Lecture 12: Policy Gradient
CS486/686 Intro to Artificial Intelligence

Pascal Poupart
David R. Cheriton School of Computer Science
CIFAR AI Chair at Vector Institute

UNIVERSITY OF WATERLOO

Outline

- Stochastic policy gradient
  - REINFORCE algorithm
- AlphaGo
- Large Language Models
  - Reinforcement Learning from Human Feedback

Model-free Policy-based Methods

- Q-learning
  - **Model-free value-based method**
  - No explicit policy representation
- Policy gradient
  - **Model-free policy-based method**
  - No explicit value function representation

Stochastic Policy

- Consider stochastic policy $\pi_\theta(a|s) = \Pr(a|s;\theta)$ parametrized by θ.

- Finitely many discrete actions

  Softmax: $\pi_\theta(a|s) = \frac{\exp(h(s,a;\theta))}{\sum_{a'} \exp(h(s,a';\theta))}$

  where $h(s,a;\theta)$ might be **linear** in θ: $h(s,a;\theta) = \sum_i \theta_i f_i(s,a)$

  or **non-linear** in θ: $h(s,a;\theta) = neuralNet(s,a;\theta)$

- Continuous actions:

  Gaussian: $\pi_\theta(a|s) = N(a|\mu(s;\theta), \Sigma(s;\theta))$

Supervised Learning

- Consider a stochastic policy $\pi_\theta(a|s)$

- Data: state-action pairs $\{(s_1, a_1^*), (s_2, a_2^*), \dots\}$

- Maximize log likelihood of the data

$$\theta^* = argmax_\theta \sum_n \log \pi_\theta(a_n^*|s_n)$$

- Gradient update

$$\theta_{n+1} \leftarrow \theta_n + \alpha_n \nabla_\theta \log \pi_\theta(a_n^*|s_n)$$

Reinforcement Learning

- Consider a stochastic policy $\pi_\theta(a|s)$

- Data: state-action-reward triples $\{(s_1, a_1, r_1), (s_2, a_2, r_2), \dots\}$

- Maximize discounted sum of rewards

$$\theta^* = argmax_\theta \boxed{\sum_n \gamma^n E_\theta[r_n|s_n, a_n]} = V_\theta(s_0)$$

- Gradient update

$$\theta_{n+1} \leftarrow \theta_n + \alpha_n \boxed{\gamma^n G_n} \nabla_\theta \log \pi_\theta(a_n|s_n)$$

where $G_n = \sum_{t=0}^{\infty} \gamma^t r_{n+t}$

Stochastic Gradient Policy Theorem

- Stochastic Gradient Policy Theorem

$$\boxed{\nabla_\theta V_\theta(s_0) \propto \sum_s \mu_\theta(s) \sum_a \nabla_\theta\, \pi_\theta(a|s)\, Q_\theta(s,a)}$$

$\mu_\theta(s)$: stationary state distribution when executing policy parametrized by θ

$Q_\theta(s,a)$: discounted sum of rewards when starting in s, executing a and following the policy parametrized by θ thereafter.

Derivation

$$
\begin{aligned}
&\nabla_\theta V_\theta(s_0) = \nabla_\theta\left[\sum_{a_0} \pi_\theta(a_0|s_0)\, Q_\theta(s_0,a_0)\right] \qquad \forall s_0 \in S \\
&= \sum_{a_0}[\nabla\, \pi_\theta(a_0|s_0)\, Q_\theta(s_0,a_0) + \pi_\theta(a_0|s_0)\, \nabla Q_\theta(s_0,a_0)] \\
&= \sum_{a_0}\left[\nabla\, \pi_\theta(a_0|s_0)\, Q_\theta(s_0,a_0) + \pi_\theta(a_0|s_0)\, \nabla \sum_{s_1,r_0} \Pr(s_1,r_0|s_0,a_0)\left(r_0+\gamma V_\theta(s_1)\right)\right] \\
&= \sum_{a_0}\left[\nabla\, \pi_\theta(a_0|s_0)\, Q_\theta(s_0,a_0) + \pi_\theta(a_0|s_0) \sum_{s_1} \gamma \Pr(s_1|s_0,a_0)\, \nabla \mathrm{V}_\theta(s_1)\right] \\
&= \sum_{a_0}[\nabla\, \pi_\theta(a_0|s_0)\, Q_\theta(s_0,a_0) + \pi_\theta(a_0|s_0) \sum_{s_1} \gamma \Pr(s_1|s_0,a_0) \\
&\qquad\qquad \sum_{a_1}[\nabla\, \pi_\theta(a_1|s_1)\, Q_w(s_1,a_1) + \pi_\theta(a_1|s_1) \sum_{s_2} \gamma \Pr(s_2|s_1,a_1)\, \nabla \mathrm{V}_\theta(s_2)] \\
&= \sum_{s\in S} \sum_{n=0}^{\infty} \gamma^n \underbrace{\Pr(s_0 \to s; n, \theta)}_{\text{Probability of reaching } s \text{ from } s_0 \text{ at time step } n} \sum_a \nabla\, \pi_\theta(a|s)\, Q_\theta(s,a)
\end{aligned}
$$

Since $\mu_\theta(s) \propto \sum_{n=0}^{\infty} \gamma^n \Pr(s_0 \to s; n, \theta)$ then

$$
\propto \sum_s \mu_\theta(s) \sum_a \nabla_\theta\, \pi_\theta(a|s)\, Q_\theta(s,a)
$$

REINFORCE: Monte Carlo Policy Gradient

random variable

- $\nabla_\theta V_\theta(s_0) = \sum_{s\in S}\sum_{n=0}^{\infty}\gamma^n \Pr(s_0 \to s; n,\theta)\sum_a \nabla_\theta\, \pi_\theta(a|s)\, Q_\theta(s,a)$

$$= E_\theta[\textstyle\sum_{n=0}^{\infty}\gamma^n \sum_a Q_\theta(S_n, a)\nabla_\theta\, \pi_\theta(a|S_n)]$$

$$= E_\theta\left[\sum_{n=0}^{\infty}\gamma^n \sum_a \pi_\theta(a|S_n)\, Q_\theta(S_n,a)\,\frac{\nabla_\theta\, \pi_\theta(a|S_n)}{\pi_\theta(a|S_n)}\right]$$

$$= E_\theta\left[\sum_{n=0}^{\infty}\gamma^n\, Q_\theta(S_n, A_n)\,\frac{\nabla_\theta\, \pi_\theta(A_n|S_n)}{\pi_\theta(A_n|S_n)}\right]$$

$$= E_\theta\left[\sum_{n=0}^{\infty}\gamma^n\, G_n\,\frac{\nabla_\theta \pi_\theta(A_n|S_n)}{\pi_\theta(A_n|S_n)}\right]$$

$$= E_\theta[\textstyle\sum_{n=0}^{\infty}\gamma^n\, G_n \nabla_\theta \log \pi_\theta(A_n|S_n)]$$

