CS310 – AI Foundations
(Andrew Abel)
February 2024

Week 5: Monte Carlo Tree Search

1

MCTS - Introduction

- Ordinary game-tree search has limitations
  - If it is not a tiny game, we need a good evaluation function
  - Complex games like Go or Chess have a lot of moves and a large branching factor
- Introduction to Monte-Carlo Tree Search
- Bandit Problems
- UCB Formula

2

Monte-Carlo Tree Search

- Revolutionised Computer Go
- Only applied to deterministic games fairly recently (since the 2000s)
- Explosion of interest due to non-game applications:
  - (motion) planning
  - Optimisation
  - Finance
  - Energy Management

3

Go and MCTS

- MCTS is an important technique that led to the success of AlphaGo in creating a strong AI player
  - Developed by Google
  - First computer Go program to beat a human professional (without a handicap) in 2015
  - Replaced by AlphaGo Zero, and then AlphaZero, and then MuZero
- Rough Idea: Simulate games systematically in order to “learn” the value of states and moves
- Other techniques in AlphaGo include:
  - deep neural networks – Help to evaluate moves and positions
  - Reinforcement learning (train strongest AI player against itself)
  - MCTS is a form of reinforcement learning in some ways

4

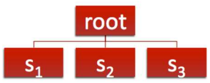
Recall: Basic Monte Carlo Search (Monte Carlo Roll Outs)

- If no evaluation function present
  - Simulate game using random moves
  - Score game at the end and keep stats of wins and losses
  - Move to position with best winning chances
  - Repeat…
- This results in an evaluation function with the hope that the sampling preserves the differences between good and bad moves

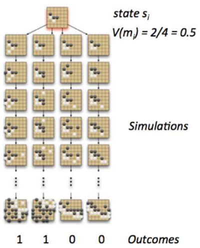
5

Basic Monte Carlo Search

- What value do we give the state?



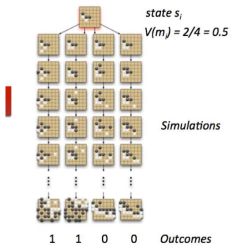
1 ply tree
root = current position
s_1 = state after move m_1
s_2 = …



6

Basic Monte Carlo Search

- What value do we give the state?
- We run 4 random games, 2 won, 2 lost
- We can get the value of the state this way
- 2/4
- However, this relies on a lot of randomness…



7

Naive Approach

- Use Monte Carlo roll-outs as evaluation function for depth-limited minimax with α ß pruning
- Problems
  - Single roll-out is very noisy (0/1 signal)
  - Running many roll-outs for for one evaluation is very slow
- Example
  - typical chess program: 1 million moves/sec
  - Go: 1 million moves/sec, 400 moves/roll-out, 100 roll-outs/eval ->25 eval/sec
- Consequence: Monte-Carlo was ignored for over 10 years in Go
- If you take the naïve approach, you might have to evaluate a lot of nodes
  - This can be expensive
  - You need to choose your nodes wisely!

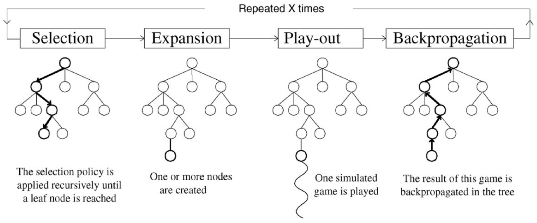
8

Monte Carlo Tree Search: High-level

- MCTS builds a statistics tree detailing the value of nodes
  - Start from the root and build a subtree
  - Has statistics attached to it giving a value of the nodes
- Statistics tree guides the AI to focus on the most interesting nodes in the game tree
  - Can be used for further exploration
- value of nodes determined by Monte-Carlo roll-outs

9

The 4 Stages of Monte Carlo Tree Search



Source: T. Pepels, M. H. M. Winands and M. Lanctot, "Real-Time Monte Carlo Tree Search in Ms Pac-Man," in IEEE Transactions on Computational Intelligence and AI in Games. (2014)

10

Monte Carlo Tree Search: Exploitation vs Exploration

- Use results of roll-outs to guide growth of game tree
- **Exploitation** focus on promising moves
- **Exploration** focus on moves that are not sufficiently explored yet
- trade-off between exploitation and exploration!
- Idea: use the theory of bandit problems!
  - Finding the right balance

11

Bandit Problems

12

Multi-Armed Puggy problem

- Assumptions
  - Choice of several arms
  - each arms pull is independent of other pulls
  - each arm has a fixed, unknown average payoff
- Which arm has the best average pay-off?

