Thinking Fast and Slow with Deep Learning and Tree Search

Thomas Anthony, Zheng Tian, and David Barber
University College London

Alex Adam and Fartash Faghri
CSC2547

Hex

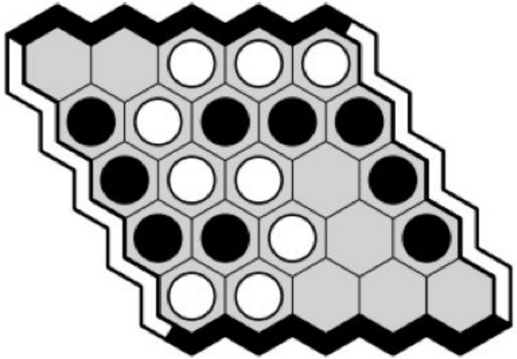


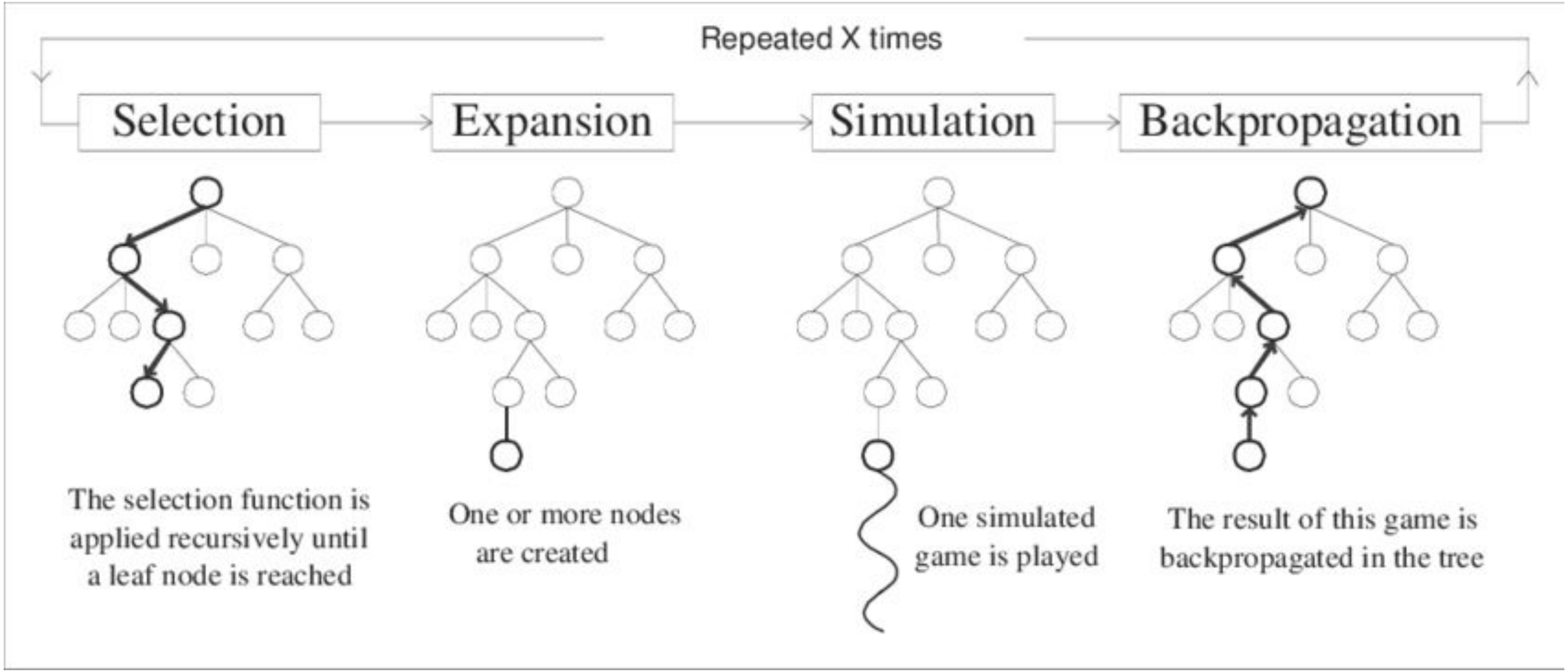
Figure 1: A 5×5 Hex game, won by white. Figure from Huang et al. [8].

What is MCTS

- Tree search algo that addresses limitations of Alpha-Beta Search
- Alpha-Beta worst case explores O(B^D) nodes
- MCTS approximates Alpha-Beta by exploring promising actions and using simulations

1. Select nodes according to $\frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}$
2. At leaf node
   a. If node has not been explored, simulate until end of game
   b. If node has been explored, add child states to tree, then simulate from random child state
3. Update UCT values of nodes along path from leaf to root

MCTS in Action



Why not REINFORCE?