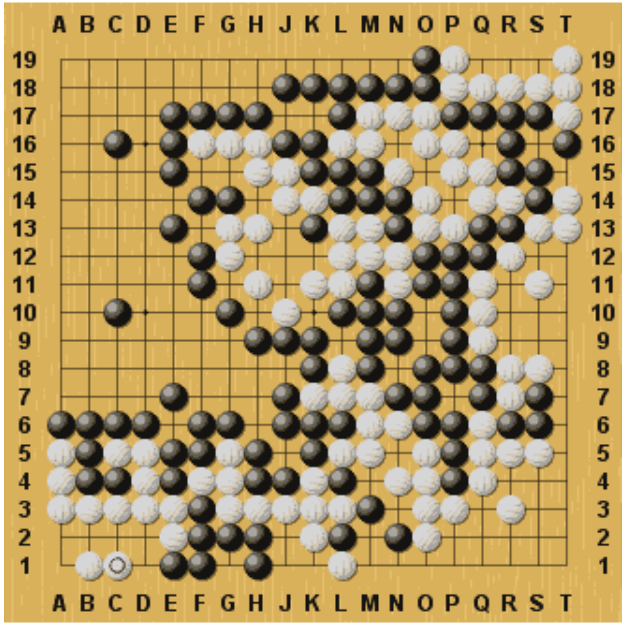
- Stochastic gradient at time step n
  $\nabla V_\theta \approx\ \gamma^n G_n \nabla_\theta \log \pi_\theta(a_n|s_n)$

REINFORCE Algorithm (stochastic policy)

```
REINFORCE($s_0$)
    Initialize $\pi_\theta$ to anything
    Loop forever (for each episode)
        Generate episode $s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T$ with $\pi_\theta$
        Loop for each step of the episode $n = 0, 1, \ldots, T$
            $G_n \leftarrow \sum_{t=0}^{T-n} \gamma^t \, r_{n+t}$
            Update policy: $\theta \leftarrow \theta + \alpha \, \gamma^n G_n \nabla_\theta \log \pi_\theta(a_n|s_n)$
    Return $\pi_\theta$
```

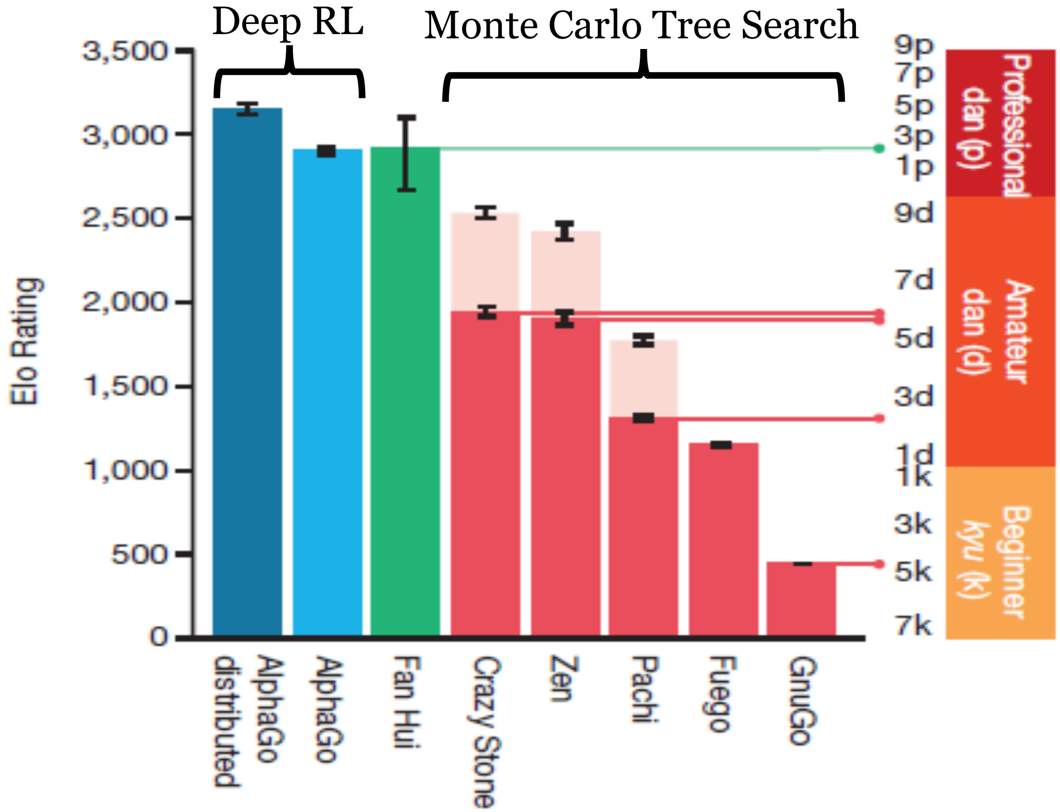
Example: Game of Go

- (simplified) rules:
  - Two players (black and white)
  - Players alternate to place a stone of their color on a vacant intersection.
  - Connected stones without any liberty (i.e., no adjacent vacant intersection) are captured and removed from the board
  - Winner: player that controls the largest number of intersections at the end of the game



Computer Go

October 2015:



Computer Go

- March 2016: AlphaGo defeats Lee Sedol (9-dan)

“[AlphaGo] can’t beat me” Ke Jie (world champion)

- May 2017: AlphaGo defeats Ke Jie (world champion)

“Last year, [AlphaGo] was still quite humanlike when it played.
But this year, it became like a god of Go” Ke Jie (world champion)