13

Consider three slot machines

- Each pull is either a win (payoff 1) or a loss (payoff 0)
- A is the best arm but we do not know that!
- How can you figure out the best machine without losing too much money playing them all?

A

P(A wins) = 60%

B

P(B wins) = 55%

C

P(C wins) = 40%

14

Exploration vs Exploitation

- Want to explore all arms - but exploration is costly
- Want to **exploit** promising arms more often - but might miss better opportunity!
- want to minimize regret = loss from playing non-optimal machine
- Need to balance exploration/exploitation!

15

Which machine/arm should we play next?

- a **policy** is a strategy for choosing the next machine to play at time *t*
- this uses selections & outcomes so far
- Examples:
    1. uniform policy: play the machine that has been played least so far
    2. greedy policy: play the machine with the highest observed payoff so far
- ties are broken randomly
- a better policy?

16

Upper confidence bound (Auer et al 2002)

- Policy
    - First, try each slot machine once.
    - Then, at each step, choose slot machine that maximises the UCB1 formula for the "upper confidence bound'":
    - C = parameter you can tune

$$\underbrace{v_i}_{\text{value estimate}} + \underbrace{C}_{\text{tunable parameter}} \times \sqrt{\frac{\ln(\overbrace{N}^{\text{total number of trials}})}{\underbrace{n_i}_{\text{num trials for arm i}}}}$$

17

UCB intuition

- Pick machine that maximises

$$\underbrace{v_i}_{\text{value estimate}} + \underbrace{C}_{\text{tunable parameter}} \times \sqrt{\frac{\ln(\overbrace{N}^{\text{total number of trials}})}{\underbrace{n_i}_{\text{num trials for arm i}}}}$$

- Higher observed reward v_i is better -> Exploitation!
- we expect the "true value" of machine i in some confidence interval around v_i

18

UCB Intuition II

- Pick machine that maximises

$$\boxed{\underbrace{v_i}_{\text{value estimate}}} + \underbrace{C}_{\text{tunable parameter}} \times \sqrt{\frac{\ln(\underbrace{N}_{\text{total number of trials}})}{\underbrace{n_i}_{\text{num trials for arm i}}}}$$

- Value estimate is the expected reward, i.e. the estimate.
- Total/number of attempts
- If played 5 times, with rewards of 10, 20, -5, 5, 10
  - n = 5, t = 40, v = 8
- This is exploitation, looking for best value

19

UCB Intuition II

- Pick machine that maximises

$$\underbrace{v_i}_{\text{value estimate}} + \underbrace{C}_{\text{tunable parameter}} \times \boxed{\sqrt{\frac{\ln(\underbrace{N}_{\text{total number of trials}})}{\underbrace{n_i}_{\text{num trials for arm i}}}}}$$

- N is total number of visits, i.e. time, number of times all tree has been explored
- n is number of visits of that particular node
- When n = 0, answer is infinity
- This is exploration
- Higher value for nodes that have been explored less

20

UCB Intuition II

- Pick machine that maximises

$$\underbrace{v_i}_{\text{value estimate}} + \boxed{\underbrace{C}_{\text{tunable parameter}}} \times \sqrt{\frac{\ln(\underbrace{N}_{\text{total number of trials}})}{\underbrace{n_i}_{\text{num trials for arm i}}}}$$

- C is our tuneable parameter
- Balances between exploration and exploitation
- If C = 0, then we don't care about exploring nodes at all
- As C-> infinity, takes more account of exploration

