Lecture 14: Monte Carlo Tree Search

Emma Brunskill

CS234 Reinforcement Learning.

- With many slides from or derived from David Silver

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

T

T

- in MAB setting this is true in RL it usually fails

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.

Class Structure

- Last time: Fast / sample efficient Reinforcement Learning
- **This Time: MCTS**
- Next time: Rewards in RL

AlphaZero and Monte Carlo Tree Search

- 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: can prioritize some additional local computation to make a better decision for right now

Simple Monte-Carlo Search

✓ dynamics & reward model

- 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

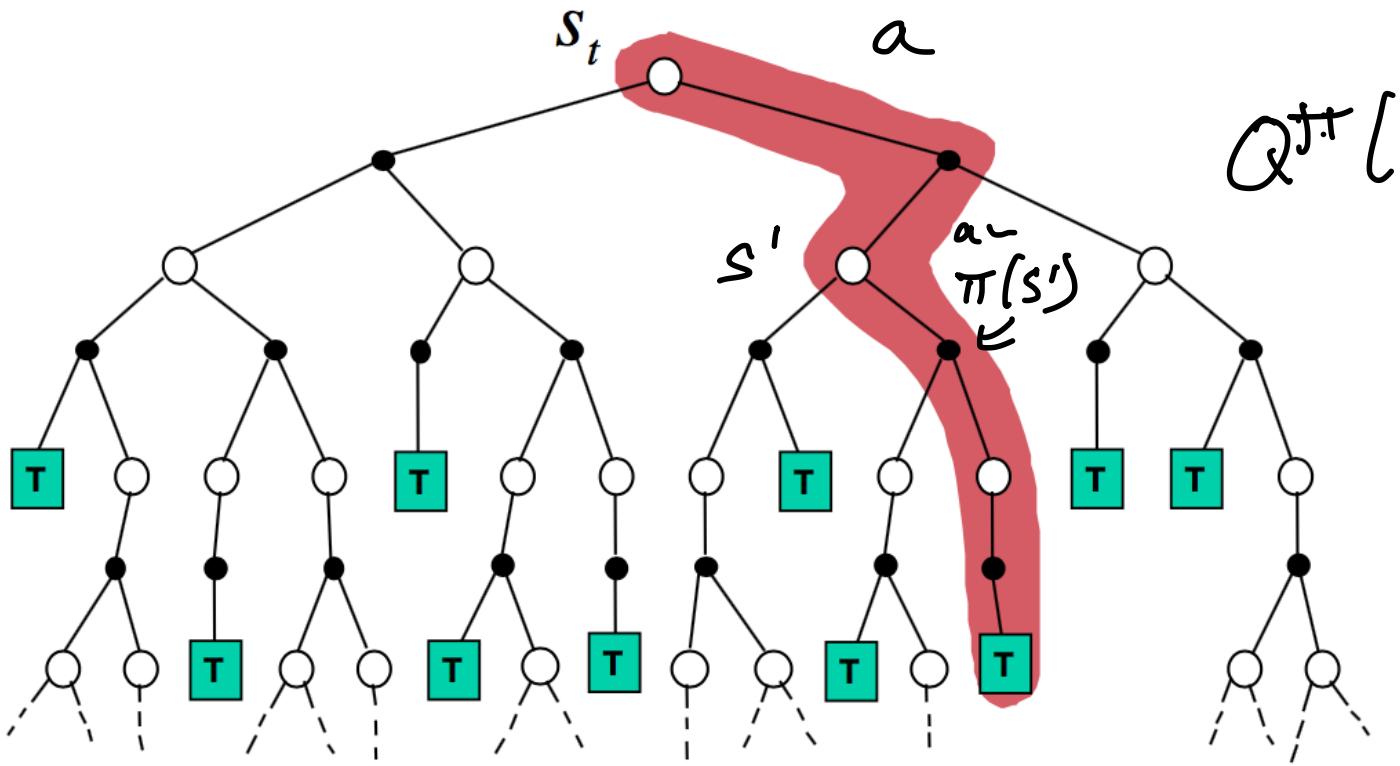
$$a_t = \underset{a \in A}{\operatorname{argmax}}\, Q(s_t, a)$$

π

- This is essentially doing 1 step of policy improvement

Simulation-Based Search

- Simulate episodes of experience from now with the model
- Apply model-free RL to simulated episodes

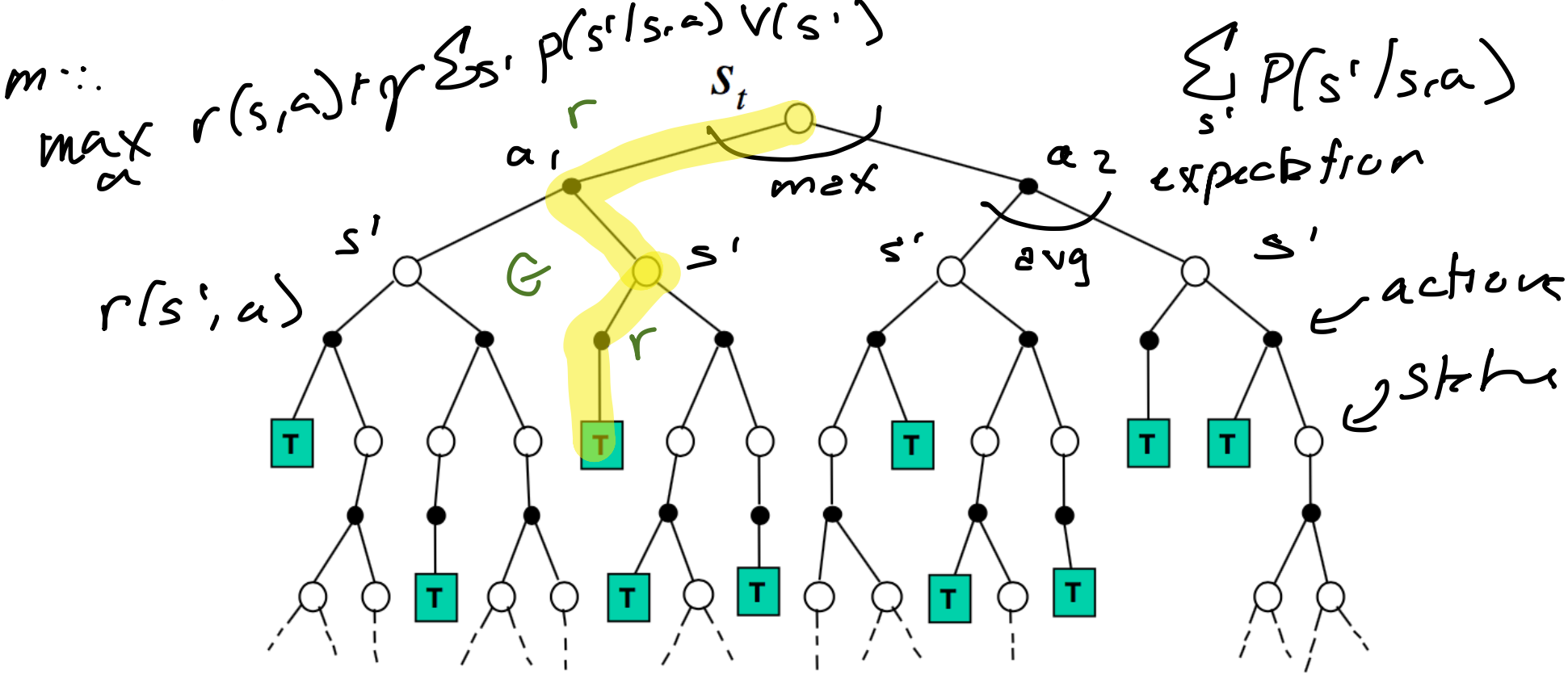


Expectimax Tree

- 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)

approximations expectations w/ averages

- 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

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

- Check your understanding: How does this differ from Monte Carlo Simulated Search?

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

F

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 maybe false
3. Given a MDP, MCTS may be a good choice for long horizon problems with a large state space and small action space

T

4. Not sure

approximating expectation by sample

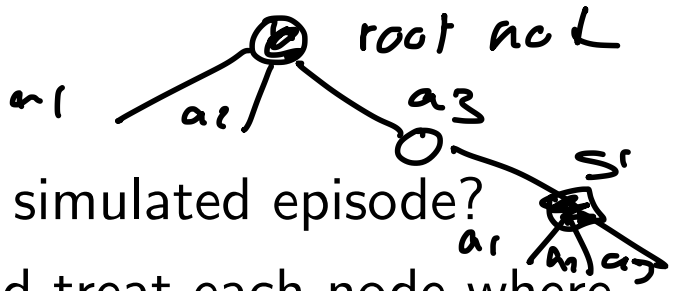
just do Bellman backups

$S^2 A \cdot H$

Upper Confidence Tree (UCT) Search

- How to select what action to take during a simulated episode?

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 i

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

- 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 (unlike e.g. DP)
- Uses sampling to break curse of dimensionality
- Works for "black-box" models (only requires samples)
- Computationally efficient, anytime, parallelisable

UCT to help w/ large action spaces

Table of Contents

AlphaGo

- AlphaGo trailer link

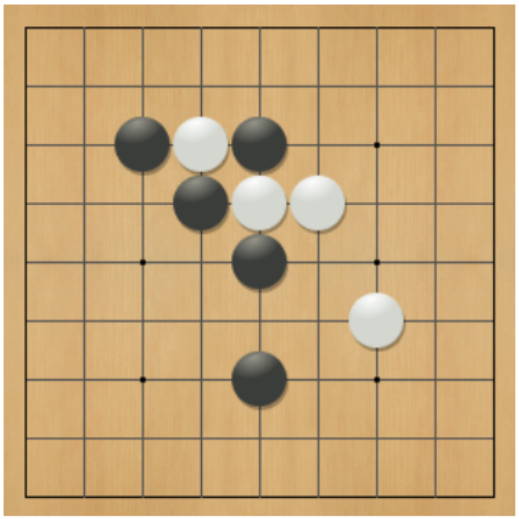
Case Study: the Game of Go

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

Rules of Go

- Usually played on 19x19, also 13x13 or 9x9 board
- Simple rules, complex strategy
- Black and white place down stones alternately
- Surrounded stones are captured and removed
- The player with more territory wins the game

Zero-one game



AlphaGo and AlphaZero

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

Self Play for Go

- Key idea: have agent play itself
- Game operates by computing best move at current state, then, for opponent move, doing the same
- 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?

Self Play for Go: Solution

- Key idea: have agent play itself
- Game operates by computing best move at current state, then, for opponent move, doing the same
- 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.

