Lecture 13: Monte Carlo Tree Search

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2026

- With some slides from or derived from David Silver

Class Structure

- Last time: Fast / sample efficient Reinforcement Learning
- **This Time: MCTS and guest lecture Shane Gu from DeepMind on World Models**
- Next time: Rewards in RL

AlphaZero and Monte Carlo Tree Search

→ Alpha Tensor

- Responsible in part for one of the greatest achievements in AI in the last decade– becoming a better Go player than any human
- Incorporates a number of interesting ideas

Table of Contents

Computing Action for Current State Only

- So far in class, compute a policy for whole state space
- Key idea: Use additional local computation to make a better decision for right now

Simple Monte-Carlo Search

- Given a model $\mathcal{M}_v$ and a simulation policy π
- For each action $a \in \mathcal{A}$
  - Simulate K episodes from current (real) state s_t

$$\{s_t, a, R^k_{t+1}, ..., S^k_T\}^K_{k=1} \sim \mathcal{M}_v, \pi$$

  - Evaluate actions by mean return (Monte-Carlo evaluation)

$$Q(s_t, a) = \frac{1}{K} \sum_{k=1}^{K} G_t \xrightarrow{P} q_\pi(s_t, a) \tag{1}$$

- Select current (real) action with maximum value

$$a_t = \underset{a \in A}{\operatorname{argmax}}\, Q(s_t, a)$$

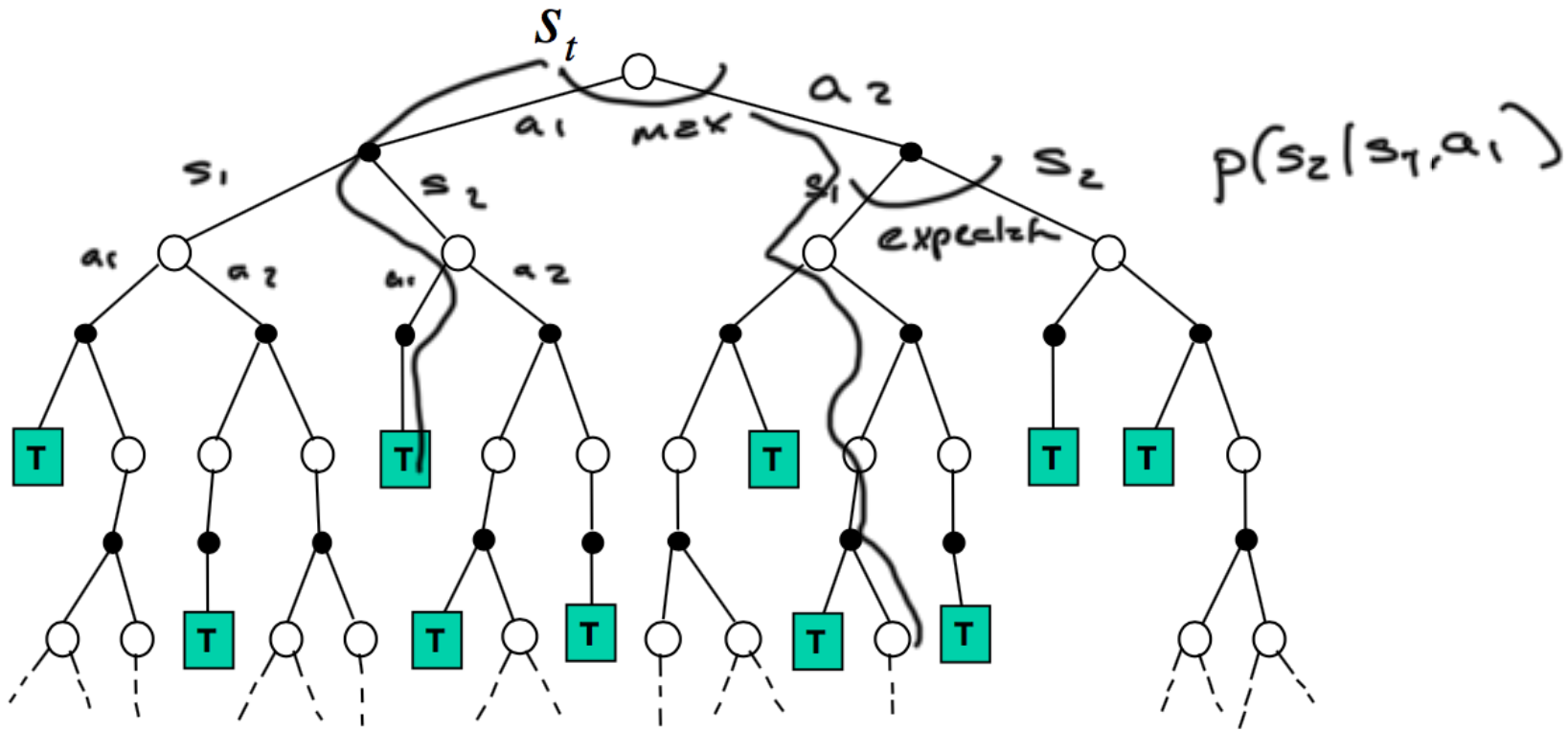
- This is essentially doing 1 step of policy improvement