21

UCB Intuition II

- Pick machine that maximises

$$\underbrace{v_i}_{\text{value estimate}} + \underbrace{C}_{\text{tunable parameter}} \times \sqrt{\frac{\ln(\overbrace{N}^{\text{total number of trials}})}{\underbrace{n_i}_{\text{num trials for arm i}}}}$$

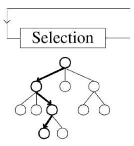
- Confidence interval is large while n_i is small, shrinks in proportion to $\sqrt{n_i}$
- Low confidence leads to high uncertainty about i which leads to more Exploration!
- Explore if number of trials n_i is small compared to number N of all trials!

22

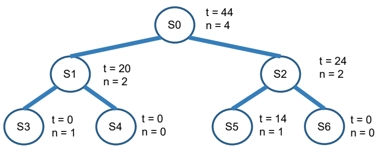
Monte Carlo Tree Search - Example

23

The 4 Stages of Monte Carlo Tree Search

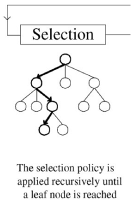


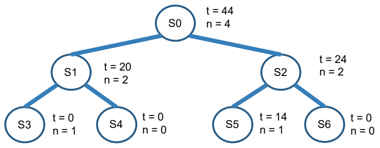
The selection policy is applied recursively until a leaf node is reached



24

The 4 Stages of Monte Carlo Tree Search





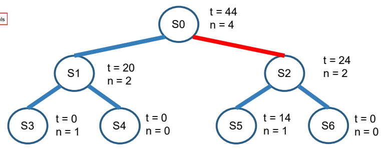
Navigate through tree to find best leaf node, starting at root (S0)

Assume C = 2 for this example

25

The 4 Stages of Monte Carlo Tree Search

$$\underbrace{v_i}_{\text{value estimate}} + \underbrace{C}_{\text{tunable parameter}} \times \sqrt{\frac{\ln(\overbrace{N}^{\text{total number of trials}})}{\underbrace{n_i}_{\text{num trials for arm i}}}}$$



S1 = (20/2) + 2 ($\sqrt{\frac{\ln(4)}{2}}$)

= 10 + 2 ($\sqrt{\frac{1.386}{2}}$)

= 10 + 2($\sqrt{0.69}$)

= 11.665

S2 = (24/2) + 2 ($\sqrt{\frac{\ln(4)}{2}}$)

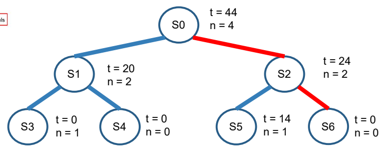
= 13.665

So we choose S2

26

The 4 Stages of Monte Carlo Tree Search

$$\underbrace{v_i}_{\text{value estimate}} + \underbrace{C}_{\text{tunable parameter}} \times \sqrt{\frac{\ln(\overbrace{N}^{\text{total number of trials}})}{\underbrace{n_i}_{\text{num trials for arm i}}}}$$



S5 = (14/1) + 2 ($\sqrt{\frac{\ln(4)}{1}}$)

S5 = 16.355

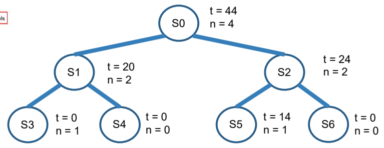
So we choose S6, expand the non explored node

S6 = n = 0 so = Infinty

27

The 4 Stages of Monte Carlo Tree Search

$$\underbrace{v_i}_{\text{value estimate}} + \underbrace{C}_{\text{tunable parameter}} \times \sqrt{\frac{\ln(\overbrace{N}^{\text{total number of trials}})}{\underbrace{n_i}_{\text{num trials for arm i}}}}$$



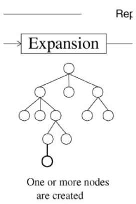
S1 = (20/2) + 2 ($\sqrt{\frac{\ln(4)}{2}}$)