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

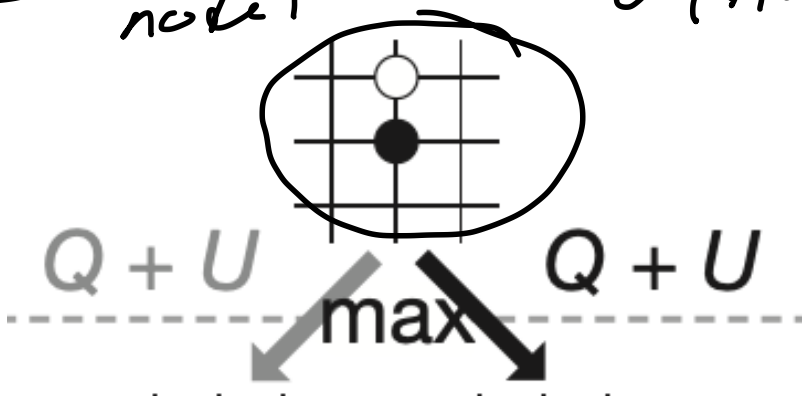
neural network $s \Rightarrow \hat{V}(s), P(s)$

policy network

- Inspired by Upper Confidence Tree Search but many changes

$Q = \frac{1}{N(i,a)} \sum_{s'} V(s')$

node i



$U(i,a) \propto \frac{P(s,a)}{1+N(i,a)}$

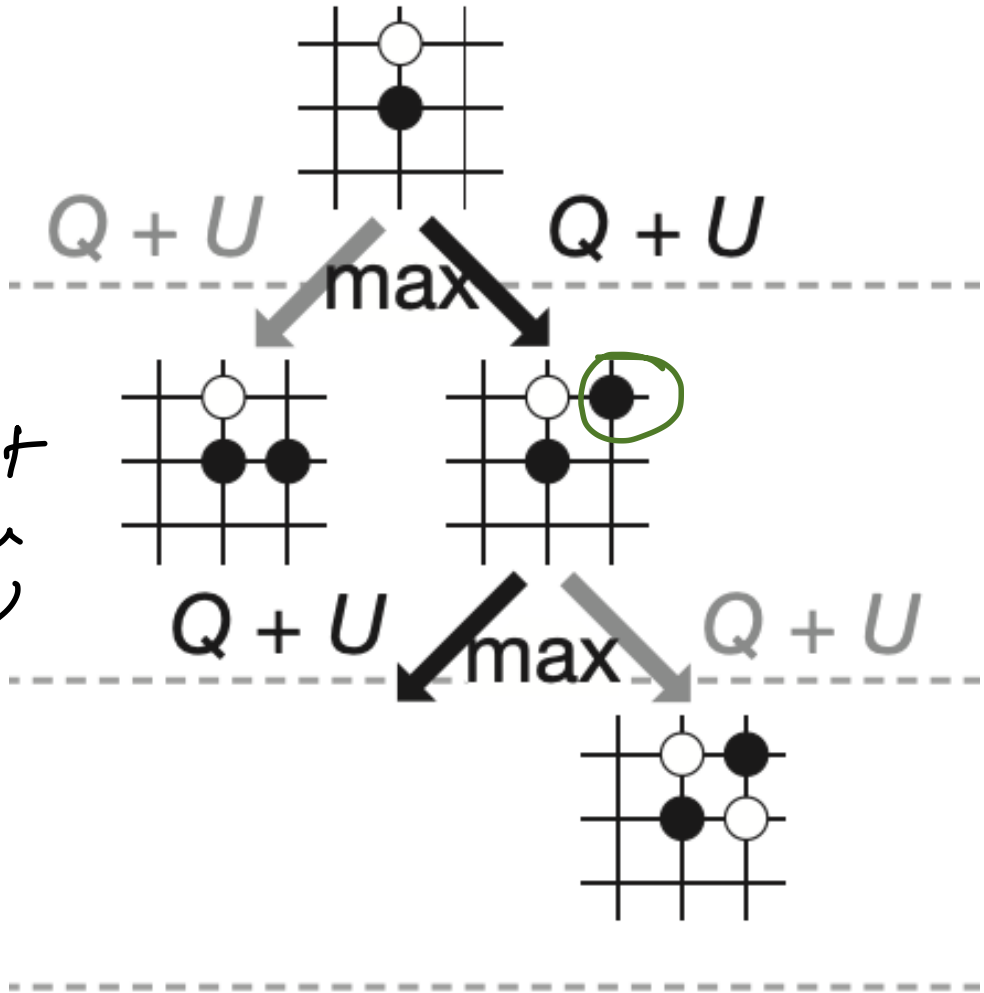
node visits

recall UCT

$U \propto \frac{1}{\sqrt{N(i,a)}}$

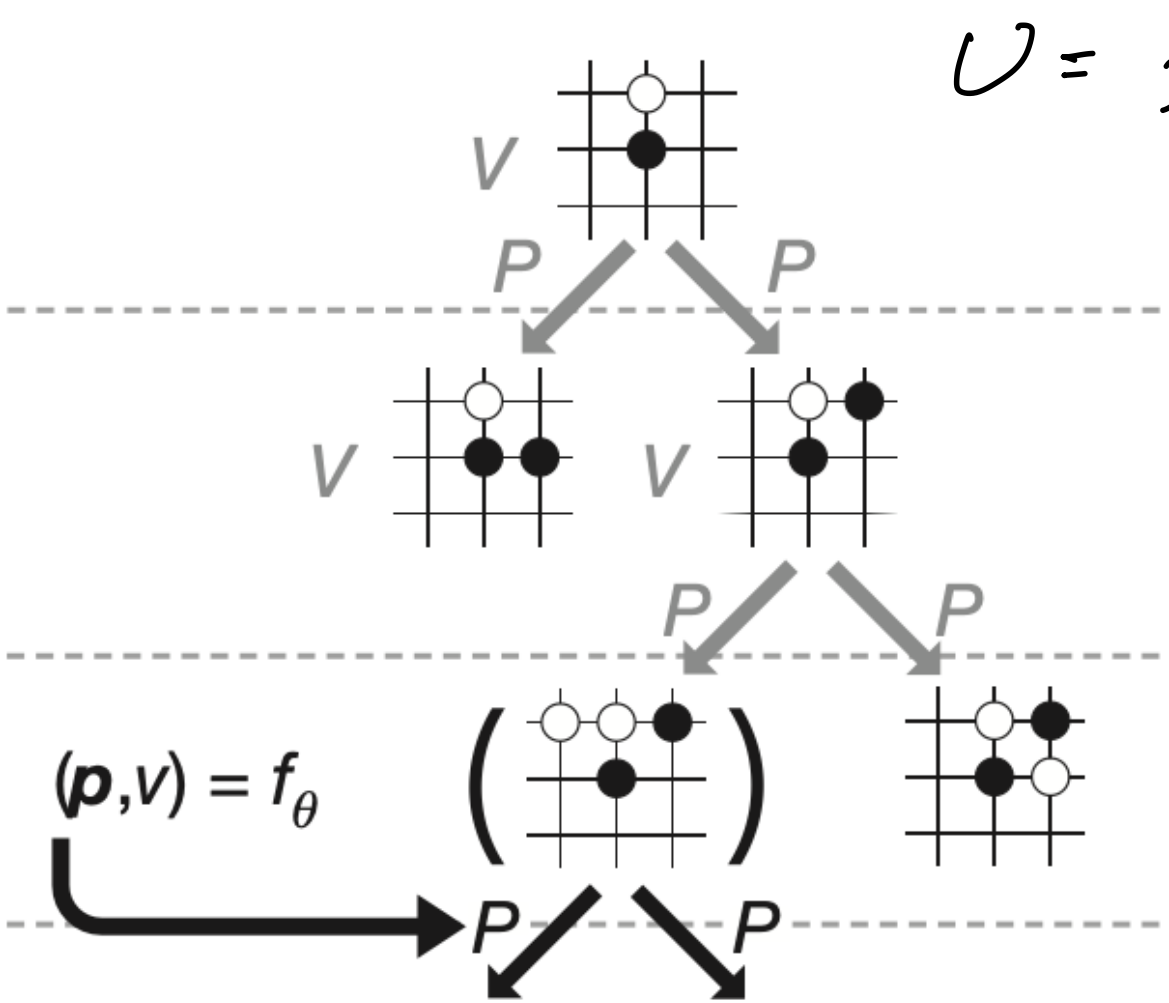
[1] Images from Silver et al. Nature 2017

