Lecture 14: Monte Carlo Tree Search

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2026

- With some slides from or derived from David Silver

Class Structure

- Last time: MCTS and guest lecture Shane Gu from DeepMind on World Models
- **This Time: MCTS and Ethics and Society Guest Lecture Part 2**
- Next time: Quiz

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

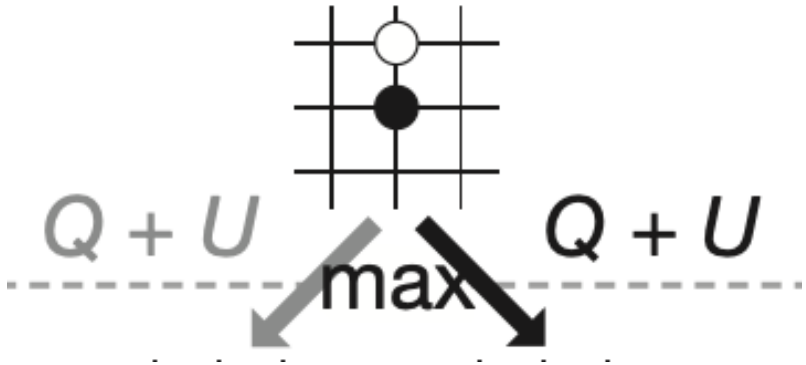
Table of Contents

Case Study: the Game of Go

- Go is 2500 years old
- Hardest classic board game
- Grand challenge task (John McCarthy)
- Traditional game-tree search has failed in Go
- Trailer: ▸ AlphaGo trailer link
- Check your understanding: does playing Go involve learning to make decisions in a world where dynamics and reward model are unknown?

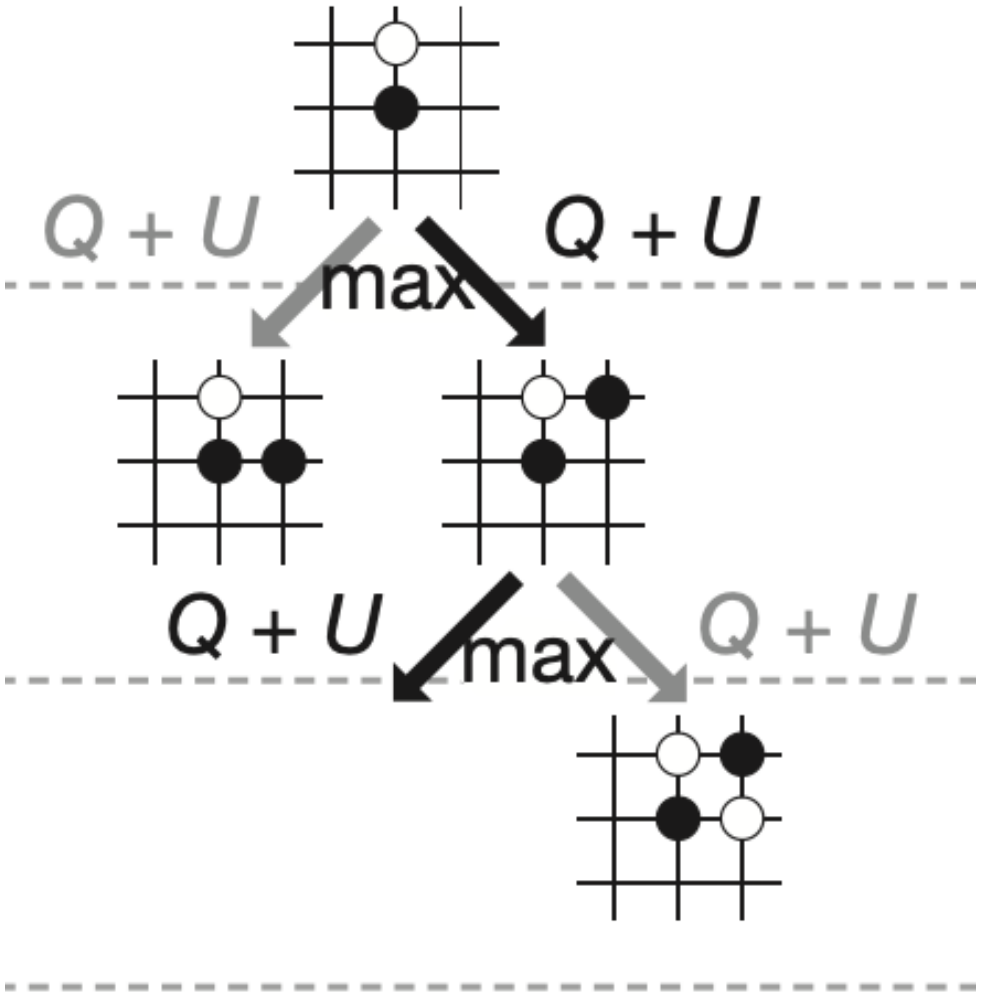
Selecting a Move in a Single Game: Start at Root[1]

- Inspired by Upper Confidence Tree Search but many changes. Use PUCT.