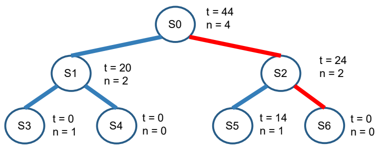
= 10 + 2 ($\sqrt{\frac{1.386}{2}}$)

= 10 + 2($\sqrt{0.69}$)

= 11.665

The 4 Stages of Monte Carlo Tree Search

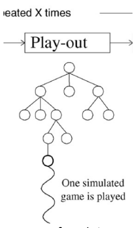


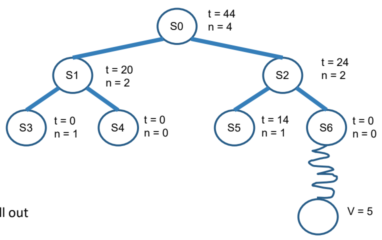


If not played, roll out, otherwise, do an expansion

We do a rollout

The 4 Stages of Monte Carlo Tree Search

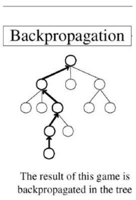




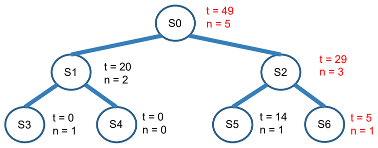
V = 5 after doing our monte carlo roll out

(picked randomly)

The 4 Stages of Monte Carlo Tree Search

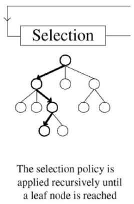


Update our totals

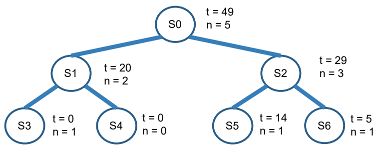


31

The 4 Stages of Monte Carlo Tree Search

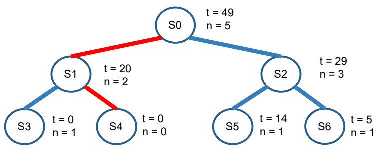


Around again!



32

The 4 Stages of Monte Carlo Tree Search



S1 = $(20/2) + 2\left(\sqrt{\frac{\ln(5)}{2}}\right)$

= 11.79

Note, a little higher!

S2 = $(29/3) + 2\left(\sqrt{\frac{\ln(5)}{3}}\right)$

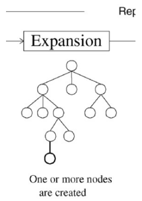
= 11.13

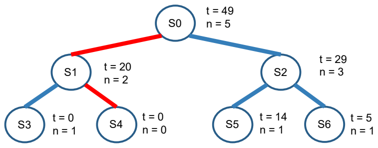
S4 n = 0, so UCB = infinity

So we choose S4

33

The 4 Stages of Monte Carlo Tree Search

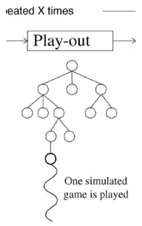


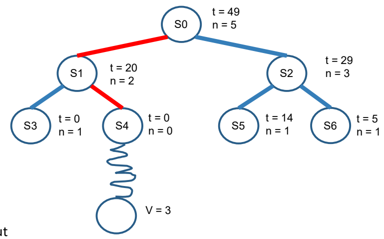


Not played, so we roll out

34

The 4 Stages of Monte Carlo Tree Search

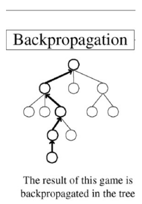


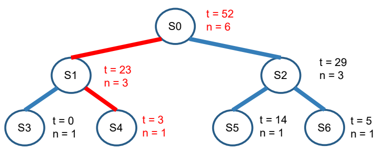


Not played, so we roll out

35

The 4 Stages of Monte Carlo Tree Search

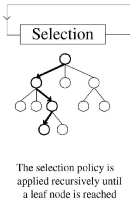


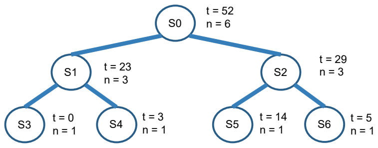


Update our totals

36

The 4 Stages of Monte Carlo Tree Search

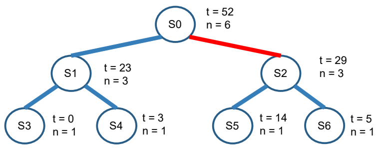




Around again!