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

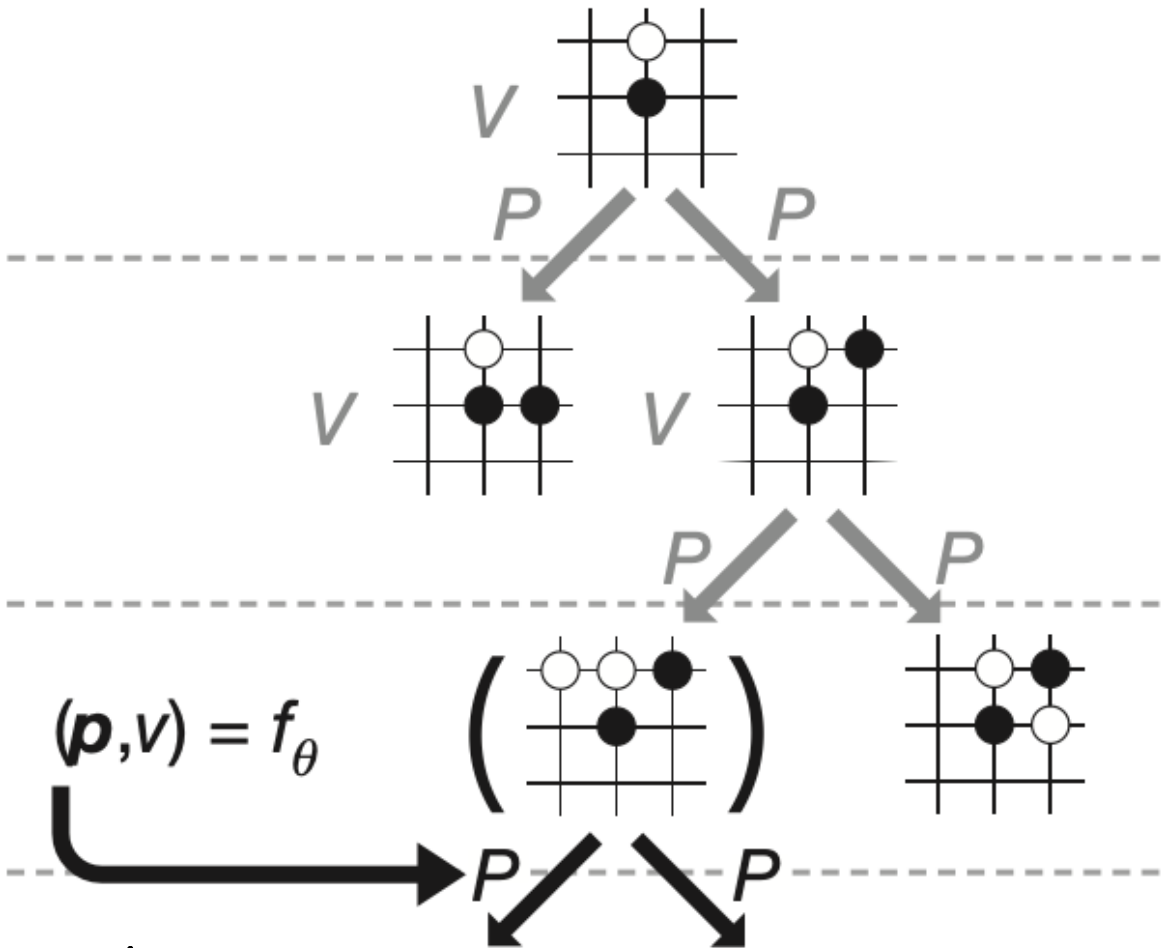


opponent selects the max Q + U

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



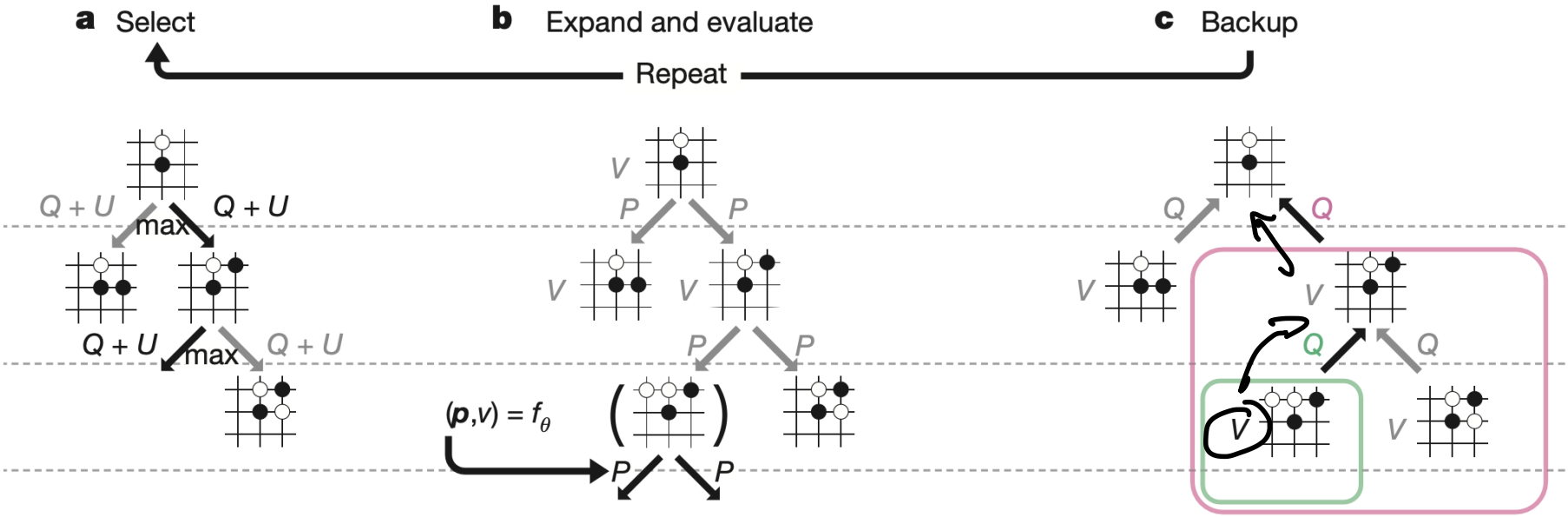
Selecting a Move in a Single Game: At Leaf, Plug in Network Predictions for Value[4]



use our NN to plug in V(s)

V(s) T

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

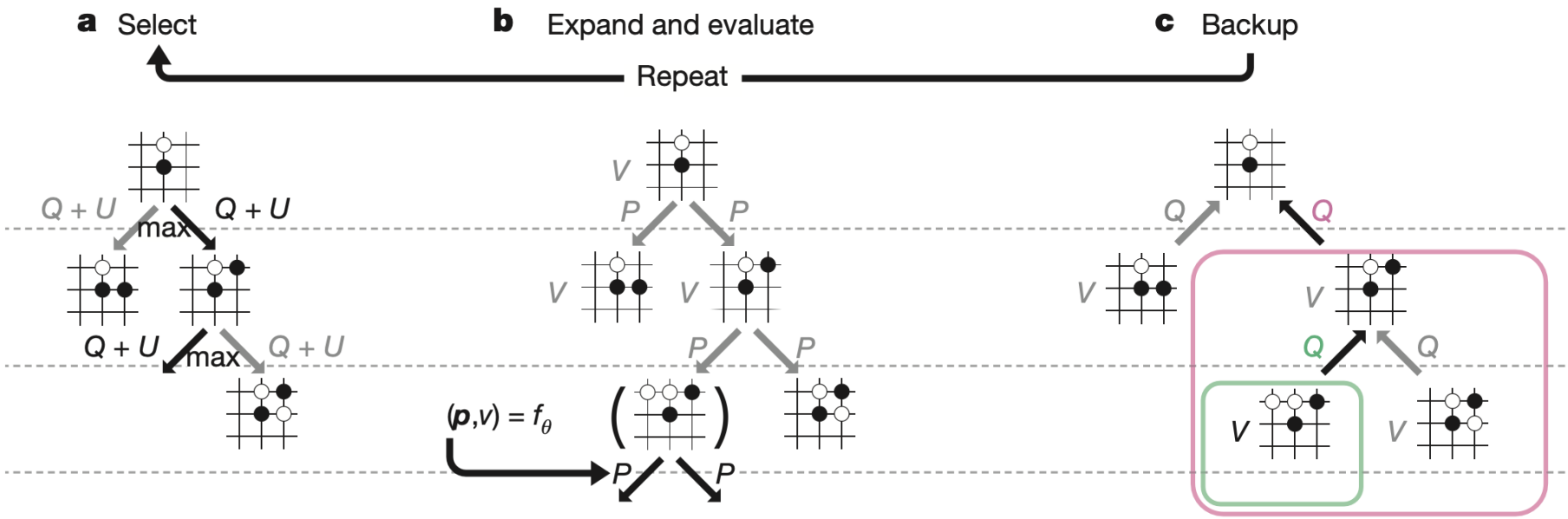


$$Q = \frac{1}{N(s,a)} \sum_{s'} V(s')$$

many many many times

[5] Images from Silver et al. Nature 2017

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

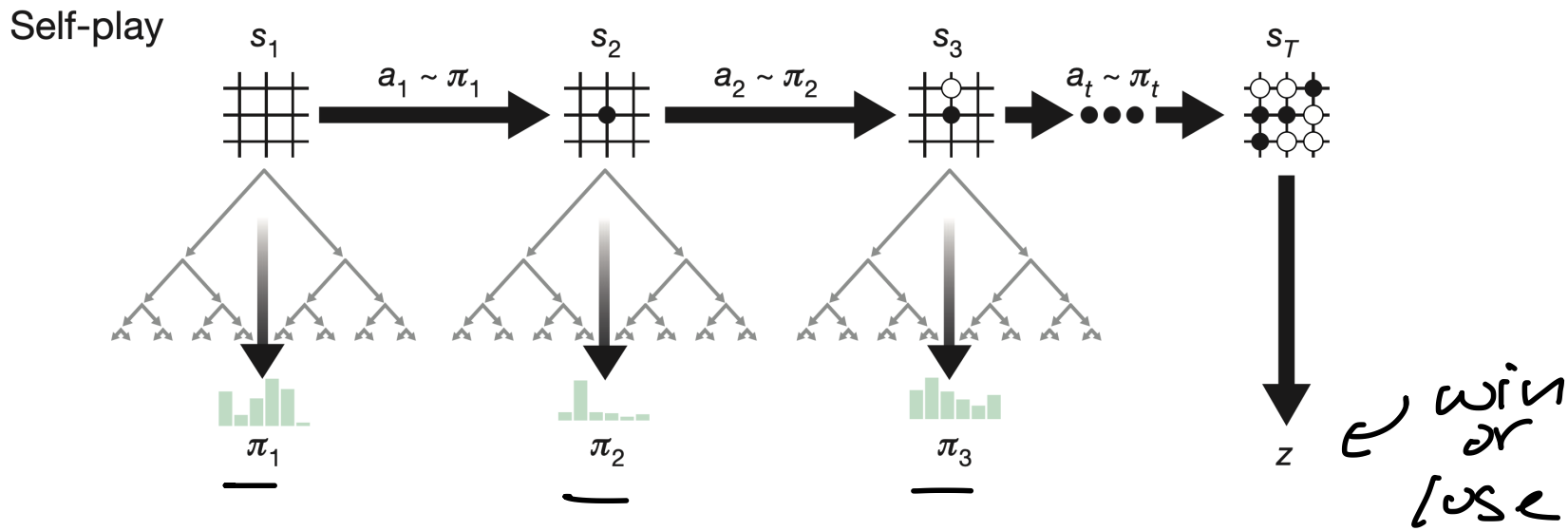


- Repeat roll out and backup process many times
- Note: inside the network alternating whether opponent or agent is "maximizing" its value. Therefore tree is mimicking a min-max tree
- At end, compute a policy for root node by

$$\pi(s) \propto N(s,a)^{\frac{1}{\tau}} \quad (2)$$

[6] Images from Silver et al. Nature 2017

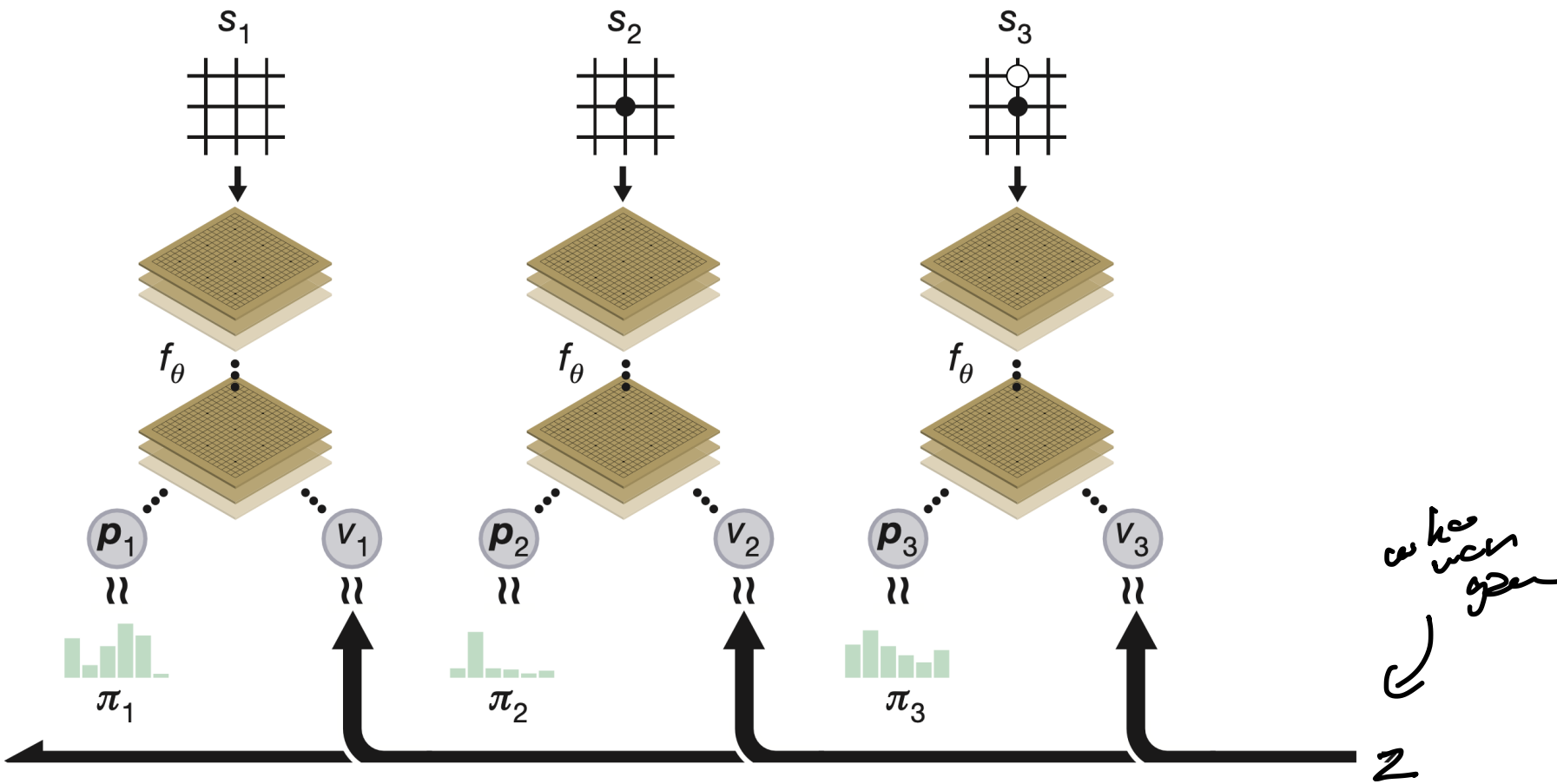
Self Play a Game[7]