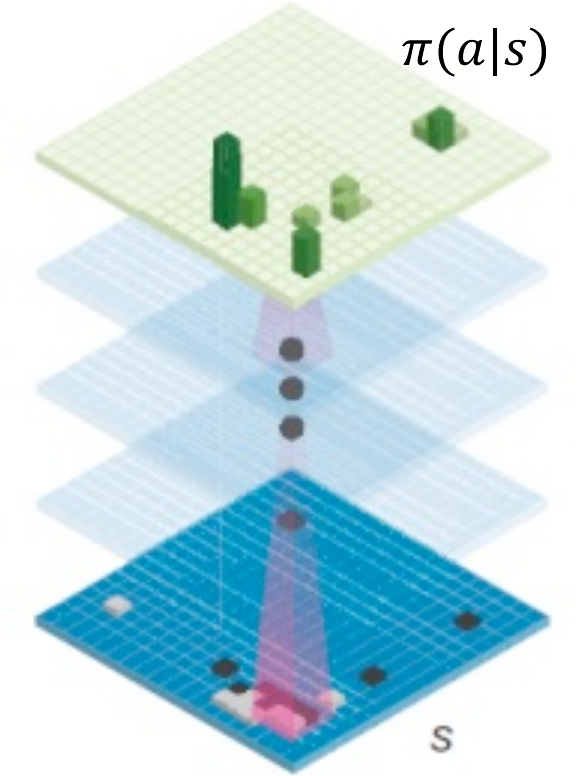
Winning Strategy

Four steps:

1. Supervised Learning of Policy Networks
2. Policy gradient with Policy Networks
3. Value gradient with Value Networks
4. Searching with Policy and Value Networks

Policy Network

- Train policy network to imitate Go experts based on a database of 30 million board configurations from the KGS Go Server.
- Policy network: $\pi(a|s)$
  - Input: state s (board configuration)
  - Output: distribution over actions a (intersection on which the next stone will be placed)



Supervised Learning of the Policy Network

- Let θ be the weights of the policy network

- Training:
  - Data: suppose a is optimal in s
  - Objective: maximize $\log \pi_\theta(\mathrm{a}|\mathrm{s})$
  - Gradient: $\nabla_\theta = \frac{\partial \log \pi_\theta(a|s)}{\partial \theta}$
  - Weight update: $\theta \leftarrow \theta + \alpha \nabla_\theta$

Policy Gradient for the Policy Network

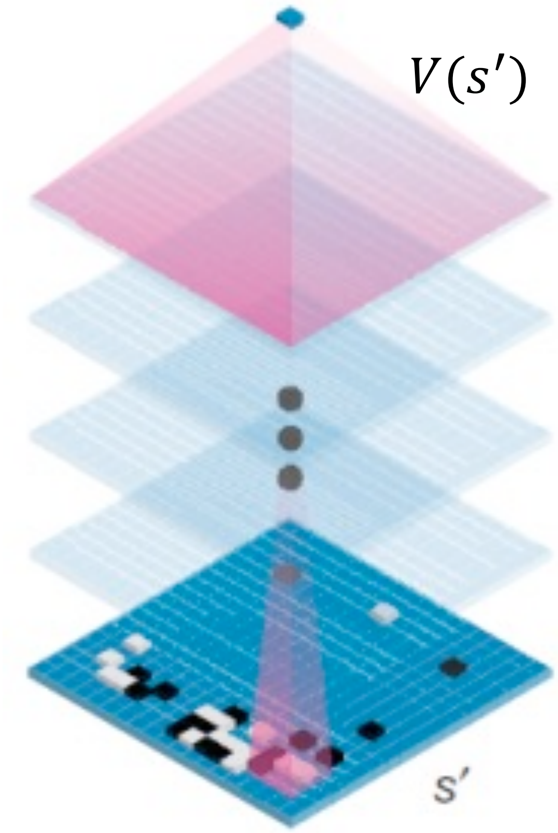
- How can we update a policy network based on reinforcements instead of the optimal action?
- Let $G_n = \sum_t \gamma^t \, r_{n+t}$ be the discounted sum of rewards in a trajectory that starts in s at time n by executing a.
- Gradient: $\nabla_\theta = \frac{\partial \, log \, \pi_\theta(a|s)}{\partial\theta} \, \gamma^n G_n$
  - Intuition: rescale supervised learning gradient by G_n
- Policy update: $\theta \leftarrow \theta + \alpha\nabla_\theta$

Policy Gradient for the Policy Network

- In computer Go, program repeatedly plays against its former self.
- For each game $G_n = \begin{cases} 1 & win \\ -1 & lose \end{cases}$
- For each (s_n, a_n) at turn n of the game, assume $\gamma = 1$ and compute
  - Gradient: $\nabla_\theta = \frac{\partial\, log\, \pi_\theta(a|s)}{\partial\theta}\, \gamma^n G_n$
  - Policy update: $\theta \leftarrow \theta + \alpha \nabla_\theta$

Value Network

- Predict $V(s')$ (i.e., who will win game) in each state s' with a value network
  - Input: state s (board configuration)
  - Output: expected discounted sum of rewards $V(s')$



Gradient Value Learning with Value Networks

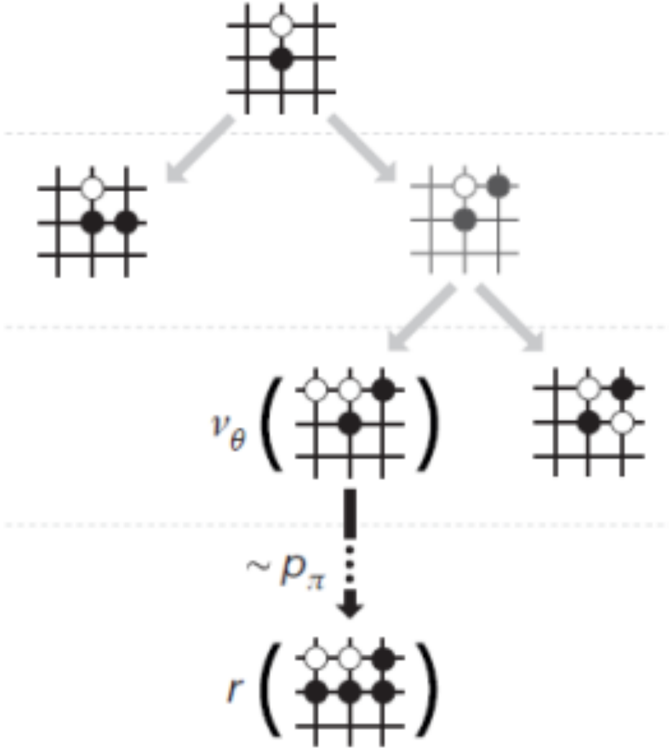
- Let $\boldsymbol{w}$ be the weights of the value network
- Training:
  - Data: (s, G) where $G = \begin{cases} 1 & win \\ -1 & lose \end{cases}$
  - Objective: minimize $\frac{1}{2}(V_{\boldsymbol{w}}(s) - G)^2$
  - Gradient: $\nabla_{\boldsymbol{w}} = \frac{\partial V_{\boldsymbol{w}}(s)}{\partial \boldsymbol{w}}(V_{\boldsymbol{w}}(s) - G)$
  - Weight update: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}}$