Maximize the expected reward: $\mathbb{E}_{\tau \sim \pi}[R] = \mathbb{E}_{\pi}[r(s, a)]$

Gradient estimator:

$$\hat{g}^{\text{REINFORCE}}[r(s, a)] = \mathbb{E}_{\pi}[r(s, a) \nabla_{\theta} \log \pi(a|s, \theta)]$$

Find policy $\pi(a|s, \theta)$ that maximizes the expected reward.

Why not REINFORCE?

Challenges:

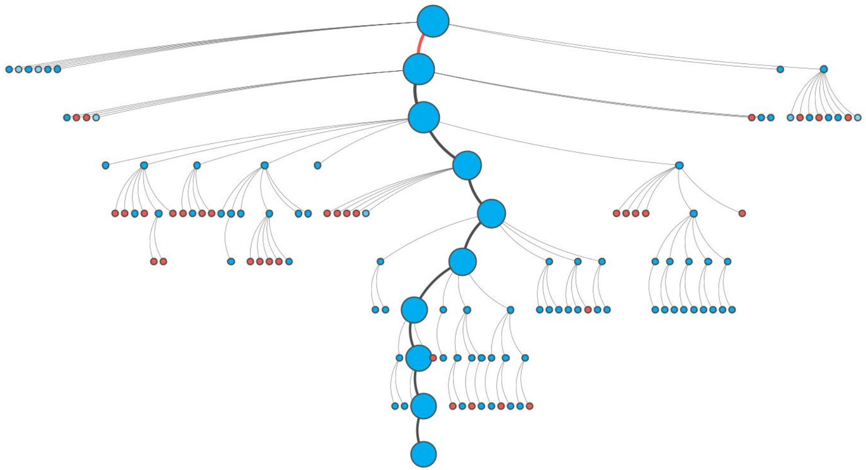
- We can only use differentiable policies $\pi(a|s, \theta)$ (Hence use MCTS!)
- High variance of REINFORCE
- Need to compute $r(s, a)$ efficiently
  - Solution 1: Do roll-outs to compute exactly (with a bit of MCTS)
  - Solution 2: Approximate r(s, a) with a neural network called Value Network

$$\hat{g}^{\mathrm{REINFORCE}}[r(s, a)] = \mathbb{E}_{\pi}[r(s, a)\nabla_{\theta} \log \pi(a|s, \theta)]$$

Imitation Learning

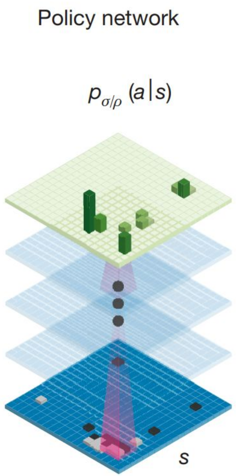
- Consists of an expert and an apprentice
- Apprentice tries to mimic expert

Expert



$$\mathcal{L}_{\text{TPT}} = -\sum_{a} \frac{n(s,a)}{n(s)} \log[\pi(a|s)]$$

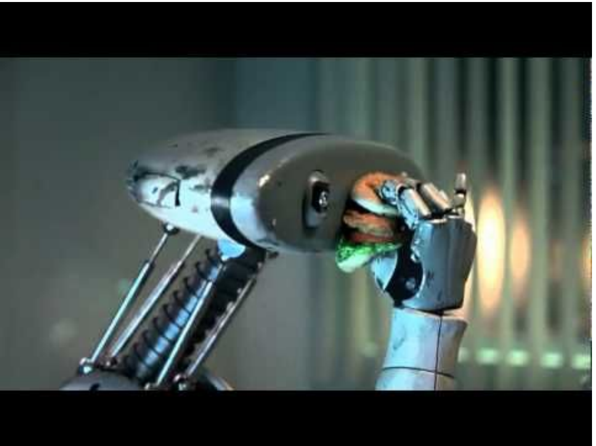
Apprentice



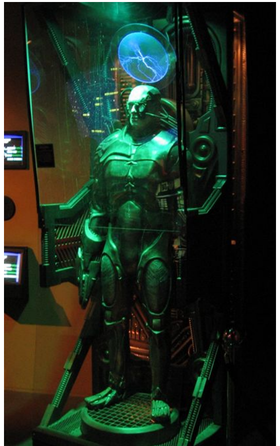
Imitation Learning Limits

- The apprentice will never exceed performance of expert
- Nothing can beat tree search given infinite resources and time
- In many domains, like game playing, expert might not be good enough

