TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2020

The AlphaZero Training Algorithm

The AlphaZero Algorithm

The AlphaZero algorithm is an RL learning method for the case where state transitions are determined by actions.

In principle, the AlphaZero algorithm could be applied to the problem of direct optimization of the BLEU score in machine translation.

Top Level Algorithm

To play a game (against yourself) select a move at each position as the game progresses.

To select a move in a game evaluate the options by growing a search tree data structure.

To grow a tree data structure we start with a tree consisting of just the current position. We then perform a series of "simulations". Each simulation adds one new leaf.

To perform a simulation recursively descend into the tree data structure by selecting a move at each node until a new node (leaf) is generated.

The Value and Policy Networks

Simulations select moves based on value and policy networks.

In AlphaZero the networks are either 20 block or 40 block ResNets and either separate networks or one dual-headed network.

Simulations

Each simulation starts from the root node (the current position) and selects a move at each node until a new leaf is generated.

During a simulation moves are selected by maximizing an upper value as in the UCT algorithm.

Each simulation returns the value $V_{\Phi}(s)$ (from the value network) for the newly generated state (position) s.

The Data Structures

Each node in the tree data structrure stores the following information which is initialized by running the value and policy networks on state s.

- $V_\Phi(s)$ — the value network value for the position s.
- The policy probabilities $\pi_\Phi(s,a)$ for each legal action a.
- The number $N(s,a)$ of simulations that have tried move a from s. This is initially zero.
- For $N(s,a) > 0$, the average $\hat{\mu}(s,a)$ of the values of the simulations that have tried move a from position s.

Simulations and Upper Confidence Bounds

In descending into the tree, a simulation selects the move $\mathrm{argmax}_a\, U(s,a)$ where we have

$$U(s,a) = \begin{cases} \lambda_u\, \pi_\Phi(s,a) & \text{if } N(s,a) = 0 \\ \\ \hat{\mu}(s,a) + \lambda_u\, \pi_\Phi(s,a)/N(s,a) & \text{otherwise} \end{cases}$$

Upper Confidence Bounds

$$U(s,a) = \begin{cases} \lambda_u \; \pi_\Phi(s,a) & \text{if } N(s,a) = 0 \\ \\ \hat{\mu}(s,a) + \lambda_u \; \pi_\Phi(s,a)/N(s,a) & \text{otherwise} \end{cases}$$

The hyperparameter λ_u should be selected so that $U(s,a)$ will typically **decrease** as $N(s,a)$ increases.

In this regime for λ_u, we can think of $U(s,a)$ as an upper confidence bound in the UCT algorithm.

Root Action Selection

When the search is completed, we must select a move from the root position to make actual progress in the game. For this we use a post-search stochastic policy

$$\pi_{s_{\mathrm{root}}}(a) \propto N(s_{\mathrm{root}}, a)^{\beta}$$

where β is a temperature hyperparameter.

Constructing a Replay Buffer

We run a large number of games.

We construct a replay buffer of triples (s, π_s, R) where

- s is a position encountered in a game and hence a root position of a tree search.

- π_s is the distribution on a defined by $P(a) \propto N(s,a)^\beta$.

- $R \in \{-1, 1\}$ is the final outcome of the game for the player whoes move it is at position s.

The Loss Function

Training is done by SGD on the following loss function.

$$\Phi^* \;=\; \operatorname*{argmin}_{\Phi}\; E_{(s,\pi,R)\sim\mathrm{Replay},\; a\sim\pi} \left(\begin{array}{l} (V_\Phi(s) - R)^2 \\ \\ -\lambda_\pi \log \pi_\Phi(a|s) \\ \\ +\lambda_R\; ||\Phi||^2 \end{array}\right)$$

The replay buffer is periodically updated with new self-play games.

END