BOSTON UNIVERSITY

Deep Learning for Data Science DS 542

https://dl4ds.github.io/fa2025/

Deep Reinforcement Learning

Original slides by Tom Gardos,
other content from Understanding Deep Learning unless otherwise cited

Plan for Today

- What is reinforcement learning?
- Policies and value functions
- Policy gradient methods
- AlphaGo
- Gran Turismo Sophy
- Language model tuning

Reinforcement Learning (RL)

An intelligent agent uses reinforcement learning to maximize a sequence of rewards arising from actions over time.

Loosely speaking, maximize expected discounted rewards

$$E\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}\right]$$

Where r_{t+1} is the reward immediately after time t, γ is a discount factor, and the reward is usually driven by state s_t and picking actions a_t...

Example: Data Center Cooling

"Despite the impressive advances in reinforcement learning (RL) algorithms, their deployment to real-world physical systems is often complicated by unexpected events and the potential for expensive failures. In this paper we describe an application of RL "in the wild" to the task of regulating temperatures and airflow inside a large-scale data center (DC). Adopting a data-driven model-based approach, we demonstrate that an RL agent is able to effectively and safely regulate conditions inside a server floor in just a few hours, while improving operational efficiency relative to existing controllers."

"Data Center Cooling using Model-predictive Control" (2018)

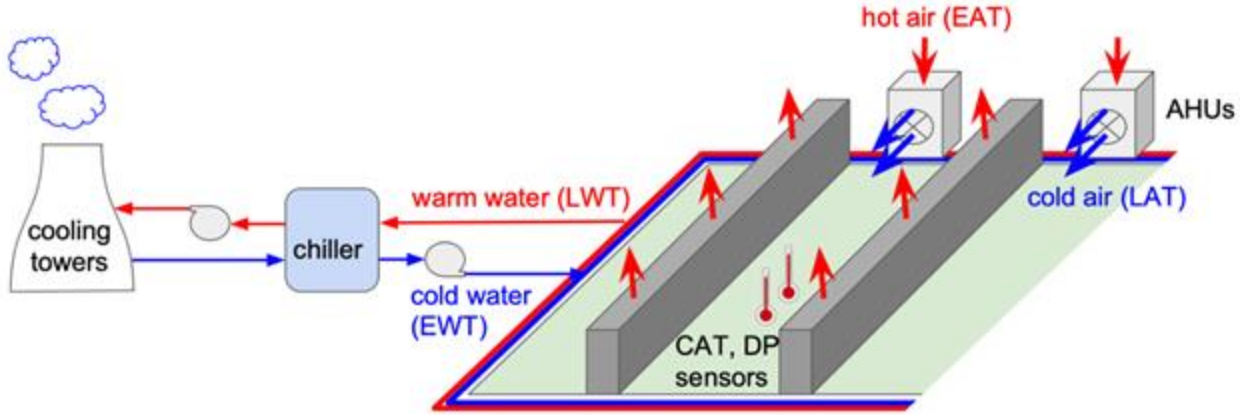
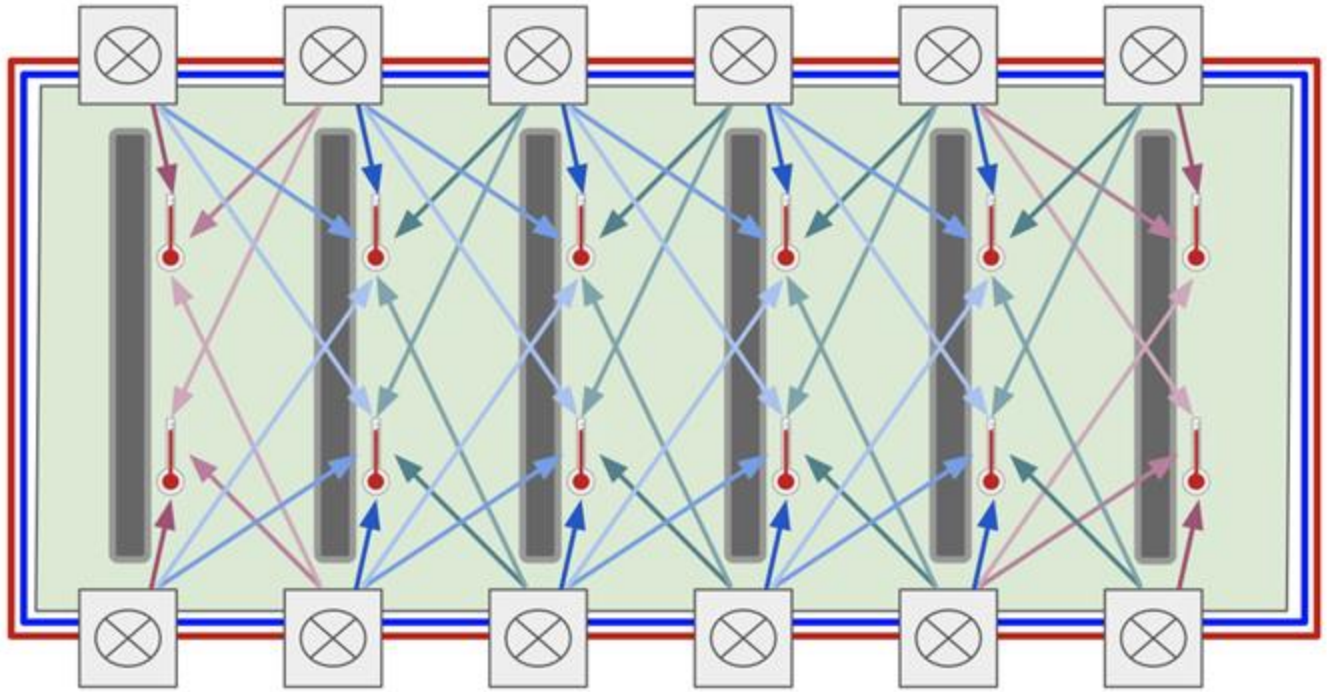