- 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

Train Neural Network to Predict Policies and Values footnoteImages from Silver et al. Nature 2017

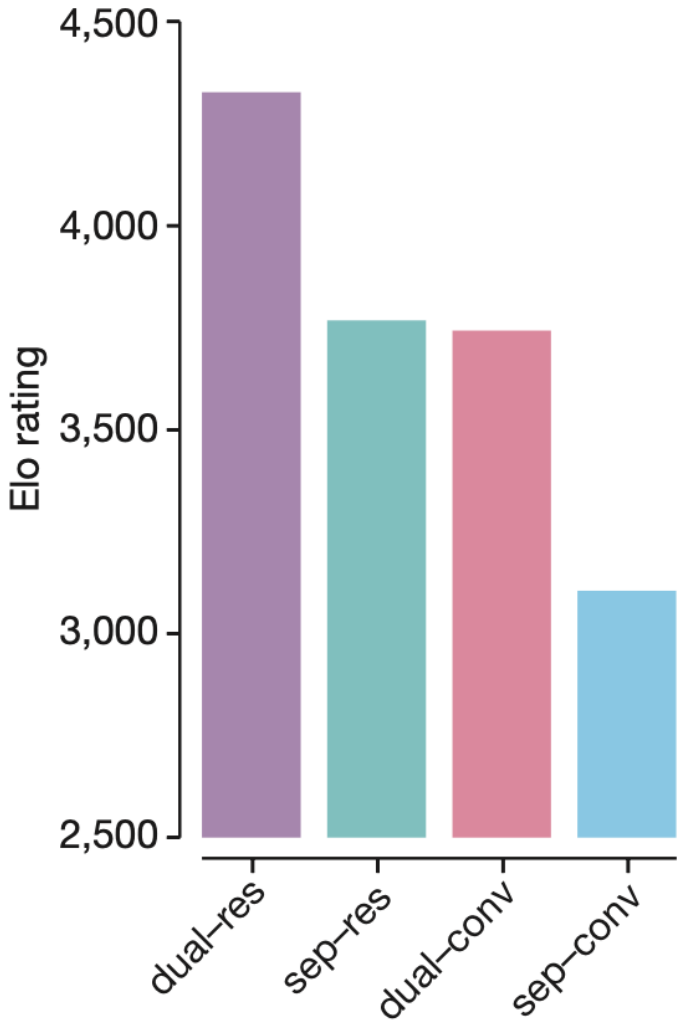


AlphaGo and AlphaZero: Recap and Evaluation

still know game rules

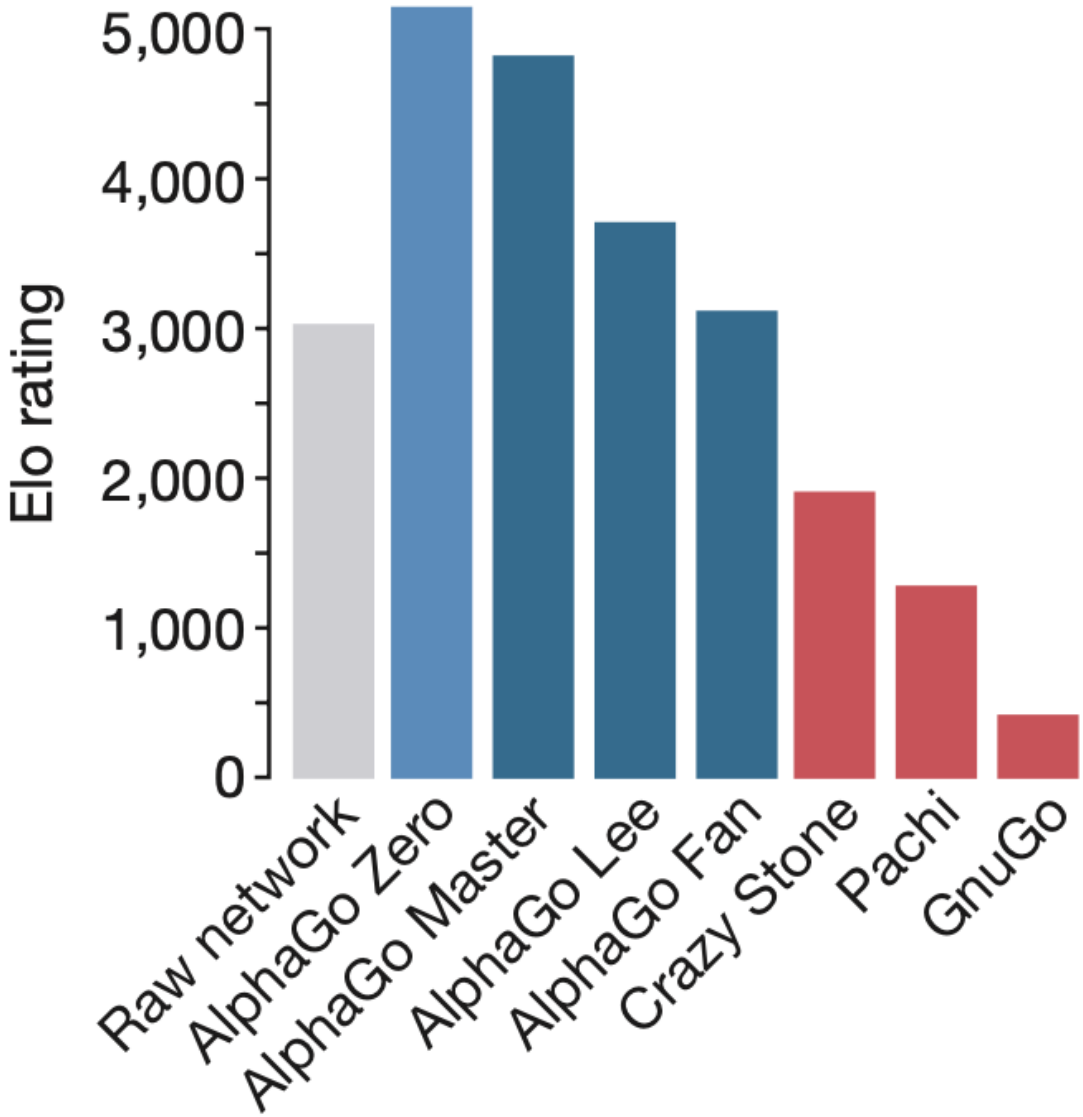
- Features:
  - Self Play
  - Strategic Computation
  - Highly selective best-first search
  - Power of Averaging
  - Local Computation
  - Learn and Update Heuristics
- Evaluation Questions
  - What is the influence of architecture?
  - What is the impact of using MCTS (on top of learning a policy /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

Impact of Architecture[8]



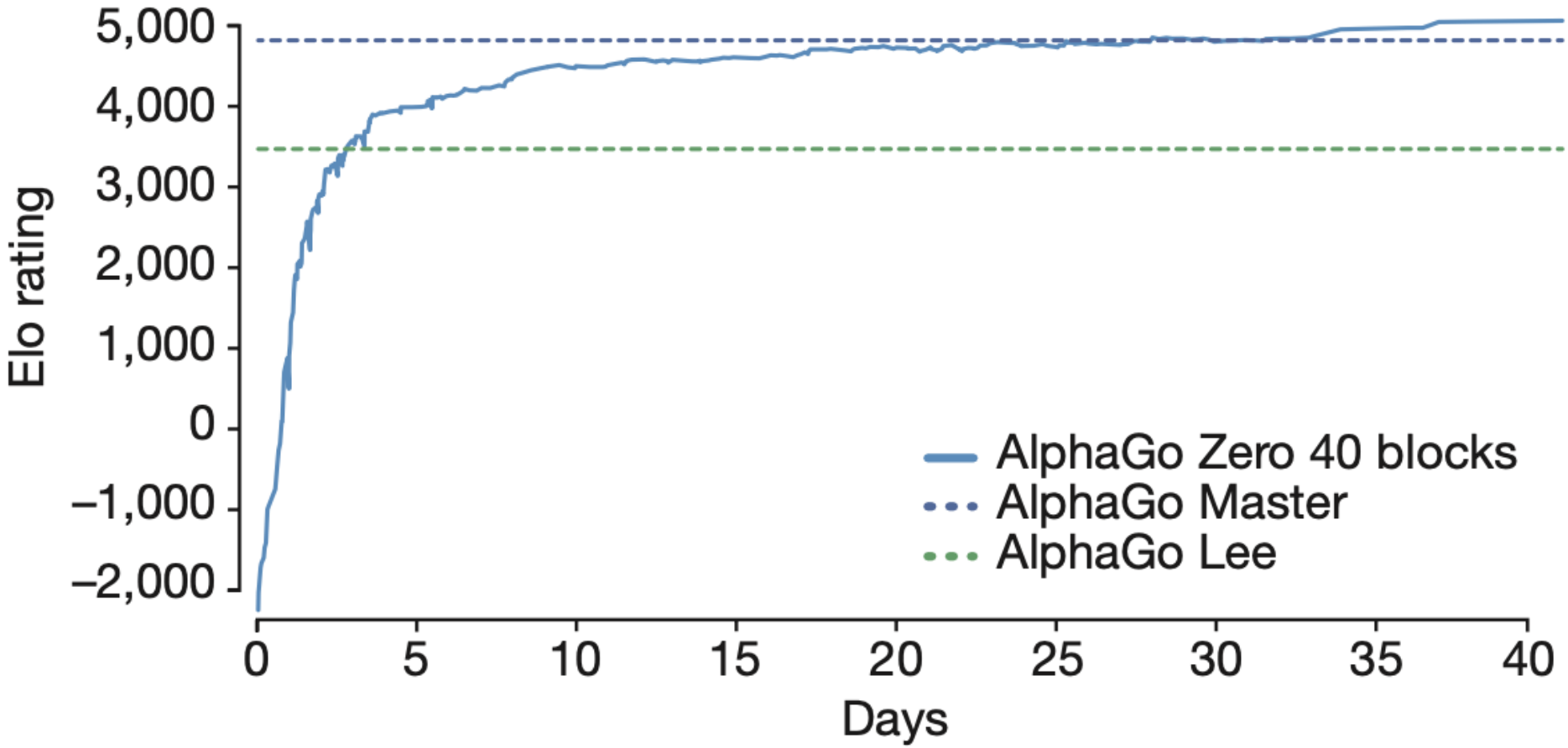
[8] Images from Silver et al. Nature 2017

