Foundations of Artificial Intelligence

44. Monte-Carlo Tree Search: Advanced Topics

Malte Helmert and Gabriele Röger

University of Basel

May 22, 2017

Board Games: Overview

chapter overview:

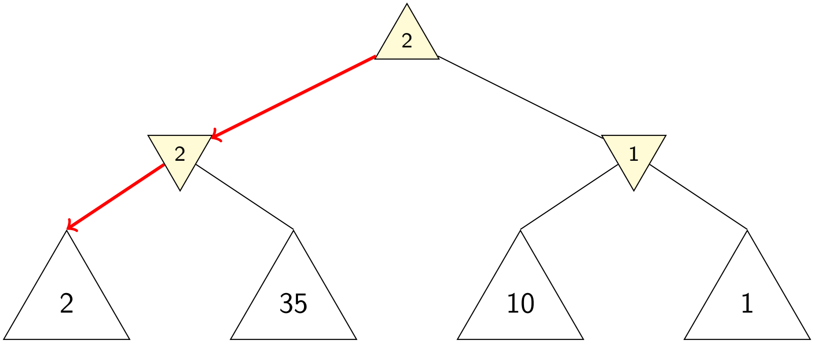
Optimality of MCTS

Reminder: Monte-Carlo Tree Search

- as long as time allows, perform iterations
  - selection: traverse tree
  - expansion: grow tree
  - simulation: play game to final position
  - backpropagation: update utility estimates
- execute move with highest utility estimate

Optimality

complete "minimax tree" computes optimal utility values Q^*



Asymptotic Optimality

Asymptotically Optimality

An MCTS algorithm is asymptotically optimal if $\hat{Q}^k(n)$ converges to $Q^*(n)$ for all $n \in \text{succ}(n_0)$ with $k \to \infty$.

Asymptotic Optimality

Asymptotically Optimality

An MCTS algorithm is asymptotically optimal if $\hat{Q}^k(n)$ converges to $Q^*(n)$ for all $n \in \text{succ}(n_0)$ with $k \to \infty$.

Note: there are MCTS instantiations that play optimally even though the values do not converge in this way
(e.g., if all $\hat{Q}^k(n)$ converge to $\ell \cdot Q^*(n)$ for a constant $\ell > 0$)

Asymptotic Optimality

For a tree policy to be asymptotically optimal, it is required that it

- explores forever:
  - every position is expanded eventually and visited infinitely often (given that the game tree is finite)
  - after a finite number of iterations, only true utility values are used in backups
- is greedy in the limit:
  - the probability that the optimal move is selected converges to 1
  - in the limit, backups based on iterations where only an optimal policy is followed dominate suboptimal backups

Tree Policy

Objective

tree policies have two contradictory objectives:

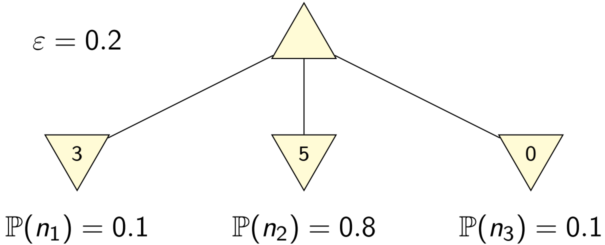
- explore parts of the game tree that have not been investigated thoroughly
- exploit knowledge about good moves to focus search on promising areas

central challenge: balance exploration and exploitation

ε-greedy: Idea

- tree policy with constant parameter ε
- with probability $1 - \varepsilon$, pick the greedy move (i.e., the one that leads to the successor node with the best utility estimate)
- otherwise, pick a non-greedy successor uniformly at random

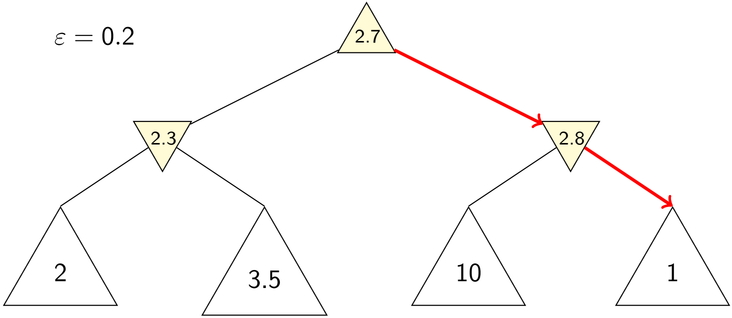
ε-greedy: Example



ε-greedy: Asymptotic Optimality

Asymptotic Optimality of ε-greedy

- explores forever
- not greedy in the limit
- $\Rightarrow$ not asymptotically optimal