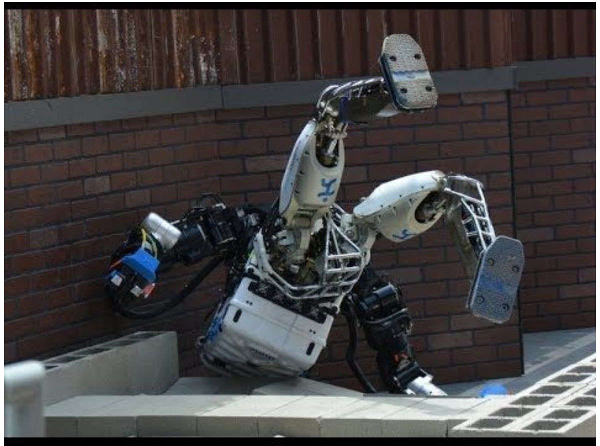
Eat



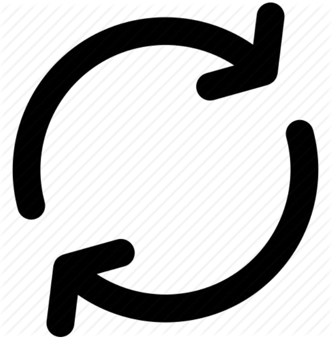
Sleep



Fail



Repeat



ExIt Pseudocode

Algorithm 1 Expert Iteration

$\hat{\pi}_0$ = initial_policy()
π_0^* = build_expert($\hat{\pi}_0$)
for i = 1; i $\leq$ max_iterations; i++ **do**
    S_i = sample_self_play($\hat{\pi}_{i-1}$)
    $D_i = \{(s,$ imitation_learning_target$(\pi^*_{i-1}(s)))|s \in S_i\}$
    $\hat{\pi}_i$ = train_policy(D_i)
    π_i^* = build_expert($\hat{\pi}_i$)
end for

The Minimal Policy Improvement Technique

MCTS as a policy improvement operator $\boldsymbol{\pi}(\boldsymbol{p})$

Define the goal of learning as finding policy p* s.t. $\boldsymbol{\pi}(\boldsymbol{p}^*) = \boldsymbol{p}^*.$

Gradient descent to solve this:

$$\begin{aligned}\theta_{t+1} &\leftarrow \theta_t + h\boldsymbol{v}(\boldsymbol{p})\frac{\partial \boldsymbol{p}}{\partial \theta} \\ &= \theta_t + h\,(\boldsymbol{\pi} - \boldsymbol{p})\,\frac{\partial \boldsymbol{p}}{\partial \theta}\end{aligned}$$

Instead of minimizing the norm of $\boldsymbol{\pi} - \boldsymbol{p}$ minimize: $\mathcal{L}(\theta) = KL(\boldsymbol{\pi}\|\boldsymbol{p}_\theta) = \boldsymbol{\pi}^T \log \boldsymbol{p}_\theta - c,$

Learning Targets

- Chosen-action Targets (CAT) loss: $\mathcal{L}_{\text{CAT}} = -\log[\pi(a^*|s)]$

  Where $a^* = \text{argmax}_a(n(s,a))$ is the move selected by MCTS.

- Tree-Policy Targets (TPT) loss: $\mathcal{L}_{\text{TPT}} = -\sum_a \frac{n(s,a)}{n(s)} \log[\pi(a|s)]$

  Where n(s, a) is the number of times an edge has been traversed.

Expert Improvement

Upper confidence bounds for trees:

$$\mathrm{UCT}(s,a) = \frac{r(s,a)}{n(s,a)} + c_b \sqrt{\frac{\log n(s)}{n(s,a)}}$$

Bias MCTS tree policy:

$$\mathrm{UCT}_{\mathrm{P-NN}}(s,a) = \mathrm{UCT}(s,a) + w_a \frac{\hat{\pi}(a|s)}{n(s,a)+1}$$