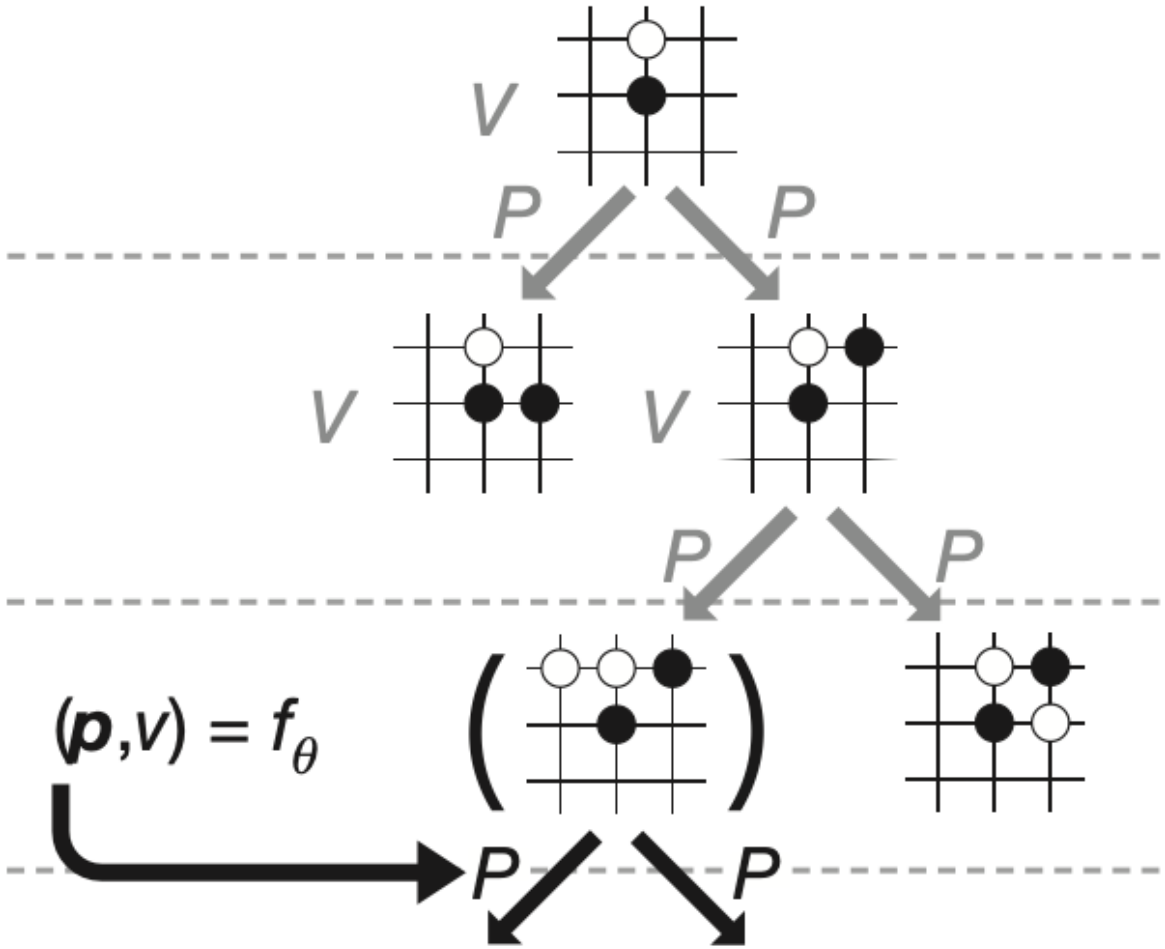


[1] Images from Silver et al. Nature 2017

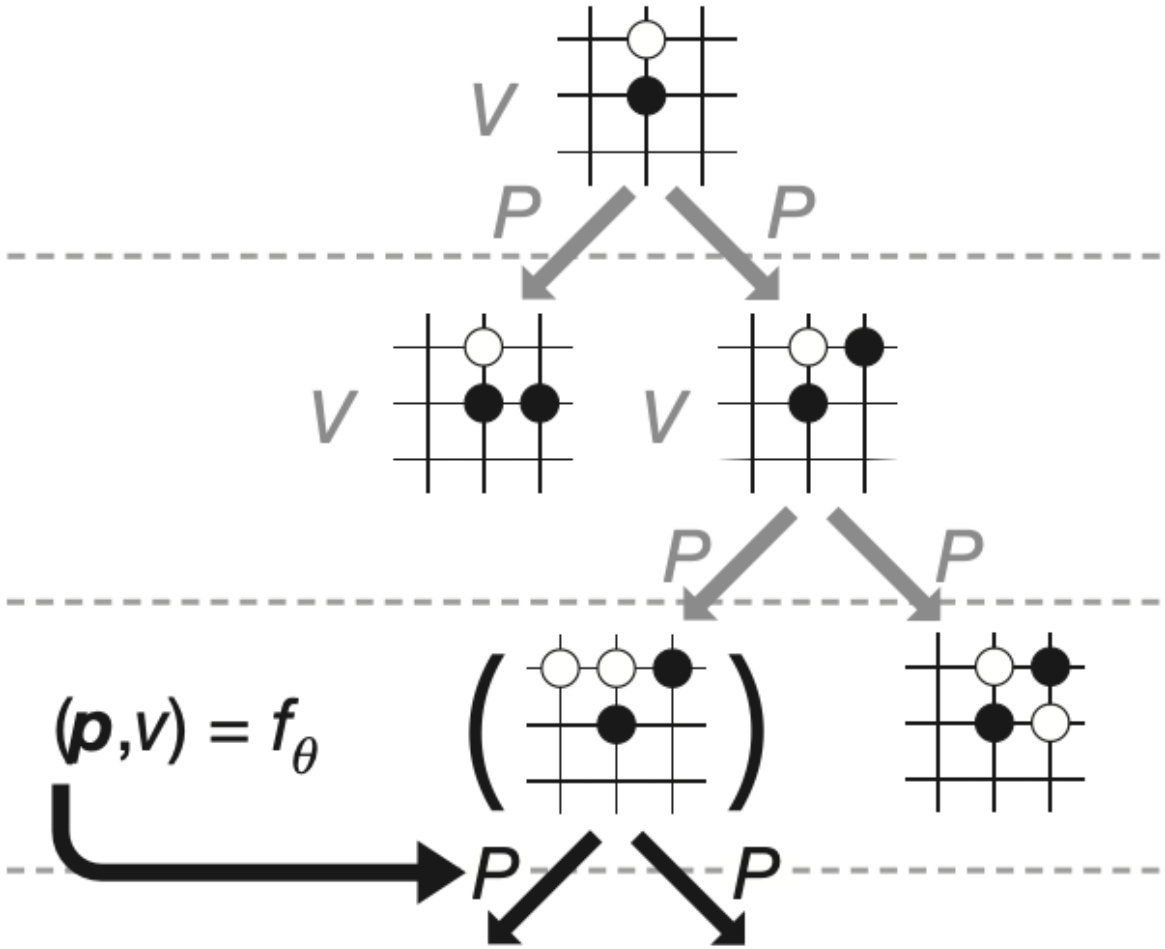
Selecting a Move in a Single Game: Repeatedly Expand[2]



