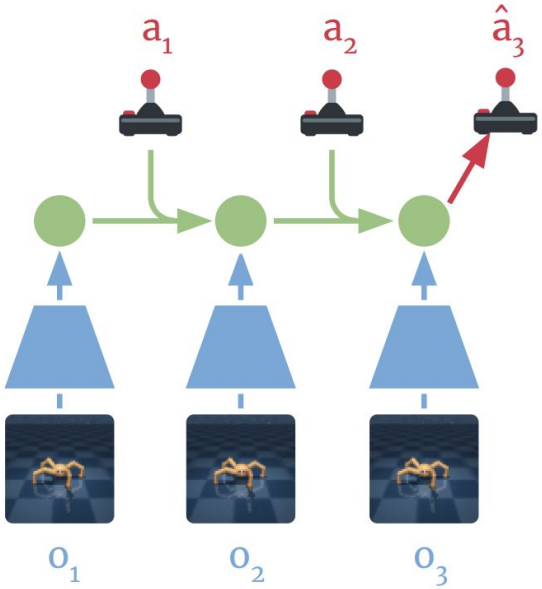
Such models can be used in a variety of ways. We highlight two:

1. *Dreamer* learns a model of the world in order to be able to rollout experience in imagination and then learn behaviour from these fictitious experiences.
2. *MuZero* builds a model of the world that is tailored to the requirements of MCTS, enabling planning in a learned state space.

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

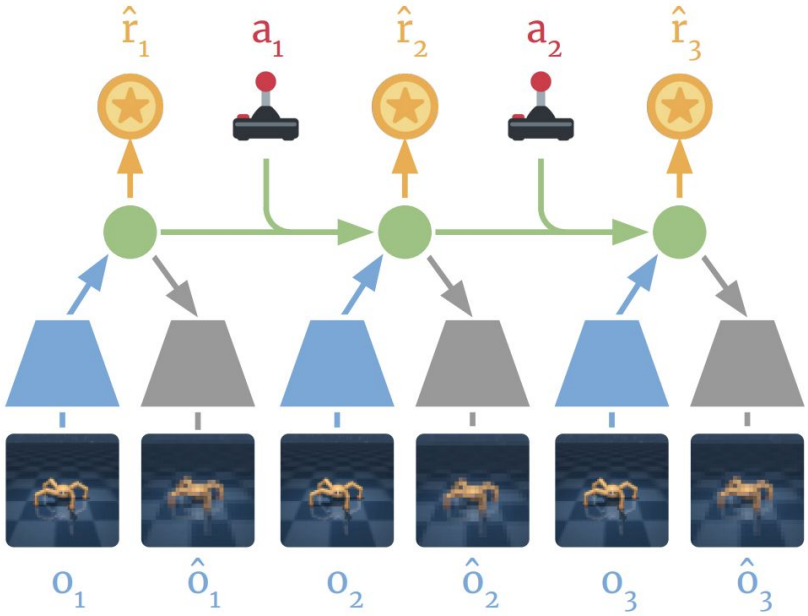
Hafner, Lillicrap, Ba, Norouzi *arXiv*, 2020

Dreamer: Act in the environment



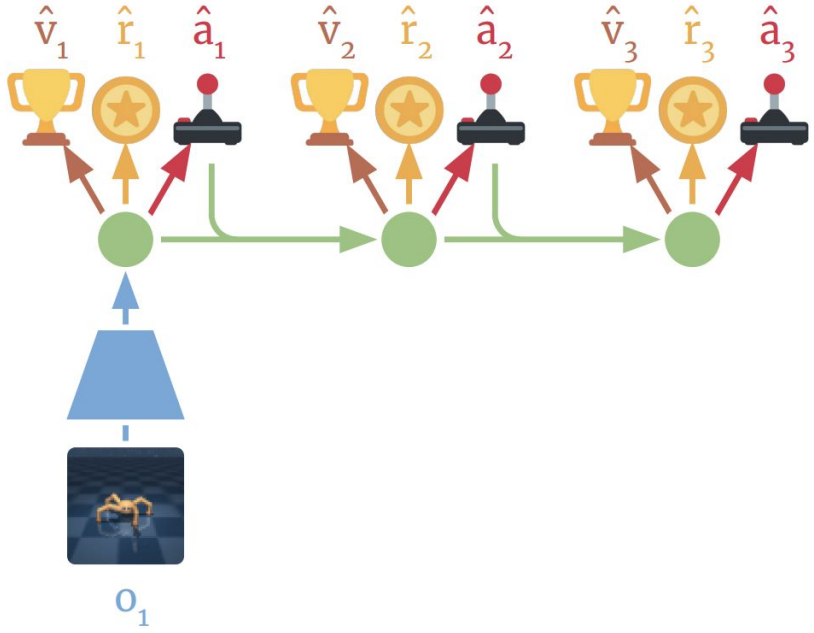
Hafner, Lillicrap, Ba, Norouzi *arXiv*, 2020