Expectimax Tree

- Can we do better than 1 step of policy improvement?
- If have a MDP model $\mathcal{M}_v$
- Can compute optimal $Q(s, a)$ values for current state by constructing an **expectimax** tree

Forward Search Expectimax Tree

- Forward search algorithms select the best action by lookahead
- They build a search tree with the current state st at the root
- Using a model of the MDP to look ahead



- No need to solve whole MDP, just sub-MDP starting from now

Expectimax Tree

- Can we do better than 1 step of policy improvement?
- If have a MDP model $\mathcal{M}_v$
- Can compute optimal $q(s, a)$ values for current state by constructing an expectimax tree
- Limitations: Size of tree scales as $(|S||A|)^H$

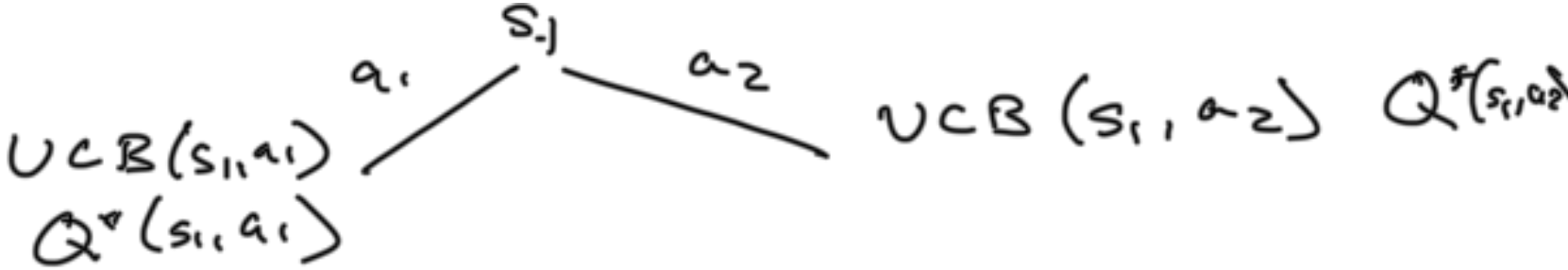
Monte-Carlo **Tree** Search (MCTS)

- Given a model $\mathcal{M}_v$
- Build a **search tree** rooted at the current state s_t
- Samples actions and next states
- Iteratively construct and update tree by performing K simulation episodes starting from the root state
- After search is finished, select current (real) action with maximum value in search tree

$$a_t = \underset{a \in A}{\operatorname{argmax}}\, Q(s_t, a)$$

Upper Confidence Tree (UCT) Search

- How to select what action to take during a simulated episode?
- UCT: borrow idea from bandit literature and treat each node where can select actions as a multi-armed bandit (MAB) problem
- Maintain an upper confidence bound over reward of each arm



Upper Confidence Tree (UCT) Search

- How to select what action to take during a simulated episode?
- UCT: borrow idea from bandit literature and treat each **node** where can select actions as a multi-armed bandit (MAB) problem
- Maintain an upper confidence bound over reward of each arm at a node

$$Q(s,a,i) = \frac{1}{N(i,a)} \sum_{k=1}^{N(i,a)} G_k(i,a) + c\sqrt{\frac{O(\log N(i))}{N(i,a)}}$$

- where $N(i,a)$ is the number of times selected arm a at node i, $G_k(i,a)$ is the k-th return (discounted sum of rewards) from node i following action a, and
- For simulated episode k at node i, select action/arm with highest upper bound to simulate and expand (or evaluate) in the tree

$$a_{ik} = \arg\max Q(s,a,i)$$

- This implies that the policy used to simulate episodes with (and expand/update the tree) can change across each episode

Advantages of MC Tree Search

- Highly selective best-first search
- Evaluates states dynamically
- Uses sampling to break curse of dimensionality
- Works for "black-box" models (only requires samples)
- Computationally efficient, anytime, parallelisable