Figure 1: Data center cooling loop. AHUs on the server floor regulate the air temperature through air-water heat exchange. Warmed water is cooled in the chiller and evaporative cooling towers.



Example: Gran Turismo Sophie

- Racing fast is the easy part.
  - Well-behaved driving is much more difficult.
  - Do not ride the walls.
  - Stop crashing to knock other cars off the course.

https://www.gran-turismo.com/jp/gran-turismo-sophy/

Re: name origin

Curiously, this area is named after the early solving tactics, not the problem definition…

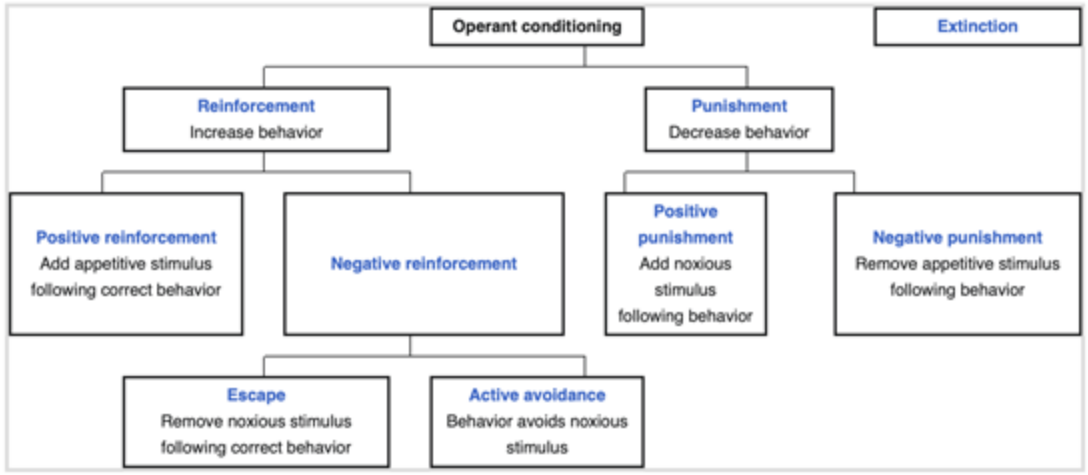


Image source: https://en.wikipedia.org/wiki/Operant_conditioning

Why is Reinforcement Learning Hard?