Searching with Policy and Value Networks

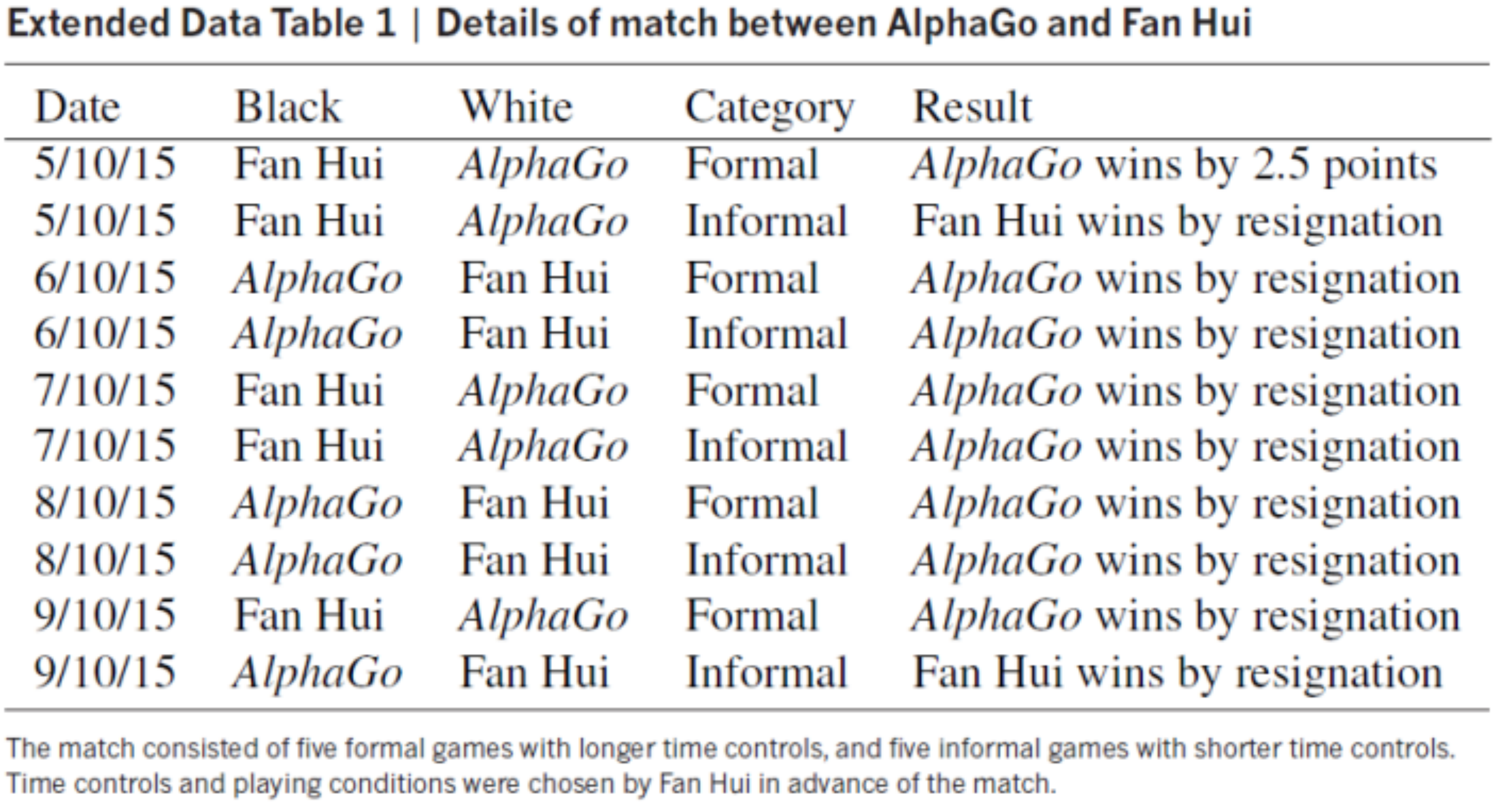
- AlphaGo combines policy and value networks into a **Monte Carlo Tree Search (MCTS)** algorithm