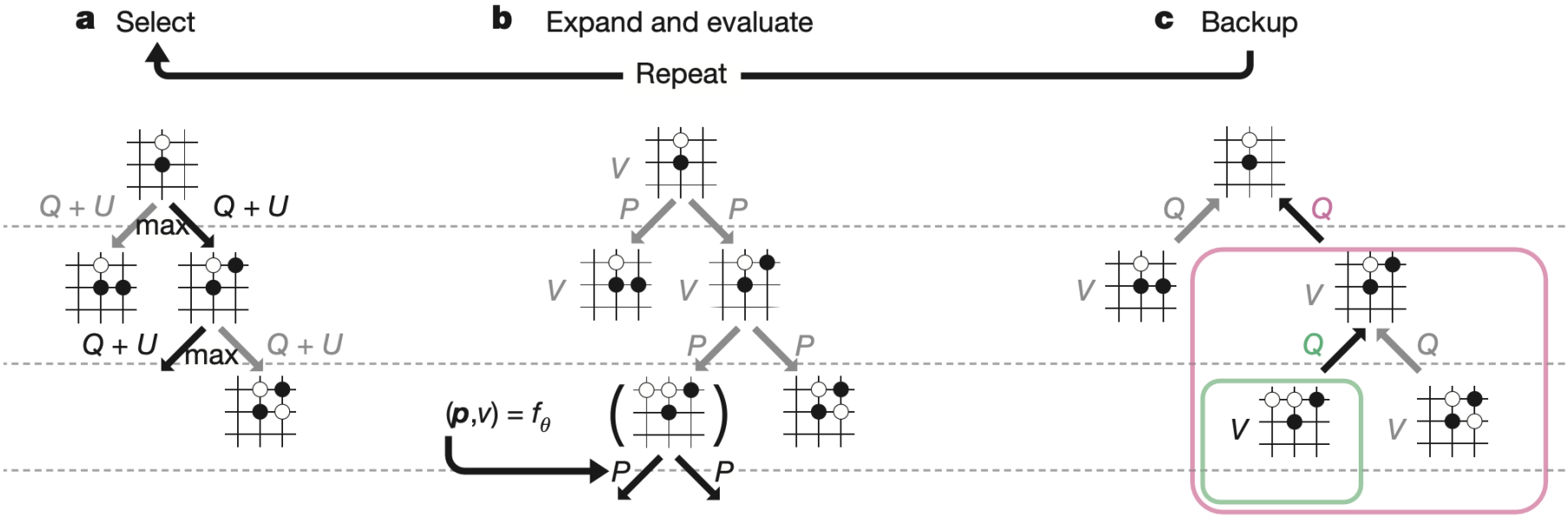
Selecting a Move in a Single Game: Note Using Network Predictions for Action Probabilities[3]



Selecting a Move in a Single Game: At Leaf, Plug in Network Predictions for Value[4]