- Rewards due to an action may be delayed arbitrarily long.
- Distributions of rewards may change due to agents changing their behavior.
- Probabilistic scenarios make systematic coverage more difficult.
- Opponents (e.g. in 2 player games) actively try to minimize agent rewards.

Which of these moves were good or bad?

PGN

1. e4 c5 2. Nc3 Nc6 3. Bc4 e6 4. Nf3 Nf6 5. O-O d5 6. exd5 exd5 7. Bb5 Bd7 8. Re1+ Be7 9. Bxc6 Bxc6 10. Ne5 Qc7 11. d4 O-O 12. Nxc6 bxc6 13. Bg5 h6 14. Bxf6 Bxf6 15. dxc5 Rad8 16. Qh5 Qa5 17. a3 Bxc3 18. bxc3 Qxc3 19. Qe5 Qxc5 20. c3 Qd6 21. Qxd6 Rxd6 22. Re7 a6 23. Rd1 Rb8 24. h4 Rb3 25. c4 Rxa3 26. c5 Re6 27. Rxe6 fxe6 28. Rb1 Kf7 29. Rb6 Ke7 30. Rxc6 Rc3 31. Rxa6 Rxc5 32. Ra7+ Kf6 33. g4 d4 34. Kf1 e5 35. Ke2 g5 36. h5 e4 37. f3 e3 38. Ra6+ Ke5 39. Rxh6 Rc2+ 40. Kd3 e2 41. Rh8 e1=N#

Only non-zero reward.

Source: https://www.reddit.com/r/chess/comments/25y7o5/cool_underpromotion_checkmate_in_a_game_i_just/

DS 543 Introduction to Reinforcement Learning

We actually have a whole course about reinforcement learning.

	Topics
Chapter 1	**Fundamentals**: Markov Decision Processes
Chapter 2	**Planning in MDPs**: Policy and Value Itertions
Chapter 3	**Model-based RL**: MPC, Dreamer, MuZero
Chapter 4	**Value-based RL**: FQI, Q-learning
Chapter 4	**Value-based RL**: Bellman completeness, DQN
Chapter 5	**Policy-based RL**: Policy Gradient Theorem, Reinforce
Chapter 5	**Policy-based RL**: Actor-Critic, Importance Sampling, DPG
Chapter 5	**Policy-based RL**: NPG, TRPO, PPO
Chapter 6	**Imitation Learning**: Behavior Cloning
Chapter 6	**Imitation Learning**: Dagger
Chapter 7	**Exploration**: Exploration in MAB
Chapter 7	**Exploration**: Exploration in MAB
Chapter 8	**Exploration**: Exploration in MDPs
Chapter 8	**Exploration**: Exploration in Deep RL
Chapter 9	**Offline RL**: FQI and naive methods
Chapter 9	**Offline RL**: Learning without full data coverage
Chapter 9	**Offline RL**: LCB and Empirical Methods
Chapter 10	**Multi-agent RL**: Game Theory Basics
Chapter 10	**Multi-agent RL**: Markov Games and Planning in MG
Chapter 10	**Multi-agent RL**: Online Learning in MGs
Chapter 10.5	**Mechanism Design**: Going beyond being a player in the game

Any Questions?

???

Moving on

- What is reinforcement learning?
- Policies and value functions
- Policy gradient methods
- AlphaGo
- Gran Turismo Sophy
- Language model tuning

Reinforcement Learning Terminology

- States
- Actions
- Reward
- Value
- Episode
- Policy
- Optimal state value
- Optimal state-action value
- Optimal policy

Rough equivalences between policies and values

- Given state values, use Monte Carlo simulations of one step to estimate value of each action. Use highest action-value for policy.
  - Requires a simulator.

- Given state action values, use max for state value and policy.

- Given policy, use Monte Carlo simulations to estimate state or state-action value over many episodes.

Which version is easier to learn?

Without knowing or assuming dynamics of environment, have "model-free" methods to learn each of these.

- TD-learning for state values
- Q-learning for state-action values
- Policy gradient methods for policies

Any Questions?

???

