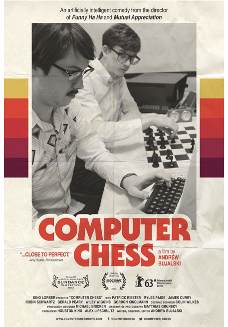
Deep Pepper: An Expert Iteration Based Chess Agent in the RL Setting

Sai Krishna, Kyle Goyette, Ahmad Shamseddin, Breandan Considine, Giancarlo Kerg

Introduction/Motivation

- Computer chess widely studied for more than 40 years.
- Alpha Zero plays chess learning entirely by self-play
  - Tabula rasa shows the power of generalized AI
  - But could embedding knowledge lead to a better chess player overall?
- How can we accelerate training?
  - **Custom feature representation**
    - Substantially reduced feature representation size
    - Uses hand-crafted features developed by human experts
  - **Pretraining the networks**
    - Thanks to Stockfish
  - **Early termination of games**
    - Using Oracle (Stockfish)

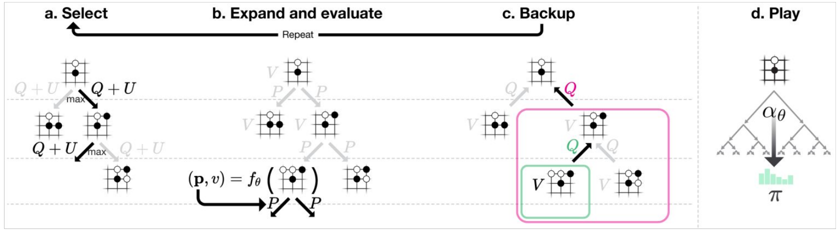


Sources: Bujalski, Computer Chess(Film) 2013,
Albert Silver, “The future is here – AlphaZero learns chess” December 2017

Monte Carlo Tree Search

Each edge stores:

- visit count **N(s,a)**
- Total action-value **W(s,a)**
- Mean action value **Q(s,a)**
- Prior Probabilities **P(s,a)**



Action selection:

$$a_t = argmax_a(Q(s_t, a) + U(s_t, a))$$

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

Expand and Evaluate

- Evaluation: from value network
- Expansion: Initialize edge stats from leaf node

Back up

$$N(s_t, a_t) = N(s_t, a_t) + 1$$

$$W(s_t, a_t) = W(s_t, a_t) + v$$

$$Q(s_t, a_t) = \frac{W(s_t, a_t)}{N(s_t, a_t)}$$

From: Mastering the Game of Go Without Human Knowledge, Credit: DeepMind

Training phase

Algorithm:
Repeat

1. Game Generation via MCTS
   a. Save MCTS policies and game outcome for each state
2. Train network, via backpropagation
3. Test new network against last iteration

$$l = (z - v)^2 - \boldsymbol{\pi}^\top \log \mathbf{p} + c\|\theta\|^2$$

Result at the end of each game

From value network

Strengthened policy obtained from MCTS

From policy network

Regularization

This is roughly Classification-Based Approximate Policy Iteration

Policy Improvement: Each MCTS call strengthens the current policy (predictions in the network) and strengthened policy from MCTS is projected back into network functional space via training.

Policy Evaluation: True game scores using the strengthened policy. Projected into function space via network training.

Mathematical Details

- Relation to CAPI
- **Generalization to multiple actions**

$$\hat{L}_n^{\pi_k}(\pi) \triangleq \int_{\mathcal{X}} \mathbf{g}_{\hat{Q}^{\pi_k}}(x)\mathbb{I}\{\pi(x) \neq \operatorname*{argmax}_{a\in\mathcal{A}} \hat{Q}^{\pi_k}(x,a)\}\, \mathrm{d}\nu_n$$

$$\mathbf{g}_Q(x) \triangleq |Q(x,1) - Q(x,2)| \qquad \text{for all } x \in \mathcal{X}.$$

$$\mathbb{P}_\nu\left(0 < \mathbf{g}_{Q^\pi}(X) \leq \varepsilon\right) \triangleq \int_{\mathcal{X}} \mathbb{I}\{0 < \mathbf{g}_{Q^\pi}(x) \leq \varepsilon\}\, \mathrm{d}\nu(x) \leq c_g\, \varepsilon^{\zeta}.$$

Algorithm CAPI(Π, ν, K)

Input: Policy space Π, State distribution ν, Number of iterations K

Initialize: Let $\pi_{(0)} \in \Pi$ be an arbitrary policy

for $k = 0, 1, \ldots, K-1$ **do**

Construct a dataset $\mathcal{D}_n^{(k)} = \{X_i\}_{i=1}^n,\ X_i \overset{\text{i.i.d.}}{\sim} \nu$

$\hat{Q}^{\pi_k} \leftarrow \text{PolicyEval}(\pi_k)$

$\pi_{k+1} \leftarrow \text{argmin}_{\pi\in\Pi} \hat{L}_n^{\pi_k}(\pi)$ (action-gap-weighted classification)

end for

Experiments And Future Work

- **Experimental Results:**
  - **Trained network for 100 games:**
    - **Wins 76% of the time against a pretrained model**
  - **Pretrained model**
    - **Wins 64% of games against randomly initialized model**
- **Future Work**
  - **Approximate ELO rating**
  - **More training**
  - **Using opening books**
    - Custom opening book from online databases
  - **Gradual shift from MCTS to alpha-beta pruning**
  - **Parallelize MCTS algorithm to speed up gameplay**

Merci Beaucoup