37

The 4 Stages of Monte Carlo Tree Search



$S1 = (23/3) + 2\left(\sqrt{\frac{\ln(6)}{3}}\right)$

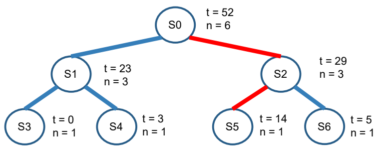
$= 9.21$

$S2 = (29/3) + 2\left(\sqrt{\frac{\ln(6)}{3}}\right)$

$= 11.21$

38

The 4 Stages of Monte Carlo Tree Search



$S5 = (14/1) + 2\left(\sqrt{\frac{\ln(6)}{1}}\right)$

$= 16.68$

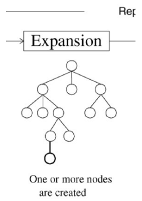
$S6 = (5/1) + 2\left(\sqrt{\frac{\ln(6)}{1}}\right)$

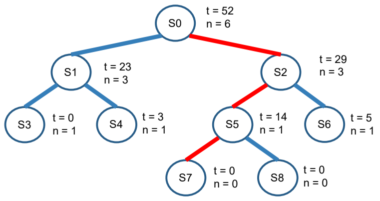
$= 7.67$

Makes sense. Highest value, both played same number of times.

39

The 4 Stages of Monte Carlo Tree Search



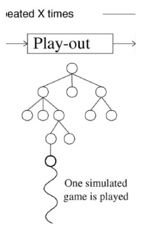


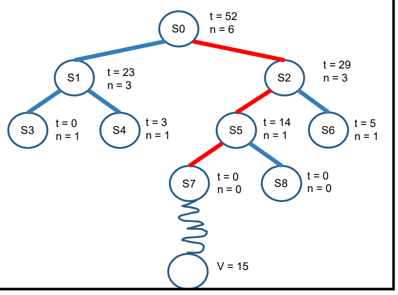
Its been played, so this time we expand the tree

Then take first node (any one we want)

40

The 4 Stages of Monte Carlo Tree Search

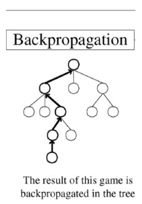


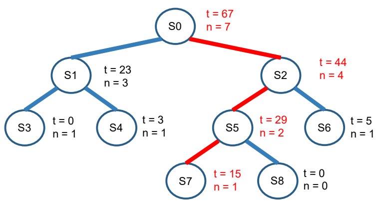


Do a rollout

41

The 4 Stages of Monte Carlo Tree Search





Update our totals

42

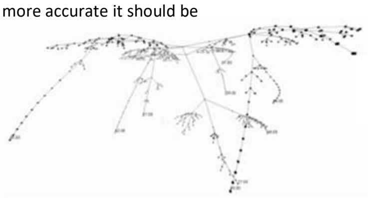
Findings

- As we play, if one branch is producing consistently better results, tree will be biased towards it
- Will not explore tree evenly
- Some nodes will remain unexplored

43

How long to play?

- X number of iterations (2000 iterations?)
- T time (10 seconds of thinking time?)
- Depends on how long you configure
- The more you run it, the more accurate it should be



44

Theoretical properties of UCB

- Main question: rate of convergence to optimal arm
- Intuition: speedy conversion leads to low regret
- Typical goal: regret is O(log N) for N trials.
- For many problems this is the optimum.
- UCB1 is a simple algorithm that achieves this optimal bound for many input distributions!

45

Summary

- This week
  - Pruning and optimisation
  - Depth limited minimax with pruning
  - Handling uncertainty and chance
  - Monte-Carlo Tree Search