Moving on

- What is reinforcement learning?
- Policies and value functions
- Policy gradient methods
- AlphaGo
- Gran Turismo Sophy
- Language model tuning

Parameterized Policies

- Given a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$,

$$\pi(a|s, \boldsymbol{\theta}) = \Pr[A_t = a | S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}]$$

- What is the functional form?
  - Deep Blue used a linear function of various hand-crafted features.
  - TD-Gammon used a neural network with a mix of features.
  - Alpha Go used deep neural networks.
  - This is a design choice specific to a problem.

Tip: Avoid 0-1 Probabilities in Policies.

- Most training methods will try actions based (in part) on current policy probabilities.

- Actions with probability zero will never be explored.
- Actions with probability one will shut out the others.

Degen Parameterized Policies

- If the state-action space is not too big, define a tabular function

$$h(s, a, \boldsymbol{\theta}) \in \mathbb{R}$$

Then set

$$\pi(a|s, \boldsymbol{\theta}) = \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}}$$

What formula is this? Why is it useful?

Softmax Benefits

- Base functionality
  - Converts logits (scores) to usable probability distribution.
  - Differentiable.

- If all the logits are finite, then $0 < \underset{i}{\text{softmax}}(\mathbf{z}) < 1$.
  - So never deterministic with 0-1 probabilities.
  - But can get arbitrarily close.

- Changes smoothly with logits and parameters (assuming smooth/continuous functional form)

Policy Gradient Methods

Given some performance measure $J(\boldsymbol{\theta})$ that we want to maximize, update parameters using gradient ascent.

$$\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J(\theta_t)}$$

where $\widehat{\nabla J(\theta_t)}$ is an estimate of the gradient.

- $\widehat{\nabla J(\theta_t)}$ is ideally unbiased but may be a more tractable approximation.

What Performance Measure?

- What should $J(\boldsymbol{\theta})$ measure?
  - Probability of win?
  - Estimated score?
  - Dollars?

Episodic Performance Measure

If we have a designated initial state s_0, define

$$J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(s_0)$$

Where $v_{\pi_{\boldsymbol{\theta}}}$ is the value function using the policy induced by $\boldsymbol{\theta}$.

- Intuitive definition, but tricky to analyze since policy changes change actions which change state distributions which change action distributions which change reward distributions…

Policy Gradient Methods for Reinforcement Learning with Function Approximation (Sutton et al, 1999)

For any Markov decision process,

$$\nabla J(\boldsymbol{\theta}) = \sum_{s} d_{\pi}(s) \sum_{a} q_{\pi}(s,a) \nabla \pi(s,a|\boldsymbol{\theta})$$

where

- $d_{\pi}(s)$ is the average discounted number times in state s under policy π.
- $q_{\pi}(s,a)$ is the average discounted reward taking action a in state s and then following policy π.

Proportional version

For any Markov decision process,

$$\nabla J(\boldsymbol{\theta}) \propto \sum_{s} \mu(s) \sum_{a} q_{\pi}(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

where

- $\mu(s)$ is the conditional distribution of states given $\boldsymbol{\theta}$.
- The difference from the previous equation is dropping a constant scaling factor based on the average episode length.

Sampling version

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta}) \\
&= \mathrm{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta}) \right] \\
&= \mathrm{E}_\pi \left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \right] \\
&= \mathrm{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right] \\
&= \mathrm{E}_\pi \left[G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right]
\end{aligned}
$$

Simple statistical gradient-following algorithms for connectionist reinforcement learning (Williams, 1992)

First policy gradient method REINFORCE -

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \boldsymbol{\theta})}$$

- Change is proportional to G_t (rewards after time t).
- Change is proportional to gradient of action probability.
- Change is inversely proportional to probability.

Log Formula Tweak

Standard implementation and pseudocode use different formula.

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$$

Based on identity

$$\nabla \ln x = \frac{\nabla x}{x}$$

Full pseudocode

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

Source: Reinforcement Learning by Sutton and Barto

Many policy gradient methods since then…

- Mostly trying to make training process run smoother and faster.
- Actor-Critic methods
  - Estimate values simultaneously.
- PPO = Proximal Policy Optimization
  - Limit changes per step for stability.

Any Questions?

???