- Idea: construct a search tree
  - Node: s
  - Edge: a

- We will discuss MCTS next lecture



Competition

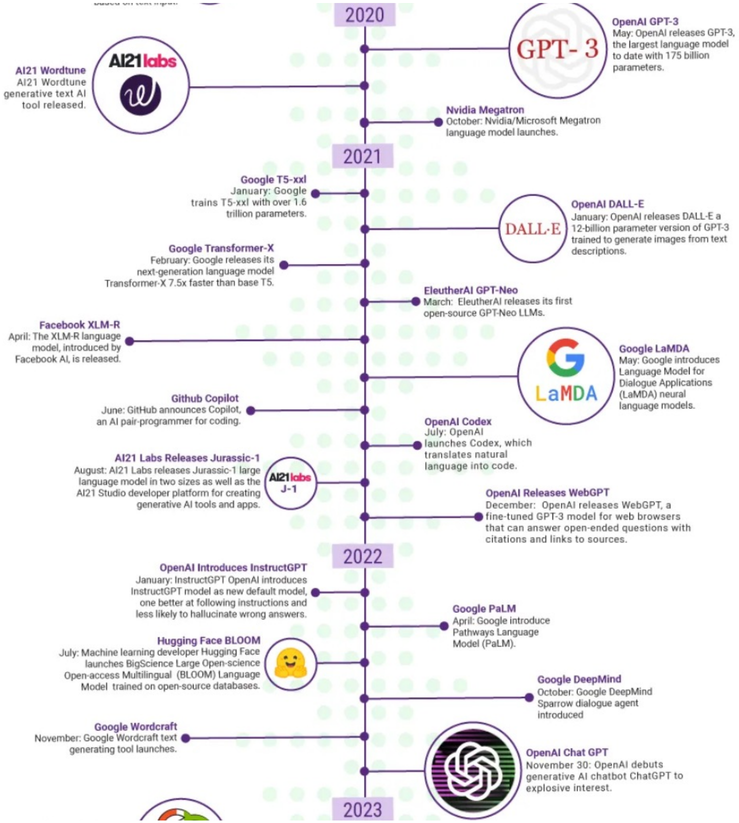


Extended Data Table 1 | Details of match between AlphaGo and Fan Hui

Date	Black	White	Category	Result
5/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by 2.5 points
5/10/15	Fan Hui	*AlphaGo*	Informal	Fan Hui wins by resignation
6/10/15	*AlphaGo*	Fan Hui	Formal	*AlphaGo* wins by resignation
6/10/15	*AlphaGo*	Fan Hui	Informal	*AlphaGo* wins by resignation
7/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by resignation
7/10/15	Fan Hui	*AlphaGo*	Informal	*AlphaGo* wins by resignation
8/10/15	*AlphaGo*	Fan Hui	Formal	*AlphaGo* wins by resignation
8/10/15	*AlphaGo*	Fan Hui	Informal	*AlphaGo* wins by resignation
9/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by resignation
9/10/15	*AlphaGo*	Fan Hui	Informal	Fan Hui wins by resignation

The match consisted of five formal games with longer time controls, and five informal games with shorter time controls. Time controls and playing conditions were chosen by Fan Hui in advance of the match.

Large Language Models (up to Sept 2023)



2020

OpenAI GPT-3
May: OpenAI releases GPT-3, the largest language model to date with 175 billion parameters.

AI21 Wordtune
AI21 Wordtune generative text AI tool released.

Nvidia Megatron
October: Nvidia/Microsoft Megatron language model launches.

2021

Google T5-xxl
January: Google trains T5-xxl with over 1.6 trillion parameters.

OpenAI DALL-E
January: OpenAI releases DALL-E a 12-billion parameter version of GPT-3 trained to generate images from text descriptions.

Google Transformer-X
February: Google releases its next-generation language model Transformer-X 7.5x faster than base T5.

EleutherAI GPT-Neo
March: EleutherAI releases its first open-source GPT-Neo LLMs.

Facebook XLM-R
April: The XLM-R language model, introduced by Facebook AI, is released.

Google LaMDA
May: Google introduces Language Model for Dialogue Applications (LaMDA) neural language models.

Github Copilot
June: GitHub announces Copilot, an AI pair-programmer for coding.

OpenAI Codex
July: OpenAI launches Codex, which translates natural language into code.

AI21 Labs Releases Jurassic-1
August: AI21 Labs releases Jurassic-1 large language model in two sizes as well as the AI21 Studio developer platform for creating generative AI tools and apps.

OpenAI Releases WebGPT
December: OpenAI releases WebGPT, a fine-tuned GPT-3 model for web browsers that can answer open-ended questions with citations and links to sources.

2022

OpenAI Introduces InstructGPT
January: InstructGPT OpenAI introduces InstructGPT model as new default model, one better at following instructions and less likely to hallucinate wrong answers.

Google PaLM
April: Google introduce Pathways Language Model (PaLM).

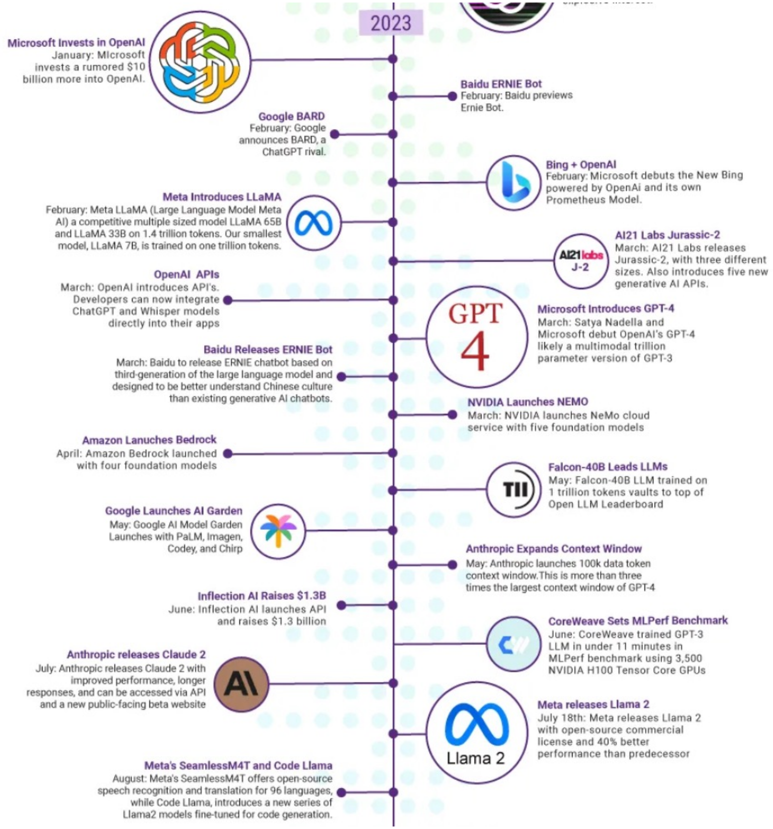
Hugging Face BLOOM
July: Machine learning developer Hugging Face launches BigScience Large Open-science Open-access Multilingual (BLOOM) Language Model trained on open-source databases.

Google DeepMind
October: Google DeepMind Sparrow dialogue agent introduced

Google Wordcraft
November: Google Wordcraft text generating tool launches.

OpenAI Chat GPT
November 30: OpenAI debuts generative AI chatbot ChatGPT to explosive interest.

2023



2023

Microsoft Invests in OpenAI
January: Microsoft invests a rumored $10 billion more into OpenAI.

Baidu ERNIE Bot
February: Baidu previews Ernie Bot.

Google BARD
February: Google announces BARD, a ChatGPT rival.

Bing + OpenAI
February: Microsoft debuts the New Bing powered by OpenAi and its own Prometheus Model.