Dreamer: Learn dynamics from experience



Hafner, Lillicrap, Ba, Norouzi *arXiv*, 2020

Dreamer: Learn behaviour in imagination



Learn behaviour in imagination

Hafner, Lillicrap, Ba, Norouzi *arXiv*, 2020

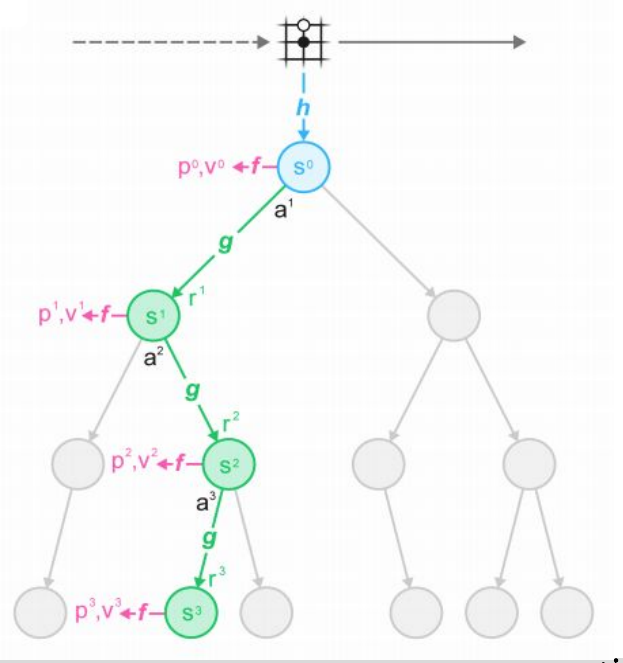
MuZero: Learning a model for planning

AlphaGo, AlphaGo Zero & AlphaZero use perfect models of the environment to plan.

- MuZero instead builds a model of the environment that can be used to search.
- Unlike most model-based work MuZero does not try to model environment transitions.
- Rather, it focuses on predicting the future reward, value, and policy.
- These are the essential components required by MCTS.

Schrittwieser, Antonoglou, Hubert, et al. *arXiv*, 2020

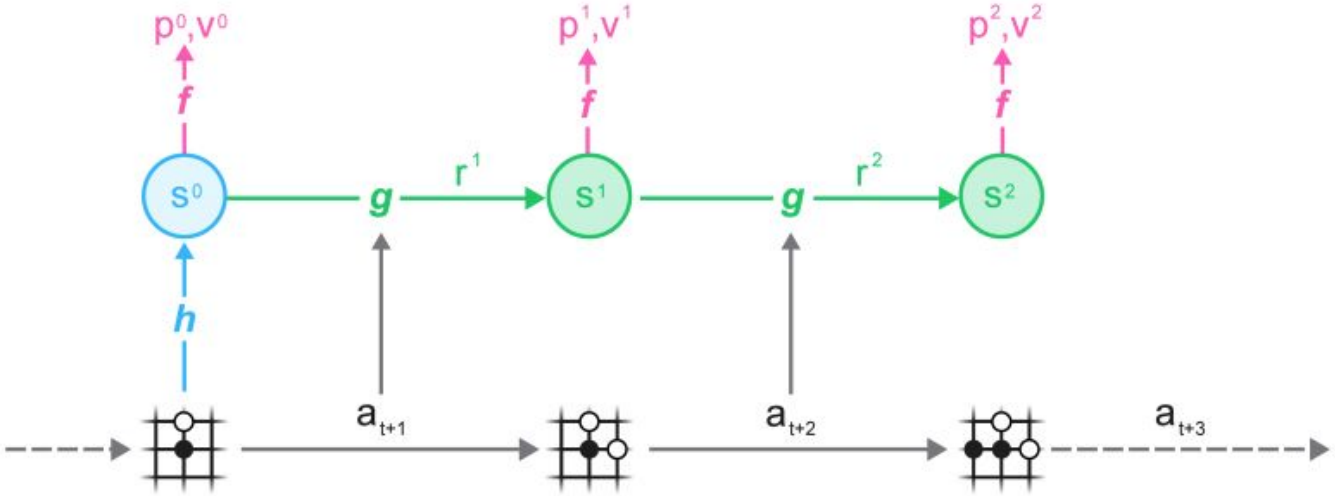
MuZero: Model design



Note that the transposition tables used in *AlphaGo* and *AlphaZero* are not possible in *MuZero*.