Moving on

- What is reinforcement learning?
- Policies and value functions
- Policy gradient methods
- AlphaGo
- Gran Turismo Sophy
- Language model tuning

Mastering the game of Go with deep neural networks and tree search (Silver et al, 2016)

- Trained a Go playing agent that beat many of the top-ranked players. (No agreed world-champion like chess).

- Go was previously considered way out of reach due to large state space and high branching factor.

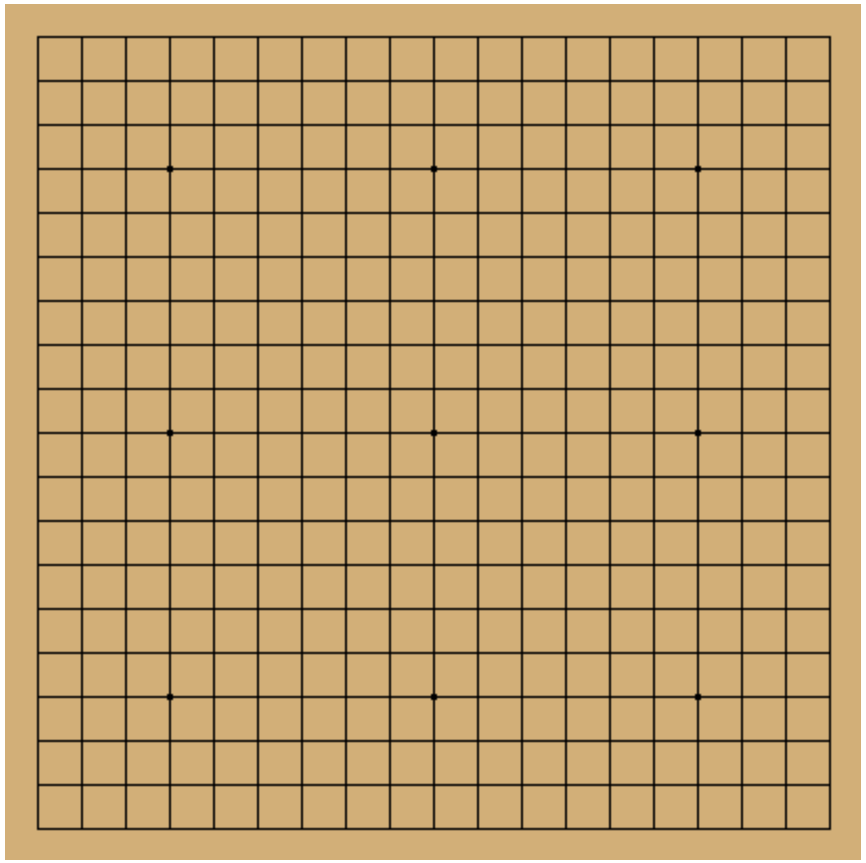


Image source: https://en.wikipedia.org/wiki/Go_(game)

What are the (main) options for learning a policy?

- Learn a state value function.
  - Requires knowledge of available actions and transitions.
- Learn a state-action value function.
  - Requires knowledge of available actions.
- Learn a policy.
  - Usually, a probability distribution over available actions.
- Can tradeoff certain knowledge of available actions for chance of making an invalid action.

What are the (main) options for learning a policy?