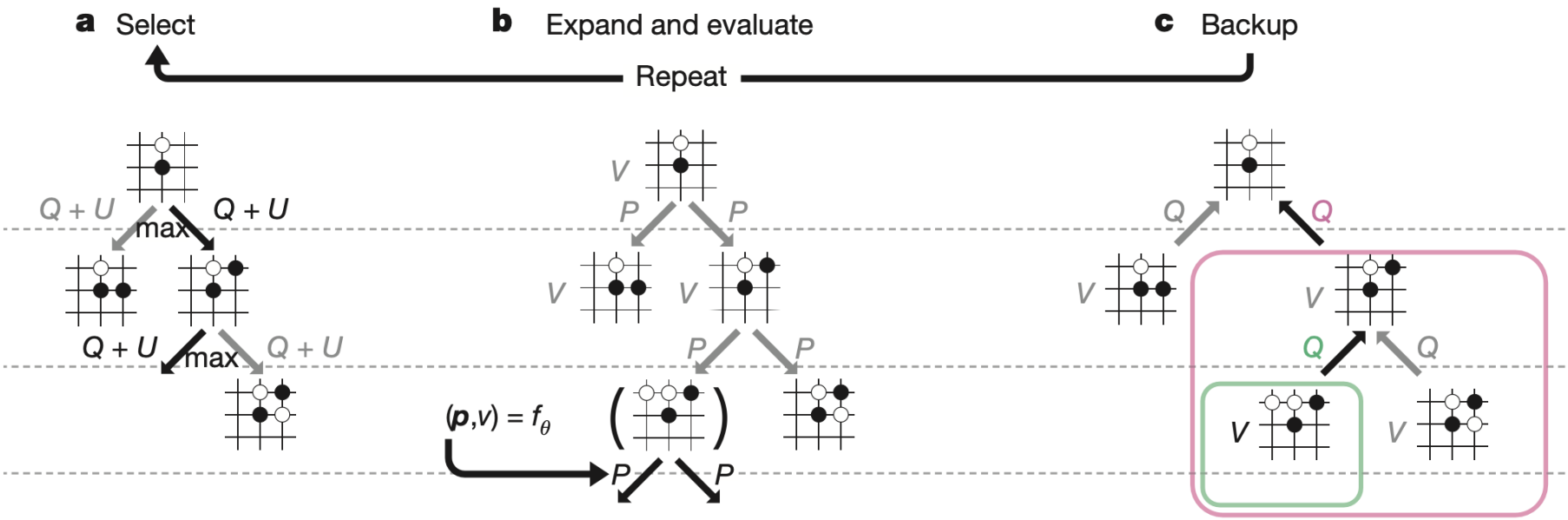


Selecting a Move in a Single Game: Update Ancestors[5]



[5]Images from Silver et al. Nature 2017

Selecting a Move in a Single Game: Repeat Many Times[6]

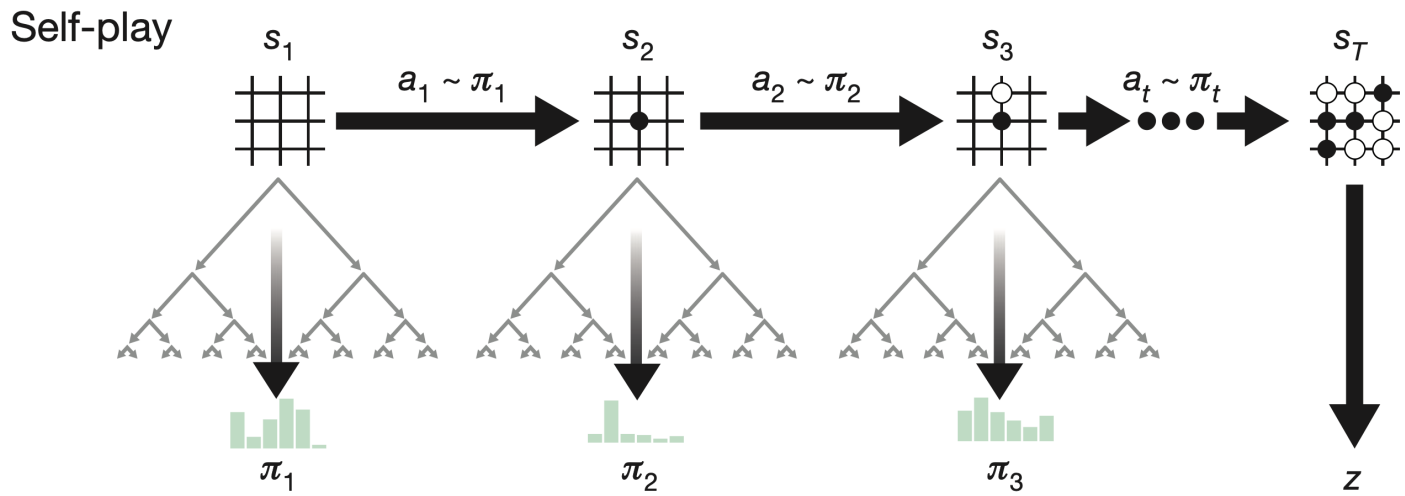


- Repeat roll out and backup process many times
- Note: inside the network alternating whether opponent or agent is "maximizing" its value. Therefore tree is mimicking a min-max tree
- At end, compute a policy for root node by

$$\pi(s) \propto N(s,a)^{\frac{1}{\tau}} \tag{1}$$

[6] Images from Silver et al. Nature 2017

Self Play a Game[7]



- Select an action according to root policy, take action, and repeat whole process
- Repeat until game ends* and observe a win or loss

