MCTS 3:
UCB & Further Improvements

CMPUT 355: Games, Puzzles, and Algorithms

Lecture Outline

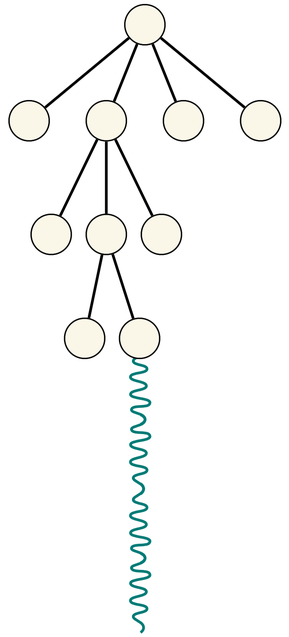
1. Logistics & Recap
2. UCB Proof Sketch
3. Some other MCTS improvements

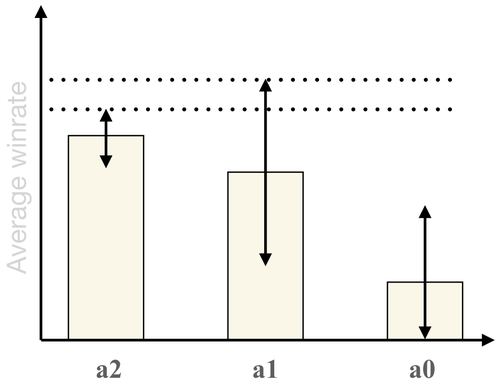
Logistics

- **Practice questions #5** will be available on **Friday** (Mar 20)
- Quiz #3 marks have been posted
  - Solutions are available on the website
- Quiz #4 marking is in progress

Recap: Monte Carlo Tree Search

- MCTS maintains a **search tree** which grows as the search progresses
- Leaves are evaluated by rapid, random **simulations**
- Search tree is traversed using **UCT** to balance **exploration** and **exploitation**
- Basic algorithm:
    1. **Traverse:** Starting from the root, traverse the search tree by iteratively choosing children using UCT
    2. If the resulting leaf node has at **least one simulation**:
        1. **Expand:** add all of the leaf's child states to the search tree
        2. **Rollout:** simulate play from one of the child states until the end of the game
    3. Otherwise:
        1. **Rollout:** simulate play from the leaf until the end of the game
    4. **Backpropagate:** Add results of the simulation to performance statistics on every node on the path back to the root
        - E.g., average **score** and number of **visits**





Recap: Solving & MCTS

- If the **search tree** reaches enough terminal nodes, MCTS can prove that a move is optimal
- When a search node is itself a terminal node, it has a score of (+/-) **infinity**
  - A node with a score of positive infinity is a guaranteed win for its parent
  - No need to expand sibling nodes
- When **all** of a node's children have scores of negative infinity, then the node is an **optimal move**
  - Can set its own score to positive infinity!

Stochastic Bandit Setting

- In a **stochastic bandit setting**, there are K **arms**. Each arm has its own associated distribution. At each timestep $1 \leq t \leq T$, the player must choose an arm $A(t)$, and will receive a reward drawn i.i.d. from the **unknown** distribution associated with the chosen arm.
- Let $X_i(t)$ be the reward that arm i would deliver on timestep t if it is pulled
    - i.e., at time t, the player will receive reward $X_{A(t)}(t)$
- At each timestep, the player incurs **regret**: the difference between the **best** arm they could have chosen at that timestep, and the arm they **actually** chose:

$$r(t) = \max_i X_i(t) - X_{A(t)}(t)$$

- We want to minimize the **cumulative, expected regret**:

$$R(T) = \sum_{t=1}^{T} \mathbb{E}[r(t)]$$

UCB Theorem

Theorem: In a stochastic multi-armed bandit setting, choosing arm

$$A_t = \arg\max_i \left[\hat{\mu}_i(t) + \sqrt{\frac{2 \ln t}{n_i(t)}} \right]$$

at each timestep t, where

- $\hat{\mu}_i(t)$ is the average reward received from arm i before time t, and
- $n_i(t)$ is the number of times arm i has been pulled before time t,

guarantees that after T rounds, the expected average regret approaches 0:

$$\lim_{T \to \infty} \frac{R(T)}{T} = O\left(\frac{\ln T}{T}\right) = 0.$$

UCB Theorem Proof Sketch

$$A_t = \arg\max_i \left[\hat{\mu}_i(t) + U_i(t)\right]$$

$$\text{where } U_i(t) = \sqrt{\frac{2\ln t}{n_i(t)}}$$

Proof sketch:

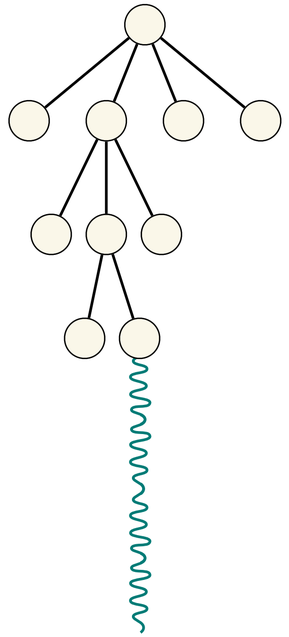
1. The true mean value of arm i will be less than $\hat{\mu}_i(t) + U_i(t)$ with probability $1 - t^{-4}$
   - We choose $U_i(t)$ specifically to **make this true** by plugging into **Hoeffding's Inequality**
2. The total regret is $R(T) = \sum_{i=1}^{K} (\mu^* - \mu_i)\mathbb{E}[n_i(T)]$
3. We pull the "wrong arm" when $\hat{\mu}_i(t) + U_i(t)$ is larger than $\hat{\mu}^*$ (i.e., when $U_i(t) > \mu^* - \mu_i$):
4. Total regret $R(T) = O(\ln T)$

$$\sqrt{\frac{2\ln t}{n_i(t)}} \approx (\mu^* - \mu)$$

$$\frac{2\ln t}{n_i(t)} \approx (\mu^* - \mu)^2$$

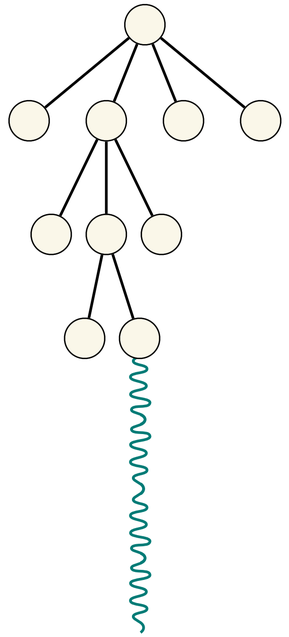
$$n_i(t) \approx \frac{\ln t}{(\mu^* - \mu)^2}$$

MCTS Improvements: Child selection



- UCT is just applying UCB algorithm to selecting children within the search tree during **traversal**
- Once we have **expanded** a leaf node, we can also improve which child we select for the initial **simulation**:
    - Game-specific heuristics
    - Neural network predictions
- **Question:** Why is this useful?
- **Question:** Don't we need this selection to be random?

MCTS Improvement: Pattern Detection in Simulations



- Purely random simulations give an accurate sense of "chances to win" (**why?**)
- But they may not realistically reflect how opponents would **really play**
- Certain moves are **obvious**:
  - Block win-threat in tic-tac-toe
  - "Saving a bridge" in Hex
- Biasing simulations toward obvious moves can greatly improve their **accuracy**
- Trade-offs:
  - Need to be able to simulate **rapidly** (argues against heavy-weight improvements)
  - Don't want simulations to **only** choose optimal moves (**why not?**)

Summary

- UCB algorithm guarantees that **average regret** in a stochastic multi-armed bandit will "vanish" (infinite limit is 0)
  - **UCT** treats each child as an **arm** of a stochastic multi-armed bandit
- MCTS has many **variants**!
  - Can improve **child selection** for initial **simulation**
  - Can improve the **quality** of the simulations themselves
  - But, it can actually be better to not always simulate **optimal** moves