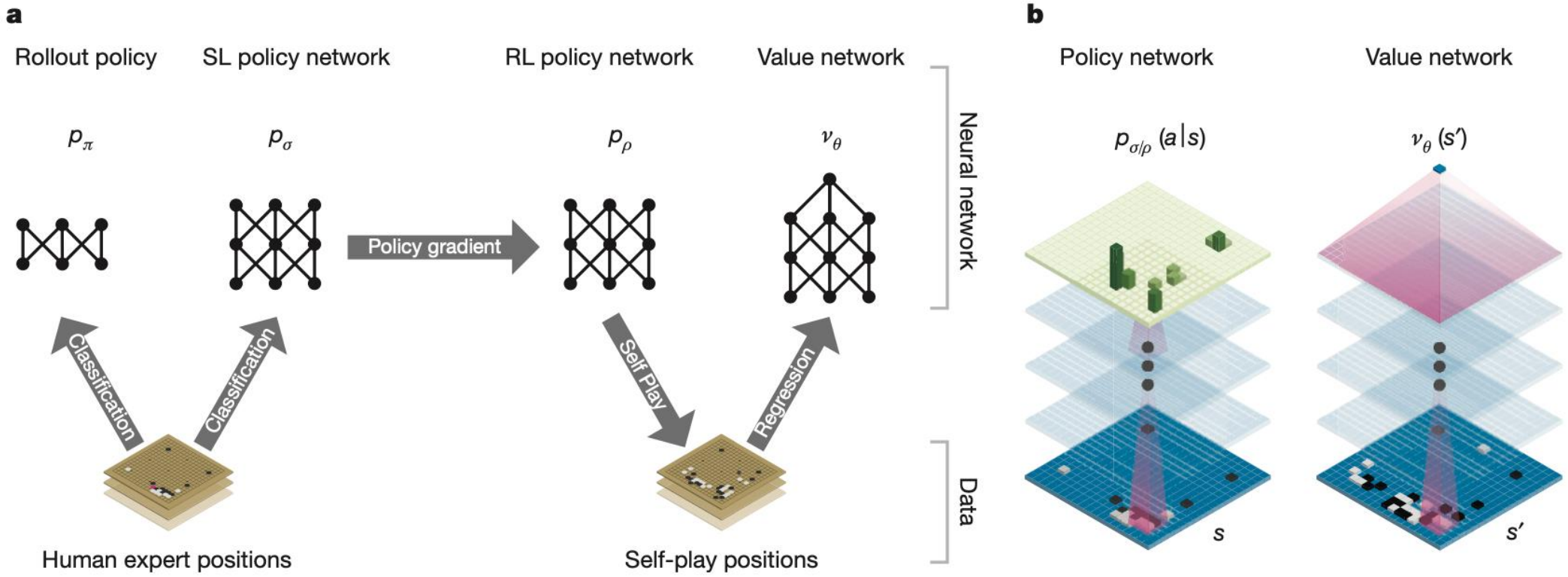
- Learn a state value function
  - Requires knowl
- Learn a state-ac
  - Requires knowl
- Learn a policy.
  - Usually, a probability distribution over available actions.

AlphaGo:

"Why not all of them?"

- Can tradeoff certain knowledge of available actions for chance of making an invalid action.

Things Alpha Go Learned with Deep Neural Networks



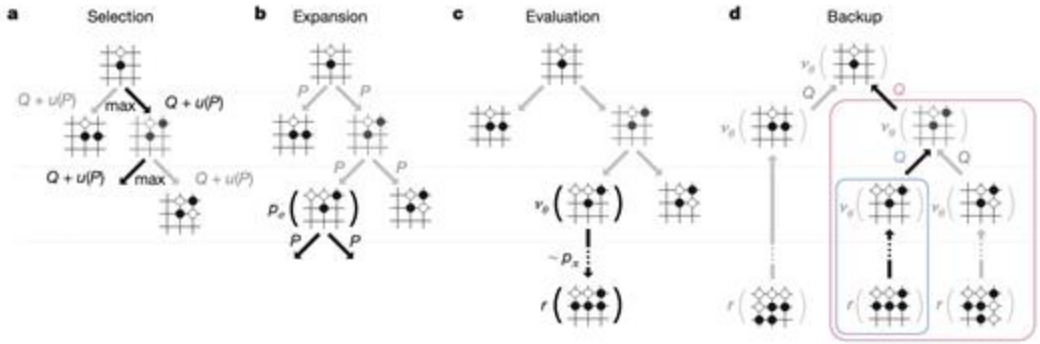
Things Alpha Go Learned with Deep Neural Networks

- Supervised learning policy trained with expert moves.
- Fast “rollout” policy trained from supervised learning policy.
- Reinforcement learning policy trained by self-play.
  - Used rollout policy to speed up process.
- Value function trained from using reinforcement policy.

Monte Carlo Tree Search (MCTS)

TLDR: play a lot of games (in your head) remembering which moves worked out.

Alpha Go used it as a “policy improvement” operator since it could catch some mistakes.



Mastering the game of Go with deep neural networks and tree search (Silver et al, 2016)

Any Questions?