Impact of MCTS[9]



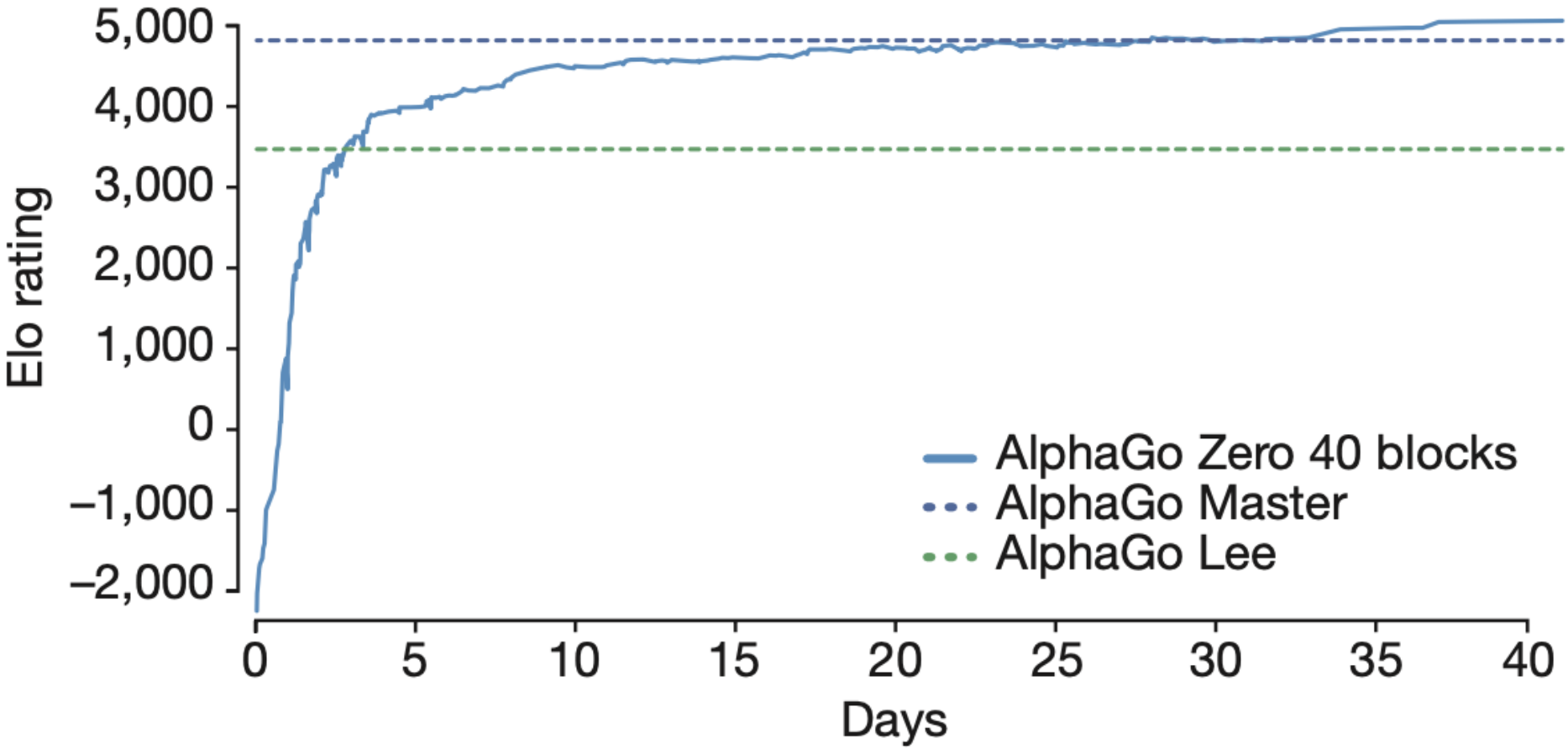
[9]Images from Silver et al. Nature 2017

Overall performance[10]



[10] Images from Silver et al. Nature 2017

Need for Human Data?[11]



[11] Images from Silver et al. Nature 2017

In more depth: Upper Confidence Tree (UCT) Search

- 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?[12]

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

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 (but not a good idea)

In more depth: Upper Confidence Tree (UCT) Search

- 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
- 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)

Class Structure

- Last time: Fast Learning
- **This Time: MCTS**
- Next time: Rewards in RL

Lecture 15 Rewards in RL

Emma Brunskill

CS234 Reinforcement Learning.

Refresh Your Understanding

Select all that are true:

- Direct Preference Optimization assumes human preferences follow a Bradley Terry model
- RLHF can be used with reward models learned from preferences or reward models learned from people labeling rewards
- Asking people to provide preference pair rankings is likely to be an efficient way to learn the reward model for board games
- DPO and RLHF can be used with extremely large policy networks
- Not sure

Refresh Your Understanding

Select all that are true:

- Direct Preference Optimization assumes human preferences follow a Bradley Terry model
- RLHF can be used with reward models learned from preferences or reward models learned from people labeling rewards
- Asking people to provide preference pair rankings is likely to be an efficient way to learn the reward model for board games
- DPO and RLHF can be used with extremely large policy networks
- Not sure

T. T. F. T

Class Structure

- Last time: MCTS
- Today: Rewards in RL
-

Refresh Your Understanding 2

Select all that are true:

- Monte Carlo Tree Search approximates a forward search tree
- MCTS tackles the action branching function through sampling
- AlphaZero uses two networks, one to help prioritize across actions, and one to provide an estimate of the value at leaves
- Doing additional guided Monte Carlo tree search when computing an action significantly improved the test time performance of AlphaZero
- Self play provides a form of curriculum learning
- Not sure

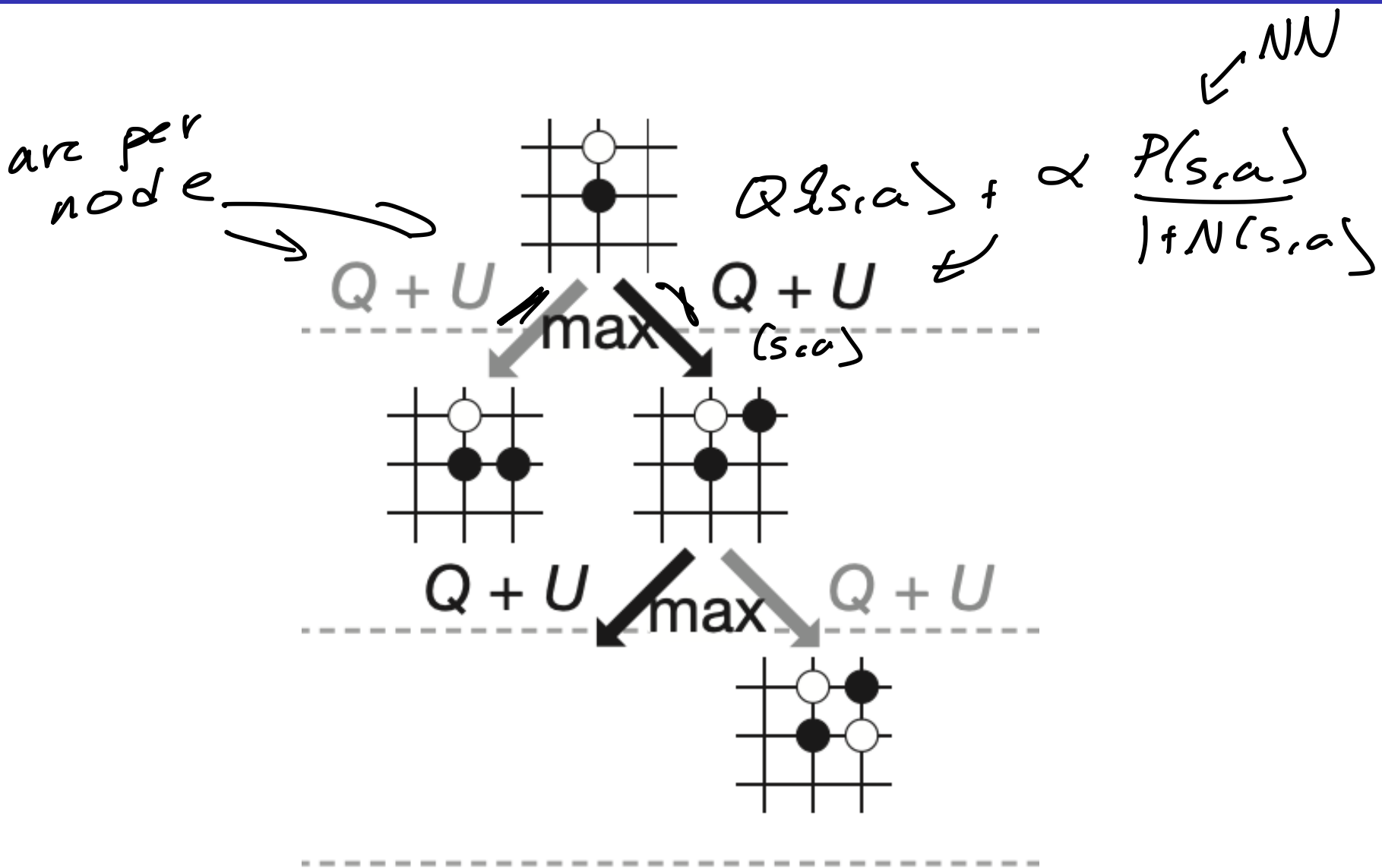
Refresh Your Understanding 2

Select all that are true:

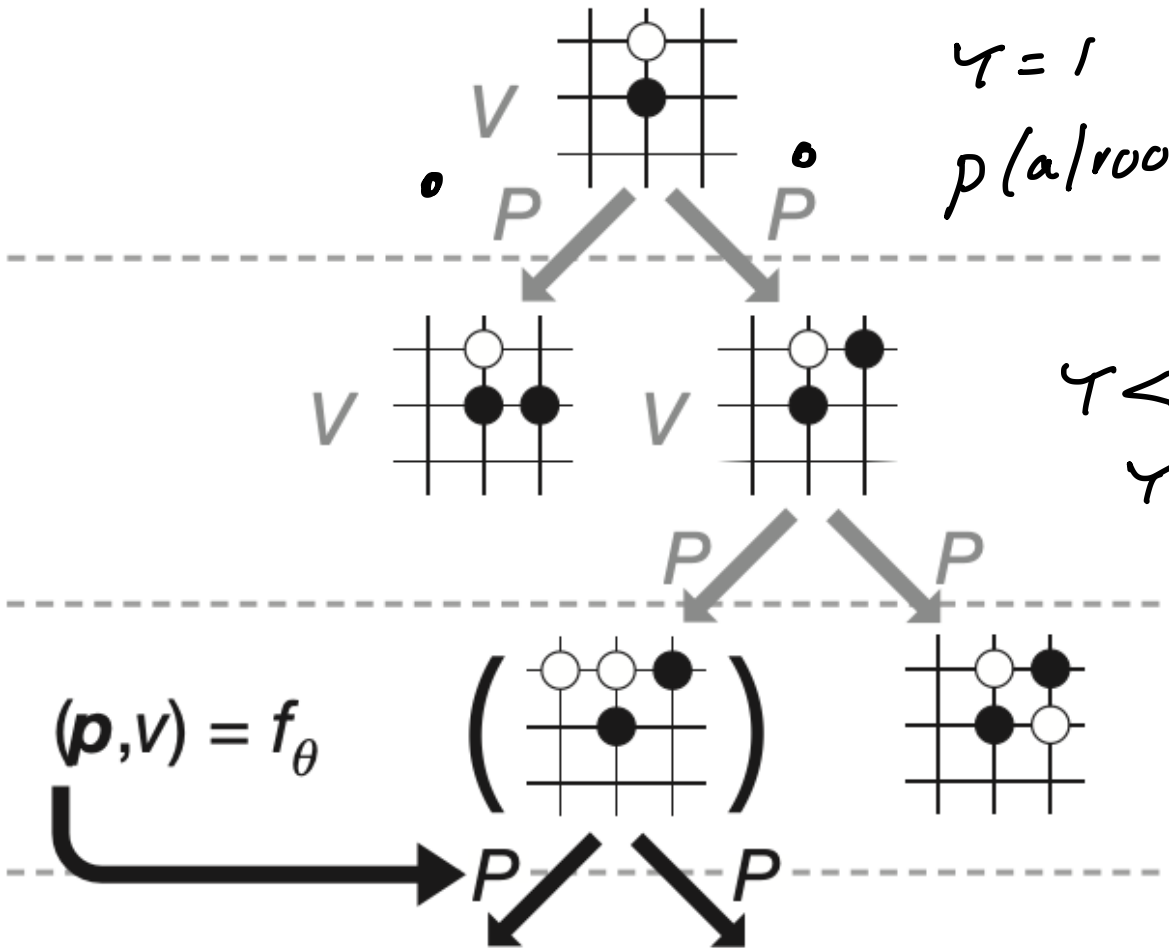
- Monte Carlo Tree Search approximates a forward search tree
- MCTS tackles the action branching function through sampling
- AlphaZero uses two networks, one to help prioritize across actions, and one to provide an estimate of the value at leaves
- Doing additional guided Monte Carlo tree search when computing an action significantly improved the test time performance of AlphaZero
- Self play provides a form of curriculum learning
- Not sure

True. False. False. True. True

Selecting a Move in a Single Game: Repeatedly Expand[1]



Selecting a Move in a Single Game: Note Using Network Predictions for Action Probabilities[2]



$\pi(root) \propto N(s,a)^{1/\tau}$

$\tau = 1$

$p(a|root) = \frac{N(a,root)}{\sum_a N(a,root)}$

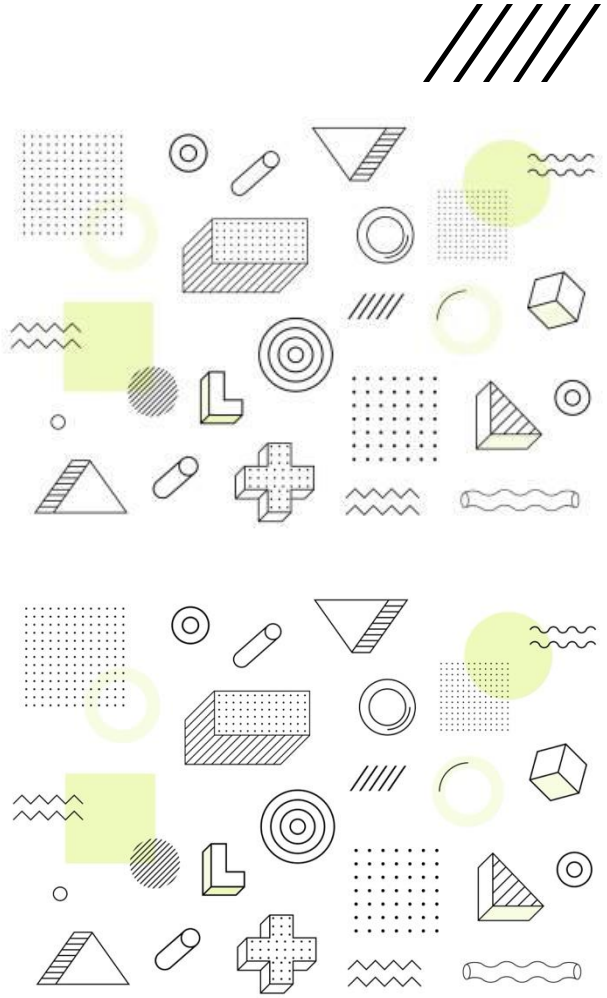
$\tau < 1$

$\tau = 0.5$

$\frac{N(a,root)^2}{\sum_{a'} N(a',root)^2}$

VALUE ALIGNMENT

DAN WEBBER, PHD



Wait, who's this "Dan" guy?

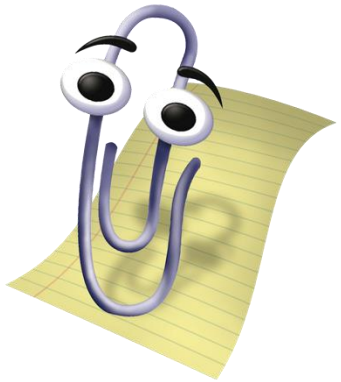
- Postdoc, HAI and EIS at Stanford

  - Embedding ethics into CS courses like this one!

- PhD in Philosophy, University of Pittsburgh
  - Dissertation on moral theory
    - Basically, trying to think systematically about **value**

- BA in Computer Science, Amherst College
  - Plus a few years as a software developer in fintech and e-commerce

Value (mis)alignment: an example



Paperclip AI (Bostrom 2016): "An AI, designed to manage production in a factory, is given the final goal of maximizing the manufacture of paperclips…"

"… and proceeds by converting first the Earth and then increasingly large chunks of the observable universe into paperclips."

Even a less powerful AI might pursue this goal in surprising ways.

Value alignment: the problem

How do we design AI agents that will do what we **really want**?

What we **really** want is often much more nuanced than what we **say** we want. Humans work with many background assumptions that are (1) hard to formalize and (2) easy to take for granted.

It's hard to solve this problem just by giving better instructions!

- Compare the difficulty in manually specifying reward functions.
- Even worse for AI that takes instructions from non-expert users!

Precisifying the problem

There are several ways of interpreting “what we really want”!

First, value alignment might be the problem of designing AI agents that do what we really **intend** for them to do.

If this is right, Paperclip AI is an example of value misalignment because the AI failed to derive the user’s **true intention** (maximize production subject to certain constraints) from their **instruction** (maximize production).

Aligning to user *intentions*

The solution, then, would be to design AI systems that successfully translate from underspecified instructions to fully specified intentions (incl. unspoken constraints, conditions, etc.)

"This is a significant challenge. To really grasp the intention behind instructions, AI may require a complete model of human language and interaction, including an understanding of the culture, institutions, and practices that allow people to understand the implied meaning of terms." (Gabriel 2020)

Aligning to user *intentions*

A philosophical problem: our intentions don't always track what we really want.

Classic cases: incomplete information, imperfect rationality

Suppose I intend for the AI to maximize paperclip production (subject to constraints) because I want to maximize return on my investment in the factory. If the AI knows that I would get a better return by producing something else, has it given me what I really want if it does what I intend?

Aligning to revealed *preferences*

Second interpretation: AI agent is value-aligned if it does what the user **prefers**.

- Paperclip AI is misaligned because I *prefer* it not destroy the world!

Problem: How can the AI know what the user prefers when that differs from the intentions expressed by the user?

Solution: The AI could infer the user's preferences from the user's **behavior** or **feedback**.

Aligning to revealed *preferences*

Technical challenges:

- Requires agent to train on observation of user or from user feedback
- Infinitely many preference/reward functions consistent with finite behavior/feedback
- Hard to infer preferences about unexpected situations (e.g., emergencies)

Philosophical problem:

- Just as my intentions can diverge from my preferences, my preferences can diverge from what is actually good for me.

Aligning to user's *best interests*

Third interpretation: AI agent is value-aligned if it does what is in the user's **best interests**, objectively speaking.

- Paperclip AI is misaligned because it is *objectively bad for me* for the world to be destroyed.

Technical/philosophical problem: Unlike the intended meaning of my instruction or my revealed preferences, my objective best interests can't be determined *empirically*. What's objectively good for me is a *philosophical* question, not a *scientific* one.

Aligning to user's *best interests*

The bad news is that philosophers *disagree* about what's objectively good for a person:

- Is it just the person's own *pleasure* or *happiness*?
- … or the satisfaction of the person's *desires* or *preferences*?
- … or are things like health, safety, knowledge, relationships, etc. objectively good for us even if we *don't* enjoy or prefer them?

The good news is that there's a lot of *agreement*:

- Health, safety, liberty, knowledge, social relationships, purpose, dignity, happiness… almost everyone agrees that these things are at least usually good for the person who has them

Aligning to user's *best interests*

One thing that is widely thought to be good for a person is **autonomy**: the ability to choose for yourself how to live your life, even if you don't always make the best choice.

We want to avoid **paternalism**: choosing what you think is best for someone rather than letting her choose for herself.

Even if we align to users' best interests, then, users' interests in autonomy might give us reason to consider their intentions or preferences, even when these conflict with their other interests.

Part-way recap

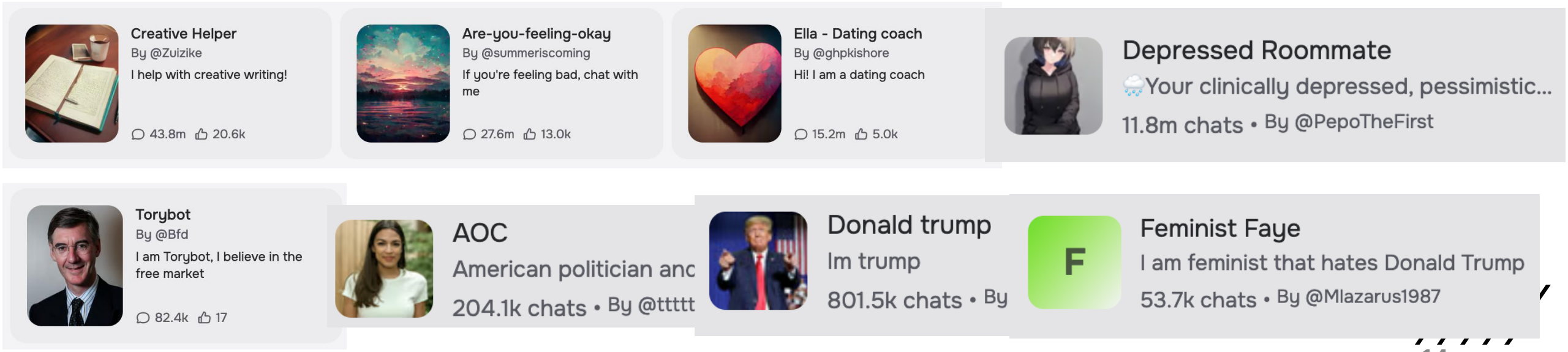
Value alignment is the problem of designing AI agents that will do what we **really want** them to do.

This could mean doing what we really **intend**, or what we really **prefer**, or what would really be in our **best interest**.

These are not always the same thing, and each option poses unique technical and philosophical problems for alignment.

