Model Predictive Control and Reinforcement Learning
– Technologies behind AlphaGo –

Joschka Boedecker and Moritz Diehl

University Freiburg

October 5, 2022

Go

Lecture Overview

Acknowledgement

Slide contents are partially based on *Reinforcement Learning: An Introduction* by Sutton and Barto and the Reinforcement Learning lecture by David Silver.

Lecture Overview

Multi-armed Bandits

- A multi-armed Bandit is a tuple $\langle \mathcal{A}, p \rangle$, where:
  - $\mathcal{A}$ is a finite set of actions (or *arms*) and
  - $p(r|a) = \Pr\{R_t = r | A_t = a\}$ is an unknown probability distribution over rewards
- In each time step t, the agent selects an action A_t
- The environment then generates reward R_t
- The agent aims at maximizing the cumulative reward

Multi-armed Bandits

- The value of action a is the expected reward $q(a) = \mathbb{E}[R_t | A_t = a]$
- If the agent knew q, it could simply pick the action with highest value to solve the problem
- We can estimate q given samples by $Q_T(a) = \frac{1}{N_T(a)} \sum_{t=1}^{T} R_t \mathbb{1}_{A_t=a}$, where $N_T(a)$ is the number of times a was taken in t time steps.

Incremental and Running Mean

- We can compute the mean of a sequence $x_1, x_2, \ldots$ incrementally:

$$\begin{aligned}
\mu_k &= \frac{1}{k} \sum_{j=1}^{k} x_j \\
&= \frac{1}{k} \left(x_k + \sum_{j=1}^{k-1} x_j \right) \\
&= \frac{1}{k} (x_k + (k-1)\mu_{k-1}) \\
&= \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1})
\end{aligned}$$

Estimation of q

- Let R_i now denote the reward received after the ith selection of this action, and let Q_n denote the estimate of its action-value after it has been selected $n-1$ times.
- Equally, we can use an incremental implementation:

$$Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$$

- Generally, we can use some step size α:

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

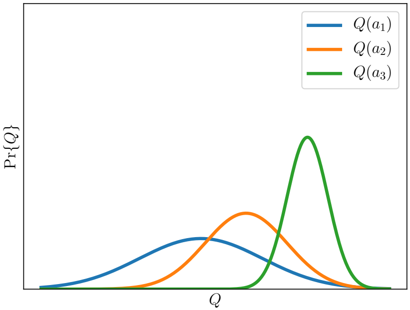
- A constant α can also be used for non-stationary problems (i.e. problems for which the reward probabilities change over time)
- Having an estimate of q, we can do greedy action selection by:

$$A_t = \arg\max_a Q_t(a)$$

ϵ-greedy

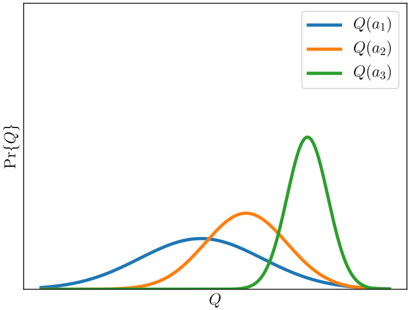
- One of the simpliest ways to explore:
  - With probability $(1 - \epsilon)$ select the greedy action
  - With probability ϵ select a random action
- Can be coupled with a decay schedule for ϵ, so as to explore less when the estimate is quite accurate after some time
- Exemplary schedule: $\epsilon_{t+1} = (1 - \frac{t}{T})\epsilon_{\text{init}}$, where t is the current round and T is the maximum considered number of rounds

Optimism in the Face of Uncertainty Principle



Which action should we pick?

Optimism in the Face of Uncertainty Principle



Which action should we pick?

Optimism in the Face of Uncertainty Principle

The more uncertain we are about an action-value, the more important it is to explore that action – since it could turn out to be the best action.

