TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2020

AlphaZero Background Algorithms

AlphaGo Fan (October 2015)

AlphaGo Defeats Fan Hui, European Go Champion.

AlphaGo Lee (March 2016)

AlphaGo Zero vs. Alphago Lee (April 2017)

AlphaGo Lee:

- Trained on both human games and self play.
- Trained for Months.
- Run on many machines with 48 TPUs for Lee Sedol match.

AlphaGo Zero:

- Trained on self play only.
- Trained for 3 days.
- Run on one machine with 4 TPUs.
- Defeated AlphaGo Lee under match conditions 100 to 0.

AlphaZero Defeats Stockfish in Chess (December 2017)

AlphaGo Zero was a fundamental algorithmic advance for general RL.

The general RL algorithm of AlphaZero is essentially the same as that of AlphaGo Zero.

Some Background

Early Computer Chess

First computer chess algorithm (min-max tree search) — Claude Shannon, 1949

α-β pruning — various originators (including John McCarthy) circa 1960.

α-β pruning was the backbone of all computer chess before AlphaGo 2015.

The game of Go is clearly not approachable by these methods.

Monte-Carlo Tree Search (MCTS)

Brugmann (1993)

First major advance in computer Go.

To estimate the value of a position (who is ahead and by how much) run a cheap stochastic policy to generate a sequence of moves (a rollout) and see who wins.

Select the move with the best rollout value.

(One Armed) Bandit Problems

Robbins (1952)

Bandit problems were studied in an independent line of research.

Consider a set of choices. The standard example is a choice between different slot machines with different but unknown expected payout. The "phrase one-armed bandit" refers to a slot machine.

But another example of choices might be the moves in a game.

Bandit Problems

Consider a set of choices where each choice gets a stochastic reward.

We can select a choice and get a reward as often as we like.

We would like to determine which choice is best using a limited number of trials.

The Upper Confidence Bound (UCB) Algorithm

Lai and Robbins (1985)

For each action choice (bandit) a, construct a confidence interval for its average reward based on n trials for that acrtion.

$$\mu(a) \in \hat{\mu}(a) \pm 2\sigma(a)/\sqrt{n(a)}$$

Always select

$$\underset{a}{\operatorname{argmax}} \;\; \hat{\mu}(a) + 2\sigma(a)/\sqrt{n(a)}$$

The Upper Confidence Tree (UCT) Algorithm

Kocsis and Szepesvari (2006), Gelly and Silver (2007)

The UCT algorithm grows a tree by running "simulations".

Each simulation descends into the tree to a leaf node, expands that leaf, and returns a value.

In the UCT algorithm each move choice at each position is treated as a bandit problem.

We select the child (bandit) with the lowest upper bound as computed from simulations selecting that child.

Bootstrapping from Game Tree Search

Vaness, Silver, Blair and Uther, NeurIPS 2009

In bootstrapped tree search we do a tree search to compute a min-max value $V_{\mathrm{mm}}(s)$ using tree search with a static evaluator $V_\Phi(s)$. We then try to fit the static value to the min-max value.

$$\Delta\Phi = -\eta\nabla_\Phi \left(V_\Phi(s) - V_{\mathrm{mm}}(s)\right)^2$$

This is similar to minimizing a Bellman error between $V_\Phi(s)$ and a rollout estimate of the value of s but where the rollout estimate is replaced by a min-max tree search estimate.

END