[7] Images from Silver et al. Nature 2017

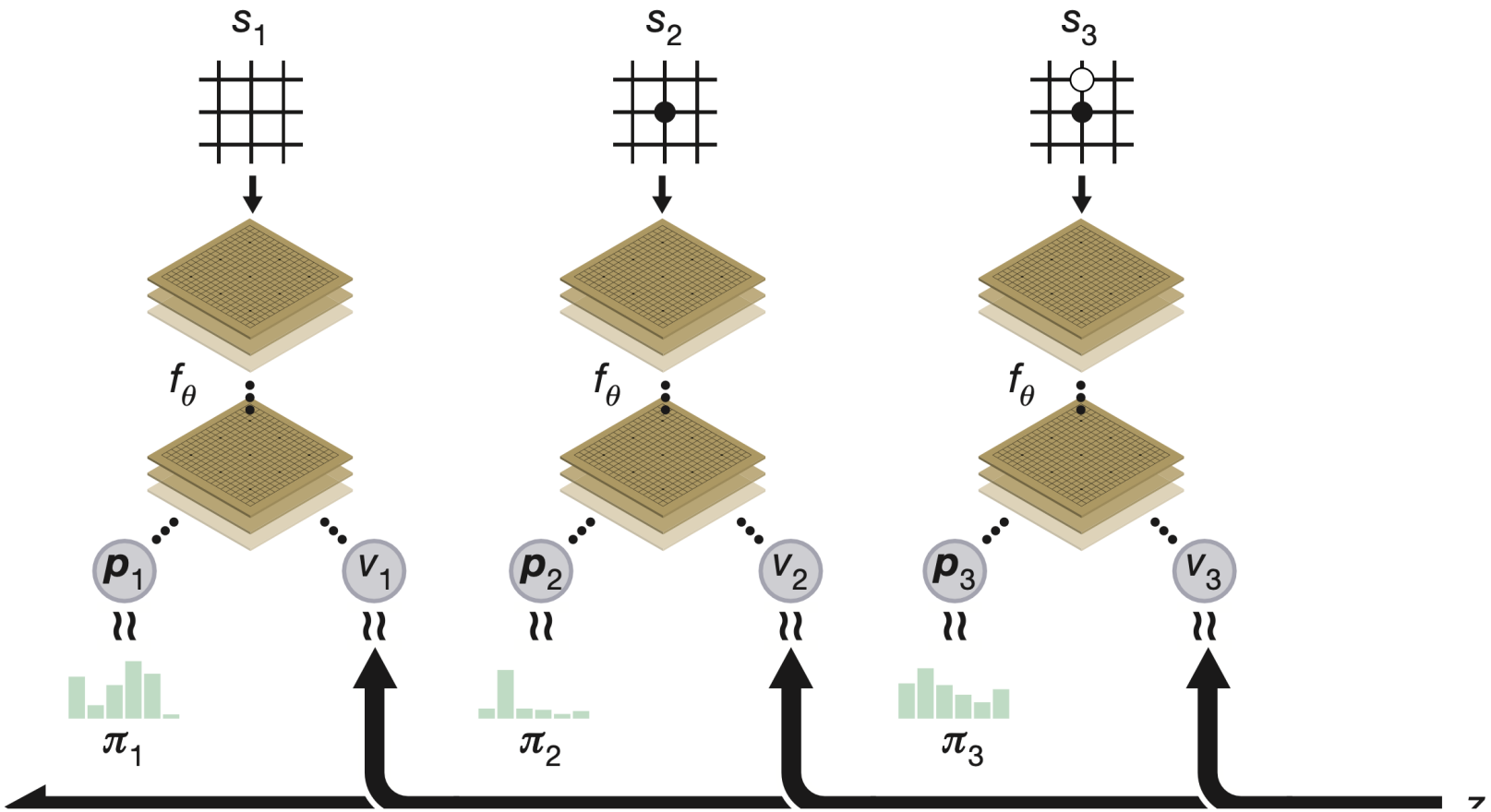
Advantages of Self Play for Go

- Bottleneck is only computation, no humans needed
- Self-play also provides a well-matched player
- Optional check your understanding: how does this help with policy training? What is the reward density?

Self Play for Go: Solution

- Bottleneck is only computation, no humans needed
- Self-play also provides a well-matched player
- Check your understanding: how does this help with policy training? What is the reward density?
  Rewards will be quite dense as both players are evenly matched. This provides a form of curriculum learning.

Train Neural Network to Predict Policies and Values [8]



[8] Images from Silver et al. Nature 2017

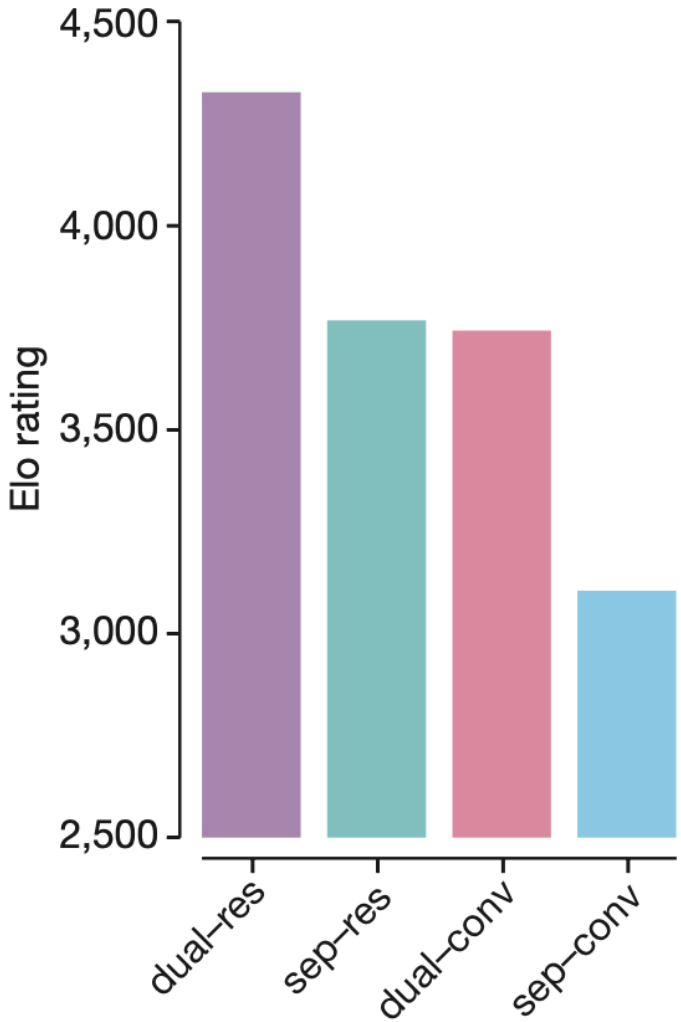
AlphaGo and AlphaZero

- Self Play
- Strategic Computation
- Highly selective best-first search
- Power of Averaging
- Local Computation
- Learn and Update Heuristics