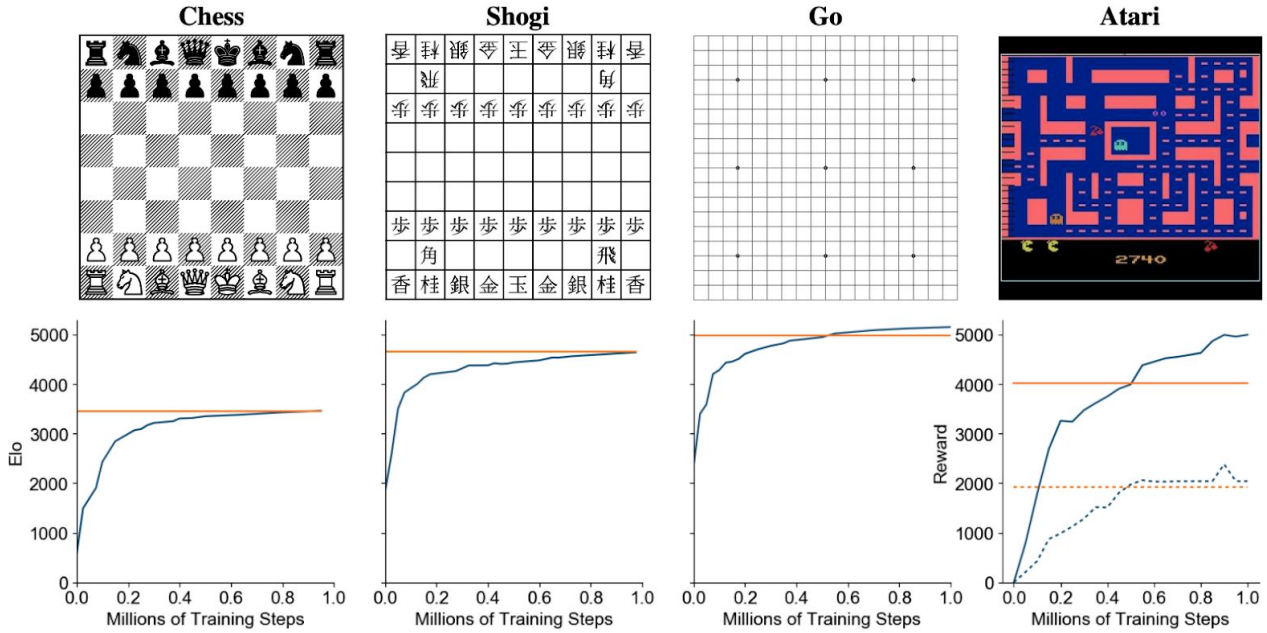
Schrittwieser, Antonoglou, Hubert, et al. *arXiv*, 2020

MuZero: How it learns



Schrittwieser, Antonoglou, Hubert, et al. *arXiv*, 2020

MuZero: Results



Schrittwieser, Antonoglou, Hubert, et al. *arXiv*, 2020

Some lessons learned

1. There is are no neat distinction between model-free and model-based reinforcement learning. There's a huge space of possibilities and we're just beginning to explore what can be done.

2. It's unclear precisely why model-based algorithms are starting to work well. Larger networks appear to be part of the equation.

3. The environment/problem studied is crucial for results. Planning at run-time has been convincingly demonstrated to be important for games like Go, chess, and shogi. The advantage of planning is less clear for games like Atari.

Where to next?

We're still missing algorithms that have the flexibility of human planning.

Most of our algorithms still rely on forward-mode planning and rollouts.

And, none of them exhibit the kinds of state, action, and temporal abstraction exhibited during human planning and behaviours.

These are exciting research questions with deep connections to the neurosciences!

Tutorial #8
Plan with MCTS

Dendritic imaginations!

Goal: Understand the core ideas behind Monte Carlo Tree Search.

A tree structure can improve planning

Naive Monte Carlo rollouts are conceptually simple, but can be wasteful and don't take into account the fact that the other opponent may be maximizing:

$$s_t \xrightarrow{\pi_\theta(a|s)} a_t \xrightarrow[p(s'|s,a)]{} s_{t+1} \xrightarrow{\pi_\theta(a|s)} a_{t+1} \xrightarrow[p(s'|s,a)]{} s_{t+2} \rightarrow V(s_{t+2})$$

$$s_t \xrightarrow{\pi_\theta(a|s)} a_t \xrightarrow[p(s'|s,a)]{} s_{t+1} \xrightarrow{\pi_\theta(a|s)} a_{t+1} \xrightarrow[p(s'|s,a)]{} s_{t+2} \rightarrow V(s_{t+2})$$