???

Moving on

- What is reinforcement learning?
- Policies and value functions
- Policy gradient methods
- AlphaGo
- Gran Turismo Sophy
- Language model tuning

Gran Turismo talk

https://webdocs.cs.ualberta.ca/~jonathan/ICGA/CG2024/3_CG2024_Keynote_1

Timestamps

- 6:08 (what)
- 13:00 (how)
- 17:15 (results)
- 18:33 (reward design)

Any Questions?

???

Moving on

- What is reinforcement learning?
- Policies vs value functions
- Policy gradient methods
- AlphaGo
- Gran Turismo Sophy
- Language model tuning

Language Model Tuning

Previously covered

- Pre-training (on the Internet)
- Supervised fine-tuning (on human-written examples)
- Reinforcement learning (???)

Aligning language models to follow instructions (OpenAI, 2022)

hundreds to low thousands

tens of thousands

hundred thousands or millions

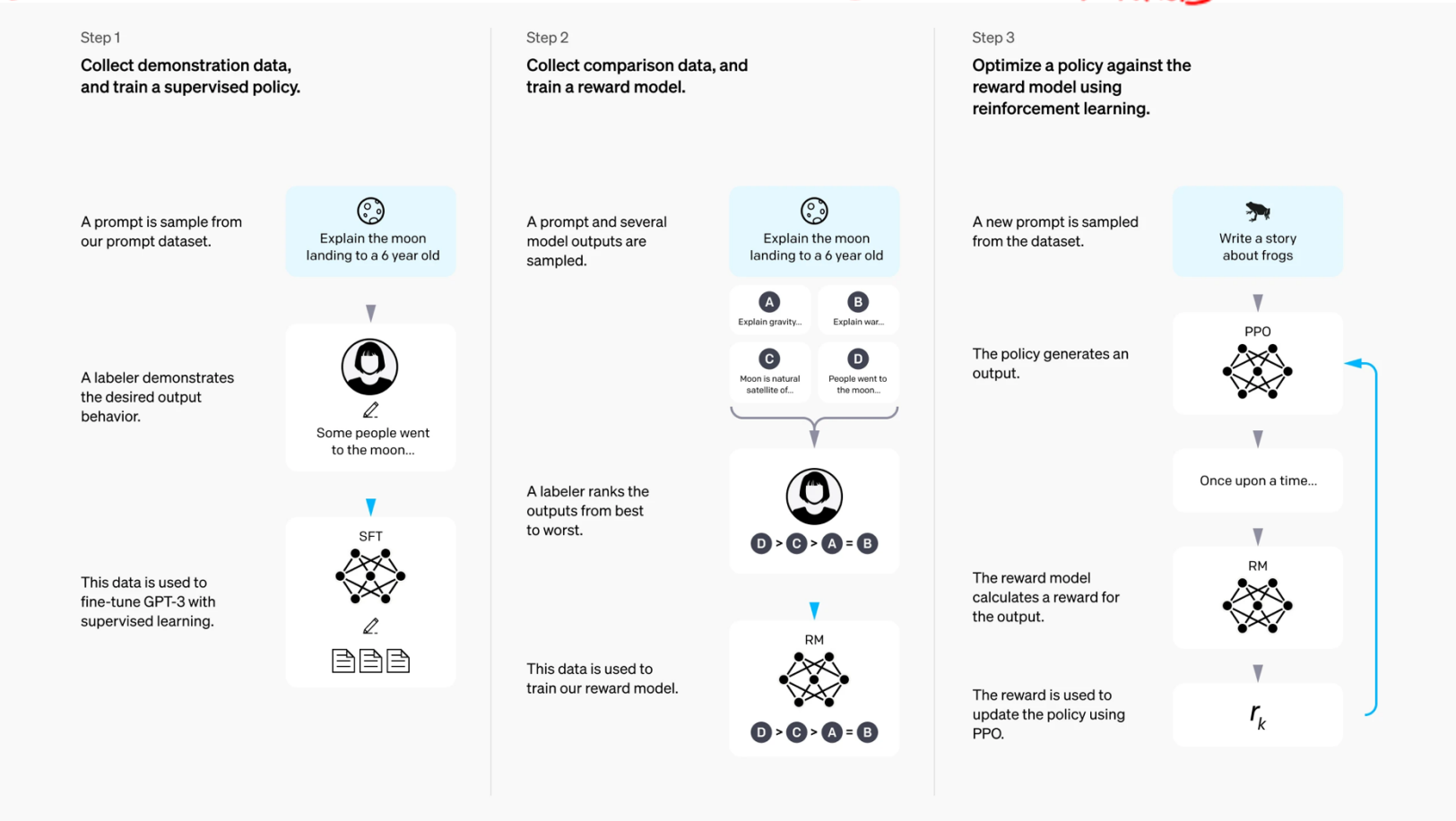


Image source: https://openai.com/index/instruction-following/

Paper: Training language models to follow instructions with human feedback (Ouyang et al, 2022)

Supervised fine-tuning (SFT)

- Generate a lot of prompts.
- Human-generate an answer for each one.
- Fine-tune language model to produce the human answer given each prompt. (Loss only computed from answer, not prompt.)
- This is nicer than the pre-trained model, but not yet amazing.

Reward modeling

- For a separate database of prompts, generate K=4 to 9 answers from the SFT model.