ε-greedy: Asymptotic Optimality

Asymptotic Optimality of ε-greedy

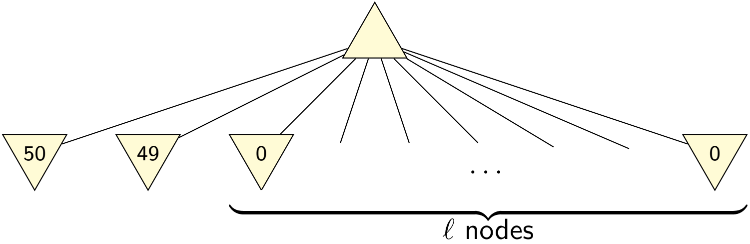
- explores forever
- not greedy in the limit
- $\Rightarrow$ not asymptotically optimal

asymptotically optimal variants:

- use decaying ε, e.g. $\varepsilon = \frac{1}{k}$
- use minimax backups

ε-greedy: Weakness

Problem:
when ε-greedy explores, all non-greedy moves are treated equally

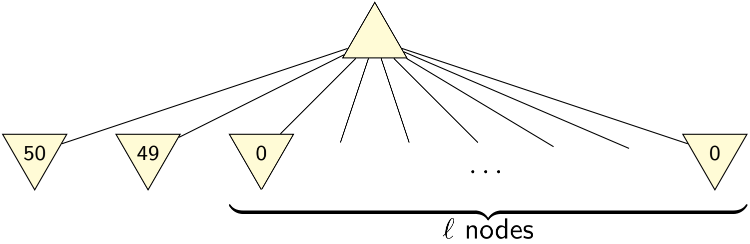


e.g., $\varepsilon = 0.2, \ell = 9$: $\mathbb{P}(n_1) = 0.8$, $\mathbb{P}(n_2) = 0.02$

Softmax: Idea

- tree policy with constant parameter τ
- select moves proportionally to their utility estimate
- Boltzmann exploration selects moves proportionally to $\mathbb{P}(n) \propto e^{\frac{\hat{Q}(n)}{\tau}}$

Softmax: Example



e.g., $\tau = 10, \ell = 9$: $\mathbb{P}(n_1) \approx 0.51$, $\mathbb{P}(n_2) \approx 0.46$

Boltzmann Exploration: Asymptotic Optimality

Asymptotic Optimality of Boltzmann Exploration

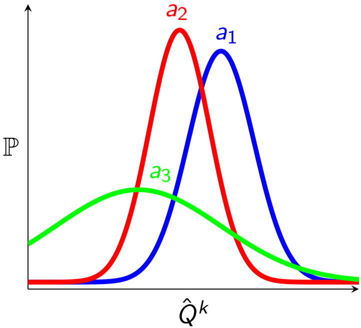
- explores forever
- not greedy in the limit
  (probabilities converge to positive constant)
- $\Rightarrow$ not asymptotically optimal

asymptotically optimal variants:

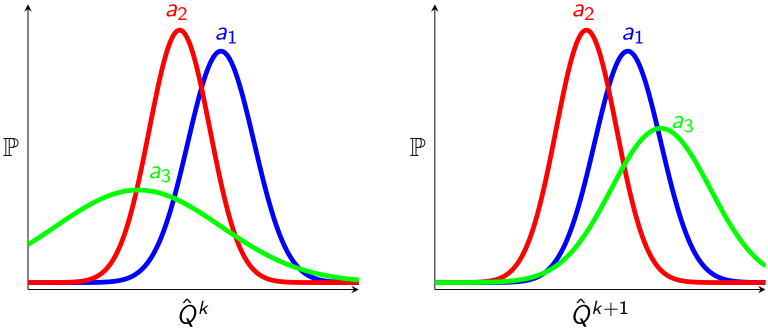
- use decaying τ
- use minimax backups

careful: τ must not decay faster than logarithmical to explore infinitely

Boltzmann Exploration: Weakness



Boltzmann Exploration: Weakness



Upper Confidence Bounds: Idea

balance exploration and exploitation by preferring moves that

- have been successful in earlier iterations (exploit)
- have been selected rarely (explore)