AlphaGo and AlphaZero: Recap and Evaluation

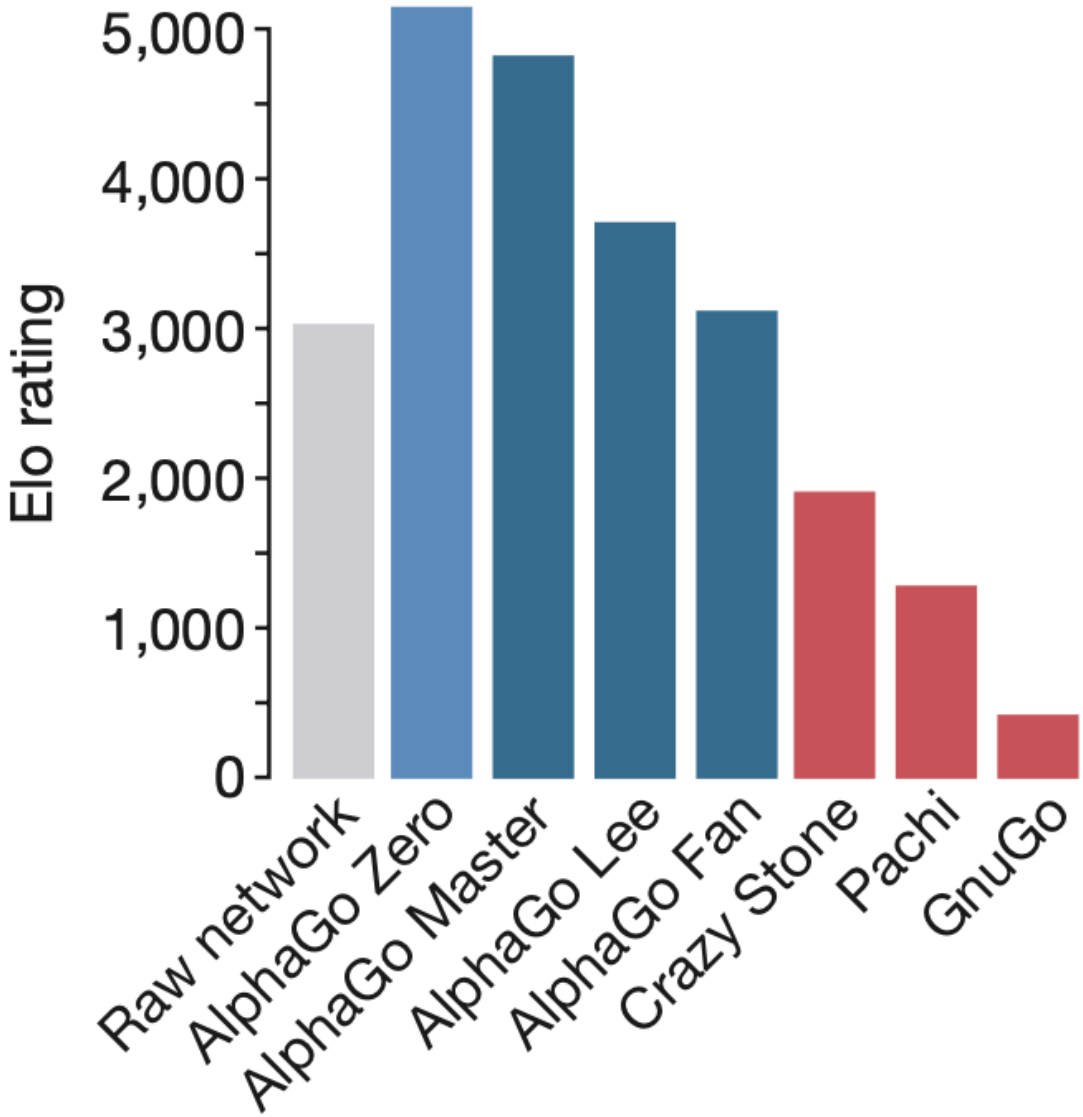
- Features:
  - Self Play
  - Strategic Computation
  - Highly selective best-first search
  - Power of Averaging
  - Local Computation
  - Learn and Update Heuristics
- Evaluation Questions
  - What is the influence of architecture?
  - What is the impact of using MCTS (on top of learning a policy / value function)?
  - How does it compare to human play or using human play?