Meta Introduces LLaMA
February: Meta LLaMA (Large Language Model Meta AI) a competitive multiple sized model LLaMA 65B and LLaMA 33B on 1.4 trillion tokens. Our smallest model, LLaMA 7B, is trained on one trillion tokens.

AI21 Labs Jurassic-2
March: AI21 Labs releases Jurassic-2, with three different sizes. Also introduces five new generative AI APIs.

OpenAI APIs
March: OpenAI introduces API's. Developers can now integrate ChatGPT and Whisper models directly into their apps

Microsoft Introduces GPT-4
March: Satya Nadella and Microsoft debut OpenAI's GPT-4 likely a multimodal trillion parameter version of GPT-3

Baidu Releases ERNIE Bot
March: Baidu to release ERNIE chatbot based on third-generation of the large language model and designed to be better understand Chinese culture than existing generative AI chatbots.

NVIDIA Launches NEMO
March: NVIDIA launches NeMo cloud service with five foundation models

Amazon Lanuches Bedrock
April: Amazon Bedrock launched with four foundation models

Falcon-40B Leads LLMs
May: Falcon-40B LLM trained on 1 trillion tokens vaults to top of Open LLM Leaderboard

Google Launches AI Garden
May: Google AI Model Garden Launches with PaLM, Imagen, Codey, and Chirp

Anthropic Expands Context Window
May: Anthropic launches 100k data token context window.This is more than three times the largest context window of GPT-4

Inflection AI Raises $1.3B
June: Inflection AI launches API and raises $1.3 billion

CoreWeave Sets MLPerf Benchmark
June: CoreWeave trained GPT-3 LLM in under 11 minutes in MLPerf benchmark using 3,500 NVIDIA H100 Tensor Core GPUs

Anthropic releases Claude 2
July: Anthropic releases Claude 2 with improved performance, longer responses, and can be accessed via API and a new public-facing beta website

Meta releases Llama 2
July 18th: Meta releases Llama 2 with open-source commercial license and 40% better performance than predecessor

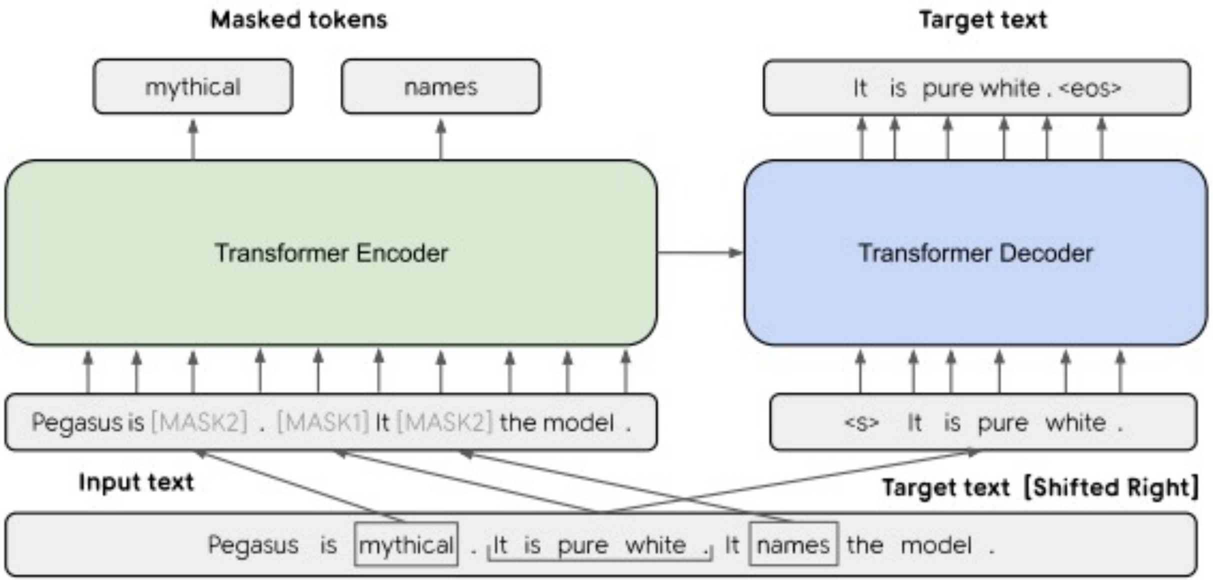
Meta's SeamlessM4T and Code Llama
August: Meta's SeamlessM4T offers open-source speech recognition and translation for 96 languages, while Code Llama, introduces a new series of Llama2 models fine-tuned for code generation.

Credit: https://synthedia.substack.com/p/a-timeline-of-large-language-model

UNIVERSITY OF WATERLOO

Self-Supervised Learning in LLMs

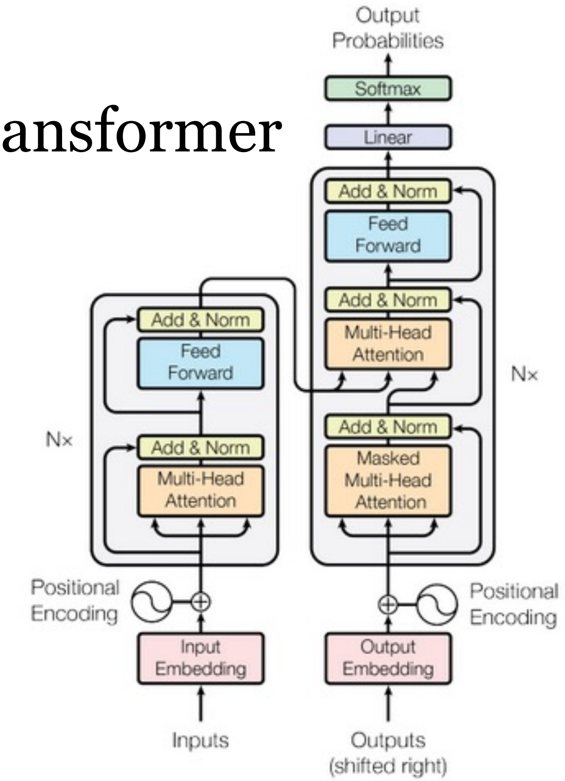
Encoder and Decoder



Masked tokens
mythical
names
Target text
It is pure white . <eos>
Transformer Encoder
Transformer Decoder
Pegasus is [MASK2] . [MASK1] It [MASK2] the model .
<s> It is pure white .
Input text
Target text [Shifted Right]
Pegasus is mythical . It is pure white . It names the model .