Upper Confidence Bounds: Idea

Upper Confidence Bounds

- select successor n' of n that maximizes $\hat{Q}(n') + \hat{U}(n')$
- based on utility estimate $\hat{Q}(n')$
- and a bonus term $\hat{U}(n')$
- select $\hat{U}(n')$ such that $Q^*(n') \leq \hat{Q}(n') + \hat{U}(n')$ with high probability
- $\hat{Q}(n') + \hat{U}(n')$ is an upper confidence bound on $Q^*(n')$ under the collected information

Upper Confidence Bounds: UCB1

- use $\hat{U}(n') = \sqrt{\frac{2 \cdot \ln N(n)}{N(n')}}$ as bonus term
- bonus term is derived from Chernoff-Hoeffding bound:
  - gives the probability that a sampled value (here: $\hat{Q}(n')$)
  - is far from its true expected value (here: $Q^*(n')$)
  - in dependence of the number of samples (here: $(N(n')$)
- picks the optimal move exponentially more often

Upper Confidence Bounds: Asymptotic Optimality

Asymptotic Optimality of UCB1

- explores forever
- greedy in the limit
- $\Rightarrow$ asymptotically optimal

Upper Confidence Bounds: Asymptotic Optimality

Asymptotic Optimality of UCB1

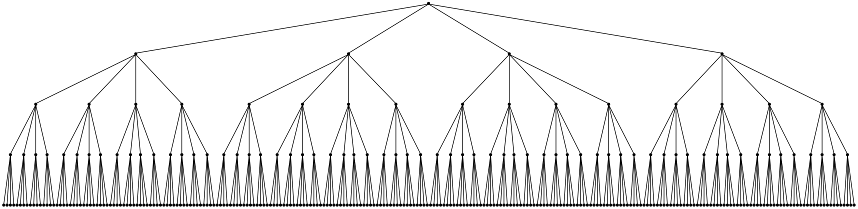
- explores forever
- greedy in the limit
- $\Rightarrow$ asymptotically optimal

However:

- no theoretical justification to use UCB1 in trees or planning scenarios
- development of tree policies active research topic

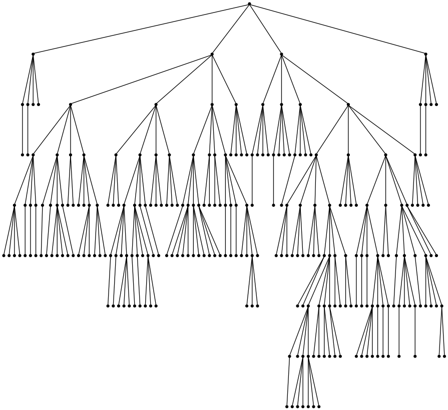
Tree Policy: Asymmetric Game Tree

full tree up to depth 4



Tree Policy: Asymmetric Game Tree

UCT tree (equal number of search nodes)



Other Techniques

Default Policy: Instantiations

default: Monte-Carlo Random Walk

- in each state, select a legal move uniformly at random
- very cheap to compute
- uninformed
- usually not sufficient for good results

Default Policy: Instantiations

default: Monte-Carlo Random Walk

- in each state, select a legal move uniformly at random
- very cheap to compute
- uninformed
- usually not sufficient for good results

only significant alternative: domain-dependent default policy

- hand-crafted
- offline learned function

Default Policy: Alternative

- default policy simulates a game to obtain utility estimate
- $\Rightarrow$ default policy must be evaluated in many positions
- if default policy is expensive to compute, simulations are expensive
- solution: replace default policy with heuristic that computes a utility estimate directly

Other MCTS Enhancements

there are many other techniques to increase information gain from iterations, e.g.,

- All Moves As First
- Rapid Action Value Estimate
- Move-Average Sampling Techique
- and many more

Literature: A Survey of Monte Carlo Tree Search Methods
Browne et. al., 2012

Expansion

- to proceed deeper into the tree, each node must be visited at least once for each legal move
- $\Rightarrow$ deep lookaheads not possible
- rather than add a single node, expand encountered leaf node and add all successors
  - allows deep lookaheads
  - needs more memory
  - needs initial utility estimate for all children

Summary

Summary

- tree policy is crucial for MCTS
  - ϵ-greedy favors the greedy move and treats all other equally
  - Boltzmann exploration selects moves proportionally to their utility estimates
  - UCB1 favors moves that were successful in the past or have been explored rarely
- there are applications for each where they perform best
- good default policies are domain-dependent and hand-crafted or learned offline
- using heuristics instead of a default policy often pays off