Case study: LLM chatbot personalization

Everyone who talks to ChatGPT is talking to the same chatbot. But many chatbot providers now offer a wide range of different chatbots with different personas. Often these personas are crafted by users:



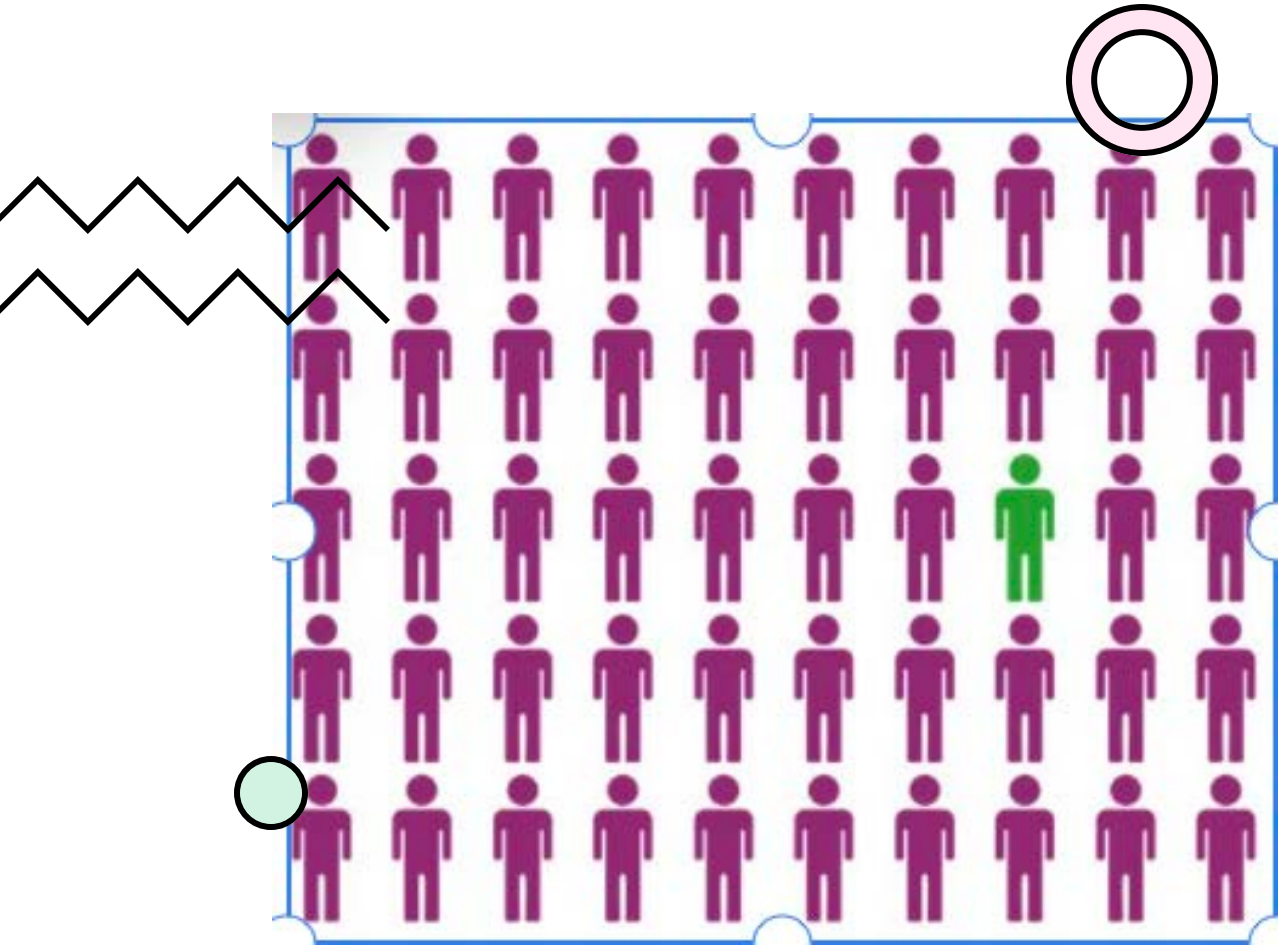
Case study: LLM chatbot personalization

Imagine you are building an LLM chatbot to serve as a source of news for users.

- In what ways might you make the chatbot personalizable if you wanted to align to users' **preferences**?
- In what ways might you make the chatbot personalizable if you wanted to align to users' **best interests**?
- What would be the **pros and cons** of each approach?

Discuss!

WHAT'S BEEN MISSING FROM OUR DISCUSSION SO FAR?



PEOPLE OTHER THAN THE USER!

Aligning to *morality*

Fourth interpretation: AI agent is value-aligned if it does what is **morally right**.

- Paperclip AI is misaligned because it's bad *for everyone* if the world is destroyed!

This interpretation emphasizes the **we** in "what we really want."

What the user intends, prefers, or even what's in her interest might be bad for others!

Aligning to *morality*

But it wasn't just a waste of time to start by focusing on the user!

Even though we want to align to morality, we also want to align to what the user wants when what the user wants is morally acceptable.

So it still matters how we think about what the **user** really wants, even if we need to think about it in the larger moral context.

Aligning to *morality*

A philosophical problem: Which things really *are* morally right?

There's a lot of disagreement on this one too!

- Is it right to lie to spare someone's feelings?
- Is it right to pirate copyrighted material?
- Is it right to buy luxuries when you could donate to charity instead?
- Is it right to kill one person to save five?

… a thousand?

… a million?

Aligning to the *best moral theory*

A **moral theory** is a systematic account of morality that aims to answer questions like these.

- For example, *consequentialism* says that an act is right iff it produces the greatest net good of any act available.
- Is it right to lie to spare someone's feelings? It could be, if you can get away with it!

Idea: align AI agents to the **correct** or **best** moral theory.

- If that theory were consequentialism, value alignment would be about training AI agents to do what they can to maximize net good.

Aligning to the *best moral theory*

A now familiar problem: philosophers *disagree* about what the best moral theory is, or even if there is one!

Prioritarianism: Produce the greatest *weighted* sum of good, with the interests of those who are worse off given more weight.

Maximin: Make things as good as possible for the worst off.

Aligning to the *best moral theory*

Satisficing: Produce a *sufficiently great* (weighted) sum of good.

Deontology: Even acts with good consequences can be wrong if they violate certain moral *rules* or *rights*.

- Common deontological rules: don't murder anyone, don't steal, don't lie, keep your promises, etc.
- Rules/rights themselves might be justified by their consequences!

Aligning to the *best moral theory*

Another problem: even if we knew the best moral theory, it might be bad to design AI agents to act on moral values that their users **don't share**.

This might be bad for moral reasons (avoiding paternalism) or more practical ones (users might not trust AI agent, might try to oppose it, etc.)

Aligning to *common-sense morality*

There's lots of moral *disagreement*, but also a lot of *agreement*.

Idea #2: align AI agents to what we might call **common-sense morality**: the common-sense moral ideas that most people agree on.

Instead of trying to make AI morally *perfect*, we just try to make it make moral decisions like a *regular person* would.

- This probably ends up being pretty deontological and satisficing!

Common-sense morality and *predictability*

One advantage of aligning to common-sense morality rather than a particular moral theory is that moral theories often have *surprising implications*.

- Can you think of a surprising implication of the consequentialist requirement to maximize net good?
- How about a deontological rule against lying?

An AI agent aligned to a particular moral theory might act on surprising implications we haven't even noticed yet!

Common-sense morality and *predictability*

By contrast, an AI agent aligned to common-sense morality would behave more predictably, making moral decisions like a regular human.

But it might be unpredictable in *edge cases*, where common sense arguably runs out.

- Would an AI aligned to common-sense morality kill one person to save a million?

Is it bad if the AI is unsure about cases we're also unsure about?

Recap

Value alignment is the problem of designing AI agents that will do what we **really want** them to do.

This could mean doing what we really **intend**, or what we really **prefer**, or what would really be in our **best interest**.

- We saw what the difference might look like in chatbot development.

It definitely means doing what's **morally right**.

- But this could mean doing what's morally right **in theory**, or just abiding by the **common-sense morality** that most people share.

Reinforcement Learning

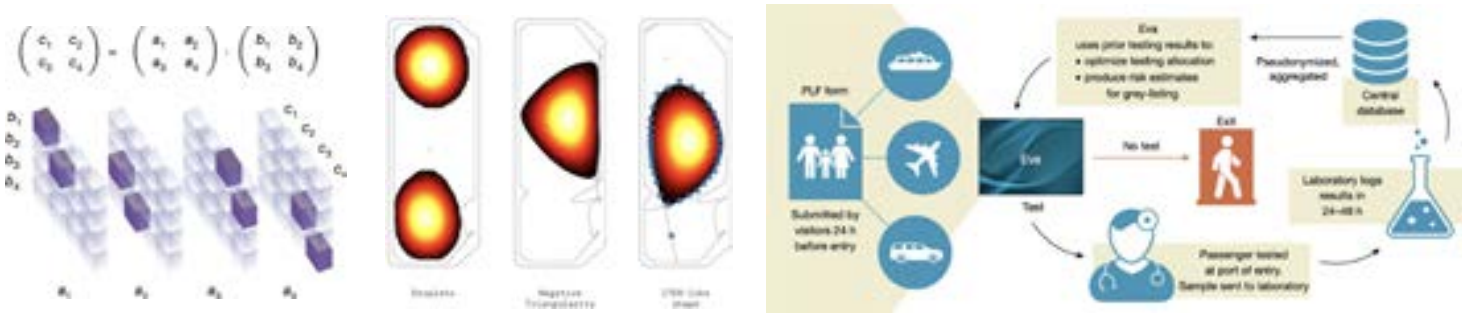
Learning through experience/data to make good decisions under uncertainty

High Level Learning Goals[1]

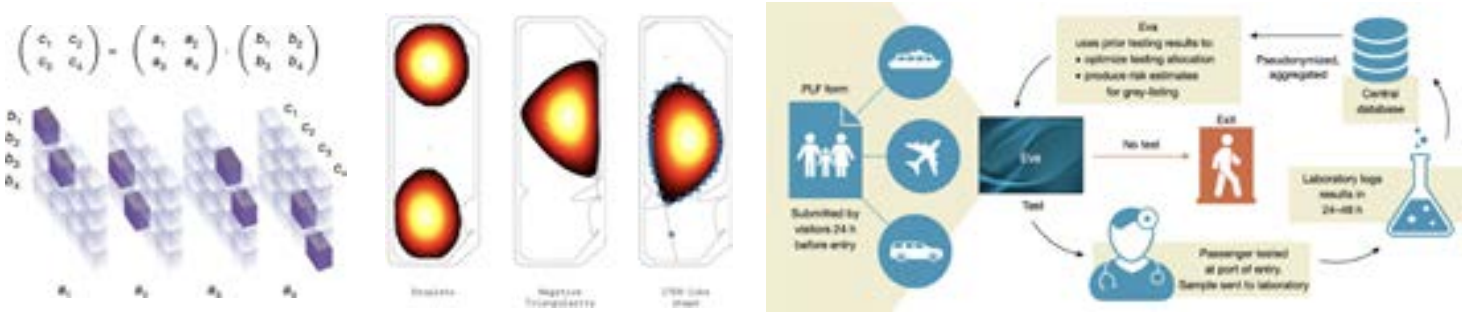
- Define the key features of RL
- Given an application problem know how (and whether) to use RL for it
- Implement (in code) common RL algorithms
- Describe (list and define) multiple criteria for analyzing RL algorithms and evaluate algorithms on these metrics: e.g. regret, sample complexity, computational complexity, empirical performance, convergence, etc.
- Describe the exploration vs exploitation challenge and compare and contrast at least two approaches for addressing this challenge (in terms of performance, scalability, complexity of implementation, and theoretical guarantees)