Impact of Architecture[9]



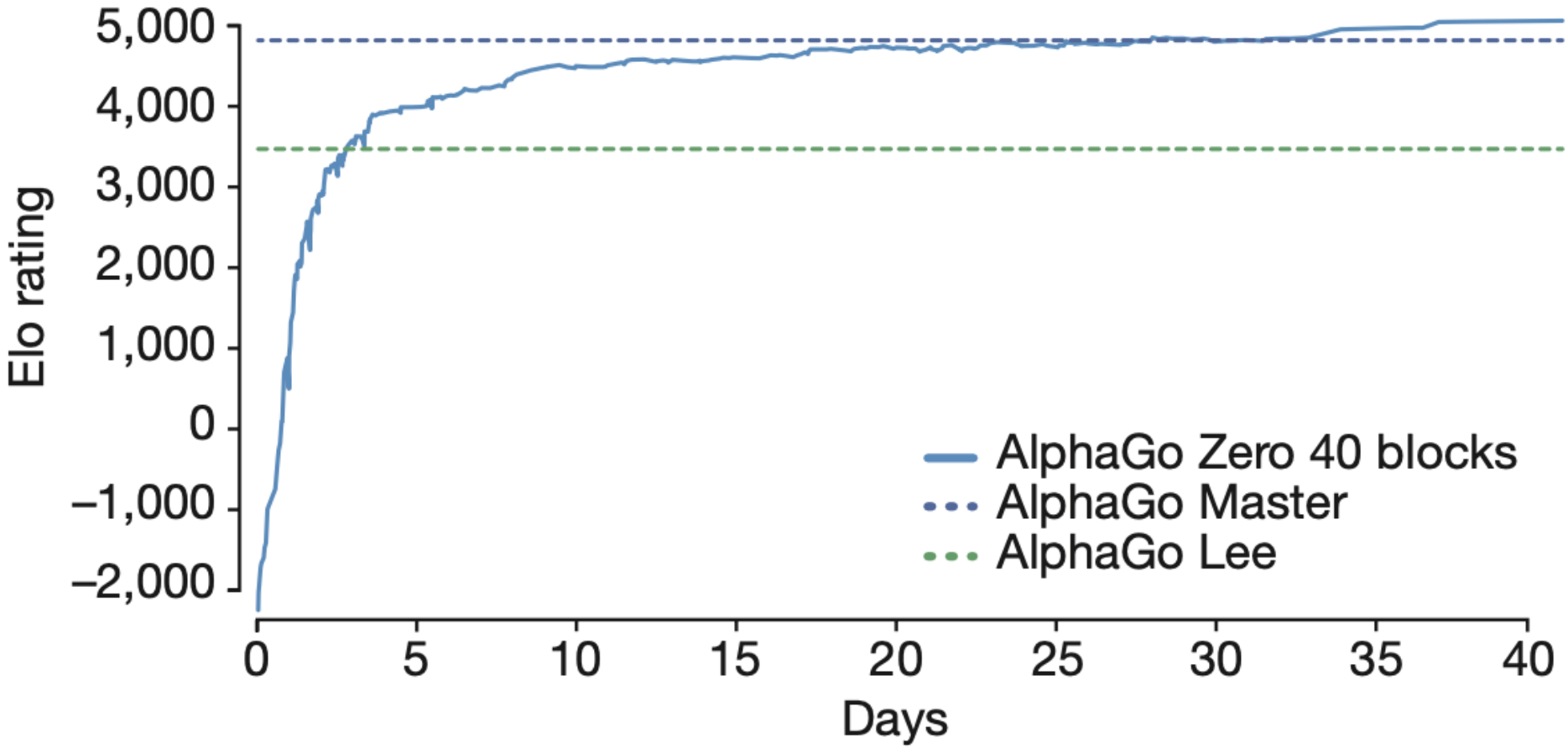
[9] Images from Silver et al. Nature 2017

Impact of MCTS[10]



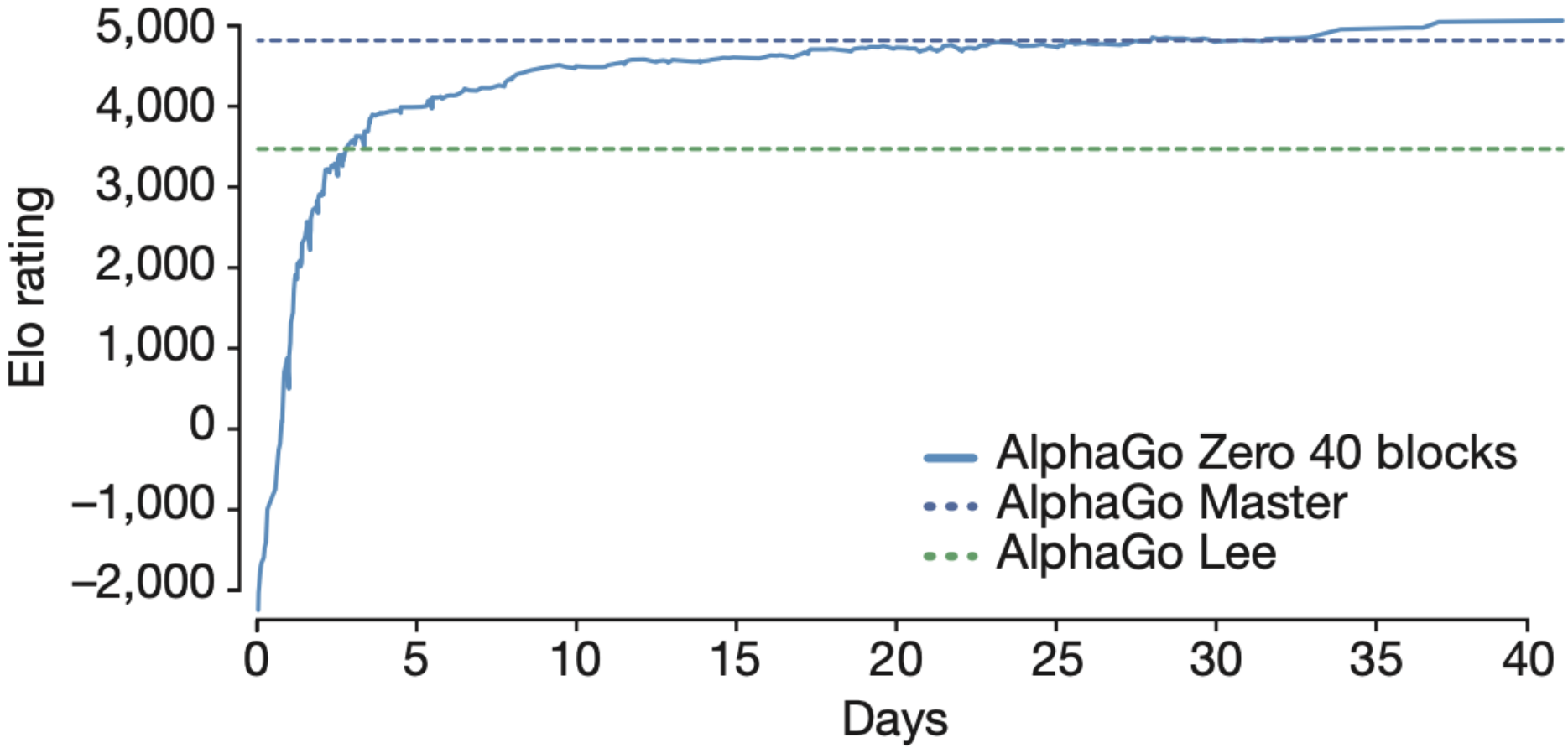
[10] Images from Silver et al. Nature 2017