Credit: Zhang et al., 2020

Transformer



Credit: Vaswani et al., 2017

GPT Models

InstructGPT (GPT-3)
January 27, 2022

Reinforcement Learning from human feedback

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

ChatGPT (GPT-3.5)
November 30, 2022

Multi-turn conversations

No techreport

GPT-4
March 14, 2023

Multi-modal (text and images)

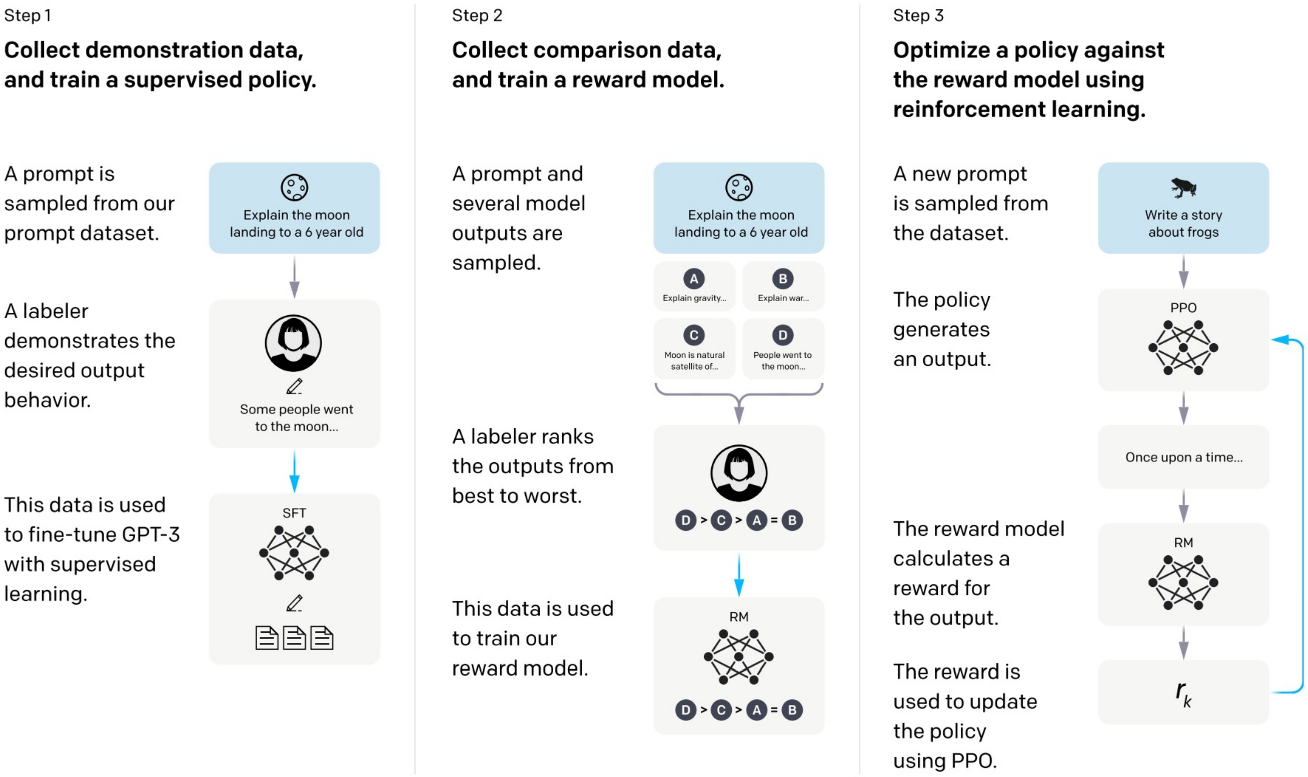
GPT-4 Technical Report (2023)

GPT Omni GPT-4o
May 13, 2024

End-to-end Multi-modal (text, audio and images)

https://openai.com/index/hello-gpt-4o/

Learning from Human Feedback



Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

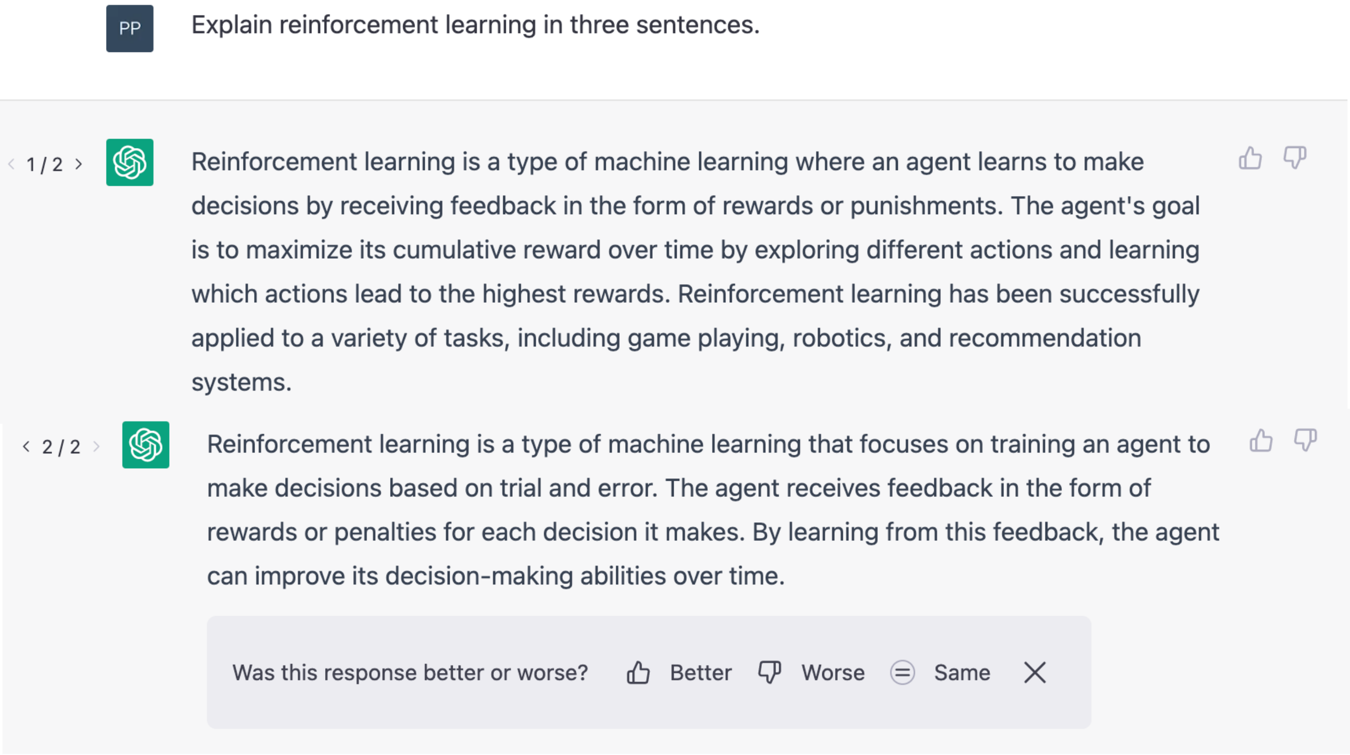
The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