Upper Confidence Bound

- Exploration is needed because there is always uncertainty about the accuracy of the action-value estimates
- Idea: take into account how close their estimates are to being maximal and the uncertainties in those estimates
- For example by the *Upper Confidence Bound* (UCB) action selection:

$$A_t = \arg\max_a Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$$

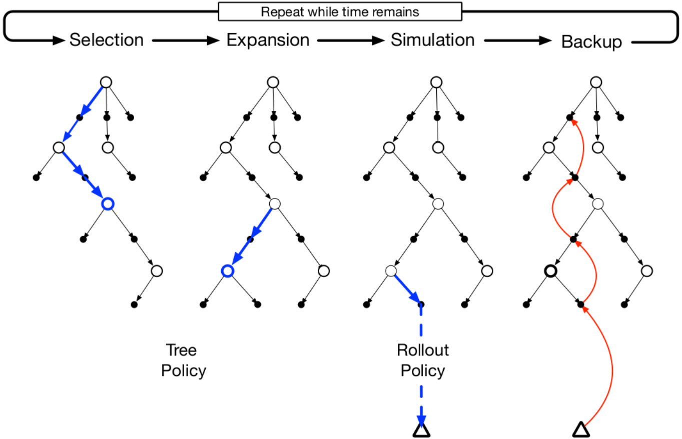
- Each time a is selected, the uncertainty is presumably reduced
- On the other hand, each time an action other than a is selected, t increases but $N_t(a)$ does not
- One implementation of the *Optimism in the Face of Uncertainty Principle*

Lecture Overview

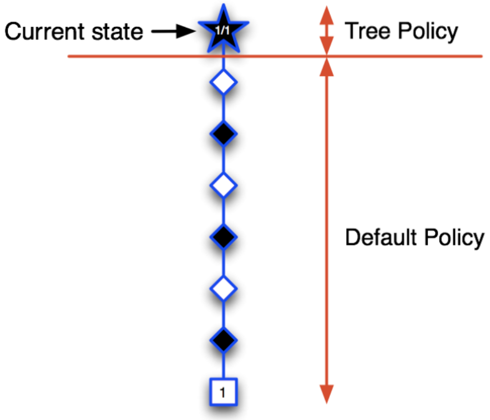
Introduction

- Extending Multi-armed Bandits to Markov Decision Processes by balancing exploration and exploitation for tree search.
- Different to minimax tree search methods, MCTS searches assymmetricaly.
- Proposed by Kocsis and Szepesvári in 2006 for improving computer Go players.
- The online search can be stopped at anytime providing the currently best action.

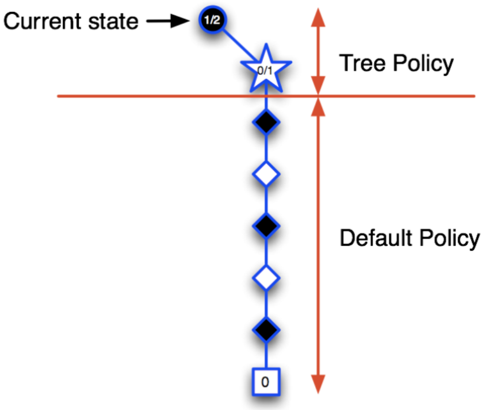
Monte Carlo Tree Search



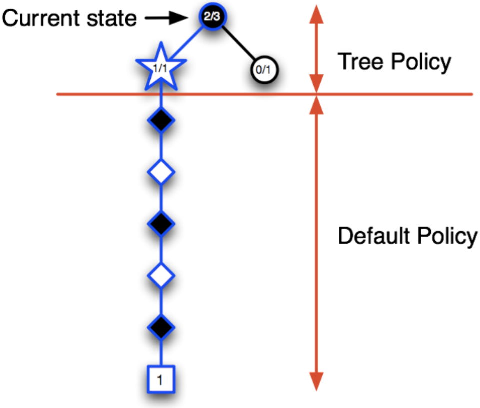
Applying Monte Carlo Tree Search (1)



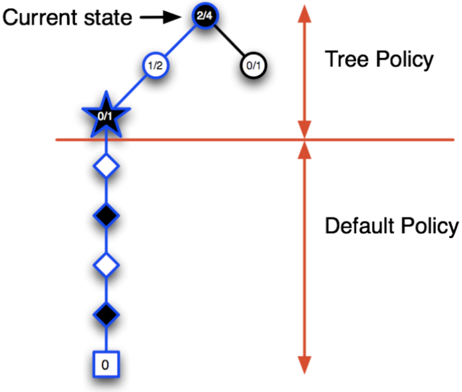
Applying Monte Carlo Tree Search (2)