Value Network and AlphaGo Zero

Value Networks can do better than random rollouts if trained with enough data

$$\mathcal{L}_{\mathrm{V}} = -z \log[V(s)] - (1 - z) \log[1 - V(s)]$$

AlphaGo Zero is very similar with a slight difference in the loss function

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

Results: ExIt vs REINFORCE

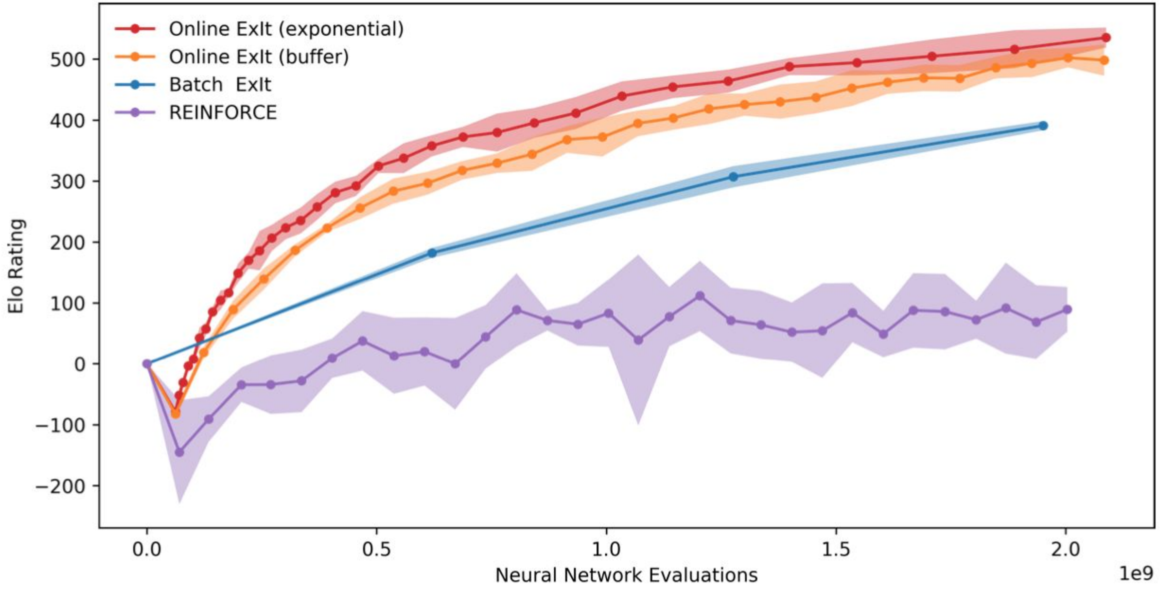


Figure 2: Elo ratings of policy gradient network and EXIT networks through training. Values are the average of 5 training runs, shaded areas represent 90% confidence intervals. Time is measured by number of neural network evaluations made. Elo calculated with BayesElo [14]

Results: Value and Policy ExIt vs MoHEX

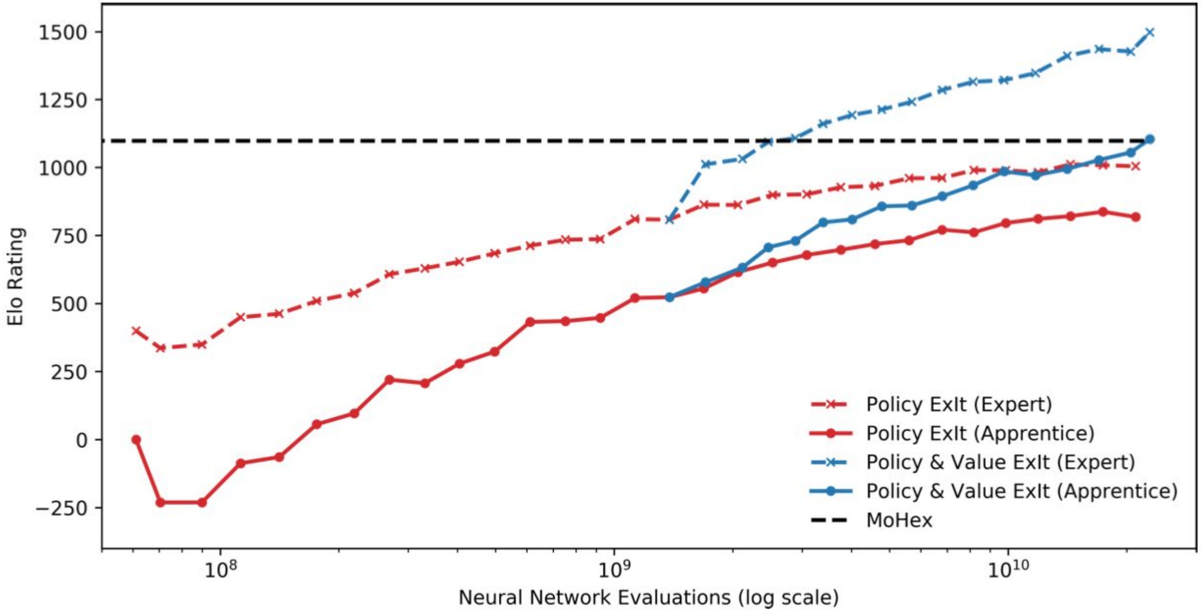


Figure 3: Apprentices and experts in distributed online EXIT, with and without neural network value estimation. MOHEX's rating (10,000 iterations per move) is shown by the black dashed line.

References

- Anthony, Thomas, Zheng Tian, and David Barber. "Thinking fast and slow with deep learning and tree search." Advances in Neural Information Processing Systems. 2017.
- Silver, David, et al. "Mastering the game of go without human knowledge." Nature 550.7676 (2017): 354.
- http://www.inference.vc/alphago-zero-policy-improvement-and-vector-fields/
- Farquhar, Gregory, et al. "TreeQN and ATreeC: Differentiable Tree Planning for Deep Reinforcement Learning." arXiv preprint arXiv:1710.11417 (2017).