Explain the moon landing to a 6 year old
Some people went to the moon...
SFT
Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B
Write a story about frogs
PPO
Once upon a time...
RM
r_k

Response Ranking



Explain reinforcement learning in three sentences.

Reinforcement learning is a type of machine learning where an agent learns to make decisions by receiving feedback in the form of rewards or punishments. The agent's goal is to maximize its cumulative reward over time by exploring different actions and learning which actions lead to the highest rewards. Reinforcement learning has been successfully applied to a variety of tasks, including game playing, robotics, and recommendation systems.

Reinforcement learning is a type of machine learning that focuses on training an agent to make decisions based on trial and error. The agent receives feedback in the form of rewards or penalties for each decision it makes. By learning from this feedback, the agent can improve its decision-making abilities over time.

Was this response better or worse?

Reward Model

Stiennon, Ouyang, Wu, Ziegler, Lowe Voss, Radford, Amodei, Christiano (2020) **Learning to summarize from human feedback**, *NeurIPS*.

- $\boldsymbol{s}$: user prompt
- $\boldsymbol{a}$: system response
- Reward function: $\boldsymbol{r_\theta(s, a)} = real\ number$
- Consider several possible responses $a_1 \succcurlyeq a_2 \succcurlyeq \cdots \succcurlyeq a_k$ ranked by annotator
- Training reward function to be consistent with the ranking:

$$\boxed{Loss(\theta) = -\frac{1}{\binom{k}{2}} E_{(s,a_i,a_j)\in Dataset} \log \sigma\left(r_\theta(s, a_i) - r_\theta(s, a_j)\right)}$$

UNIVERSITY OF WATERLOO

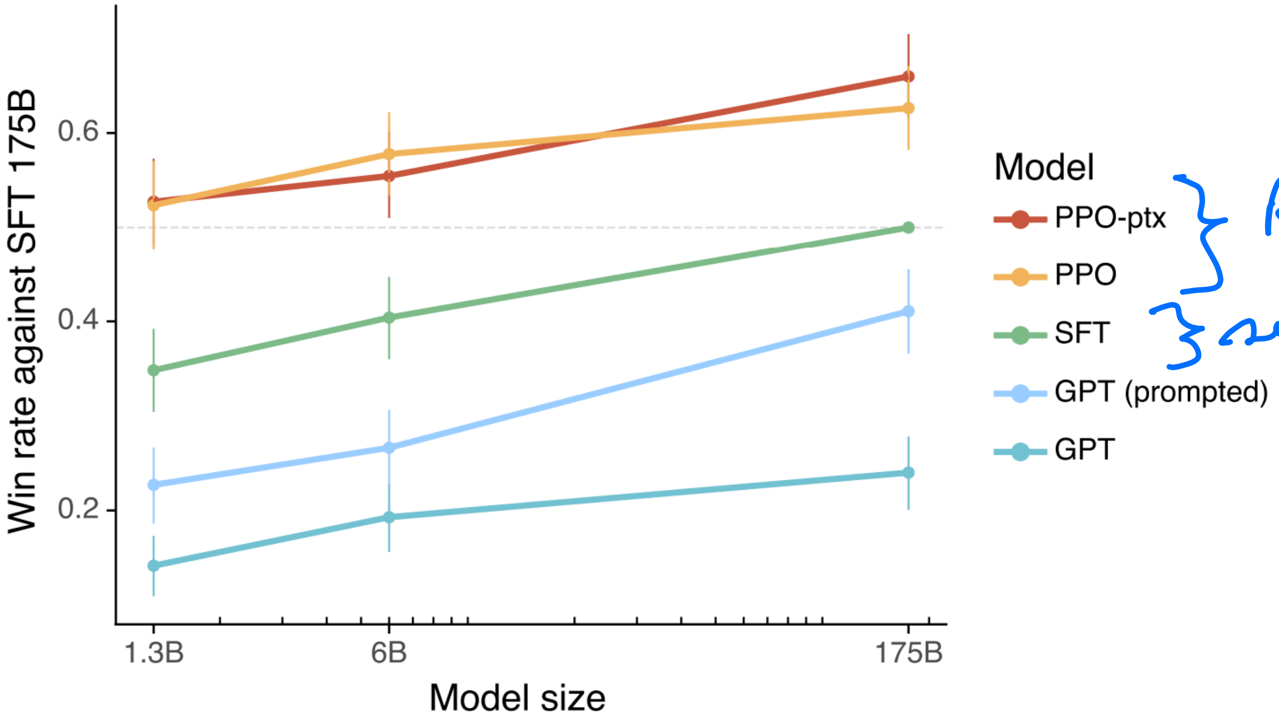
Reinforcement Learning

Ouyang, Wu, Jiang, Wainwright, et al. (2022) **Training language models to follow instructions with human feedback**, *NeurIPS*.

- Pretrain language model (GPT-3)
- Fine-Tune GPT-3 by RL to obtain InstructGPT
    - Policy (language model): $\pi_\phi(s) = a$
    - Optimize $\pi_\phi(s)$ by policy gradient (PPO)

$$\boxed{\max_\phi E_{s \in Dataset}\left[E_{a\sim\pi_\phi(a|s)}[r_\theta(s,a)] - \beta\, KL\left(\pi_\phi(\cdot\,|s)\middle|\pi_{ref}(\cdot\,|s)\right)\right]}$$

InstructGPT Results



} RL

} supervised learning