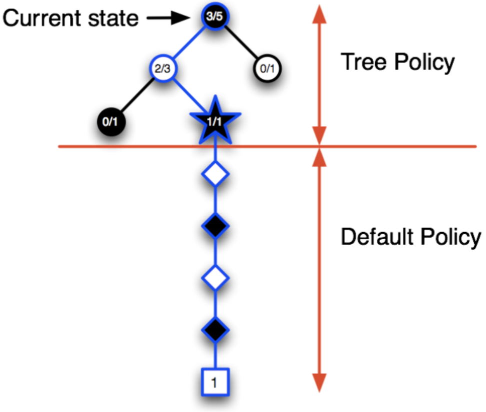
Applying Monte Carlo Tree Search (3)



Applying Monte Carlo Tree Search (4)



Applying Monte Carlo Tree Search (5)



Monte Carlo Tree Search

- During the selection and expansion phase, we keep track of the visit count

$$N(s,a) = \sum_{i=1}^{n} \mathbb{1}(s,a,i),$$

where $\mathbb{1}(s,a,i)$ indicates whether the ith selection or expansion phase visited state s and took action a.

- **Selection**: starting at the root, traverse to a leaf node following the *tree policy*
- An example of a popular tree policy is to adapt UCB for tree search:

$$A_t = \arg\max_a Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}$$

Monte Carlo Tree Search

- **Expansion**: expand the tree by a child node when reaching a new leaf node s_L^i.
- **Simulation**: simulate an episode from the new node following the *rollout policy*.
- With the rollout we get a Monte Carlo estimate $V(s_L^i)$ of the value of the new node.
- **Backup**: update the action-values for all nodes visited **in the tree**

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} \mathbb{1}(s,a,i) V(s_L^i).$$

For the sake of simplicity, we assume that there is only a terminal reward.

Lecture Overview

Can we beat professional go players with MCTS?

No.

- **Problem 1**: High branching factor
  - At each new node MCTS always starts from scratch and has to try every possible move.
  - At the start of the game, there are 19^2 possible moves.
- **Problem 2**: Monte Carlo rollouts
  - Simulation rollouts, especially with random rollout policies, have high variance.
  - With suboptimal rollout polices the result can be even biased on average.

Guiding MCTS with Reinforcement Learning

- AlphaGo Zero differs from normal MCTS by leveraging a single neural network that predicts both a value and a policy for a given state s by

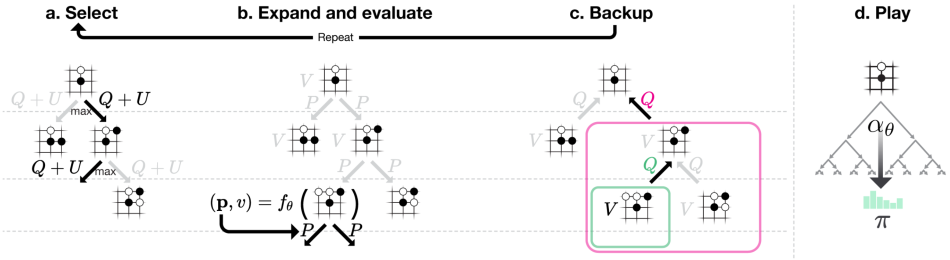
$$(\mathbf{p}, v) = f_\theta(s)$$

- The value part v is used to estimate the value of leaf states instead of an high-variance Monte Carlo rollout.
- The policy part $\mathbf{p}$ is used as a prior $P(s,a) = \mathbf{p}(s,a)$ to guide the MCTS search in a good direction right from the start

$$A = \arg\max_a Q(s,a) + c\frac{P(s,a)}{1 + N(s,a)}$$

- The effect of the prior is reduced over time by the number of visits $N(s,a)$.