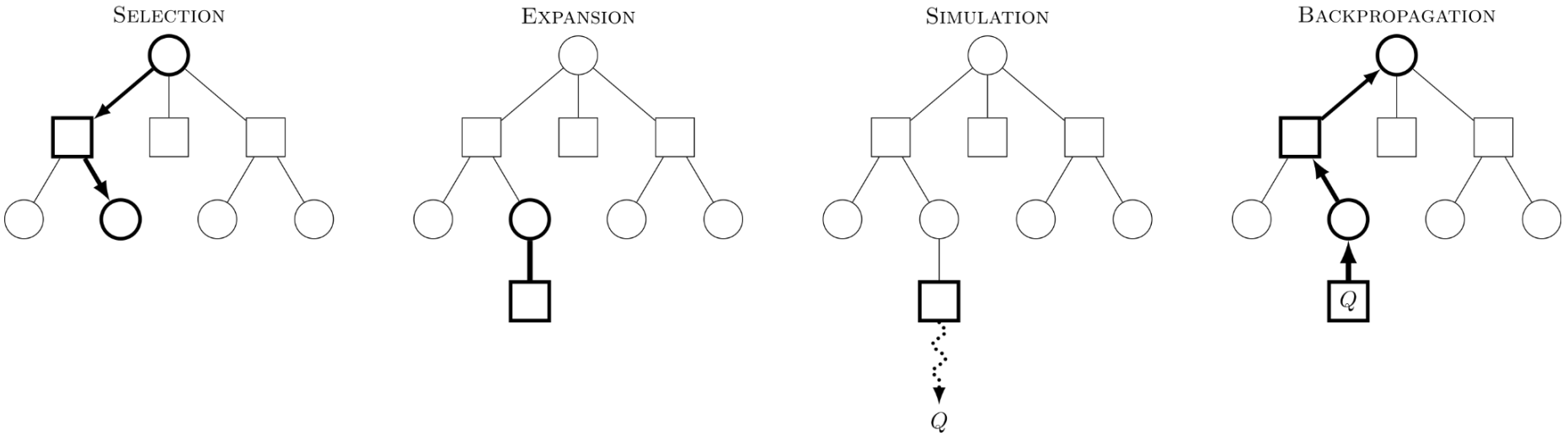
$$s_t \xrightarrow{\pi_\theta(a|s)} a_t \xrightarrow[p(s'|s,a)]{} s_{t+1} \xrightarrow{\pi_\theta(a|s)} a_{t+1} \xrightarrow[p(s'|s,a)]{} s_{t+2} \rightarrow V(s_{t+2})$$

A tree structure can improve planning

Monte Carlo Tree Search can be used to:

1. Re-use previously computed policy and value information.
2. Take into account the fact that the opponent is adversarial.
3. Make use of explore / exploit trade-offs during search.

A tree structure can improve planning

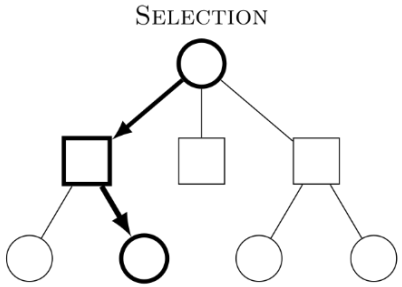


Choosing actions during selection

MCTS efficiently trades exploration and exploitation over the course of search:

$$\arg\max_a (Q(s_t, a) + u(s_t, a))$$

$$u(s, a) = c_{\text{puct}} P(s, a) \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$$



- Initially prefers actions with higher prior and low counts.
- Asymptotically prefers actions with high estimated value.
- UCB variant algorithm developed in the context of multi-armed bandits.

Exercise

- Finish the MCTS planner by using UCT to select actions to build the tree.
- Deploy the MCTS planner to build a tree search for a given board position, producing value estimates and action counts for that position.

Tutorial #9
Play with MCTS

Beat all the players!

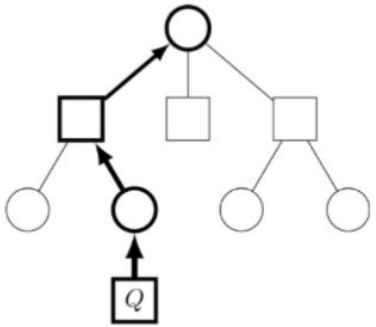
Goal: Understand how to use the results of MCTS to play games.

Using MCTS to play

For every simulation down the tree, we count the number of times that we have visited a particular state and action.

Search focuses on those states and actions that are deemed most likely to lead to high values in the future.

So, to create an agent, we only have to choose the action with the largest count from those available at the root state!



Exercise

- Create an agent that plans by selecting the root action with the highest simulation count following MCTS simulations.
- Play games against other agents.
- If time permits, vary the amount of thinking time allowed (i.e. the number of simulations / rollouts) to plan a move.