[1] For more detailed descriptions, see website

Revisiting Motivating Domains from First Lecture

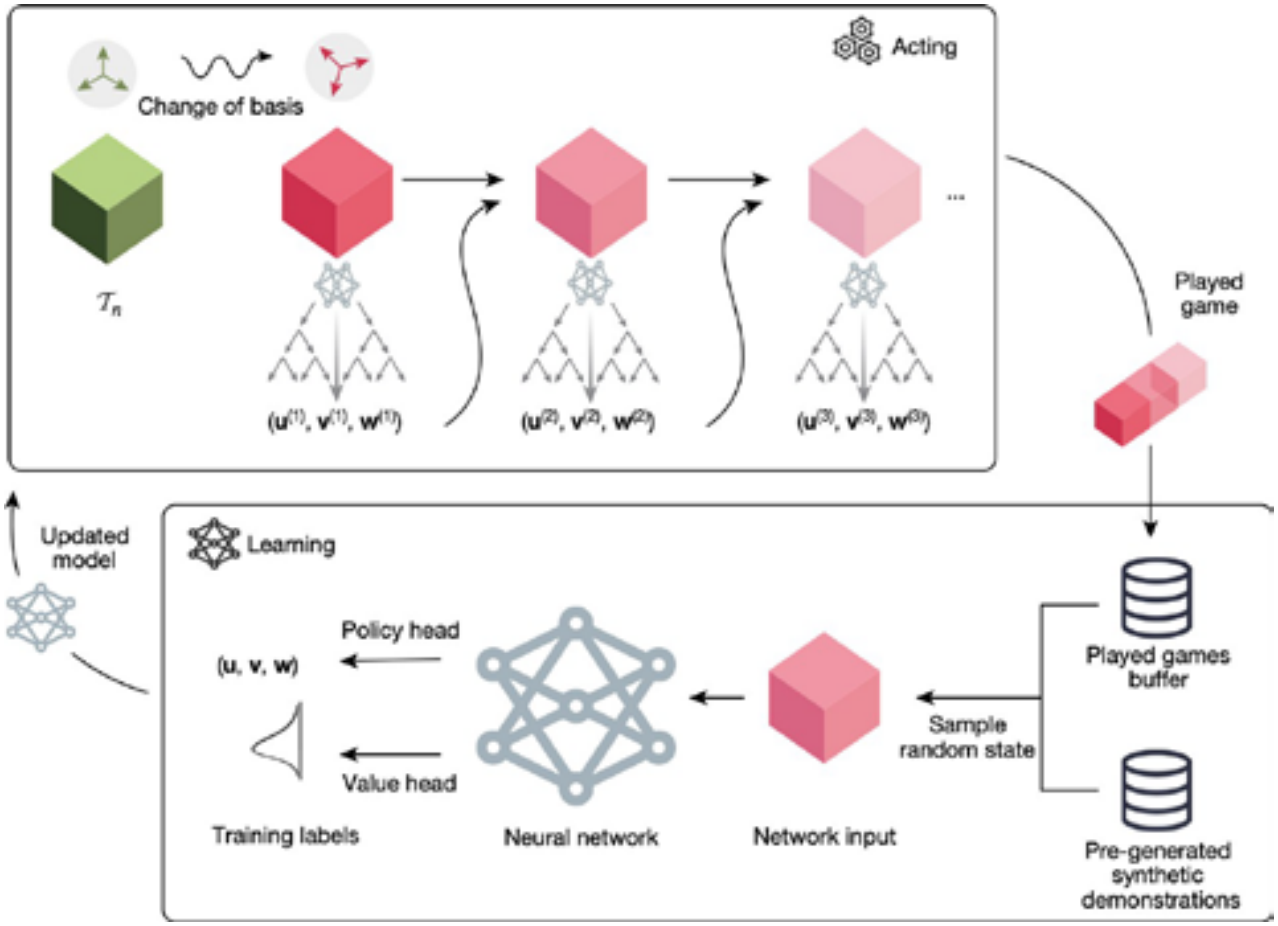


CYU: Answer For One of These Domains

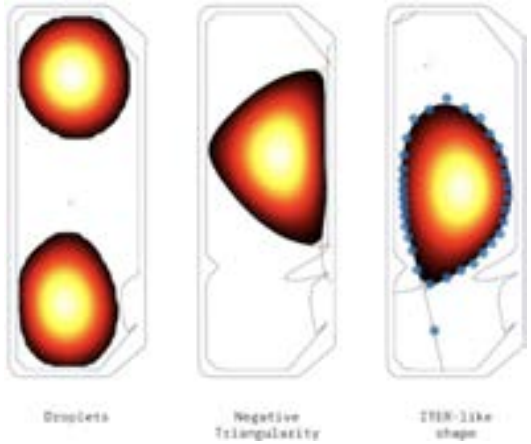


- Which domain are you choosing?
- Is this problem a bandit? A multi-step RL problem?
- Is the problem online / offline or some combination?
- What might the state / action / rewards be?
- What algorithms might be useful here?

AlphaTensor. Fawzi et al. 2022

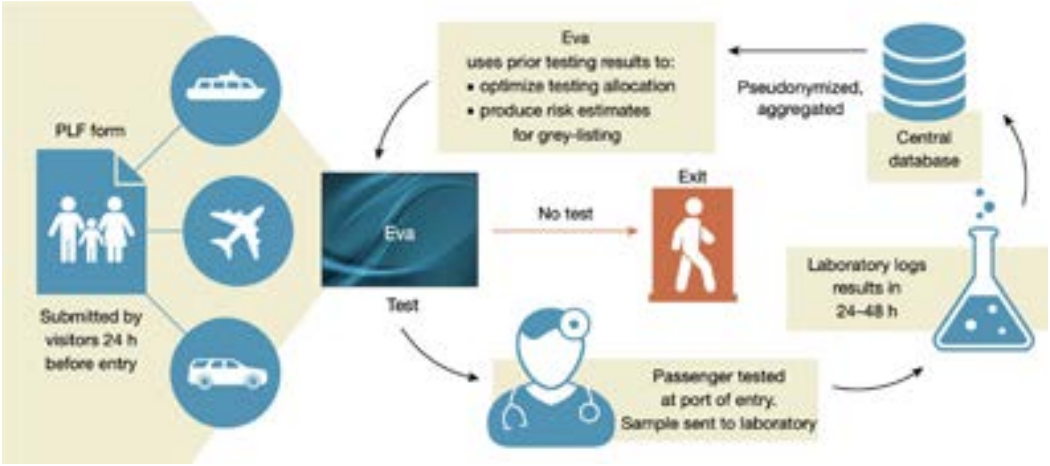


Revisiting: Learning Plasma Control for Fusion Science[2]



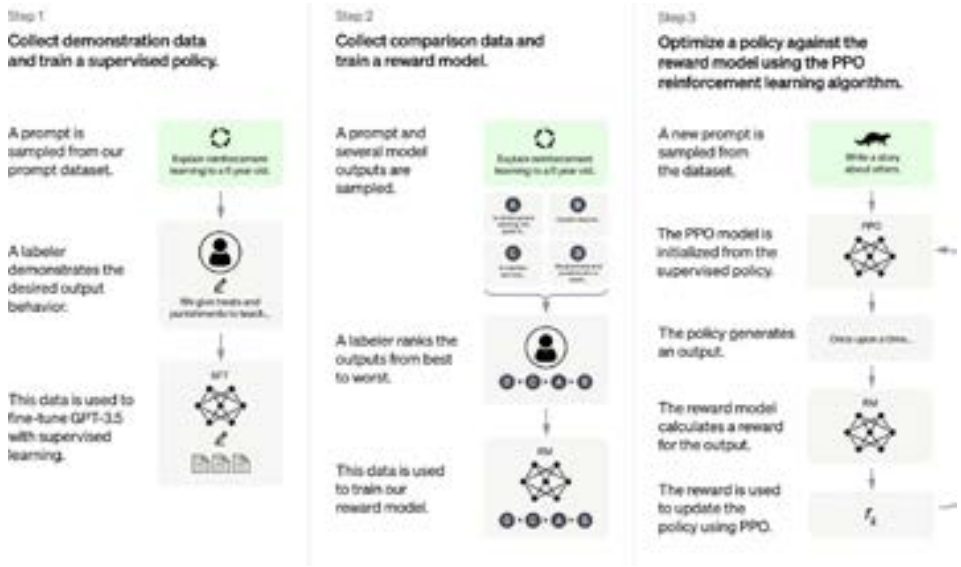
[2]Image credits: DeepMind & SPC/EPFL. Degrave et al. Nature 2022 https://www.nature.com/articles/s41586-021-04301-9

Revisiting: Efficient and targeted COVID-19 border testing via RL [3]



[3] Bastani et al. Nature 2021

Revisiting: ChatGPT (https://openai.com/blog/chatgpt/)



Reinforcement Learning

- Learn a policy $\pi(a|s)$ from data to optimize future expected reward
- Optimization, delayed consequences, exploration, generalization
- Actions impact data distribution: rewards observed and states reached

Reinforcement Learning: Standard Settings

- State dependence
  - Bandits: next state independent of prior state and action
  - General decision process: next state depends on prior states and actions
- Online/Offline
  - Offline / batch: Learn from historical data only
  - Online: Agent / algorithm can actively gather its own data

Reinforcement Learning: Core Ideas[4]

- Function approximation + Offpolicy learning is a key challenge
  - New policy introduces new distribution over (s,a,r)
  - Important because want data efficient RL in complex domains
  - PPO: Control with clipping
  - DAGGER: mitigate by obtaining more expert labels
  - Pessimistic Q Learning / CQL / MOPO: introduce pessimism into offline RL

[4]These align closely with many of the core points of Chelsea Finn's Deep RL course summary slides

Reinforcement Learning: Core Ideas[5]

- Function approximation + Offpolicy learning is a key challenge
  - New policy introduces new distribution over (s,a,r)
  - Important because want data efficient RL in complex domains
  - PPO: Control with clipping
  - DAGGER: mitigate by obtaining more expert labels
  - Pessimistic Q Learning / CQL / MOPO: introduce pessimism into offline RL
- Models, values and policies
  - Models: easier to represent uncertainty (why?), useful for MCTS
  - Q function: summarizes performance of policy & and implies policy
  - Policies: the main target of most RL applications
- Computational vs Data Efficiency
  - Data efficient techniques often very computationally intensive
  - In some domains, data = computation (e.g. simulated settings)

[5]These align closely with many of the core points of Chelsea Finn's Deep RL course summary slides

Open Challenges

- Practical, robust RL
  - Robust/stable: Need for automatic hyperparameter tuning, model selection, and generally robust methods for off-the-shelf RL
  - Efficiency: Need for data and computationally efficient methods
  - Hybrid offline-online:
- Framing the problem
  - Alternate formulations to Markov decision processes?
  - Multi-task vs single task?
  - Alternate forms of feedback?
  - Stochastic vs adversarial vs cooperative decision process?
  - Continuous learning + planning vs system identification then planning?
- Advancing data-driven decision making in domains that could benefit