Guiding MCTS with Reinforcement Learning

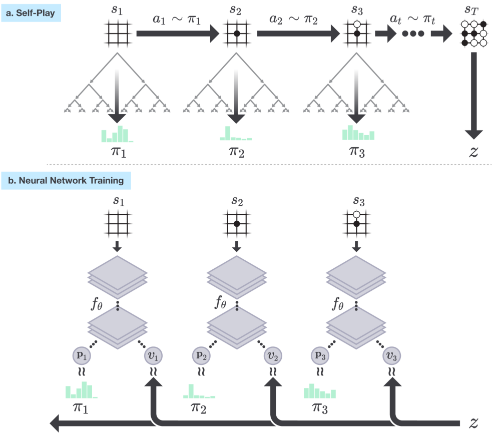


Training the policy and value network by self-play

- First play a game by self-play and then update the value and policy network.
- For self-play do at each step MCTS and obtain the visitation counts $N(s, a)$ of each action at the root state.
- From the visitation counts we derive a policy

$$\pi(a) \propto N^{1/\tau}$$

where τ is a temperature parameter.

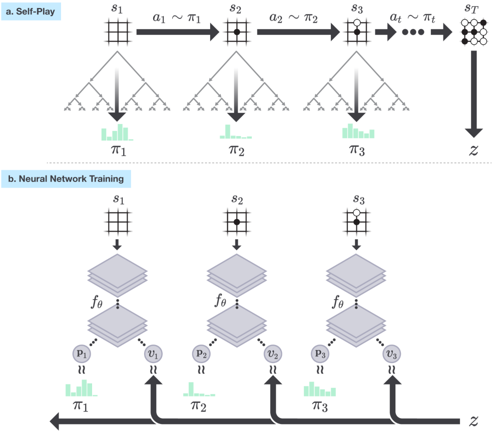


Training the policy and value network by self-play

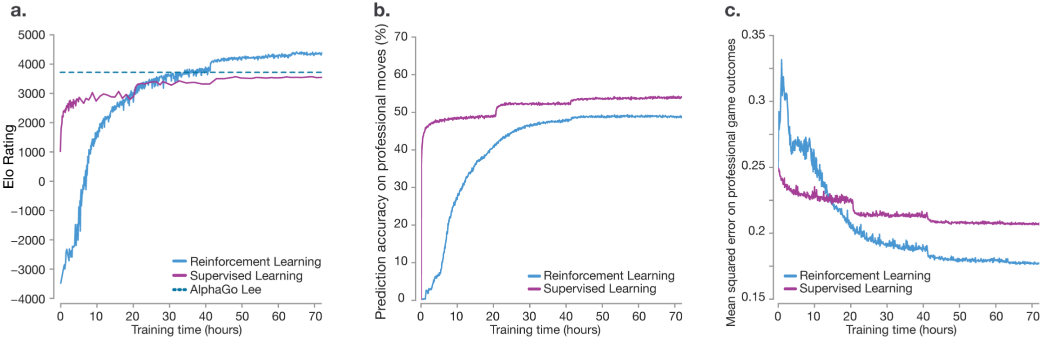
- The network $(\mathbf{p}, v) = f_\theta(s)$ is updated by

$$l = (z - v)^2 - \pi^T \log \mathbf{p} + \alpha \|\theta\|^2,$$

  where z is the outcome of the game.
- The policy part of the network $\mathbf{p}$ learns the probablities of the MCTS search π via a cross-entropy loss.



Results


a.
Elo Rating
5000
4000
3000
2000
1000
0
−1000
−2000
−3000
−4000
0 10 20 30 40 50 60 70
Training time (hours)
Reinforcement Learning
Supervised Learning
AlphaGo Lee
b.
Prediction accuracy on professional moves (%)
70
60
50
40
30
20
10
0
0 10 20 30 40 50 60 70
Training time (hours)
Reinforcement Learning
Supervised Learning
c.
Mean squared error on professional game outcomes
0.35
0.3
0.25
0.2
0.15
0 10 20 30 40 50 60 70
Training time (hours)
Reinforcement Learning
Supervised Learning

Some Notable Iterations of AlphaGo

1. *AlphaGo* (Silver et al. 2016):
   - Seperate value and policy network trained with supervised learning from expert data.
   - Policy and value network improved with reinforcement learning.
2. *AlphaGo Zero* (Silver et al. 2017):
   - No human knowledge used for training.
   - Joined value and policy network trained online with MCTS search.
3. *MuZero* (Schrittwieser et al. 2020):
   - Rules are learned by a model.
   - MCTS in latent space.
   - Achieved also state of the art in Atari Games.

Trivia

- The last game a human won against AlphaGo was in 2016 by Lee Sedol: https://youtu.be/WXuK6gekU1Y?t=3934