- Ask humans to sort these answers.
- Train a reward model to rank these answers.
  - Usually starting from a smaller language model r_θ with last layer replaced like for classification.
  - Loss function based on cross entropy using sigmoid and reward difference…

$r_\theta(\,)$ = reward function

Want

$r_\theta(\text{better}) > r_\theta(\text{worse})$

for all better/worse pairs [0,1] output. interpret as P(w>l)

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

reward of winning answer

reward of losing answer

sigmoid

cross entropy loss

$-\log[\cdots]$

y_w winning text

y_l losing text

x prompt

Training language models to follow instructions with human feedback (Ouyang et al, 2022)

Reinforcement Learning from Human Feedback (RLHF)

Repeat a lot:

- Sample new prompts.
- Generate answer using the language model.
- Calculate a reward using the reward model.
- Update the language model with favorite policy gradient method (they used PPO).

No humans in the loop.

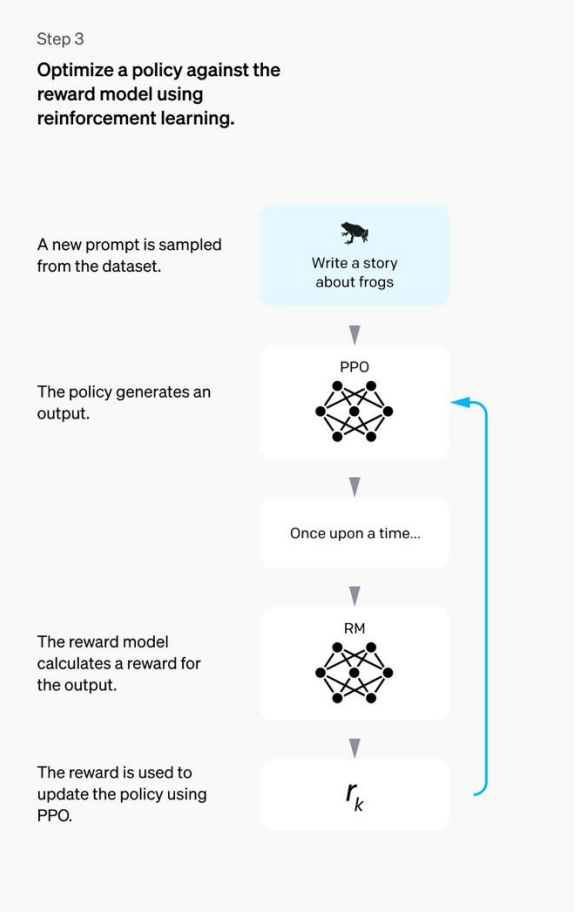


Image source: https://openai.com/index/instruction-following/
Paper: Training language models to follow instructions with human feedback (Ouyang et al, 2022)

Any Questions?

???

- What is reinforcement learning?
- Policies vs value functions
- Policy gradient methods
- AlphaGo
- Gran Turismo Sophy
- Language model tuning