Overall performance[11]



[11] Images from Silver et al. Nature 2017

Need for Human Data?[12]



[12] Images from Silver et al. Nature 2017

AlphaZero

- These ideas are useful beyond Go. For chess, shogi, other games...
- For discovering faster matrix multiplication (AlphaTensor)
- For discovering faster sorting algorithms (AlphaDev)
- Beautiful insight: using RL to vastly speed up problems can represent as (extremely large) search problems

Class Structure

- Last time: Guest lecture
- **This Time: MCTS**
- Next time: Quiz

Optional Check Your Understanding: MCTS

- MCTS involves deciding on an action to take by doing tree search where it picks actions to maximize $Q(S, A)$ and samples states. Select all
  1. Given a MDP, MCTS may be a good choice for short horizon problems with a small number of states and actions.
  2. Given a MDP, MCTS may be a good choice for long horizon problems with a large action space and a small state space
  3. Given a MDP, MCTS may be a good choice for long horizon problems with a large state space and small action space
  4. Not sure

Optional Check Your Understanding: MCTS Solutions

- MCTS involves deciding on an action to take by doing tree search where it picks actions to maximize $Q(S, A)$ and samples states. Select all
  1. Given a MDP, MCTS may be a good choice for short horizon problems with a small number of states and actions.
  2. Given a MDP, MCTS may be a good choice for long horizon problems with a large action space and a small state space
  3. Given a MDP, MCTS may be a good choice for long horizon problems with a large state space and small action space
  4. Not sure

F F T

In more depth: Upper Confidence Tree (UCT) Search. Optional CYU

- UCT: borrow idea from bandit literature and treat each tree node where can select actions as a multi-armed bandit (MAB) problem
- Maintain an upper confidence bound over reward of each arm and select the best arm
- Check your understanding: Why is this slightly strange? Hint: why were upper confidence bounds a good idea for exploration/ exploitation? Is there an exploration/ exploitation problem during simulated episodes?[13]

[13] Relates to metalevel reasoning (for an example related to Go see "Selecting Computations: Theory and Applications", Hay, Russell, Tolpin and Shimony 2012)

Optional Check Your Understanding: UCT Search

- In Upper Confidence Tree (UCT) search we treat each tree node as a multi-armed bandit (MAB) problem, and use an upper confidence bound over the future value of each action to help select actions for later rollouts. Select all that are true
  1. This may be useful since it will prioritize actions that lead to later good rewards
  2. UCB minimizes regret. UCT is minimizing regret within rollouts of the tree. (If this is true, think about if this a good idea?)
  3. Not sure

Optional Check Your Understanding: UCT Search

- In Upper Confidence Tree (UCT) search we treat each tree node as a multi-armed bandit (MAB) problem, and use an upper confidence bound over the future value of each action to help select actions for later rollouts. Select all that are true
  1. This may be useful since it will prioritize actions that lead to later good rewards
  2. UCB minimizes regret. UCT is minimizing regret within rollouts of the tree. (If this is true, think about if this a good idea?)
  3. Not sure

T. T

Refresh Your Understanding

Select all that are true:

- Upper confidence bounds are used to balance exploration and leveraging the acquired information to achieve high reward
- These algorithms can be used in bandits and Markov decision processes
- If the reward model is known, there is no benefit to using an upper confidence bound algorithm

Refresh Your Understanding

Select all that are true:

- Upper confidence bounds are used to balance exploration and leveraging the acquired information to achieve high reward
- These algorithms can be used in bandits and Markov decision processes
- If the reward model is known, there is no benefit to using an upper confidence bound algorithm

True. True. Depends on setting. In bandits, no additional gain. In RL, if the dynamics model is not known, there will be a gain.